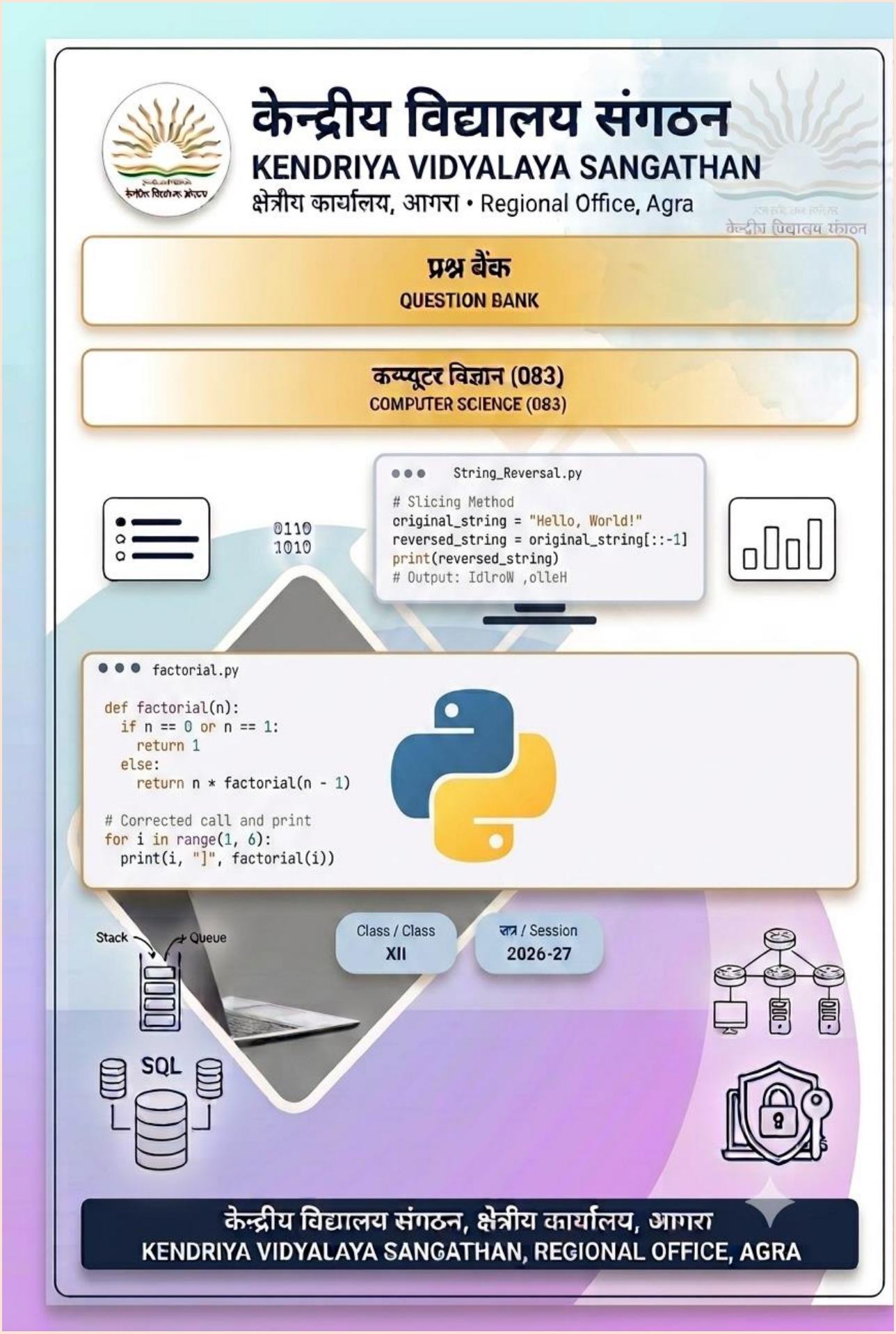

केन्द्रीय विद्यालय संगठन
KENDRIYA VIDYALAYA SANGATHAN
क्षेत्रीय कार्यालय, आगरा • Regional Office, Agra
प्रश्न बैंक
QUESTION BANK
कम्प्यूटर विज्ञान (083)
COMPUTER SCIENCE (083)
String_Reversal.py
# Slicing Method
original_string = "Hello, World!"
reversed_string = original_string[::-1]
print(reversed_string)
# Output: IdlroW ,olleH
0110
1010
factorial.py
def factorial(n):
if n == 0 or n == 1:
return 1
else:
return n * factorial(n - 1)
# Corrected call and print
for i in range(1, 6):
print(i, "]", factorial(i))
Stack
Queue
SQL
Class / Class
XII
सत्र / Session
2026-27
केन्द्रीय विद्यालय संगठन, क्षेत्रीय कार्यालय, आगरा
KENDRIYA VIDYALAYA SANGATHAN, REGIONAL OFFICE, AGRA

KENDRIYA VIDYALAYA SANGATHAN

(AGRA REGION)

QUESTION BANK / VIDEO LECTURE LINKS

COMPUTER SCIENCE

CLASS XII

SESSION: 2026-27

संदेश

मुझे यह जानकर अत्यंत हर्ष हो रहा है कि केन्द्रीय विद्यालय संगठन, क्षेत्रीय कार्यालय आगरा के मार्गदर्शन में ***कक्षा 12 कंप्यूटर विज्ञान प्रश्न बैंक*** तैयार किया गया है। यह प्रश्न बैंक विद्यार्थियों की शैक्षणिक आवश्यकताओं को ध्यान में रखते हुए तैयार किया गया है ताकि वे आत्मविश्वास के साथ बोर्ड परीक्षा में सफलता प्राप्त कर सकें।

कंप्यूटर विज्ञान एक ऐसा विषय है जिसमें सैद्धांतिक समझ के साथ-साथ व्यावहारिक ज्ञान भी अत्यंत आवश्यक है। इस प्रश्न बैंक के माध्यम से विद्यार्थियों को विभिन्न प्रकार के प्रश्नों का अभ्यास करने का अवसर मिलेगा, जिससे उनकी तार्किक क्षमता, समस्या समाधान कौशल तथा विश्लेषणात्मक सोच का विकास होगा।

मैं इस प्रश्न बैंक के निर्माण में योगदान देने वाले सभी विषय विशेषज्ञों, शिक्षकों एवं अकादमिक दल के प्रति हार्दिक आभार व्यक्त करती हूँ, जिनके समर्पण और परिश्रम से यह संकलन संभव हो पाया है।

मैं सभी विद्यार्थियों से अपेक्षा करती हूँ कि वे इस प्रश्न बैंक का नियमित अभ्यास करें, विषय को जिज्ञासा एवं लगन के साथ समझें तथा आत्मविश्वास के साथ परीक्षा में सफलता प्राप्त करें। मुझे पूर्ण विश्वास है कि यह प्रयास विद्यार्थियों को उत्कृष्ट परिणाम दिलाने में सहायक सिद्ध होगा।

सभी विद्यार्थियों को उज्ज्वल भविष्य के लिए मेरी शुभकामनाएँ।

(श्रीमती इंदिरा मुद्गल)
उपायुक्त
केन्द्रीय विद्यालय संगठन
क्षेत्रीय कार्यालय, आगरा

PATRON

Smt. Indira Mudgal, Deputy Commissioner, KVS RO AGRA

CO-PATRON

Smt. Rani Dange, Assistant Commissioner, KVS RO AGRA
Shri Raj Kumar, Assistant Commissioner, KVS RO AGRA

CO-ORDINATOR

Shri Manoj Kumar, Principal, PM SHRI KV MURADNAGAR

EDITORS

Smt. Shilpi Singh, Vice Principal, PM SHRI KV No.2 HINDAN

Shri Deepak Kumar Sharma, PGT CS, PM SHRI KV OEF HAZRATPUR

COVER DESION

Shri Sachin Bhardwaj, PM SHRI KV NO. 1 AFS HINDAN

OVERALL COMPILATION COORDINATOR

Shri Pankaj Singh, PGT CS, PM SHRI KV No 3 JHANSI

CONTENT COMPILATION TEAM

PGT COMPUTER SCIENCE	KENDRIYA VIDYALAYA
SHRI. PANKAJ SINGH	PM SHRI KENDRIYA VIDYALAYA No 3 JHANSI
SHRI. YOGESH BAJAJ	PM SHRI KENDRIYA VIDYALAYA AIR FORCE STATION SARSAWA
SHRI. DEEPAK KUMAR SHARMA	PM SHRI KENDRIYA VIDYALAYA OEF HAZRATPUR

CONTENT PREPARATION TEAM

SNo	PGT COMPUTER SCIENCE WITH KV NAME	TOPICS / CHAPTERS NAME
1	SMT. SUMAN VERMA KV NTPC DADRI	REVISION OF PYTHON TOPICS COVERED IN CLASS XI.
2	SMT. ARCHANA GUPTA PM SHRI KV NO.3 AGRA CANTT	FUNCTIONS AND EXCEPTION HANDLING
3	SMT. SANGEETA PAL PM SHRI KV NO.2 AFS HINDAN	TEXT FILE HANDLING AND STACK
4	SHRI. SACHIN BHARDWAJ PM SHRI KV NO.1 AFS HINDAN	BINARY FILE HANDLING AND CSV FILE HANDLING
5	SMT. KIRAN SINGH PM SHRI KV KNN GHAZIABAD SHIFT (I)	COMPUTER NETWORKS
6	SMT. NAUSHEEN KHAN PM SHRI KV CRPF RAMPUR	DATABASE MANAGEMENT AND DATABASE CONNECTIVITY

DISTRIBUTION OF MARKS

CLASS XII: COMPUTER SCIENCE (083) (THEORY-70 MARKS)

UNIT NO.	UNIT NAME	MARKS
1	COMPUTATIONAL THINKING AND PROGRAMMING – 2	40
2	COMPUTER NETWORKS	10
3	DATABASE MANAGEMENT	20

CLASS XII: COMPUTER SCIENCE (083) (PRACTICAL-30 MARKS)

S. NO.	COMPONENT / DESCRIPTION	MARKS
1	LAB TEST — Python Program (60% Logic + 20% Documentation + 20% Code Quality)	8
	SQL Queries (4 queries based on one or two tables)	4
2	REPORT FILE — Minimum 15 Python programs; SQL Queries: minimum 5 sets using one or two tables; Minimum 4 programs based on Python–SQL Connectivity	7
3	PROJECT (Using concepts learnt in Classes 11 and 12)	8
4	VIVA VOCE	3

Approximate Chapter-wise Weightage

(Based on previous year paper analysis – not officially released by CBSE)

CHAPTER	APPROX. MARKS
Revision Tour	12
Functions	8
File Handling	14
Stack	3
Network Handling	14
Database Management System (DBMS)	13
Python Connectivity	6

Video Lecture Links

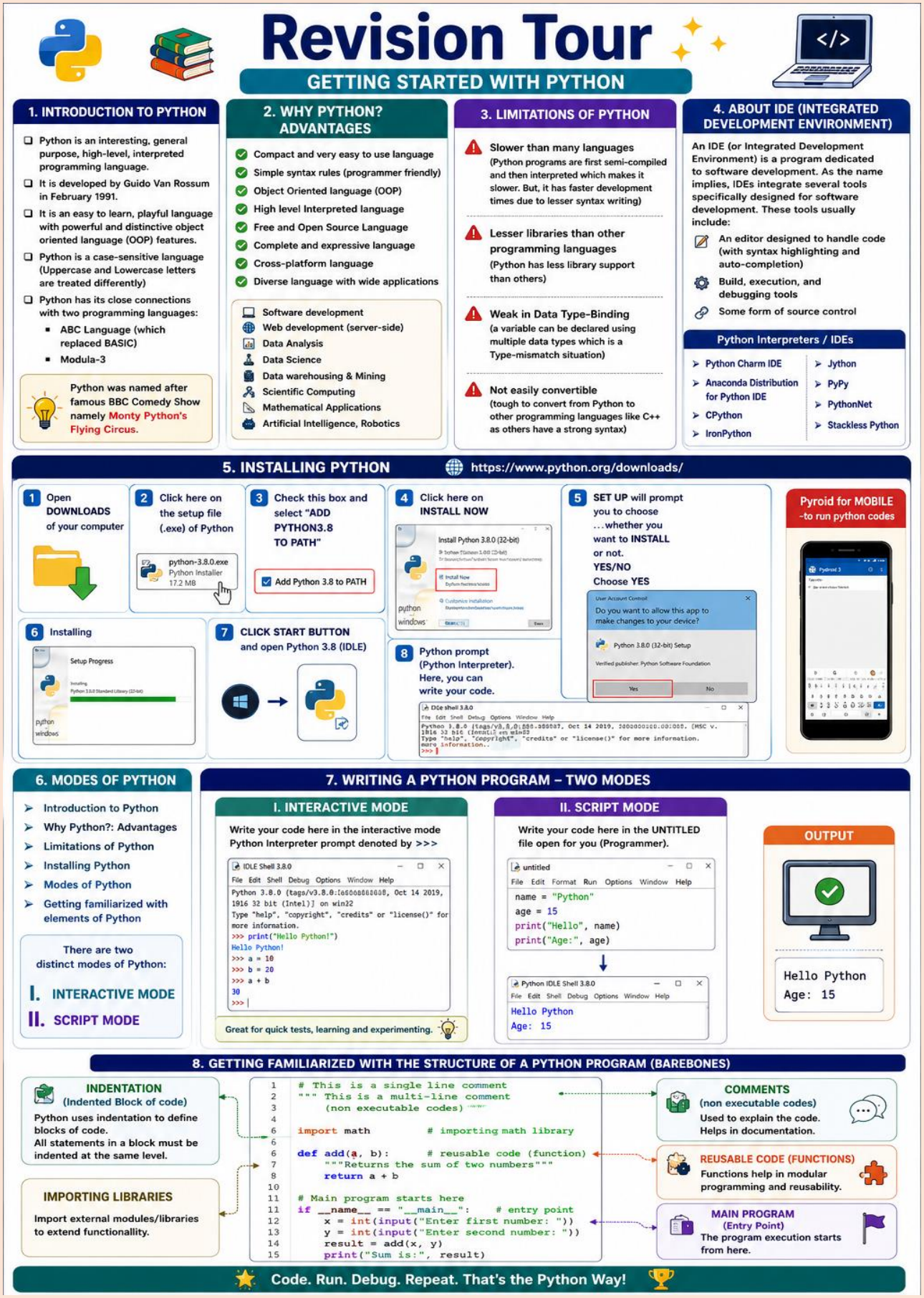
CLASS: XII SUBJECT: (083) COMPUTER SCIENCE

Sno	Name of Teacher With KV	Topic Allotted	Link of Video Lecture
1	Ms. Nidhi Yagnik Surajpur	Revision of Class XI Python	https://www.youtube.com/watch?v=81wY45_aEjQ (Part 1 of 2) https://www.youtube.com/watch?v=sVbTQMTHb9s (Part 2 of 2)
2	Mr. Pankaj Singh KV No 3 Jhansi	Functions	https://www.youtube.com/watch?v=mMyGs70P4wc&t=7s (Part 1 of 4) https://www.youtube.com/watch?v=pgzXH74fuG4&t=4s (Part 2 of 4) https://www.youtube.com/watch?v=KuRRapermGc&t=4s (Part 3 of 4) https://www.youtube.com/watch?v=JM47fhH84SQ&t=8s (Part 4 of 4)
3	Mr. Naveen K. Yadav PL Meerut	Text File Handling	https://www.youtube.com/watch?v=Ix1gie3wl4I

4	Mr. L. K. Tyagi Sec 24 Noida	Stack	https://drive.google.com/drive/folders/1-vCVoS0SgbUJvm5gtTVY3ZL10V58IekS
5	Ms. Sunita Gupta Muradnagar	Binary File Handling	https://drive.google.com/file/d/1dzOR_OvV1B-2VJb8jsIoLiqIHlxXZOcq/view
6	Mr. Sunit Kumar Muzaffarnagar	CSV File Handling	https://drive.google.com/file/d/1EBksxKfo1NfCEyCLwgKJt_jAibpP8X42/view?usp=sharing
7	Mr. Sachin Bhardwaj No 1 AFS Hindon	Computer Networks	https://youtu.be/i5PrPW-Xck4?si=5chthadJywM7FI_S (Part 1 of 3) https://youtu.be/NtXwdwNvM9o?si=V9GFEl148AUTnurk (Part 2 of 3) https://youtu.be/w3H3oSzZZPc?si=FdbqEiIy-lQbAm4j (Part 3 of 3)

8	Ms. Rajni Kardam Grt. Noida	DBMS / SQL	https://youtu.be/XoUF5VyG4-M?si=hEVbMq2sPM46ss_B (DBMS Concepts) https://youtu.be/IrewSI3ylUE?si=-M-7xc9CnDpqR10F (SQL Revision)
	 		
9	Mr. Pankaj Singh KV No 3 Jhansi	Database Connectivity	https://www.youtube.com/watch?v=fufEOuzpEio&t=3 (Part 1 of 2) https://www.youtube.com/watch?v=Lpj9OHzqD0I&t=6s (Part 2 of 2)
	 		

CHAPTER NAME: PYTHON REVISION TOUR



Revision Tour

1. PYTHON CHARACTER SET

A set of valid characters recognized by Python. Python uses the traditional ASCII character set. The latest version recognizes the Unicode character set. The ASCII character set is a subset of the Unicode character set.

- **Letters** A-Z , a-z
- **Digits** 0-9
- **Special symbols** Special symbol available over keyboard
- **White spaces** blank space, tab, carriage return, new line, form feed
- **Other characters** Unicode

2. TOKENS (KILPO)

Smallest individual unit in a program is called a token. They are:

- K Keywords
- I Identifiers
- L Literals
- P Punctuators
- O Operators

My funda of remembering Tokens: KILPO → it's just a shortcut given by me for remembrance ;)

3. KEYWORDS

Keywords are the words that convey a special meaning to the language compiler/interpreter. They are reserved for special purpose and must not be used as normal identifier names.

and	exec	not
as	finally	or
assert	for	pass
break	from	print
class	global	raise
continue	if	return
def	import	try
del	in	while
elif	is	with
else	lambda	yield
except		

Python is a case sensitive language.

4. IDENTIFIERS

Identifiers are the fundamental building blocks of a program which are used to identify a variable, function name, class name, module name or any object.

RULES OF WRITING AN IDENTIFIER

- An identifier starts with a letter A to Z, a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Identifier name can NOT start with a digit.
- Python does NOT allow special characters other than underscore (_).
- Identifier must NOT be a keyword of Python.
- Python is a case sensitive programming language that is uppercase letters (capital letters) and lowercase letters (small letters) are treated differently. Eg. Marks and marks are two different identifiers in Python.

Some valid identifiers
Mymarks , file12 , tv9r , Avg_2 , _no

Some invalid identifiers
4Qtr , global , XI-CS , E mail , %age
Starts with digit · keyword · Special characters - (hyphen) space %

5. LITERALS / VALUES

Literals are data items that have a fixed value. Python allows several kind of literals:

A. Numeric Literals

- Int (signed integers) → Positive or Negative whole numbers with no decimal point
  Eg. 10, -96, 1234, 0 — 123
- Floating point / Real values → float represent real numbers and are written with a decimal point dividing the integer and fractional part.
  Eg. 2.0, 54.7, -12.0, -0.075, .4 — 3.14
- Complex (complex numbers) → a+bj , where a and b are floats and J(or j) represents √-1 , which is an imaginary number. a is the real part and b is the imaginary part of the number.
  Eg. 3+4j , -2j , 0+5.5j — a+ bj

B. Boolean Literals

True and False are the only two Boolean values. — True · False

C. Special Literal

None literal represents absence of a value. That is why nothing is printed for the value of b. — None

D. String Literal

- String literal is a sequence of characters surrounded by quotes (single or double or triple).
- String literals can be written in either single quote ' ' or double quote " " or triple quotes ''' ''' or """ """.

'Hello' · "Hello" · '''Hello''' · """Hello"""

5. LITERALS / VALUES (contd.)

E. Escape Sequences / Non-graphic Characters

Escape Sequence	Represents	Description	Example (Output)
\n	↓	New Line	print("Hello\nWorld") Hello World
\t	→⁝	Horizontal Tab	print("Hi\tPython") Hi Python
\	\	Backslash	print("C:\\Python") C:\Python
\'	'	Single Quote	print('It\'s great') It's great
"	"	Double Quote	print("He said \"Hi\"") He said "Hi"
\r	↵	Carriage Return	print("Hello\rHi") Hi
\f	📄	Form Feed	print("Page1\fPage2") Page1
\b	←	Backspace	print("AB\bC") AC
\ooo	ooo	Octal value	print("\101") A
\xhh	hh	Hex value	print("\x41") A

Examples in Action

- print("Hello\nWorld") → The cursor moves to the next line after printing 'Hello'
- print("Hello\tPython") → The cursor keeps one tab space after printing 'Hello'
- print('It\'s great') → To print one ' (apostrophe) inside single quotes, use \ escape sequence.
- print("It's great") → No error when same written inside double quotes.

6. PUNCTUATORS & OPERATORS

A. Punctuators

Punctuators are certain symbols which are used in a program to denote the structure.

(	)	[	]	{
,	:	;	.	@
'	"	#	$	%
←	→	”		...

B. Operators

Operators are used to perform operations on variables and values.

1. Arithmetic Operators: + - * / // % **
2. Relational (Comparison) Operators: == != > < >= <=
3. Logical Operators: and or not
4. Assignment Operators: = += -= *= /= //= %= **=
5. Bitwise Operators: & | ^ ~ << >>
6. Membership Operators: in not in
7. Identity Operators: is is not

Keep Practicing, Keep Coding, Keep Growing! Happy Pythoning!

Revision Tour

FLOW OF CONTROL IN PYTHON

1. TYPES OF STATEMENTS

Statements are the instructions given to computer to perform any kind of action viz. data movements, making decision, repeating actions.

Statements form the smallest executable unit within a program.

Simple Statement (Single statement)

Any single executable statement is a simple statement.

```
x = 10
```

Compound Statement

A group of statements executed as a unit.

It has:

(i) Header line – begins with a keyword and ends with a colon (:)

(ii) Body – one or more indented Python statements at the same indentation level.

Header Line → `if x > 0:` Body: `print("Positive")`, `print("Non-zero")`

Empty Statement (Null operation)

A statement which does nothing.

```
pass
```

2. FLOW OF CONTROL

Flow of Control is the order in which statements are executed in a program.

Statements in a program may be executed sequentially, selectively or iteratively.

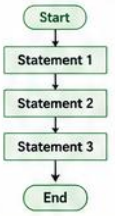
Every program language provides or supports the following constructs:

- Sequence (Sequential Flow)
- Selection / Conditional / Decisional
- Iteration / Looping (For loop, While loop)

> These constructs help us control the flow of execution and make decisions or repeat actions in a program.

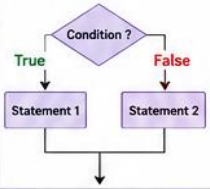
3. TYPES OF CONSTRUCTS

1 Sequence Construct / Sequential Flow



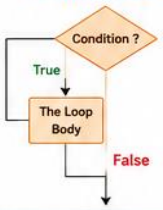
- Statements are executed sequentially.
- It represents the default flow of statements.
- Program starts from the first statement and ends at the final statement.

2 Selection / Decisional / Conditional Construct



- Execution of statement(s) depends on a condition-test.
- Helps in making decisions about which set of statements to execute.

3 Iteration / Looping Construct / Iterative Flow



- Repetition of a set of statements depending on a condition.
- The loop body is repeated till the condition is True (or False depending on the loop).
- Also called "Looping Construct".
- Eg. for loop, while loop

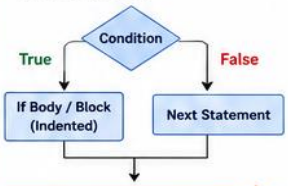
4. SELECTION / CONDITIONAL / DECISION MAKING CONSTRUCT

There are three types of decision making statements.

1. if statement
2. if-else statement
3. Nested if-else statement

A. if STATEMENT

- If statement must be provided with a condition.
- If the condition is True, the indented body / block gets executed.
- If the condition is False, then the control doesn't execute the if body/block.



NOTE

To indicate a block of code in Python, you must indent each line of the block by the same amount. In example programs, all statements inside the if block are at the same indentation level.

5. EXAMPLES OF IF STATEMENT

Example 1: Positive Number

```
n = int(input("Enter a number: "))
if n > 0:
    print("Positive number")
```

Input: 7
Output: Positive number

Example 2: Even Number

```
n = int(input("Enter a number: "))
if n % 2 == 0:
    print("Even number")
```

Input: 12
Output: Even number

Example 3: Grade Finder

```
marks = int(input("Enter marks: "))
if marks >= 90:
    print("Grade A")
if marks >= 75 and marks < 90:
    print("Grade B")
if marks >= 50 and marks < 75:
    print("Grade C")
if marks < 50:
    print("Grade D")
```

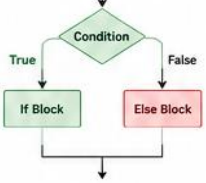
Input: 78
Output: Grade B

If condition is True, the indented block executes.
If condition is False, the next statement after the block executes.

6. OTHER DECISION MAKING STATEMENTS

A. if-else Statement

```
if condition:
    # block of code
else:
    # block of code
```

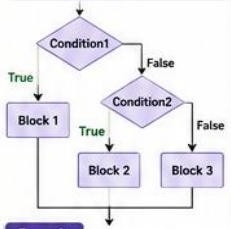


Example

```
n = int(input("Enter a number: "))
if n % 2 == 0:
    print("Even")
else:
    print("Odd")
```

B. Nested if-else Statement

```
if condition1:
    # block
else:
    if condition2:
        # block
    else:
        # block
```



Example

```
n = int(input("Enter a number: "))
if n > 0:
    print("Positive")
else:
    if n == 0:
        print("Zero")
    else:
        print("Negative")
```

7. LOOPING (ITERATION) OVERVIEW

Loops are used to repeat a set of statements multiple times.

Types of Loops in Python

1. **for loop** (Used when number of iterations is known)
2. **while loop** (Used when number of iterations is unknown)

Example: for loop

```
for i in range(5):
    print(i)
```

Output: 0 1 2 3 4

Example: while loop

```
i = 1
while i <= 3:
    print(i)
    i = i + 1
```

Output: 1 2 3

8. SUGGESTED PROGRAMS TO PRACTICE

- Calculation of Simple Interest
- Calculation of Compound Interest
- Finding Factorial of a Positive Number
- Absolute Value of a Number
- Sort Three Numbers
- Divisibility of a Number
- More Extra Programs

Practice Makes Perfect. Code More, Think Better!

Revision Tour

STRING in PYTHON – CHEAT SHEET

1 WHAT IS A STRING?

- A String is a sequence of characters enclosed in single (' '), double (" ") or triple quotes (''' ''').
- Strings are **immutable** (unmodifiable). Any change creates a new string.
- Each character has an index.

Forward Index Direction (0 to n-1)

0	1	2	3	4	5	6	7	8	10
P	y	t	h	o	n	I	s	F	u
-11	-10	-9	-8	-7	-6	-5	-4	-2	-1

Backward Index Direction (-1 to -n)

Creation Examples

Single Line Strings

```
'Hello'
"Python"
```

Multiline Strings

```
"""This is
a multiline
string"""
```

Note: Every time we perform something on the string that changes it, Python internally creates a new string rather than modifying the old string in place.

2 TRAVERSAL OF A STRING

Traversing means iterating through the elements of a string one character at a time.

```
s = "Python"
for ch in s:
    print(ch)
```

Output:

P	y	t	h	o	n

Common Ways to Traverse

Using for loop	`for ch in s:` `  print(ch)`
Using range() & index	`for i in range(len(s)):` `  print(s[i])`

Using while loop	`i = 0`
	`while i < len(s):`
	`  print(s[i])`
	`  i += 1`

★ len(s) gives the length (number of characters) in the string.

3 OPERATIONS ON STRING

1. Concatenation / Joining

'+' operator is used to join strings.

```
a = "Hello"
b = "World"
print(a + " " + b)    # Hello World
```

2. Repetition / Replication

'*' operator is used to repeat a string.

```
s = "Hi! "
print(s * 3)     # Hi! Hi! Hi!
```

3. Membership

Use 'in' and 'not in' to check presence.

```
s = "Python is fun"
print('on' in s)          # True
print('java" not in s)  # True
```

Note: Strings are case-sensitive.
Eg: 'Python' != 'python'

4 IMPORTANT STRING METHODS

Category	Method	Description	Example
Aa Case Conversion	capitalize()	First char capital rest small	'hello'.capitalize() # 'Hello'
	title()	First char of each word capital	'hello world'.title() # 'Hello World'
	upper()	All characters uppercase	'abc'.upper() # 'ABC'
	lower()	All characters lowercase	'ABC'.lower() # 'abc'
	swapcase() *	Swap case of each character	'AbC'.swapcase() # 'aBc'
Information	len(s)	Length of string	len('Python') # 6
	count(sub)	Count occurrences of substring	'banana'.count('an') # 2
	find(sub)	Index of first occurrence (-1 if not found)	'hello'.find('e') # 1
	index(sub)	Index of first occurrence (error if not found)	'hello'.index('o') # 4
Type Check	isalnum()	Alphanumeric?	'abc123'.isalnum() # True
	isalpha()	All letters?	'Python'.isalpha() # True
	isdigit()	All digits?	'12345'.isdigit() # True
	islower()	All lowercase?	'python'.islower() # True
	isupper()	All uppercase?	'PYTHON'.isupper() # True
	isspace()	All spaces?	' '.isspace() # True
Manipulation	split(sep)	Split string into list	'a,b,c'.split(',')
	partition(sep)	Split into 3 parts (before, sep, after)	'a-b-c'.partition('-') # ('a', '-', 'b-c')
	strip([chars])	Remove spaces (both sides by default)	' hi '.strip()
	lstrip([chars])	Remove from left	' hi '.lstrip() # 'hi '
	rstrip([chars])	Remove from right	' hi '.rstrip() # ' hi'
	replace(old,new)	Replace occurrences	'a-b-c'.replace('-','_') # 'a_b_c'

★ Not in curriculum (Extra)

EXTRA PROGRAM IDEA

- Palindrome of a String

```
s = input("Enter a string: ")
if s == s[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")
```

QUICK TIPS

- Indexing starts at 0 (forward) and -1 (backward).
- len(s) = number of characters in the string.
- Strings are immutable.
- Use methods – they make your code easy and readable!

Thank You!
Keep Coding ☺

Revision Tour

LIST in PYTHON – CHEAT SHEET

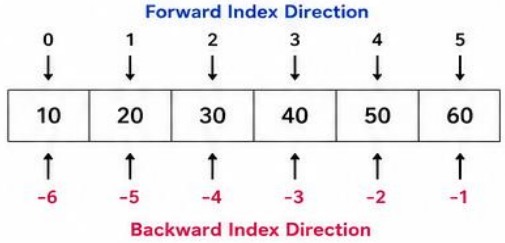
1 WHAT IS A LIST?

- A list is a collection of items formed by placing comma-separated expressions in square brackets [ ].
- Each item (element) has its own index number.
- Index of first item is 0 and last item is n-1 (n = number of items in list).
- Lists are **mutable** sequences i.e. we can change elements of a list in place.
- Lists can contain values of mixed data types.

CONTENTS

Key Point: Lists are ordered, mutable, and can hold duplicate values.

2 INDEXING IN A LIST



NOTE:

- The index (subscript) is the numbered position of an item in the list.
- In Python, indices begin from 0, 1, 2, ..., (length-1) in forward direction and -1, -2, -3, ..., (-length) in backward direction.
- Where length is the length of the list.

Example:

```
L = [10, 20, 30, 40, 50, 60]
L[0] = 10,   L[5] = 60,   L[-1] = 60,   L[-6] = 10
```

3 CREATION OF LIST

Basic List Creation

```
[10, 20, 30]          # List of numbers
['A', 'B', 'C']       # List of characters
[10, 'A', 3.5, True]  # Mixed data type
L = [1, 2, 3, 4]      # With name L
[]                    # Empty list (without name)
L = []                # Empty list (with name L)
l = list()            # Empty list using list()
```

Long List & Nested List

```
MultiTens = [10,20,30,40,50,60,70,80,90,100]   # Long list
L = [1, 'A', [10, 20, 30], ['x', 'y']]         # Nested list (List inside a List)
```

From Existing Sequences

```
list('Python')          # From string
list(input().split())   # From user input (space separated)
list((10, 20, 30))      # From tuple
list(eval(input()))     # From user input using eval()
```

`list(input())` takes input as strings. `eval(input())` identifies the type from the expression.
Example: `list('123')` → `['1','2','3']`
`eval('[1,2,3]')` → `[1, 2, 3]`

4 OPERATIONS ON LIST (OVERVIEW)

1 Concatenation / Joining	`[1,2] + [3,4]` →	`[1, 2, 3, 4]`
2 Repetition / Replication	`[1,2] * 3` →	`[1, 2, 1, 2, 1, 2]`
3 Membership	`3 in [1,2,3]` →	True
	`5 not in [1,2,3]` →	True
4 Comparison	`[1,2] == [1,2]` →	True
	`[1,2] < [1,3]` →	True

SLICING

```
L = [10, 20, 30, 40, 50, 60]
    0   1   2   3   4   5
```

Slice	Meaning	Result
L[1:4]	From index 1 to 3 (exclude 4)	[20, 30, 40]
L[:3]	From start to index 2	[10, 20, 30]
L[3:]	From index 3 to end	[40, 50, 60]
L[-3:]	Last 3 elements	[40, 50, 60]
L[::-1]	Reverse the list	[60, 50, 40, 30, 20, 10]
L[::2]	Every 2nd element	[10, 30, 50]

Note: Slicing does not modify the original list. It returns a new list.

5 FUNCTIONS / METHODS (Quick Reference)

Method / Function	Description
len(L)	Returns number of elements in list
list(iterable)	Converts an iterable to list
append(x)	Adds x at the end
extend(iterable)	Extends list by appending elements from iterable
insert(i, x)	Inserts x at index i
count(x)	Counts occurrences of x
index(x)	Returns index of first occurrence of x
remove(x)	Removes first occurrence of x
pop(i)	Removes & returns item at index i (default last item)
reverse()	Reverses the list in place
sort()	Sorts the list in ascending order
min(L), max(L), sum(L)	Minimum, Maximum & Sum of numeric items

6 IMPORTANT TASKS

Finding Maximum

```
max(L)  # Only for numeric list
```

Finding Minimum

```
min(L)  # Only for numeric list
```

Finding Mean (Average)

```
mean = sum(L) / len(L)
  # Only for numeric list
```

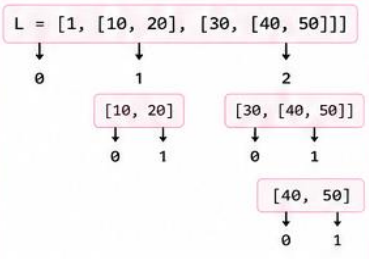
Linear Search (for x in L)

```
found = False
for item in L:
    if item == x:
        found = True
        break
```

Counting Frequency

```
from collections import Counter
freq = Counter(L)  # Returns a dict
# e.g., Counter([1,2,2,3]) → {1:1, 2:2, 3:1}
```

7 NESTED LISTS



- Access using multiple indexes: `L[2][1][0]` → 40
- Nested lists can have different lengths.

Practice more programs to master LISTS in Python! **Keep Coding!**

Revision Tour

TUPLES in PYTHON – CHEAT SHEET

1 WHAT IS A TUPLE?

- A Tuple is a collection of items formed by placing comma-separated values in round brackets ( ).
- Each item (element) has its own index number.
- Index of first item is 0 and last item is n-1 (n = number of items in Tuple).
- Tuples are **immutable** sequences – we cannot change elements in place.
- Tuples can contain values of mixed data types.

CONTENTS

Key Point: Tuples are ordered, immutable, and allow duplicate values.

2 INDEXING IN A TUPLE

Forward Index Direction

0	1	2	3	4	5
10	20	30	40	50	60
-6	-5	-4	-3	-2	-1

Backward Index Direction

NOTE:

- The index (subscript) is the numbered position of an item in the Tuple.
- In Python, indices begin from 0, 1, 2, ..., (length-1) in forward direction and -1, -2, -3, ..., (-length) in backward direction.
- Where length is the length of the Tuple.

Example:

```
T = (10, 20, 30, 40, 50, 60)
T[0] = 10,    T[5] = 60,    T[-1] = 60,    T[-6] = 10
```

3 CREATION OF TUPLE

Basic Tuple Creation

```
(10, 20, 30)              # Tuple of numbers
('A', 'B', 'C')           # Tuple of characters
(10, 'A', 3.5, True]      # Mixed data type
T = (1, 2, 3, 4)          # With name T
()                        # Empty Tuple (without name)
t = tuple()               # Empty Tuple (with name t)
```

Long & Nested Tuple

```
MultiTens = (10,20,30,40,50,60,70,80,90,100]   # Long Tuple
T = (1, 'A', (10, 20, 30), ('x', 'y'])         # Nested Tuple
                                               (Tuple inside a Tuple)
```

From Existing Sequences

```
tuple('Python')           # From string
tuple(input().split())    # From user input (space separated)
tuple((10, 20, 30))       # From tuple
tuple(eval(input()))      # From user input using eval()
```

Note: All elements are considered as string when using list(input()) or tuple(input()). eval() identifies the type by looking at the expression.

4 OPERATIONS ON TUPLE (OVERVIEW)

1	**Concatenation / Joining**	`(1,2) + (3,4)` →	`(1, 2, 3, 4)`
2	**Repetition / Replication**	`[1,2] * 3` →	`(1, 2, 1, 2, 1, 2]`
3	**Membership**	`3 in (1,2,3]` →	True
		`5 not in (1,2,3]` →	True
4	**Comparison**	`(1,2) == (1,2)` →	True
		`[1,2) < (1,3)` →	True

SLICING

```
T = (10, 20, 30, 40, 50, 60)
    0   1   2   3   4   5
```

Slice	Meaning	Result
T[1:4]	From index 1 to 3 (exclude 4)	(20, 30, 40)
T[:3]	From start to index 2	(10, 20, 30]
T[3:]	From index 3 to end	(40, 50, 60]
T[-3:]	Last 3 elements	(40, 50, 60]
T[::-1]	Reverse the Tuple	(60, 50, 40, 30, 20, 10)
T[::2]	Every 2nd element	(10, 30, 50]

Note: Slicing does not modify the original Tuple. It returns a new Tuple.

5 FUNCTIONS / METHODS (Quick Reference)

Method / Function	Description	Example
len()	Returns number of elements	len(T) → 6
tuple(iterable)	Converts iterable to Tuple	tuple([1,2]) → (1, 2)
count(x)	Counts occurrences of x	T.count(10)
index(x[, start[, end]])	Returns index of first occurrence of x	T.index(30)
sorted(T)	Returns sorted list of Tuple items	sorted(T)
min(T), max(T)	Minimum, Maximum item	min(T), max(T)
sum(T)	Sum of numeric items	sum(1,2,3)) → 6

Tuple has limited methods because it is immutable.

6 IMPORTANT TASKS

Finding Maximum / Minimum / Mean

```
T = (4, 1, 7, 2, 9)
maximum = max(T)          # 9
minimum = min(T)          # 1
mean    = sum(T) / len(T) # 23/5 = 4.6
```

Linear Search in Tuple

```
T = (10, 30, 30, 40, 50)
T = 30
key = 30
found = False
for item in T:
    if item == key:
        found = True
        break
# found → True
```

Counting Frequency of Elements

```
from collections import Counter
T = (1, 2, 2, 3, 4, 2, 1)
freq = Counter(T)   # Counter({2: 3, 1: 2, 3: 1, 4: 1})
```

7 NESTED TUPLES

```
T = (1, (10, 20), (30, (40, 50)))
0   1     2
1   (10,20)  (30,(40,50))
    0  1     0   1
    10 20    30  (40,50)
             0   1
             40  50
```

- Access using multiple indexes:
  - T[1][0] → 10
  - T[2][0] → 30
  - T[2][1][1] → 50
- Nested Tuples can have different lengths.

8 SUGGESTED PROGRAM IDEAS & QUICK TIPS

Suggested Programs

- Count the number of times each character appears in a given string using a Tuple (or Counter).
- Accept numbers from user and store them in a Tuple. Find max, min, mean.
- Perform linear search in a Tuple.
- Merge two Tuples and remove duplicates.

Quick Tips

- Tuples are immutable.
- Use tuple() to convert from list/string.
- Tuples can store duplicate values.
- Use count() and index() to work with elements.
- Tuples are faster and memory-efficient compared to lists (read-only data).

Practice more programs to master TUPLES in Python!

Happy Coding!

Revision Tour

DICTIONARY in PYTHON – CHEAT SHEET

1 WHAT IS A DICTIONARY?

- Dictionary is an unordered collection of key-value pairs written inside curly braces { }.
- Each key is unique and maps to a value.
- Dictionary is mutable – we can add, update or delete items.
- It can contain values of mixed data types.

CONTENTS

Analogy

Think of a dictionary as a set of lockers. Each locker has a unique KEY (locker number) and stores some VALUE (items).

Example:

```
D = {10: 'CS', 'IP': 50}
      Key      Value
```

2 CREATION OF DICTIONARY

Basic Creation

Empty dictionary using dict()

```
d1 = dict()
```

Empty dictionary using {}

```
d2 = {}
```

Create from Key-Value Pairs

Directly (using {})

```
d = {'a': 1, 'b': 2}
```

From Tuple of Lists

```
d = dict([['a',1], ['b',2]])
```

From Tuple of Tuples

```
d = dict((('a',1), ('b',2)))
```

Using zip() with two lists

```
keys = ['a','b']
vals = [1,2]
d = dict(zip(keys, vals))
```

From List of Tuples

```
d = dict([('a',1), ('b',2)])
```

From List of Lists

```
d = dict([['a',1], ['b',2]])
```

Key must be immutable (str, int, float, tuple). Value can be mutable or immutable.

3 ACCESSING, ADDING & MODIFYING ITEMS

Accessing Elements

Using key in []

```
d = {'name': 'Riya', 'age': 16}
print(d['name'])   # Riya
```

Using get() method

```
print(d.get('age'))        # 16
print(d.get('city'))       # None
print(d.get('city','NA'))  # NA
```

Adding Items

```
d = {'a': 1}
d['b'] = 2      # add new item
print(d)        # {'a': 1, 'b': 2}
```

Modifying Items

```
d['a'] = 10     # update value
print(d)        # {'a': 10, 'b': 2}
```

Deleting Items

```
del d['a']      # delete specific key
d.pop('b')      # remove key & return value
d.popitem()     # remove and return last inserted pair
d.clear()       # remove all items
del d           # delete the dictionary itself
```

4 TRAVERSAL & MEMBERSHIP

Traversal

Traverse Keys

```
d ={'a':1,'b':2,'c':3}
for k in d.keys():
    print(k)
```

Output:

```
a
b
c
```

Traverse Values

```
for v in d.values():
    print(v)
```

Output:

```
1
2
3
```

Traverse Key-Value Pairs

```
for k,v in d.items():
    print(f"{k} : {v}")
```

Output:

```
a : 1
b : 2
c : 3
```

Membership Operator

Check Key

```
'a' in d  # True
'd' in d  # False
```

Check Value

```
1 in d.values()  # True
4 in d.values()  # False
```

Dictionary is unordered (in Python < 3.7). In Python 3.7+, insertion order is preserved.

5 FUNCTIONS / METHODS (Quick Reference)

Method / Function	Description	Example
len(d)	Number of key-value pairs	len(d) → 3
dict()	Creates empty dictionary	dict() → {}
keys()	Returns view of all keys	d.keys()
values()	Returns view of all values	d.values()
items()	Returns view of (key,value) pairs	d.items()
get(key,default)	Returns value for key (or default)	d.get('x','NA')
update(dict2)	Updates with pairs from dict2	d.update(d2)
fromkeys(seq, val)	Create dict with keys from seq and value = val	dict.fromkeys(['a','b'],0)
copy()	Returns shallow copy	d.copy()
pop(key[,default])	Removes key & returns value	d.pop('a')
popitem()	Removes & returns last pair	d.popitem()
setdefault(key, val)	Returns value; sets default if key not present	d.setdefault('x', 0)
clear()	Removes all items	d.clear()

Built-in Functions: max(d), min(d), sorted(d)
max/min on keys, sorted returns sorted list of keys

6 IMPORTANT TASKS

1 Count frequency of characters in a string

```
s = "banana"
freq = {}
for ch in s:
    freq[ch] = freq.get(ch, 0) + 1
print(freq)   # {'b': 1, 'a': 3, 'n': 2}
```

2 Employee Information

```
emp = {
 101: {'name': 'Amit', 'salary': 50000},
 102: {'name': 'Neha', 'salary': 62000},
 103. {'name': 'Ravi', 'salary': 48000}
}
for eid, info in emp.items():
   print(eid, info['name'], info['salary'])
```

7 SUGGESTED PROGRAM IDEAS

- Count the number of times each character appears in a given string using a dictionary.
- Create a dictionary with employee no., names of employees, their salary and display them.

8 QUICK TIPS

- Dictionary keys are unique and immutable.
- Values can be of any data type.
- Use get() to avoid KeyError.
- Use items() for iterating over both keys and values.
- Use update() to merge dictionaries.

```
d1 = {'a':1}
d2 = {'b':2}
d1.update(d2)   # {'a':1, 'b':2}
```

Common Errors

KeyError: accessing non-existing key (print(d['x']))

TypeError: using mutable type as key (e.g. list)

Practice more programs to master DICTIONARY in Python!

Happy Coding!

Question Type: MCQ

1	State True or False In Python, a dictionary is an ordered collection of items (key: value pairs).	1
2	State True or False: "In a Python loops can also have else clause"	1
3	State True or False "continue" keyword skips remaining part of an iteration in a loop	1
4	State if the following statement is True or False: Using the statistics module, the output of the below statements will be 20: import statistics statistics. median([10, 20, 10, 30, 10, 20, 30])	1
5	State True or False: In Python, tuple is a mutable data type	1
6	State True or False: A list cannot contain another list as an element.	1
7	State True or False: The keys of a dictionary must be of immutable types.	1
8	State True or False: "In a Python program, if a break statement is given in a nested loop, it terminates the execution of all loops in one go."	1
9	What will be the output of the following code snippet? a , b = "10" , "20" print (int (a + b) + int(a) * int(b)) A) 1220 B) 410 C) 620 D) 30	1
10	What will be the output of the following code? x = 5 y = "2" print (x * int(y) + len(y + str(x))) A) 11 B) 12 C) 15 D) 10	1
11	Which of the following code snippets correctly checks if a string s is a palindrome (case insensitive)? A) s == s[::-1] B) s.lower( ) == s.upper( )[::-1] C) s.lower( ) == s.lower( )[::-1] D)s== s.reverse()	1

12	Which of the following code snippets will raise an error for s = "Python"? A) print(s[10]) B) print(s[1:10]) C) print (s [-1: -3]) D) print(s[:2])	1
13	Identify the valid Python identifier from the following: A) 2user B)user@2 C) user_2 D) user 2	1
14	What will be the output of following python code? s = 'Programming' print(s.split("m")) A) ['Progra', ' ', 'ing'] B) ['Progra', 'ing']. C) ['Progra', 'm', 'ing'] D) [Progra', 'ming']	1
15	Select the correct output of the code: S="Amrit Mahotsav @ 75" n=S.split(" ",2) print(n) A) ('Amrit', 'Mahotsav', '@', '75') B) ['Amrit', 'Mahotsav', '@ 75'] C) ('Amrit', 'Mahotsav', '@ 75') D) ['Amrit', 'Mahotsav', '@', '75']	1
16	Select the output of the code: s = “Bring it on” l = s.split() s_new = “#”.join([l[0].lower(), l[1], l[2].title()]) print(s_new) A) bring#it#ON B) bring#it#on C) Bring#it#On D) bring#it#On	1
17	What is the output of the expression? s='All the Best' p=s.split("t") print(p) A) ['All ', 'he Bes', ''] B) (‘All ', 'he Bes', '') C) ['All ', 't', 'he', 'Bes', 't'] D) Error	1
18	What is the output of the expression? St1=”abc@pink@city” print (St1.split("@")) A) (“abc”, “@”, “pink”, “@”, “city”) B) [“abc”, “@”, “pink”,”@’,”city”] C) [“abc”, “pink”, “city”] D) Error	1
19	What will be the output of the following Python statement? print(10-3**2**2+144/12)	1

20	What will be the output of the following statement? print (6+5/4**2//5+8) A)-14.0 B)14 C)-14 D) 1	1
21	What will be the output of following code if? a = “abcde” a [1:1] == a [1:2] type (a[1:1]) == type (a[1:2])	1
22	Predict the output of the following Python statements: import statistics as s s.mode ([10, 20, 10, 30, 10, 20, 30]) A)30 B) 20 C)10 D) 18.5	1
23	Select the correct output of the code: a = "foobar" a = a.partition("o") print(a) A) ["fo","","bar"] B) ["f","oo","bar") C) ["f","o","bar"] D) ("f","o","obar")	1
24	Identify the output of the following code snippet: str = "KENDRIYA VIDYALAYA" str=str.replace('YA','*') print(str) A) KENDRIYA VIDYALAYA B) KENDRI*A VID*ALAYA C) KENDRI* VID*LA* D) * KENDRI* VID*LA*	1
25	Select the correct output of the code: S = "text#next" print(S.strip("t")) A) ext#nex B) ex#nex C) text#nex D) ext#next	1
26	Select the correct output of the code: s = "Question paper 2022-23" s= s.split('2') print(s) A) ['Question paper ', '0', '', '-', '3'] B) ('Question paper ', '0', '', '-', '3') C) ['Question paper ', '0', '2', '', '-', '3'] D) ('Question paper ', '0', '2', '', '-', '3')	1

27	Select the correct output of the code: S = "text#next" print(S.strip("t")) A) ext#nex B) ex#nex C) text#nex D) ext#next	1
28	Consider the statements given below and then choose the correct output from the given options: Game="World Cup 2023" print(Game[-6::-1]) A) CdrW B) ce o C) puC dlroW D) Error	1

29	Which of the following statement(s) would give an error after executing the following code?
	S="Welcome to class XII" #Statement 1
	print(S) # Statement 2
	S="Thank you" #Statement 3
	S[0]= '@' #Statement 4
	S=S+"Thank you" #Statement 5
	A) Statement 3 B) Statement 4
	C) Statement 5 D) Statement 4 and 5
		1

30	What will be output of the following code snippet? Msg='Wings of Fire!' print (Msg[-9: :2])	1
31	Given s1="Hello". Which of the following statements will give an error? A) print(s1[4]) B) s2=s1 C) s1=s1[4] D) s1[4]= "Y"	1
32	What will be the output of the following code snippet? message= "Satyamev Jayate" print(message[-2::-2])	1
33	What will be the output of the following Python code? str= "Soft Skills" print(str[-3::-3]) A) lSf B) Stkl C) StKi D) l	1

34	Select the correct output of the code:
	>>> St="Computer"
	>>> print(St[4:])
	A) "Comp" B) "pmoC" C) "uter" D) "mput"
		1

35	Find the output: A=”MISSISSIPPI” print(A[:4]+’#’+A[-5:-1]) A) MISSI#SIPPI B) MISS#SIPPI C) MISS#IPPIS D) MISSI#PPIS	1

36	What is the output of the following code?
	Text = 'Happy hour12-3'
	L= " "
	for i in range(len(Text)):
	if Text[i].isupper():
	L=L+Text[i].lower()
	elif Text[i].islower():
	L=L+Text[i].upper()
	elif Text[i].isdigit():
	L=L+(Text[i]*2)
	else:
	L=L+'#'
	print(L)
	(A)hAPPY#HOUR1122#33 (B)Happy#hOUR12#3
	(C)hAPPY#HOUR112233 (D)Happy Hour11 22 33 #
		1

37	What will be the output of following python code? t1 = 1,2,3 t2 = (1,2,3) print (t1 is t2) A) True B) 1 C) False D) 0	1
38	Which of the following is not a tuple in python? A) (10, 20) B) (10,) C) (10) D) All are tuples.	1
39	Given a tuple tupl=(10,20,30,40,50,60,70,80,90) What will be the output of print (tupl[3:7:2])? A) (40,50,60,70,80) B) (40,50,60,70) C) [40,60] D) (40,60)	1
40	What will be the output of the following code? Tuple1= (10,) Tuple2=Tuple1*2 print(Tuple2) A) 20 B) (20,) C) (10,10) D) Error	1

41	Identify the output of the following code snippet: T=(100) print(T*2) A) Syntax error B) (200,) C) 200 D) (100,100)	1
42	Given a Tuple tup1= (10, 20, 30, 40, 50, 60, 70, 80, 90). What will be the output of print (tup1 [-2: -5])? A) (80,70,60) B) ( ) C) (60,70,80) D) Error	1
43	Which of the following options will not result in an error when performed on types in python where tp = (5,2,7,0,3) ? A) Tp[1] = 2 B) tp.append(2) C) tp1 = tp+tp D) tp.sum( )	1
44	Given a tuple tupl=(10,20,30,40,50,60,70,80,90) What will be the output of print (tupl[3:7:2])? A) (40,50,60,70,80) B) (40,50,60,70) C) [40,60] D) (40,60)	1
45	What will be output of the following code: d1={1:2,3:4,5:6} d2=d1.get(3) print(d2) A) 4 B)3 C) 5 D) 6	1
46	What will be the output of the following Python code? my_dict = {"name": "Alicia", "age": 27, "city": "DELHI"} print(my_dict.get("profession", "Not Specified")) A) Alicia B)DELHI C)None D)Not Specified	1
47	What will be the output of the following python dictionary operation? data = {'A':2000, 'B':2500, 'C':3000, 'A':4000} print(data) A) {'A':2000, 'B':2500, 'C':3000, 'A':4000} B) {'A':2000, 'B':2500, 'C':3000} C) {'A':4000, 'B':2500, 'C':3000} D) It will generate an error	1
48	Given the following dictionaries dict_stud = {"rno" : "53", "name" : 'Rajveer Singh'} dict_mark = {"Accts" : 87, "English" : 65} Which statement will merge the contents of both dictionaries? A) dict_stud + dict_mark B) dict_stud.add(dict_mark) C) dict_stud.merge(dict_mark) D) dict_stud.update(dict_mark)	1

49	my_dict is a dictionary as defined below, then which of the following statements will raise an exception? my_dict = {'aman': 10, 'sumit': 20, 'suresh': 30} A) my_dict.get('suresh') B) print(my_dict['aman', 'sumit']) C) my_dict['aman']=20 D) print(str(my_dict))	1

50	Identify the output of the following python code: 1
	D={1:"one",2:"two", 3:"three"}
	L=[]
	for k,v in D.items():
	  if 'o' in v:
	  L.append(k)
	print(L)
	
	A) [1,2] B) [1,3] C)[2,3] D)[3,1]
		1

51	_____ method is used to delete a given element from the list.	1
52	What will be the output of the following code? L = ["India", "Incredible", "Bharat"] print(L[1][0] + L[2][-1]) A) IT B) it C) It D) It	1
53	Which of the following can delete an element from a list if the index of the element is given? A) pop( ) B) remove( ) C) clear( ) D) all of these	1
54	Consider a declaration L=(1,'Python','3.14') Which of the following represents the data type of L? A) list B) tuple C) dictionary D) string	1
55	Consider the given expression: 17%5==2 and 4%2>0 or 15//2==7.5 Which of the following will be correct output if the given expression is evaluated? A)True B) False C)None D)Null	1
56	The following expression will evaluate to print (2+(3%5)**1**2/5+2) A) 5 B) 4.6 C) 5.8 D) 4	1

57	What will be the output? print (16*5/4*2/5-8) A) -3.33 B) 6.0 C) 0.0 D) -13	1
58	Write the output of following expression: 5>10 and not(10<15)	1
59	What will be the output of following expression? (5<10) and (10< 5) or (3<18) and not 8<18 A) True B) False C) Error D) No output	1
60	What will the following expression be evaluated to in Python? print (2**3**2) A) 64 B) 256 C) 512 D) 32	1
61	Consider the given expression: print (19<11 and 29>19 or not 75>30) Which of the following will be the correct output of the given expression? A) True B) False C) Null D) No output	1
62	Write the output of the following Python code : for k in range(7,40,6): print (k + '-')	1
63	Consider a list L= [10,20,30,40], which of the following statement will result in an error: A) L [0] =L [0] +2 B) L=L+2 C) L=L*2 D) L[1]=50	1
64	Suppose a tuple T is declared as T = (10, 12, 43, 39), which of the following is incorrect? A) print(T.count(12)) B)print(any(T)) C) print(sum(T)) D)T.sort()	1
65	What possible output is expected to be displayed on the screen at the time of execution of the Python program from the following code? import random L=[10,30,50,70] Lower=random.randint(2,2) Upper=random.randint(2,3) for K in range(Lower, Upper+1): print(L[K], end="@") A) 50@70@ B) 90@ C) 10@30@50@ D) 10@30@50@70@	1

66	State True or False : In Python, data type of 74 is same as the data type of 74.0.	
67	Identify the output of the following code snippet : s = "the Truth" print(s.capitalize()) A) The truth B) THE TRUTH C) The Truth D) the Truth	
68	Which of the following expressions in Python evaluates to True ? (A) 2>3 and 2<3 B) 3>1 and 2 (C) 3>1 and 3>2 (D) 3>1 and 3<2	
69	What is the output of the following code snippet ? s='War and Peace by Leo Tolstoy' print(s.partition("by")) A) ('War and Peace ', 'by', ' Leo Tolstoy') B) ['War and Peace ', 'by', ' Leo Tolstoy'] C) ('War and Peace ', ' Leo Tolstoy') D) ['War and Peace ', ' Leo Tolstoy']	
70	What will be the output of the following statement ? print("PythonProgram"[–1:2:–2])	
71	What will be the output of the following code snippet ? t = tuple('tuple') t2 = t[2], t += t2 print(t) A) ('tuple') B) ('tuple' , 'p') C) ('t', 'u', 'p', 'l', 'e', 'p') D) ('t', 'u', 'p', 'l', 'e')	
72	Which of the following statements is true about dictionaries in Python ? A) A dictionary is an example of sequence datatype. B) A dictionary cannot have two elements with same key. C) A dictionary cannot have two elements with same value. D) The key and value of an element cannot be the same.	
73	If L is a list with 6 elements, then which of the following statements will raise an exception ? A) L.pop(1) B) L.pop(6) C) L.insert(1,6) D) L.insert(6,1)	

ANSWERS

1	False	2	True	3	True	4	True	5	False
6	False	7	True	8	False	9	A	10	B
11	C	12	A	13	C	14	A	15	D
16	D	17	A	18	C	19	-59.0	20	B
21	True	22	C	23	D	24	C	25	A
26	A	27	A	28	C	29	B	30	so ie
31	D	32	tyJvmya	33	A	34	C	35	B
36	A	37	A	38	C	39	D	40	C
41	C	42	B	43	C	44	D	45	A
46	D	47	C	48	D	49	B	50	A
51	remove()	52	C	53	A	54	B	55	B
56	ii	57	C	58	False	59	B	60	C
61	False	62	Error *	63	ii	64	4	65	A
66	False	67	A	68	B/C	69	A	70	mroPo
71	C	72	B	73	B				

*62 Error as unsupported operand type(s) for +: 'int' and ‘str'

Assertion/Reasoning Type Questions

Directions: In the following questions, a statement of Assertion (A) is followed by a statement of Reason (R). Mark the correct choice as:

(A) Both A and R are true and R is the correct explanation of A

(B) Both A and R are true and R is not the correct explanation of A

(C) A is true but R is false

(D) A is false but R is true

1	Assertion (A): Lists can store only elements of the same data type. Reason (R): Python is a dynamically typed language	1
2	Assertion (A): List is an immutable data type. Reasoning(R): When an attempt is made to update the value of an immutable variable, the old variable is destroyed and a new variable is created by the same name in memory	1
3	Assertion (A): Dictionaries are mutable; hence its keys can be easily changed. Reason(R): Mutability means a value can be changed in place without having to create new storage for the changed value.	1
4	Assertion (A): You can add an element in a dictionary using key: value pair. Reasoning (R): A new (key: value) pair is added only when the same key doesn't exist in the dictionary. If the key is already present, then the existing key gets updated and the new entry will be made in the dictionary.	1
5	Assertion (A): Tuples hold a sequence of homogenous elements. Reason (R): Tuples are immutable	1
6.	Assertion (A) : [1,2,3]+'123' is an invalid expression in Python. Reason (R) : In Python, a list cannot be concatenated with a string.	

ANSWERS

1	D	2	D	3	D	4	A	5	D
6	A								

Question Type: Find Output / Short Answer / Long Answer

1	Write difference between mutable and immutable property, Explain it with its example.	2
2	Which of the following can be used as valid identifier(s) in Python? (i) Total (ii) @selute (iii) Que$tion (iv) great (v) 4th Sem (vi) li1 (vii) No# (viii) _Data	2
3	i) Write the names of any two data types available in python. (ii)Write any 2 operators name used in python	2
4	(i) Explain the difference between explicit and implicit type conversion in Python with a suitable example. (ii) Explain the difference between break and continue statements in Python with a suitable example.	2
5	Give two examples of each of the following: (i)logical operators (ii) Relational operators	2
6	Give two examples of each of the following (i) Assignment operators (ii) Logical operators	2
7	Consider the following list L1 and write Python statement for the following questions: L1=['english','physics','chemistry','cs','biology'] (i) (A) To insert subject "maths" as last element (B) To display the list in reverse alphabetical order (ii) (A) To remove the first element of list (B) To Find the index position of element 'cs'	2
8	Write the Python statement for each of the following tasks using BUILT-IN functions/methods only: (i) To insert an element 400 at the fourth position, in the list L1. (ii) To check whether a string named, message ends with a full stop/ period or not.	2
9	A list named employee salary stores salary of employees. Write the Python command to import the required module and (using built-in function) to display the most common salary value from the given list.	2

10	Write the Python statement for each of the following tasks using BUILT-IN functions/methods only: (i) To delete an element 10 from the list lst. (ii) To replace the string "This" with "That" in the string str1	2

11	Rewrite the following code in Python after removing all syntax error(s). Underline each correction done in the code.
	p=30
	for c in range(0,p)
	if c%4==0:
	print (c*4)
	Elseif c%5==0:
	print (c+3)
	else
	print(c+10)
		2

12	Predict the output of the Python code given below:
	T = (9,18,27,36,45,54)
	L=list(T)
	L1 = [ ]
	for i in L:
	if i%6==0:
	L1.append(i)
	T1 = tuple(L1)
	print(T1)
		2

13	Predict the output of the Python code given below:
	List1 = list("Examination")
	List2 =List1[1: -1]
	new_list = []
	for i in List2:
	j=List2.index(i)
	if j%2==0:
	List1.remove(i)
	print(List1)
		2

14	Predict the output of the following code: d={"IND":"DEL","SRI”:"COL","CHI":"BEI"} str1="" for i in d: str1=strl+str(d[i])+"@" str2=str1[:–1] print (str2)	2
15	(i)Given is a Python string declaration: myexam="@@PRE BOARD EXAMINATION 2024@@" Write the output of: print(myexam[::-2]) (ii)Write the output of the code given below: my_dict = {"name": "Aman", "age": 26} my_dict['age'] = 27 my_dict['address'] = "Delhi" print(my_dict.items())	2

16	Write the output displayed on execution of the following Python code:
	LS=["HIMALAYA","NILGIRI","ALASKA","ALPS"]
	D={}
	for S in LS :
	if len(S)%4 == 0:
	D[S] = len(S)
	for K in D:
	print(K,D[K], sep = "#")
		2

17	(i) Given is a Python string declaration: NAME = "Learning Python is Fun" Write the output of: print (NAME [-5: -10: -1]) (ii) Write the output of the code given below: dict1= {1: ["Rohit",20], 2: ["Siya",90]} dict2= {1: ["Rahul",95], 5: ["Rajan",80]} dict1.update(dict2) print(dict1.values( ))	2

18	Write the Python statement for each of the following tasks using BUILT_IN functions/ methods only: str="PYTHON@LANGUAGE" (i) To print the above string from index 2 onwards. (ii) To initialize an empty dictionary named as d.	2
19	Write the Python statement for each of the following tasks using BUILT_IN functions/ methods only: (i) s=”LANGUAGE" To convert the above string into list. (ii) To initialize an empty tuple named as t.	2
20	Write the Python statement for each of the following tasks using built-in functions/methods only: (i) To remove the item whose key is "NISHA" from a dictionary named Students. For example, if the dictionary Students contains {"ANITA":90, "NISHA":76, "ASHA":92}, then after removal the dictionary should contain{"ANITA":90,"ASHA":92} (ii) To display the number of occurrences of the substring "is" in a string named message. For example, if the string message contains "This is his book", then the output will be 3.	2
21	If str1= “foundation stone” and str2= “strong base”, then Write the Python statements for each of the following tasks using Built-in functions/methods only: (i) To count the occurrence of the letter ‘n’ in str1. (ii) To convert the string str2 in a list and each word of the string should become the element of the list	2
22	Write the Python statement for each of the following tasks using BUILT_IN functions/ methods only: (i) str="PYTHON@LANGUAGE" (A) To print the above string from index 2 onwards. (B) To initialize an empty dictionary named as d.	2

	(ii) Write the Python statement for each of the following tasks using BUILT_IN functions/ methods only: (A) s=”LANGUAGE" To convert the above string into list. (B) To initialize an empty tuple named as t.	
23	(i) Answer using Python built-in methods/functions only: (A) Write a statement to find the index of the first occurrence of the substring "good" in a string named review. (B) Write a statement to sort the elements of list L1 in descending order. (ii) Predict the output of the following Python code: text="Learn Python with fun and practice" print(text.partition("with")) print(text.count("a"))	2
24	What possible outputs are expected to be displayed on the screen at the time of execution of the program from the following code? import random temp=[10,20,30,40,50,60] c=random.randint(0,4) for i in range(0, c): print(temp[i],”#”) A) 10#20# B) 10#20#30#40#50# C) 10#20#30# D) 50#60#	2
25	Find possible o/p (s) at the time of execution of the program from the following code? Also specify the maximum values of variables Lower and Upper. import random as r AR=[20, 30, 40, 50, 60, 70]; Lower =r.randint(1,3) Upper =r.randint(2,4) for K in range(Lower, Upper +1): print (AR[K], end=”#“) A) 10#40#70# B) 30#40#50# C) 50#60#70# D) 40#50#70#	2

26	Identify the correct output(s) of the following code. Also write the minimum and the maximum possible values of the variable Lot import random word='Inspiration' Lot=2*random.randint(2,4) for i in range(Lot,len(word),3): print(word[i], end='$') A) i$ain$ B) in C) itn$ D) ain$	2
27	What possible outcome will be produced when the following code is executed? import random value=random.randint(0,3) fruit=["APPLE","ORANGE","MANGO","GRAPE"] for i in range(value): print(fruit[i], end='##') A) APPLE## B) APPLE##ORANGE## C) APPLE## ORANGE##GRAPE## D) ORANGE##MANGO##APPLE##	2
28	Find the correct output(s) of the following code. Also write the maximum and minimum values that can be assigned to variable Y. import random X=random.random() Y=random.randint(0,4) print (int(X),":", Y+int(X)) A) 0:0 B) 1:6 C) 2:4 D) 0:3	2
29	What possible outputs(s) are expected to be displayed on screen at the time of execution of the program from the following code? Also specify the maximum values that can be assigned to each of the variables FROM and TO. import random AR=[20,30,40,50,60,70] FROM=random.randint(1,3) TO=random.randint(2,4) for K in range(FROM,TO): print (AR[K],end="#") A)10#40#70# B)30#40#50# C)50#60#70# D)40#50#70#	2

30	Find possible o/p (s) at the time of execution of the program from the following code? Also specify the maximum values of variables Lower and Upper. import random as r AR=[20, 30, 40, 50, 60, 70]; Lower =r.randint(1,3) Upper =r.randint(2,4) for K in range(Lower, Upper +1): print (AR[K],end="#") (A) 10#40#70# (B) 30#40#50# (C) 50#60#70# (D) 40#50#70#	2
31	Write a Python statement to perform the following tasks : (USE BUILT_IN FUNCTIONS/METHODS ONLY) (i) To create a new list L1 containing the elements of list L arranged in ascending order, without modifying list L. (ii) A statement to check whether the given character, ch is an alphabet or a number.	
32	Assuming that D1 is a dictionary in Python, (i) (a) Write a Python expression to check if the key, 'RNo' is present in D1. OR (i) (b) Write a Python expression to check if any key in D1 has a value 12. (ii) (a) Write a single statement using a BUILT_IN function to add the key : value pair 'RNo':12, if the key 'RNo' is not present in D1. However, if the key 'RNo' is present, the function should return its value. OR (ii) (b) Write a single statement to delete all the elements from D1.	
33	What possible output(s) from the given options will NOT be displayed when the following code is executed ? Also, mention, for how many iterations the for loop in the given code will run ? import random a = [1,2,3,4,5,6] for i in range(4): j=random.randrange(i,5) print(a[j],end='–') print( ) A) 3–4–5–4– B) 2–2–4–5– C) 4–3–3–5– D) 5–1–2–4–	

ANSWERS

1	Mutable: Can be changed (e.g., lists, dictionary). Immutable: Cannot be changed (e.g., tuples, string).
2	Valid identifier(s) (i)Total (iv) great (vi) li1 (viii) _Data Invalid identifier(s) (ii) @selute (iii) Que$tion (v) 4th Sem (vii) No#
3	i) Names of any two data types available in python: int, float or any other valid datatype in python. ii) Any 2 operators name used in python: Arithmetic, Logical, Relational or any other valid operator in python.

4	A. Implicit Conversion: Python automatically converts one data type to another.
	Example:
	x = 10
	y = 3.5
	result = x + y # x is implicitly converted to float
	Explicit Conversion: The user manually converts one data type to another using functions like int(), float().
	Example:
	x = "10"
	y = int(x) # Explicit conversion from string to integer
	
	B. Break exits the loop entirely, while continue skips the current iteration and moves to the next one.
	Example of break:
	for i in range(5):
	if i == 2:
	break # Exits the loop
	print(i)
	Output:
	1
	Example of continue:
	for i in range(5):

	if i==2:
	continue # Skips printing 2
	print(i)
	Output:
	1
	3
	4
	5

5	Logical Operator: AND, OR, NOT Relational Operator : >, <, >=,<=,!=,==
6	a) Assignment Operators = += -= *= **= /= //= %= b) Logical Operators not and or (any two from each)
7	I) A) L1.append('maths') B) L1.sort(reverse=True) (II) A) L1.pop(0) B) L1.index('cs')
8	i) L1.insert(3,400) (ii) message.endswith('.')
9	import statistics print(statistics.mode(employeesalary))
10	(i) lst.remove(10) (ii) str1.replace("This","That")

11	p=30
	for c in range(0,p):
	if c%4==0:
	print (c*4)
	elif c%5==0:
	print (c+3)
	else:
	print(c+10)

12	(18, 36, 54)

13	['E', 'a', 'i', 'a', 'i', 'n']
14	DEL@COL@BEI
15	(i) @40 OTNMX RO R@ (ii) dict_items([('name', 'SAM'), ('age', 27), ('address', 'MUMBAI')])
16	HIMALAYA#8 ALPS#4
17	(i) si no (ii) dict_values([['Rahul', 95], ['Siya', 90], ['Rajan', 80]])
18	(i) str="PYTHON@LANGUAGE" print(str[2: :]) (ii) d=dict( )
19	(i) s=”LANGUAGE" l=list(s) (ii) t=tuple( )
20	(i) Students.pop("NISHA") (ii) print(message.count("is")) message.count("is")
21	(i) print(str1.count(‘n’)) (ii) L=str2.split( )
22	(i) A) str="PYTHON@LANGUAGE" print (str[2: :]) B) d=dict( ) (ii) A) s=”LANGUAGE" l=list(s) B) t=tuple( )
23	(i) (A) index = review.find("good") (B) L1.sort(reverse=True) (ii) ('Learn Python ', 'with', ' fun and practice') 3
24	A) 10#20# or B) 10#20#30#40#50# or C) 10#20#30#

25	Lower = r.randint(1,3) means Lower will have value 1,2, or 3 Upper =r.randint(2,4) means Upper will have value 2, 3, or 4 So K will be from (1, 2, 3) to (2, 3, 4) Means if K=1, then upper limit (2,3,4) If K=2, then upper limit (2,3,4) If K=3, then upper limit (2,3,4) So correct answer B) 30#40#50# Maximum values of variables Lower and Upper are 3 and 4
26	Identify the correct output(s) of the following code. Also write the minimum and the maximum possible values of the variable Lot Minimum value possible for Lot: 4 Maximum value possible for Lot: 8 Possible outputs are : A) and B)
27	A) APPLE## B) APPLE##ORANGE##
28	A) and D) are possible outputs Maximum: 0 Minimum: 3
29	(B)30#40#50# Maximum value assigned to FROM is 3 Maximum value assigned to TO is 4
30	Lower = r.randint(1,3) means Lower will have value 1,2, or 3 Upper =r.randint(2,4) means Upper will have value 2, 3, or 4 So K will be from (1, 2, 3) to (2, 3, 4) Means if K=1, then upper limit (2,3,4) If K=2, then upper limit (2,3,4) If K=3, then upper limit (2,3,4) So correct answer B) 30#40#50# Maximum values of variables Lower and Upper are 3 and 4.
31	(i) L1=sorted(L) OR L1=list(L) L1.sort()

	(II) ch.isalnum() OR ch.isdigit() or ch.isalpha()

32	(i) (a) 'RNo' in D1 OR 'RNo' in D1.keys()
	OR
	(i) (b) 12 in D1.values()
	(ii) (a) D1.setdefault('RNo',12) OR
	if D1.get('RNo'):
	print(D1['RNo']) # return D1['RNo']
	else:
	D1['RNo']=12
	OR
	(ii) (b) D1.clear() OR D1={} OR D1=dict()

33	(D) 5–1–2–4– The for loop will run for 4 iterations.

Prepared By: **Mrs. Archana Gupta, PGT (CS)**

PM SHRI KV No 3 AGRA

Updated by: **Mrs. Suman Verma, PGT (CS)**

KV NTPC DADRI

CLASS 12 COMPUTER SCIENCE (083)

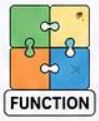
FUNCTIONS

COMPLETE REVISION – CBSE LATEST SYLLABUS

Functions are self-contained blocks of statements that perform a specific task.
They improve modularity, code reusability, readability and debugging.

1 INTRODUCTION TO FUNCTIONS

- A function is a named block of statements that performs a specific task.
- It may take input, process it and may return a value.
- Functions help in breaking a large problem into smaller modules.



Types of Functions in Python

1. **Built-in Functions** (e.g., print(), len(), type())
2. **User-defined Functions**

2 ADVANTAGES OF USING FUNCTIONS

- ✅ Promotes code reusability.
- ✅ Improves readability and reduces complexity.
- ✅ Easier testing and debugging.
- ✅ Allows problem to be divided into smaller tasks (Modularity).
- ✅ Supports abstraction (focus on what the function does, not how).

3 DEFINING A FUNCTION

The general syntax to define a function in Python is:

```
def function_name(parameters):
    """docstring (optional)"""
    # statements
    return value    # (optional)
```

→ `def` → keyword to define a function
→ `function_name` → valid identifier
→ `parameters` → input values (optional)
→ `return` → sends a value back to the caller (optional)

4 TYPES OF FUNCTIONS IN PYTHON

A. Based on Presence of Arguments and Return Value

Type	Arguments	Return Value	Description
1. No Arguments No Return Value	No	No	Function does not take any input and does not return any value.
2. No Arguments With Return Value	No	Yes	Function does not take input but returns a value.
3. With Arguments No Return Value	Yes	No	Function takes input but does not return any value.
4. With Arguments With Return Value	Yes	Yes	Function takes input and returns a value.

B. Based on Nature

1. **Built-in Functions** → Predefined in Python.
   Example: print(), input(), len(), type(), range(), max(), min(), sum()
2. **User-defined Functions** → Created by the programmer using 'def'.

5 EXAMPLES OF USER-DEFINED FUNCTIONS

1. No Arguments No Return Value

```
def greet():
    print("Hello!")

# greeting
greet()
```

2. No Arguments With Return Value

```
def get_pi():
    return 3.14159

# calling
value = get_pi()
print(value)
```

3. With Arguments No Return Value

```
def greet(name):
    print("Hello,", name)


# calling
greet("Riya")
```

4. With Arguments With Return Value

```
def add(a, b):
    return a + b

# calling
result = add(10, 20)
print(result)
```

6 PARAMETERS AND ARGUMENTS

- **Parameter** → Variable in the function definition.
- **Argument** → Value passed to the function during the call.

Example:

```
def multiply(x, y):   # x, y are parameters
    return x * y

result = multiply(5, 6)  # 5, 6 are arguments
```

7 RETURN STATEMENT

- **return** statement is used to send a value from the function back to the caller.
- A function can have multiple return statements, but only one is executed.
- If no return statement is used, the function returns None by default.

Example:

```
def square(n):
    if n < 0:
        return "Negative"
    return n * n

print(square(5))     # 25
print(square(-3))    # Negative
```

8 TYPES OF ARGUMENTS

1. Positional Arguments
Passed in correct positional order.

```
def sub(a, b):
    return a - b
print(sub(10, 4))  # 6
```

2. Keyword Arguments
Passed using parameter names.

```
print(sub(b=4, a=10))  # 6
```

3. Default Arguments
Parameters have default values.

```
def power(x, y=2):
    return x ** y
print(power(3))     # 9 (y takes default 2)
print(power(3, 3))  # 27
```

4. Variable Length Arguments

- `*args` (Non-Keyword Variable Length)

```
def add_all(*args):
    s = 0
    for i in args:
        s += i
    return s
print(add_all(1,2,3,4,5))  # 15
```

- `**kwargs` (Keyword Variable Length)

```
def show_details(**kwargs):
    for k, v in kwargs.items():
        print(k, ":", v)
show_details(name="Riya", age=17, city="Delhi")
```

9 SCOPE OF VARIABLES

- **Local Variable** → Defined inside a function; accessible only within that function.
- **Global Variable** → Defined outside a function; accessible throughout the program.
- A local variable and global variable with same name are treated separately.

Example:

```
x = 10   # global variable

def func():
    x = 5   # local variable
    print("Inside:", x)

func()
print("Outside:", x)   # Inside: 5, Outside: 10
```

10 IMPORTANCE IN PROGRAMMING

Functions make programs:

- Modular
- Easy to maintain
- Reusable
- Scalable
- Easier to test and debug

11 EXAMPLE: BUILT-IN FUNCTIONS

`print()`	→ Displays output
`len()`	→ Returns length of sequence
`type()`	→ Returns data type
`range()`	→ Generates sequence of numbers
`max()`	→ Returns maximum value
`min()`	→ Returns minimum value
`sum()`	→ Returns sum of values

12 QUICK SUMMARY

- ★ Functions are defined using 'def' keyword.
- ★ Functions can take arguments and return values.
- ★ Four types based on arguments and return value.
- ★ Arguments can be positional, keyword, default or variable length.
- ★ Variables have local and global scope.
- ★ A function can call itself (recursion).
- ★ Functions improve modularity and code quality.

WRITE FUNCTIONS • USE FUNCTIONS • REUSE FUNCTIONS • THINK MODULAR • CODE SMART

EXCEPTION HANDLING IN PYTHON

Python • Class 12 Computer Science (083) • CBSE

1 What is Exception?

An exception is an error that occurs during program execution that disrupts the normal flow of instructions.

Exception handling allows us to handle these errors gracefully without crashing the program.

Example

Dividing a number by zero raises an exception.

2 Syntax of Exception Handling

```
try:
    # Code that may raise an exception
except ExceptionType:
    # Code to handle the exception
else:
    # Code to execute if no exception occurs
finally:
    # Code that always executes
```

- **try :** Contains code that may cause an exception.
- **except :** Handles the exception.
- **else :** Executes if no exception occurs.
- **finally :** Always executes, whether exception occurs or not.

3 Example Program

```
try:
    num1 = int(input("Enter numerator: "))
    num2 = int(input("Enter denominator: "))
    result = num1 / num2
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
except ValueError:
    print("Error: Please enter a valid number.")
else:
    print("Division successful.")
finally:
    print("Thank you!")
```

Output (Sample)

```
Enter numerator: 10
Enter denominator: 0
Error: Cannot divide by zero.
Thank you!
```

4 Multiple except Blocks

```
try:
    n = int(input("Enter a number: "))
    print(10 / n)
except ZeroDivisionError:
    print("Cannot divide by zero.")
except ValueError:
    print("Invalid input! Enter a number.")
except Exception as e:
    print("Some error occurred:", e)
```

5 Common Built-in Exceptions

Exception	Description
ZeroDivisionError	Raised when division or modulo by zero occurs.
ValueError	Raised when a function receives an argument of correct type but inappropriate value.
TypeError	Raised when an operation is applied to an inappropriate type.
FileNotFoundError	Raised when a file or directory is not found.
IndexError	Raised when a sequence index is out of range.
KeyError	Raised when a dictionary key is not found.
Exception	Base class for all built-in exceptions.

6 Raising Exceptions

We can raise an exception manually using the `raise` keyword.

```
def check_age(age):
    if age < 18:
        raise ValueError("Age must be 18 or above.")
    else:
        print("Access granted.")
```

Example

check_age(16) → ValueError: Age must be 18 or above.

7 Best Practices

- ✔ Always handle specific exceptions.
- ✔ Use `else` for code that runs when no exception occurs.
- ✔ Use `finally` to release resources (files, connections, etc.).
- ✔ Avoid using bare `except:` (without specifying exception).

8 Nested try...except

```
try:
    a = int(input("Enter first number: "))
    try:
        b = int(input("Enter second number: "))
        print(a / b)
    except ZeroDivisionError:
        print("Inner: Cannot divide by zero.")
except ValueError:
    print("Outer: Invalid input!")
```

9 Quick Recall

try | except | else | finally | raise | ZeroDivisionError | ValueError | TypeError | FileNotFoundError | Exception

Exception Handling makes our programs robust and user-friendly. It helps prevent crashes and provides meaningful error messages.

Chapter Name: Function and Exception Handling

Question Type: MCQ

Q1	What is the output of the following? def f(x, y): return x % y print(f(10, 3)) a) 1 b) 3 c) 10 d) 0	
Q2	What will be the output of the following? def f(a, b=4, c=5): return a + b + c print(f(1, c=10)) a) 15 b) 20 c) Error d) 10	
Q3	Which statement is true about functions in Python? a) Functions can be nested b) Functions cannot be recursive c) Function names cannot be passed as arguments d) Functions must be defined after they are called	
Q4	What will be the output of the following? def f(x): return x * x print(f(5)) a) 5 b) 10 c) 25 d) None	
Q5	Which of the following is true about Python functions? a) A function can return multiple values b) A function cannot call another function c) Functions must always return a value d) A function definition must include parameters	
Q6	Which of the following is not a part of the python function? a) function header b) return statement c) parameter list d) function keyword	

Q7	Which of the following is not a valid function name in Python? a) my_func b) _func c) 2func d) func2	
Q8	What is the scope of variables declared inside a function? a) Global b) Local c) Universal d) None of the above	
Q9	Which of the following is the correct syntax to define a function in Python? a) function myFunc(): b) def myFunc(): c) define myFunc(): d) func myFunc():	
Q10	What is the output? def f(): return [1,2,3]; print(len(f())) a) 2 b) 3 c) 4 d) Error	
Q11	Rashmin is learning the python functions He read the topic types of python functions. He read that functions already available in the python library is called _____. Fill appropriate word in this blank : a) UDF (User Defined Function) b) Built-in Functions c) Modules d) Reusable Function	
Q12	Sam is confused between arguments and parameters. Select the fact about argument and parameter and solve her doubt a) arguments are those values being passed and parameters are those values received b) parameters are those values being passed and arguments are those values received c) arguments appear in the function header and parameters appear in the function call d) arguments can have same name and parameters can have value type.	

Q13	Which of the following function header is correct : a) def discount(rate=7,qty,dis=5) b) def discount(rate=7,qty,dis) c) def discount(rate,qty,dis=5) d) def discount(qty,rate=7,dis)	
14	What will be the output of the following Python code? i = 5 print(i,end='@@') def add(): global i i = i+7 print(i,end='##') add() print(i) a) 5@@12##15 b) 5@@5##12 c) 5@@12##12 d)12@@12##12	

15	What will be the output of the following code?
	a = 15
	def update(x):
	global a
	a += 2
	if x%2==0:
	a *= x
	else:
	a //= x
	a=a+5
	print(a, end="$")
	update(5)
	print(a)
	
	(A) 20$11 (B) 15$4 (C) 20$4 (D) 22$4	

16	What will be the output of the following code? c = 10 def increment(): global c	

	c += 5 print(c, end='@') increment() c = 30 print(c) A) 15@15@ B) 15@15 C) 15@30 D) 15@30@	
17	What will be the output of the following code? p=32 def display(): global p p=p%5 print(p, end='*') display() print(p) A. 32*2 B. 32*32 C. 2*32 D. 2*2	
18	What will be the output of the following Python code? x = 8 print(x, end='$$') def modify(): global x x = x * 2 print(x, end='@@') modify() print(x) a) 8$$16@@16 b) 8$$16@@8 c) 8$$16@@16@@ d) 8$$16@@8$$	
19	Find output p=12 q=10 def profun(q): global p p=q+5 q=p+2	

	print(p,q,end=" ") profun(p) print(p,q) (a)17 17 19 10 (b) 17 10 17 19 (c) 17 19 12 10 (d) 17 19 17 10	
20	Which of the following keywords is used to handle exceptions in Python? a) catch b) except c) error d) throw	
21	Which of the following is true about try-except-finally? a) finally block may not execute b) finally block always executes c) except block always executes d) none	
22	What will be the output of the following Python code? try: x = 10 / 0 except Exception: print("Some other error!") except ZeroDivisionError: print("Division by zero error!") a) Division by zero error! b) Some other error! c) ZeroDivisionError d) Nothing is printed	
23	What will be the output of the following Python code? try: value = int("abc") except ZeroDivisionError: print("Zero Division error!") except Exception: print("Some other error!") except ValueError: print("Conversion error!") a) Division by zero error! b) Some other error! c) Conversion error! d) Nothing is printed	

24	State whether the following statement is True or False "ZeroDivisionError" is an example of syntax error.	
25	What will be output of the following python code? try: z=3+'a' except ValueError: print("Value Error") except TypeError: print("Type Error") A. Value Error B. Type Error C. 3a D. Index Error	
26	When the given value is not suitable to perform an operation, Python will generate an error message which is known as: a. NameError b. ValueError c. TypeError d. IndexError	
27	What will be the output of the following code? try: print(5/2) print(undefined_var) except ZeroDivisionError: print("Zero Division Error") except NameError: print("Name Error") a. 2.5 followed by Zero Division Error b. 2.5 followed by Name Error c. Only Zero Division Error d. Only Name Error	
28	What will happen if an exception occurs inside the try block and there is no matching except block? A. The program ignores the exception. B. The program terminates and displays an error message. C. The finally block is skipped. D. The else block is executed.	
29	_____ is the block of exception handling that gets executed invariably whether exception occurs or not.	
30	A _____ occurs when an undefined variable is used.	

Answers

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
a)	a)	a)	c)	a)	d)	c)	b)	b)	b)
Q11	Q12	Q13	Q14	Q15	16	17	18	19	20
b)	a)	c)	c)	c)	C	D	A	D	B
21	22	23	24	25	26	27	28	29	30
B	B	B	False	B	B	B	B	finally	Name Error

Assertion Reasoning Question

Assertion (A) and Reason(R) based questions. Mark the correct choice as:

A) Both A and R are true and R is the correct explanation of A

B) Both A and R are true but R is not the correct explanation of A.

C) A is true but R is false.

D) A is false but R is true

1	Assertion (A):- If the arguments in a function call statement match the number and order of arguments as defined in the function definition, such arguments are called positional arguments. Reasoning (R):- During a function call, the argument list first contains positional argument(s) followed by default argument(s).	
2	Assertion (A):-The number of actual parameter in a function call may not be equal to number of formal parameter of the function. Reasoning (R) :- During a function call it is optional to pass the values to default parameter.	
3	Assertion (A): Global variables are accessible in the whole program. Reason (R) : Local variables are accessible only within a function or block in which it is declared.	
4	Assertion (A): In Python, the finally block is always executed, whether an exception occurs or not. Reason (R): The finally block is used to write the code that must be executed before a program terminates, such as closing files or releasing resources.	
5	Assertion (A): In Python, a function can return multiple values. Reason (R): In Python, functions return values as a tuple when returning multiple values.	

Answers

1	2	3	4	5	6
A	A	B	A	A	A

Question Type: Find Output / Short Answer / Long Answer

Q1	A school wants to create a program to calculate the average marks of students. def avgMarks(marks): total = sum(marks) return total / len(marks) student1 = [90, 85, 88, 92] print(avgMarks(student1)) (a) Identify the type of function used. (b) What will be the output? (c) If student2 = [], what error will occur?	
Ans	(a) Type of function → User-defined function (b) Output → (90+85+88+92)/4 = 88.75 (c) If student2 = [], error = ZeroDivisionError (division by zero)	
Q2	A program is written to read integers from the user and display the square: try: num = int(input('Enter a number: ')) print('Square:', num*num) except ValueError: print('Invalid input') (a) What will happen if the user enters 'ten'? (b) Which exception is handled here? (c) Modify the code to also handle division by zero error if we add print(10/num).	
Ans	(a) If user enters 'ten' → Output = Invalid input (ValueError occurs). (b) Exception handled = ValueError (c) Modified code: try: num = int(input('Enter a number: ')) print('Square:', num*num) print(10/num) except ValueError: print('Invalid input') except ZeroDivisionError: print('Division by zero not allowed')	

Q3	A company uses the following code to calculate bonus:
	def bonus(salary, percent=10):
	try:
	return salary * percent / 100
	except TypeError:
	return 'Invalid input'
	(a) What is the default bonus percentage?
	(b) Write the output of bonus(20000) and bonus('20000').
	(c) Explain how exception handling improves the reliability of the program.	

Ans	(a) Default bonus percentage = 10% (b)bonus(20000) → 2000.0 bonus('20000') → Invalid input (TypeError) (c) Exception handling improves reliability by preventing program crashes and providing user-friendly error messages when invalid data is given.	
Q4	Read the following code snippet carefully: def greet(name, msg='Good Morning'): return 'Hello'+name+msg print(greet('Amit')) print(greet('Riya', 'Good Night')) (a) What is the purpose of the default argument? (b) Write the output of the program. (c) Modify the function so that it raises an error if name is not a string.	

Ans	(a) Purpose of default argument → Provides a default value if not supplied.
	(b) Output:
	Hello Amit Good Morning
	Hello Riya Good Night
	(c) Modified function:
	def greet(name, msg='Good Morning'):
	if not isinstance(name, str):
	raise TypeError("Name must be a string")
	return f'Hello {name}, {msg}'	

Q5	A function is defined as follows: def calc(a, b=0, c=0): return a + b + c	

	(a) What kind of arguments are used in this function? (b) Write the output for: calc(5), calc(5, 10), calc(5, 10, 15). (c) Can we call the function as calc(b=10, a=5)? Justify.	
Ans	(a) Arguments → Positional + Default arguments (b)calc(5) → 5 calc(5, 10) → 15 calc(5, 10, 15) → 30 (c) Yes, calc(b=10, a=5) is valid because keyword arguments can be given in any order	

Q6	A bank application accepts withdrawal amount:
	def withdraw(balance, amount):
	try:
	if amount > balance:
	raise ValueError('Insufficient funds')
	return balance - amount
	except ValueError as e:
	return e
	(a) Which type of exception is being raised here?
	(b) What will be the output for withdraw(5000, 2000) and withdraw(3000, 4000)?
	(c) Suggest a real-life scenario where such exception handling is necessary.	

Ans	(a) Exception raised = ValueError (b)withdraw(5000, 2000) → 3000 withdraw(3000, 4000) → Insufficient funds (c) Real-life scenario → ATM withdrawals / Online banking transactions to prevent overdrafts.	
Q7	What are the parts of functions? Explain with a suitable example?	
Ans	The arts of functions are as follows: Function header: It starts with def followed by the function name and required parameters. Parameters are the input variables written into brackets. These parameters are supplied in the function calling statement. Function Body: This is the main part of a program. It contains the main block of the program. The set of instructions such as calculations, logical comparisons etc is written here in this part. It ends with a return statement. Indentation is very	

	important for the function body. Function calling statement: This is the final part of a function. It invokes the function and returns the output as instructed into the function body. def function_name(): ----------#statements Function_name() # calling statements	
Q8	What are the comments? What are the role comments in the program? How to write single-line comments and multi-line comments?	
Ans	Comments are the additional explanatory text written in the function to provide some description or explanation about the statement or block of code. Comments play an important role in understanding a program. It provides a descriptive explanation of the written statements. It makes the intention very clear for what the statement is used or what process is going to be done. The single-line comment is written by # followed by the text. The multiline comments start with ”’ and are followed by the text then ends with ”’.	
Q9	Write and explain the types of functions supported by python?	
Ans	Built-in Functions: Pre-defined functions of python such as len(), type(), input() etc. Functions defined in modules: Functions defined in particular modules, can be used when the module is imported. A module is a container of functions, variables, constants, and classes in a separate file which can be reused. User-Defined Functions: Function created by the programmer.	
Q10	What are the arguments supported by python? Explain each of them with a suitable example.	
Ans	Python supports four argument types: 1. Positional Arguments: Arguments passed to a function in correct positional order, no. of arguments must match with no. of parameters required. 2. Default Arguments: Assign a default value to a certain parameter, it is used when the user knows the value of the parameter, default values are specified in the function header. It is optional in the function call statement. If not provided in the function call statement then the default value is	

	considered. Default arguments must be provided from right to left. 3. Key Word Arguments: Keyword arguments are the named arguments with assigned values being passed in the function call statement, the user can combine any type of argument. 4. Variable Length Arguments: It allows the user to pass as many arguments as required in the program. Variable-length arguments are defined with the * symbol.	
Q11	Differentiate between parameters and arguments.	

Ans	Parameters refer to the variables listed in a function's declaration, defining the input that the function can accept.
	Arguments, however, are the actual values passed to the function when it is called, filling the parameters during execution.
	Def sum(a,b): # a and b parameter
	Print(a+b)
	X=5
	Y=7
	Sum(x,y) # x,y are arguments	

Q12	What is the local variable and global variable? Explain with an example.	
Ans	Global Variable: A variable that is declared in top-level statements is called a global variable. To access the value of a global variable user need to write a global keyword in front of the variable in a function. Local Variable: A name declared in a specific function body is called a local variable. message = "I am global" def my_function(): # 2. This is a LOCAL variable status = "I am local" print(message) # Works! Can read the global variable inside a function print(status) # Works! Can read the local variable inside its own function	
Q13	Every syntax error is an exception but every exception cannot be a syntax error." Justify the statement.	

Ans	A syntax error occurs when code does not follow the rules of a programming language, but every exception cannot be a syntax error because there are several types of errors that can be encountered during the execution of the program, such as division by zero or file not found errors. These errors are not syntax errors.	
Q14	Differentiate between ValueError, TypeError, and ZeroDivisionError. Give examples?	
Ans	TypeError: This exception occurs when an operation or function is applied to an object of an inappropriate or unsupported data type. The type of the operand or argument is incorrect for the operation being performed. ValueError: This exception is raised when a function or operation receives an argument of the correct data type, but the value of that argument is inappropriate or out of range for the operation. The type is correct, but the content is problematic. ZeroDivisionError: This is a specific type of arithmetic error that occurs when an attempt is made to divide a number by zero. It's a specialized form of error related to a fundamental mathematical constraint.	
Q15	Write a program to demonstrate the use of finally block even when exception occurs?	

Ans	try:
	  x = int('abc')
	except ValueError:
	  print('Error occurred')
	finally:
	  print('Finally executed')	

Q16	Write a function remove_element() in Python that accepts a list L and a number n. If the number n exists in the list, it should be removed. If it does not exist, print a message saying "Element not found"	

Ans	def remove_element(L, n):
	  if n in L:
	  L.remove(n)
	  print(L)
	  else:
	  print("Element not found")	

Q17	Write a Python function add_contact() that accepts a dictionary phone_book, a name, and a phone number. The function should add the name and phone number to the dictionary. If the name already exists, print "Contact already exists" instead of updating it.	

Ans	def add_contact(phone_book, name, number):
	  if name in phone_book:
	  print("Contact already exists")
	  else:
	  phone_book[name] = number
	  print("Contact added successfully")	

18	Rewrite the following code in Python after removing all syntax error(s) and underline each
	correction done in the code .
	define fun1():
	  30 = num
	  for k range(0,num):
	  if k%4=0 :
	  print(k*4)
	  else:
	  print(k+3)	

Ans	**def fu**n1():
	  num=30
	  for k **in** range(0,num):
	  if k%4==0 :
	  print(k*4)
	  else:
	  print(k+3)	

19	**Predict the output of the Python code given below:**
	def my_city (L,N):
	  for i in range(0,N):
	  if len(L)>4:
	  L[i]=L[i]+L[i]
	  else:
	  L[i]=L[i]
	sub=['Delhi','Jaipur','Agra','Surat','Mumbai','Bhopal']
	my_city(sub,6)
	print(sub)	

Ans	['DelhiDelhi', 'JaipurJaipur', 'AgraAgra', 'SuratSurat', 'MumbaiMumbai', 'BhopalBhopal']	
20	The code provided below is intended to calculate sum of odd numbers in a list. However, there are syntax and logical errors in the code. Rewrite the code in python after removing all error(s). Underline each correction done in the code. def Sum-Odd(): numbers = [13,8,15,20]	

	total = 0
	for val in numbers():
	if val%2==0:
	total = total + val
	print(total)
	Sum_Odd()	

Ans	def **Sum_Odd():** numbers = [13,8,15,20] total = 0 for val in **numbers:** if val%**2!=0:** total = total + val **print(total)** Sum_Odd()	
21	Write a function remove_duplicates() in Python that accepts a list L and returns a new list containing only the unique elements from L, after removing the duplicates.	

Ans	def remove_duplicates(L):
	new_list = []
	for ele in L:
	if ele not in new_list:
	new_list.append(ele)
	return new_list	

22	Write a Python function update_score() that accepts a dictionary named scores, a player name and a new score. If the player name already present in dictionary, update the score, otherwise, display the message "Player not found".	

Ans	def update_score(scores, playername, new_score):
	if playername in scores:
	scores[player] = new_score
	else:
	print("Player not found")	

23	The code provided below is intended to find the sum of all elements present in the list. Rewrite it after removing all the errors. Also, underline all the corrections made. define list_sum(numbers) total = 0 For i in numbers: total =+ i return total Print(list_sum([1,2,3,4]))	
Ans	**def** list_sum(**numbers) :** total = 0	

	for i in numbers: **total+ = i** return total **print**(list_sum([1,2,3,4]))	
24	Write a function replace_element(L, old, new) that replaces the first occurrence of old with new in list L. If old is not found, print "Element not found".	

Ans	def replace_element(L, old, new):
	if old in L:
	index = L.index(old) # Find the index of first occurrence
	L[index] = new # Replace it with new value
	print("Updated List:", L)
	else:
	print("Element not found")	

25	Write a Python function delete_contact(phone_book, name) that accepts a dictionary phone_book and a name. The function should remove a contact with the parameter 'name' from the dictionary. If the contact doesn't exist, print "Contact not found". Note: Dictionary has name as key and phone number as value.	

Ans	def delete_contact(phone_book, name):
	if name in phone_book:
	del phone_book[name] # Remove the contact
	print(f"Contact '{name}' deleted successfully.")
	else:
	print("Contact not found")	

26	Write a function double_element() in Python that accepts a list L and a number n. If the number n exists in the list, it should be updated to its double value. If it does not exist, print a message saying "Element not found".	

Ans	def double_element(L, n):
	if n in L:
	index = L.index(n)
	L[index] = n * 2
	print("Updated List:", L)
	else:
	print("Element not found")
	
	L = [2, 5, 8, 10]
	double_element(L, 5)	

27	Write a Python function add_emplyoee() that accepts a dictionary company_emplyoee, an employee number, a name and department. The function should add the employee number as key and name, department as value to the dictionary. If the employee number already exists, print "Employee already exists" instead of updating it.	
Ans	def add_employee(company_employee, emp_no, name, department): if emp_no in company_employee:	

	print("Employee already exists") else: company_employee[emp_no] = (name, department) print("Employee added successfully")	
28	Write a python function count_element() to accept a set of negative as well as positive numbers in a list. Display the sum of negative even numbers and positive odd numbers available in the list. (Both sum separately).	

Ans	def count_element(numbers):
	sum_neg_even = 0
	sum_pos_odd = 0
	for n in numbers:
	if n < 0 and n % 2 == 0:
	sum_neg_even += n
	elif n > 0 and n % 2 != 0:
	sum_pos_odd += n
	print("Sum of negative even numbers:", sum_neg_even)
	print("Sum of positive odd numbers:", sum_pos_odd)	

29	Write a Python function to create a dictionary which accepts months and days with month as key and days as value. Display the months having 31 days.	

Ans	def months():
	months_days = {}
	n = int(input("Enter number of months you want to input: "))
	for _ in range(n):
	month = input("Enter month name: ")
	days = int(input(f"Enter number of days in {month}: "))
	months_days[month] = days
	months_31_days = [month for month, days in months_days.items() if days == 31]
	print("Months having 31 days:", months_31_days)	

30	Predict the output of the following Python function snippet:
	def Show(n):
	for k in range(1,n,2):
	if(k%7==0):
	print(k**2)
	else:
	print(k+7)
	Show(10)	

Ans	8 10 12 49 16	

31	Write a Python function ZeroEnding(Scores) to add all those values in the list of Scores, which are ending with zero(0) and display the sum. For example: If the Scores contains [235,100,455,300,111,600] Then the sum should be displayed as 1000.	

Ans	def ZeroEnding(Scores): #Sum of Numbers ending with Zero in the given List
	  sumZero=0
	  for value in Scores:
	  if value%10==0:
	  sumZero+=value
	  print("Sum of Numbers ending with Zero in the given List=",sumZero)	

32	Write a Python function to create a dictionary from a given text. The new dictionary should contain each word as the key and number of characters in each word as the value in the given text. For example: If the given text is "Are you keeping well?" Then the output is {'Are': 3, 'you': 3, 'keeping': 7, 'well?': 5}	

Ans	def credict(txt):
	  words=txt.split()
	  dwords=dict()
	  for word in words:
	  dwords[word]=len(word)
	  print("Newly created dictionary from the given text is ",dwords)	

33	The code given below is intended to find the reverse of a string. But there are syntax and logical errors in the code. Rewrite the above code after removing all errors. Also underline all the corrections made.
	Def revstring(str1):
	  rstr=""
	  for i in range(len(str1)-1,-1):
	  rstr=str1[i]
	  return rstr
	
	#Main Program
	print("Python Program to Revese a String:")
	str0=input("Enter the String:")
	Print("Reverse of ",str0,"is",revstring(str0))	

Ans	def revstring(str1):
	  rstr=""
	  for i in range(len(str1)-1,-1,-1):
	  rstr+=str1[i]
	  return rstr
	
	#Main Program
	print("Python Program to Revese a String:")
	str0=input("Enter the String:")
	print("Reverse of ",str0,"is",revstring(str0))	

34	Predict the output of the Python code given below:
	def update(nums,n):
	  for value in range(n):
	  if nums[value]%3 == 0:
	  nums[value]%=4
	  if nums[value]%2 == 0:
	  nums[value]*=2
	
	lstnum = [81,82,18,48,99,60,25]
	update(lstnum,4)
	for ele in lstnum:
	  print(ele,end="#")	

Ans	1#164#4#0#99#60#25#	
35	Write a Python function update_marks() that accepts a dictionary Marks with student names as keys and their marks as values, along with a name and new_mark. The function should update the student's mark if the name exists, otherwise print "Student not found".	

Ans	def update_marks(Marks, name, new_mark):
	  if name in Marks:
	  Marks[name] = new_mark
	  print("Marks updated")
	  else:
	  print("Student not found")	

36	A. Write a function Insert_pn() in Python that accepts a list named Perfect and a number n. If the number is a perfect square, add the number to the end of the given list else display a message stating "Number is not Perfect number"	

Ana	import math
	
	def Insert_pn(Perfect, n):
	  if math.isqrt(n) ** 2 == n:
	  Perfect.append(n)
	  else:
	  print("Number is not Perfect number")	

37	Write a Python function topper() that accepts a dictionary resultsheet that contains student details { rollno: [name,mark]}. The function should return the name and rollno of the student who scored highest mark. Display as "Topper is xyz with rollno abc".	

Ans	def topper(resultsheet):
	  max_marks = -1
	  top_name = ""
	  top_roll = ""
	
	  for rollno in resultsheet:	

```
        name, marks = resultsheet[rollno]
        if marks > max_marks:
            max_marks = marks
            top_name = name
            top_roll = rollno

    return top_name, top_roll

resultsheet = {
    101: ["Rahul", 85],
    102: ["Anita", 92],
    103: ["Rohan", 88]
}

name, rollno = topper(resultsheet)
print("Topper is", name, "with rollno", rollno)
```

Name: ARCHANA GUPTA

School: PM SHRI KV NO3 AGRA CANTT

TEXT FILE HANDLING

Python • Class 12 Computer Science (083) • Revision Sheet

1 What is a Text File? stores data as ASCII / Unicode characters

- A text file stores data in **human-readable** form — each line ends with a newline character `\n`.
- Default extension is `.txt` — can be opened & read using any text editor (Notepad, VS Code).
- Python's default file-handling mode is **text mode**. On writing, `\n` is translated to the OS line separator.

2 Opening & Closing a File

```
# Syntax
file_object = open("filename", "mode")
# ... read/write operations ...
file_object.close()

# Recommended way — auto-closes the file
with open("data.txt", "r") as f:
    content = f.read()
```

3 File Opening Modes

Mode	Full Form	Description
`r`	Read	Opens for reading (default). Error if file doesn't exist.
`w`	Write	Opens for writing. Creates new / **overwrites** existing file.
`a`	Append	Opens for writing at end of file. Creates file if absent.
`x`	Exclusive	Creates a new file; errors if the file already exists.
`r+`	Read + Write	File must exist; pointer starts at beginning.
`w+`	Write + Read	Overwrites file; also allows reading.
`a+`	Append + Read	Appends and allows reading; pointer at end.

4 Reading & Writing Functions

Function	Purpose
`f.read()`	Reads the **entire file** as one string.
`f.read(n)`	Reads **n** characters/bytes from current pointer position.
`f.readline()`	Reads **one single line** including the newline.
`f.readlines()`	Reads all lines into a **list of strings**.
`f.write(str)`	Writes a string to the file (no auto newline).
`f.writelines(list)`	Writes a list of strings to the file.

5 File Pointer Methods

Method	Use
`f.tell()`	Returns current position of the file pointer (in bytes).
`f.seek(off,w)`	Moves pointer to byte **off** from reference point **w**.

w value	Reference Point
0	Beginning of file (default)
1	Current position
2	End of file

6 3 Ways to Read a File

- **Loop over file object:** `for line in f:` print(line)
- **readlines() + loop:** lines = f.readlines(); for l in lines: ...
- **read() + split():** words = f.read().split()

7 Sample Program

```
# Count words & lines in a text file
with open("story.txt","r") as f:
    lines = f.readlines()
    wc = 0
    for ln in lines:
        wc += len(ln.split())
print("Lines:", len(lines))
print("Words:", wc)
```

Exam Tip: Use `with open()` — it auto-closes the file even if an exception occurs. No need to call `close()` separately!

Memory Tip → Read Won't Ask Xtra = r · w · a · x

Common Errors to Avoid

- ✘ Forgetting to close a file opened without `with` → wastes memory / data loss.
- ✘ Using mode `'w'` when you meant `'a'` → erases existing data!
- ✘ Reading past end-of-file returns an empty string `''`, not an error.

8 Viva Voce Questions

1. Default mode of `open()` function? → **'r' (text read mode)**
2. Which function reads only one line? → **readline()**
3. What does `seek(0,2)` do? → **moves pointer to end of file**
4. Which statement auto-closes a file? → **with statement**
5. Mode to add data without erasing content? → **'a' (append)**

9 Quick Recall

open() · close() · read() · readline() · readlines() · write() · writelines() · tell() · seek() · with · EOF = ''

CHAPTER NAME: TEXT FILE HANDLING

Creating, Opening, Reading & Writing Files

1. What is a Text File?

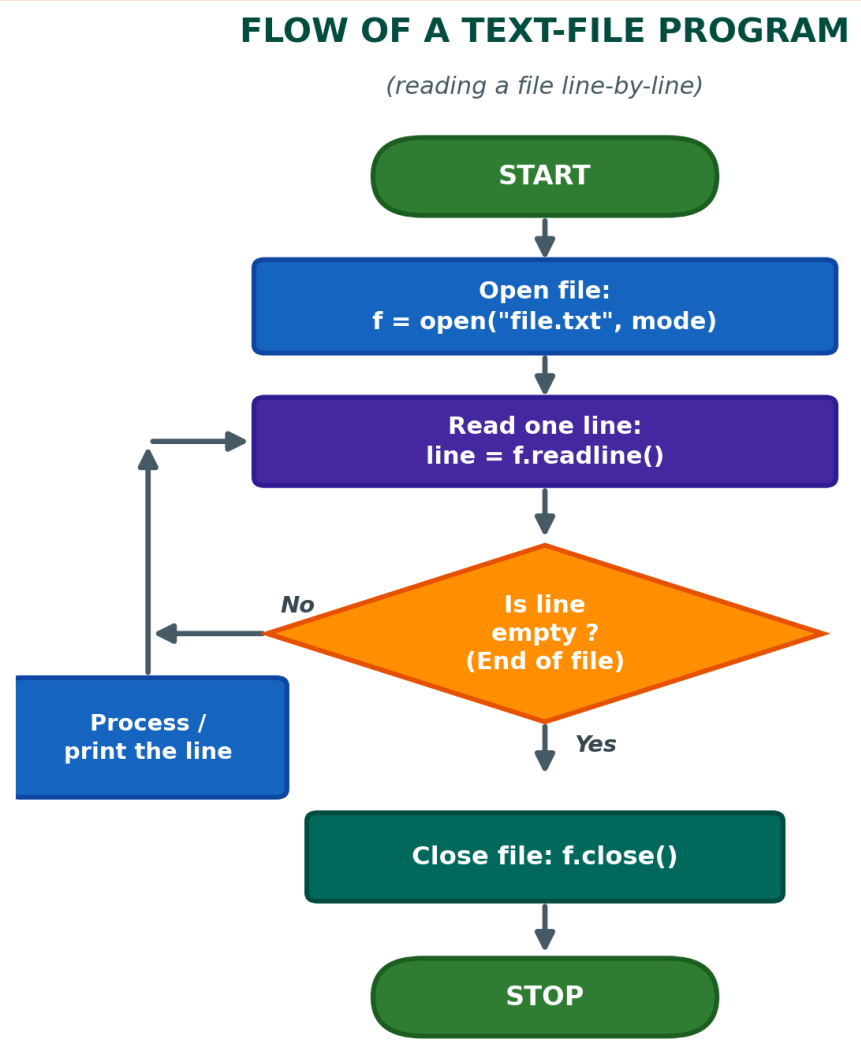
A text file stores data as plain, human-readable characters (ASCII/Unicode), organised in lines separated by a newline (\n). Python treats a text file as a stream of characters, accessed through a file object created with the built-in open() function.

Syntax: file_object = open(filename, mode)

```
f = open("data.txt", "w")
```

2. Flow of the Program

The flowchart below traces the standard logic used to process a text file: open it, read line by line in a loop until the end of file is reached, then close it.



3. File Opening Modes

Mode	Full Form	Meaning
'r'	Read	Opens for reading only. File must already exist (else FileNotFoundError). Pointer at start.
'w'	Write	Opens for writing. Creates a new file, or overwrites/truncates an existing one.
'a'	Append	Opens for writing; existing content kept, new data added at the end.
'r+'	Read + Write	Both read & write; file must exist; pointer starts at beginning.
'w+'	Write + Read	Both read & write; overwrites existing file, creates new if absent.
'a+'	Append + Read	Both read & write; pointer starts at end for writing.

4. Creating & Writing a File

```
f = open("story.txt", "w")
f.write("Hello students\n")
f.write("Learning file handling\n")
f.close()
# write() writes one string; \n needed for newline
```

4.1 writelines() — Write a List of Strings

```
lines = ["Line 1\n","Line 2\n","Line 3\n"]
f = open("story.txt", "a")
f.writelines(lines)
f.close()
# appends all list items; no auto newline added
```

4.2 Using with (Best Practice)

```
with open("story.txt", "r") as f:
   data = f.read()
   print(data)
# file auto-closes even if an error occurs
# no need to call f.close() explicitly
```

Why close a file? Closing flushes unsaved data from the buffer to disk and frees system memory. If not closed explicitly, Python closes it automatically when the program ends — but data loss risk remains if the program crashes mid-way.

5. Read Methods — read() / readline() / readlines()

```
f = open("story.txt", "r")
f.read()          # whole file as ONE string
f.read(10)        # first 10 characters only
f.readline()      # ONE line (with \n), as string
f.readlines()     # ALL lines as a LIST of strings
f.close()
```

Reading line by line till end of file:

```
f = open("story.txt", "r")
while True:
   line = f.readline()
   if not line:
      break
   print(line.strip())
f.close()
```

Method	Returns	Typical Use
read()	Single string (entire file, or n chars)	Counting characters, vowels, searching text.
readline()	Single string (one line incl. \n)	Line-by-line processing with a loop.
readlines()	List of strings (each line an element)	Counting lines, iterating with for line in list.

6. File Pointer: tell() & seek()

6.1 tell() — Where is the pointer?

```
f = open("story.txt", "r")
f.read(5)
print(f.tell())    # -> 5 (bytes read so far)
f.close()
# tell() returns the CURRENT position
# of the file pointer, in bytes from start
```

6.2 seek() — Move the pointer

```
f = open("story.txt", "r")
f.seek(0)          # go to the beginning
f.seek(10)         # jump to byte offset 10
data = f.read()    # read from that position
f.close()
# seek(offset) repositions pointer for next
# read/write operation
```

seek(offset, whence) — whence = 0 (default, from start), 1 (from current position), 2 (from end); text-mode files typically use whence = 0 only. Combining both: read a chunk, check tell(), then seek() back to re-read it.

Quick recap: read → whole text • readline → one line at a time • readlines → list of all lines • tell()/seek() → track and move the file pointer.

7. CBSE Board-Pattern Questions (2022 – 2026 Style)

S.No	Question
1	***[Board 2024 Pattern]*** Write function showInLines() that reads STORY.TXT and prints every sentence on a separate line (a sentence ends with ., ? or !).

Ans	def showInLines():
	  f = open("STORY.TXT", "r")
	  text = f.read()
	  f.close()
	  sentence = ""
	  for ch in text:
	  sentence += ch
	  if ch in ".?!":
	  print(sentence.strip())
	  sentence = ""

2	***[Frequent Pattern]*** Write a function countwords() to count and display the total number of words in the text file story.txt.

Ans	def countwords():
	  f = open("story.txt", "r")
	  data = f.read()
	  f.close()

S.No	Question
	words = data.split() print("Total words =", len(words))
3	***[Frequent Pattern]*** Write a function to count the number of lines in story.txt that start with the alphabet 'A' (or any given letter).

Ans	def countA():
	f = open("story.txt", "r")
	lines = f.readlines()
	f.close()
	count = 0
	for line in lines:
	if line.strip() and line[0].upper() == 'A':
	count += 1
	print("Lines starting with A =", count)

4	***[Frequent Pattern]*** Write function AMCount() to read each character of STORY.TXT and count occurrences of alphabets 'A'/'a' and 'M'/'m'.
Ans	def AMCount(): f = open("STORY.TXT", "r") data = f.read() f.close() a = data.count('A') + data.count('a') m = data.count('M') + data.count('m') print("A or a =", a, ", M or m =", m)
5	***[Frequent Pattern]*** Write a function DISPLAYWORDS() to read STORY.TXT and display only those words which have fewer than 4 characters.

Ans	def DISPLAYWORDS():
	f = open("STORY.TXT", "r")
	words = f.read().split()
	f.close()
	for w in words:
	if len(w) < 4:

S.No	Question
	print(w)
6	***[Conceptual]*** Differentiate readline() and readlines().
Ans	readline() returns the next single line as a string; readlines() returns all remaining lines as a list of strings — useful when you need to loop through every line by index.
7	***[Conceptual]*** Differentiate 'w' and 'a' modes; r+ and w+ modes.
Ans	'w' erases existing content before writing; 'a' preserves it and adds new data at the end. r+ needs an existing file and keeps its data; w+ overwrites or creates a fresh file — both r+ and w+ allow reading and writing.
8	***[Application]*** Accept sentences from the user until "END" is typed, store each in hello.txt, then display only the sentences that begin with an uppercase letter.

Ans	f = open("hello.txt", "w")
	while True:
	  s = input("Enter line (END to stop): ")
	  if s == "END":
	  break
	  f.write(s + "\n")
	f.close()
	
	f = open("hello.txt", "r")
	for line in f.readlines():
	  if line.strip() and line[0].isupper():
	  print(line.strip())
	f.close()

8. Board Favourites — Quick Tips

- Word / line / character counting patterns.
- Filtering lines or words by their first letter or length.
- Sentence splitting using end-of-sentence punctuation.
- Read-mode comparisons — practice each pattern with a different file or letter.

Question Type: MCQ AND DESCRIPTIVE

Q1	Which function raises an error if you try to write to a file opened in "r" mode? A. read() B. readline() C. write() D. writelines()	1
Q2	What is the output of the following code if the file is empty? file = open("data.txt", "r") print(file.read()) A. Error B. None C. Empty string D. 0	1
Q3	Which of the following correctly opens a file for appending and reading? A. "r+" B. "w+" C. "a+" D. "r"	1
Q4	What will the following code do? file = open("data.txt", "a") file.write("Hello\n") file.close() A. Overwrites the file with "Hello" B. Appends "Hello" to the end of the file C. Reads the first line of the file D. Deletes the file	1
Q5	Which of the following will raise an error if the file does not exist? A. open("data.txt", "r") B. open("data.txt", "w") C. open("data.txt", "a") D. open("data.txt", "w+")	1
Q6	Which mode allows you to open a file for reading and writing without truncating the file? A. r B. w C. r+ D. w+	1
Q7	What is returned by the tell() method? A. Number of characters in the file B. Current position of the file pointer C. List of lines in the file D. Entire file content	1
Q8	Which of the following modes opens a file for writing and truncates the file if it already exists? A. r B. w C. a D. r+	1

Q9	What happens if you try to open a non-existent file in r mode? A. A new file is created. B. An error (FileNotFoundError) is raised. C. The file is opened for writing. D. The file pointer is set to the end.	1
Q10	Which of the following iterable objects can be passed to the writelines() method? A. List of strings B. Integer C. Dictionary D. Single string	1
Q11	Which of the following modes allows both reading and writing, and creates a new file if it does not exist? A. r B. w+ C. a D. r+	1
Q12	What does the following code do when executed? file = open("data.txt", "r") file.seek(10, 1) A. Moves the file pointer to the 10th byte from the beginning of the file. B. Moves the file pointer 10 bytes forward from the current position. C. Moves the file pointer 10 bytes backward from the end of the file. D. Reads 10 bytes from the current position.	1
	ASSERTION (A) and REASONING (R) Mark the correct choice as A. Both (A) and (R) are true and (R) is the correct explanation for (A). B. Both (A) and (R) are true and (R) is not the correct explanation for (A). C. (A) is true but (R) is false. D. (A) is false but (R) is true	
Q13	**Assertion (A):** Opening a file in 'a' mode will delete its previous content. **Reason (R):** The 'a' mode appends new data at the end of the existing file content.	1
Q14	**Assertion (A):** The with statement ensures a file is properly closed after its block finishes. **Reason (R):** Using with avoids the need to explicitly call close ( ) on a file object.	1

Q15	**Assertion (A):** Files must be closed using the close ( ) function to ensure data is saved properly. **Reason (R):** Not closing a file may result in data loss or corruption.	1
Q16	**Assertion (A):** Opening a file in write mode('w') will delete its existing contents. **Reason (R):** In Python, the 'w' mode creates a new file if it doesn't exist, but preserves old content if the file exists.	1
Q17	**Assertion (A):** The writelines( ) method adds newline characters automatically after each line. **Reason (R):** writelines( ) writes a list of strings.	1

Answers:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
C	C	C	B	A	C	B	B	B	A
Q11	Q12	Q13	Q14	Q15	Q16	Q17			
B	B	D	A	A	C	D			

Question Type: Long Answer

Q1	**A text file data.txt contains some text. Write a function counter() in Python to count the number of characters, words, and lines in the file.**	3

Ans	def counter():
	  # Open the file in read mode
	  f = open("data.txt", "r")
	  lines = f.readlines()
	  num_lines = len(lines)
	  num_words = 0
	  num_chars = 0
	  for line in lines:
	  num_chars += len(line) # Count all characters including spaces
	  num_words += len(line.split()) # Count words in line
	  print("Number of lines:", num_lines) # Display the counts
	  print("Number of words:", num_words)
	  print("Number of characters:", num_chars)
	  f.close()
	# Call the function
	counter()
		3

Q2	**A text file story.txt contains some text. Write a function longest_word() in Python to display the longest word present in the file.**	3

Ans	def longest_word():
	  f = open("story.txt", "r") # Open the file in read mode
	  longest_word = ""
	  for line in f:
	  words = line.split()
	  for w in words:
	  word = w.strip(",.!?;:") # remove punctuation
	  if len(word) > len(longest_word):
	  longest_word = word
	  print("Longest word in the file is:", longest_word)
	  f.close()
	# Call the function
	longest_word()
		3

Q3	**A text file myfile.txt contains some text. Write a function counter() in Python to count the number of vowels (a, e, i, o, u) present in the file (case-insensitive).**	3

Ans	# Function to count vowels in a file
	def counter():
	  # Open the file in read mode
	  f = open("myfile.txt", "r")
	  text = f.read()
	  # Define vowels
	  vowels = "aeiouAEIOU"
	  # Count vowels
	  count = 0
	  for char in text:
	  if char in vowels:
	  count += 1
	  print("Number of vowels in the file:", count)
	# Call the function
	counter()
		3

Q4	**Write a method display() in Python to read lines from a text file DIARY.TXT and display those lines which start with the alphabets P.**	3

Ans	def display():
	  # Open the file in read mode
	  f = open("DIARY.TXT", "r")
	  # Read each line
	  for line in f:
	  # Remove leading/trailing whitespaces
	  line = line.strip()
	  # Check if the line starts with 'P'
	  if line.startswith('P'):
	  print(line)
	  # Close the file
	  f.close()
	# Call the function
	display()
		3

Q5	**Write a method Task() in Python that copies a text file "source.txt" onto "target.txt" barring the lines starting with ! sign.**	3

Ans	def Task():
	# Open source file in read mode and target file in write mode
	with open("source.txt", "r") as src, open("target.txt", "w") as tgt:
	for line in src:
	if not line.startswith('!'): # Skip lines starting with '!'
	tgt.write(line)
	# Call the function
	Task()
		3

Q6	**Write a method/function COUNTWORDS() in Python to read contents from a text file DECODE.TXT, to count and return the occurrence of those words, which are having 5 or more characters.**	3

Ans	def COUNTWORDS():
	count = 0
	with open('DECODE.TXT', 'r') as f:
	for line in f:
	words = line.split()
	for word in words:
	if len(word) >= 5:
	count += 1
	return count
		3

Q7	**Write a method/function COUNTLINES() in Python to read lines from a text file CONTENT.TXT, and display those lines, which have '@' anywhere in the line.**	3

Ans	def COUNTLINES():
	with open('CONTENT.TXT', 'r') as f:
	for line in f:
	if '@' in line:
	print(line.strip())
		3

Q8	**Write a function, c_words() in Python that separately counts and displays the number of uppercase and lowercase alphabets in a text file, Words.txt.**	3

Ans	def c_words():
	upper = 0
	lower = 0
	with open('Words.txt', 'r') as f:
	text = f.read()
	for char in text:
	if char.isupper():
	upper += 1
	elif char.islower():
	lower += 1
	print("Uppercase alphabets:", upper)
	print("Lowercase alphabets:", lower)
		3

Q9	**Write a function in Python that displays the book names having 'Y' or 'y' in their name from a text file Bookname.txt.**	3

Ans	def display_books_with_y():
	with open('Bookname.txt', 'r') as f:
	for line in f:
	if 'Y' in line or 'y' in line:
	print(line.strip())
		3

Q10	**Write a function RevString() to read a textfile Input.txt and prints the words starting with 'O' in reverse order. The rest of the content is displayed normally.**	3

Ans	def RevString():
	with open('Input.txt', 'r') as f:
	text = f.read()
	words = text.split()
	for word in words:
	if word.startswith('O'):
	print(word[::-1], end=' ')
	else:
	print(word, end=' ')
		3

Q11	**Write the definition of a Python function named LongLines() which reads the contents of a text file LINES.TXT and displays those lines having at least 10 words.**	3

Ans	def LongLines():
	  with open('LINES.TXT', 'r') as f:
	  for line in f:
	  words = line.split()
	  if len(words) >= 10:
	  print(line.strip())
		3

Q12	**Write a function count_Dwords() in Python to count the words ending with a digit in a text file Details.txt.**	3

Ans	def count_Dwords():
	  count = 0
	  with open('Details.txt', 'r') as f:
	  text = f.read()
	  words = text.split()
	  for word in words:
	  if word and word[-1].isdigit():
	  count += 1
	  print(count)
		3

Q13	**A text file story.txt contains some text. Write a Python program to count and display the number of words starting with the letter A or a.**	3

Ans	# Open the file in read mode
	f = open("story.txt", "r")
	count = 0
	for line in f:
	  words = line.split()
	  for w in words:
	  if w.startswith('A') or w.startswith('a'):
	  count += 1
	f.close()
	print("Number of words starting with A/a:", count)
		3

Q14	**A text file notes.txt contains some text. Write a Python program to count the number of lines which start with the word "The" (case-sensitive).**	3

Ans	# Open the file in read mode
	f = open("notes.txt", "r")
	count = 0
	for line in f:
	  # strip leading spaces before checking
	  if line.lstrip().startswith("The"):
	  count += 1
	f.close()
	print("Number of lines starting with 'The':", count)
		3

CSV FILE HANDLING

Python • Class 12 Computer Science (083) • Revision Sheet

1 What is a CSV File? Comma Separated Values

- A plain **text file** where each line is a data record and each field is separated by a **comma** (default delimiter).
- Widely used to exchange **tabular data** between spreadsheets, databases & programs — extension `.csv`.
- Python's built-in `csv` module makes reading/writing CSV files easy & reliable.

2 Importing & Opening

```
import csv
# always open with newline='' to avoid blank rows
with open("data.csv","w",newline='') as f:
    ...
```

3 Writing to a CSV File

Function	Purpose
`csv.writer(f)`	Returns a **writer object** bound to file f.
`w.writerow(list)`	Writes **one row** from a list of values.
`w.writerows(list_of_lists)`	Writes **multiple rows** at once.

```
writer = csv.writer(f)
writer.writerow(["Roll","Name","Marks"])
writer.writerows([[1,"Ravi",88],[2,"Isha",92]])
```

4 Reading from a CSV File

```
with open("data.csv","r",newline='') as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)   # row is a list
```

Common Mistakes to Avoid

- ✘ Forgetting `newline=''` when opening on Windows → blank rows inserted between records.
- ✘ Mismatched `fieldnames` order with actual dictionary keys in DictWriter.
- ✘ Not calling `writeheader()` before writing rows with DictWriter.

CSV vs Text File

- → CSV has a **fixed structure** (rows & columns); plain text files don't.
- → CSV is best for **tabular / record-based** data like marksheets, databases.

5 Dictionary-based CSV I/O

Class	Purpose
`csv.DictWriter(f, fieldnames)`	Writes rows from **dictionaries**; needs column headers.
`csv.DictReader(f)`	Reads each row as an **OrderedDict** using the header row as keys.

```
fields = ["Roll","Name"]
w = csv.DictWriter(f, fieldnames=fields)
w.writeheader()
w.writerow({"Roll":1,"Name":"Ravi"})
```

6 Useful Parameters

Parameter	Use
`delimiter`	Character separating fields (default `,`).
`quotechar`	Character used to quote fields with special characters.
`lineterminator`	String used to end a row (default `\r\n`).

⭐ **Exam Tip:** Always open CSV files with `newline=''` — otherwise extra blank rows appear on Windows systems!

7 Reading with DictReader

```
with open("data.csv",newline='') as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row["Name"], row["Marks"])
```

Memory Tip → CSV = Comma Separated Values — writer writes lists, DictWriter writes dicts

Advantages of CSV

- ✔ Lightweight, human-readable, opens directly in Excel / Google Sheets.
- ✔ Easy to import/export between different applications.

8 Viva Voce Questions

1. Which module is used to work with CSV files? → **csv**
2. Default delimiter used in a CSV file? → **comma (,)**
3. Function to write a single row? → **writerow()**
4. What does DictReader return for each row? → **an OrderedDict**
5. Why use newline='' while opening? → **prevents extra blank rows**

9 Quick Recall

`import csv` `writer()` `reader()` `writerow()` `writerows()` `DictWriter` `DictReader` `newline=''` `delimiter`

CSV File Handling

the csv module

Write, Read, Update, Delete records — using List rows with Python's csv module

what.md

What is a CSV file?

CSV = Comma Separated Values — a plain text file that stores tabular data (rows & columns), with each field in a row separated by a comma. Python's **csv** module reads and writes these rows as ready-to-use Python lists, instead of manually splitting/joining strings.

why.md

Why use it?

CSV files are lightweight, human-readable, and open directly in Excel/Sheets. The csv module correctly handles commas or newlines inside quoted fields and different delimiters — something a plain split(",") on a text file gets wrong.

Writing & Reading CSV Rows

WRITE — row(s) to text line(s): a Python row (list) is passed to writerow(), or a list of rows to writerows(); it becomes a comma-joined text line, written to the CSV file opened in "w" or "a" mode with newline=''.

READ — text line back to row: csv.reader() reads each text line from the CSV file (opened in "r" mode) and splits it into a list of strings.

Note: *every field read back is a string — numbers need int()/float() conversion after reading. newline='' stops blank rows being inserted on Windows.*

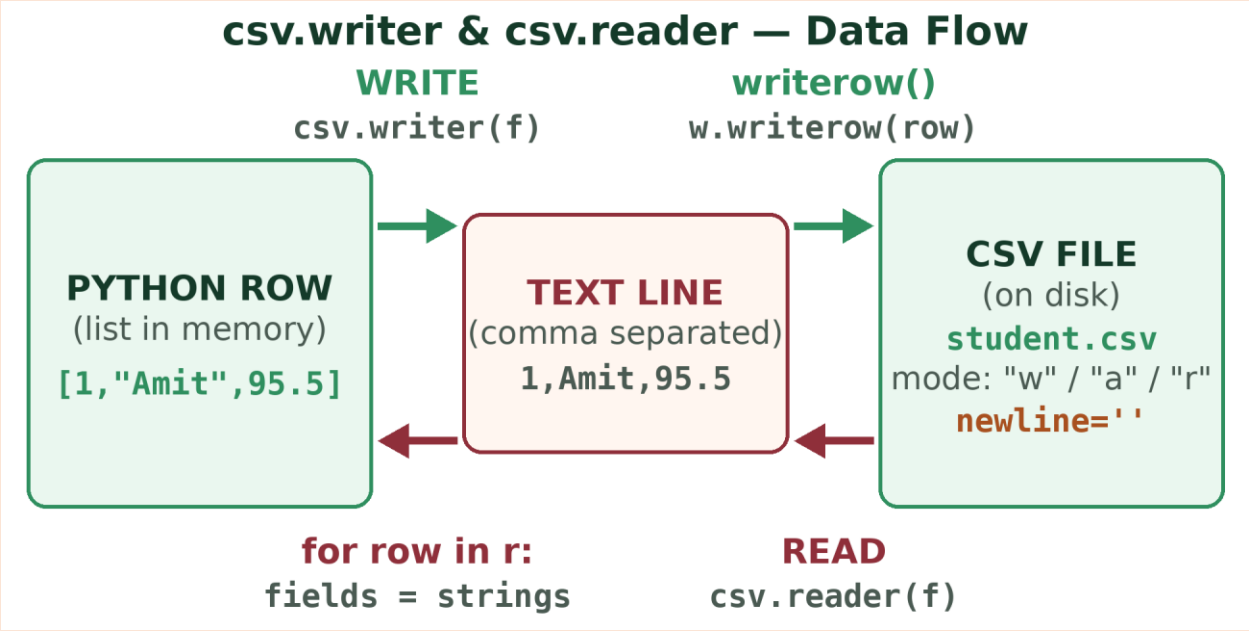


Fig: csv.writer joins a list into a comma-separated text line saved in the file; csv.reader splits each line back into a list (all fields as strings).

write_list.py WRITE

Write — header + rows

```
import csv
f = open("student.csv", "w", newline='')
w = csv.writer(f)
w.writerow(["Roll","Name","Marks"])  # heade
w.writerow([1, "Amit", 95.5])
f.close()
```

read_list.py READ

Read — loop over rows

```
f = open("student.csv", "r", newline='')
r = csv.reader(f)
for row in r:
    print(row)  # row is a list of strings
f.close()
```

1	2	3	4	5	6
import csv	open(newline='')	writer() / reader()	writerow()/writerows()	rewrite file (update/delete)	close()

Operation Summary

Operation	Method	Python example
WRITE	writer.writerow(row)	w.writerow([101,'Pen',50])
WRITE MANY	writer.writerows(rows)	w.writerows(data) # data = list of rows
APPEND	open in 'a' mode	f = open("student.csv","a",newline=''); w = csv.writer(f)
READ	for row in csv.reader(f)	for row in r: print(row)
UPDATE	read all → modify → rewrite	row[2] = newmarks, then write every row back in 'w'
DELETE	read all except target → rewrite	if row[0] != roll: w.writerow(row)

Update, Delete & writerows()

update_list.py **UPDATE**

Update — row [Roll, Name, Marks]

```
def update(r, nm):
   f = open("student.csv","r",newline='')
   rows = list(csv.reader(f))
   f.close()
   for row in rows:
      if row[0]==str(r):
         row[2]=nm
   f = open("student.csv","w",newline='')
   w = csv.writer(f)
   w.writerows(rows)
   f.close()
```

delete_list.py **DELETE**

Delete — by Roll No

```
r = input("Roll: ")
f=open("student.csv","r",newline='')
rows=list(csv.reader(f))
f.close()
f=open("student.csv","w",newline='')
w=csv.writer(f)
for row in rows:
   if row[0]!=r:
      w.writerow(row)
f.close()
```

writerows_demo.py **WRITE MANY**

writerows() — write header + all rows in one call

```
import csv
f = open("student.csv","w",newline='')
w = csv.writer(f)
data = [["Roll","Name","Marks"],
        [1,"Amit",95.5],
        [2,"Isha",88.0]]
w.writerows(data)  # one call instead of a loop
f.close()
```

★ Golden rule — import csv → open(newline='') → csv.writer()/csv.reader() → writerow()/writerows()/read loop → (update/delete: read-all, modify, rewrite) → close() ★

Board-Pattern Questions

CSV file handling · csv-module · CBSE 2023–2026

2023_board.py WRITE

Write header & record from user input

CSV file item.csv stores [Icode, Iname, Qty]. Write the header once, then append a record entered by the user.

```
import csv
f = open("item.csv","a",newline='')
w = csv.writer(f)
ic = int(input("Code: "))
nm = input("Name: ")
qt = int(input("Qty: "))
w.writerow([ic,nm,qt])
f.close()
```

2023_comptt.py READ

Display all records

Read every row of item.csv and display it, one row at a time.

```
f = open("item.csv","r",newline='')
r = csv.reader(f)
for row in r:
    print(row)
f.close()
```

2024_board.py UPDATE

Update Qty by Item code

Table product.csv holds [Item_code, Product_name, Qty]. Update quantity for a code entered by the user.

```
code=input("Item code: ")
nq=input("New qty: ")
f=open("product.csv","r",newline='')
rows=list(csv.reader(f))
f.close()
for row in rows:
    if row[0]==code: row[2]=nq
f=open("product.csv","w",newline='')
w=csv.writer(f)
w.writerows(rows)
f.close()
```

2024_comptt.py DELETE

Delete by key entered by user

Table member.csv holds [MemNo, Name, Books]. Delete the record whose MemNo is entered.

```
mno=input("MemNo: ")
f=open("member.csv","r",newline='')
rows=list(csv.reader(f))
f.close()
f=open("member.csv","w",newline='')
w=csv.writer(f)
for row in rows:
   if row[0]!=mno: w.writerow(row)
f.close()
```

2025_boardsample.py REAL PYQ · READ · 4 marks

Happiness.csv — CBSE Board Sample Paper 2024–25 (Q33)

A csv file "Happiness.csv" contains survey data. Each record: Country name, Population, Sample Size, Happy (persons who said they were happy) — e.g. ['Signiland', 5673000, 5000, 3426]. Write functions to: (i) read all data as a list and display records where population > 5000000, and (ii) count the number of records.

```
import csv

def display_high_population():
   f = open("Happiness.csv","r",newline='')
   data = list(csv.reader(f))
   f.close()
   for row in data:
      if int(row[1]) > 5000000:
         print(row)

def count_records():
   f = open("Happiness.csv","r",newline='')
   data = list(csv.reader(f))
   f.close()
   print("Total records:", len(data))
```

2026_boardsample.py REAL PYQ · WRITE · 4 marks

Sales.csv — CBSE Board Sample Paper 2025–26 (Q33)

Raj, manager of a medical store, keeps sales records in Sales.csv with columns Product_ID, Product_Name, Quantity_Sold, Price_Per_Unit. Write Accept() to accept a sales record from the user and add it to Sales.csv, and CalculateTotalSales() to calculate

and return the total sales.

```
import csv

def Accept():
    f = open("Sales.csv","a",newline='')
    w = csv.writer(f)
    pid = int(input("Product ID: "))
    pname = input("Product Name: ")
    qty = int(input("Quantity Sold: "))
    price = float(input("Price Per Unit: "))
    w.writerow([pid,pname,qty,price])
    f.close()

def CalculateTotalSales():
    f = open("Sales.csv","r",newline='')
    r = csv.reader(f)
    total = 0
    for row in r:
        total += int(row[2]) * float(row[3])
    f.close()
    return total
```

Conceptual Questions

CONCEPTUAL · Q1

Why should a CSV file be opened with newline='' when using the csv module?

Ans: In text mode, Python's universal-newline translation can add an extra carriage return on some platforms (notably Windows), which the csv module then sees as extra blank rows. Passing newline='' disables that translation so the csv module can handle line endings correctly itself.

CONCEPTUAL · Q2

Why is the csv module preferred over reading/writing a plain text file for tabular data?

Ans: The csv module automatically handles delimiters and quoting for fields that themselves contain commas or newlines, and returns each row as a ready-to-use list — a plain split(",") on text would break on such fields and needs manual parsing.

Board favourites — writerow()/writerows() for header + data, read loop with csv.reader(), update & delete via read-all → modify → rewrite, all on list rows. Q33 (2025 & 2026) reproduced from official CBSE Computer Science Board Sample Papers.

BINARY FILE HANDLING

Python • Class 12 Computer Science (083) • Revision Sheet

1 What is a Binary File? data stored as it is in memory (0s & 1s)

- Stores data in the **same format** used internally by the program — not human readable (opens as garbled text).
- Faster & more efficient to store **complex objects** — lists, dictionaries, tuples, class objects.
- Python uses the `pickle` module to convert objects → byte stream (**serialization**) & back (**deserialization**).

2 Text File vs Binary File

Text File	Binary File
Stores data as readable characters	Stores data as 0s & 1s (raw bytes)
Can be opened in Notepad	Appears as unreadable symbols in Notepad
Each line ends with `\n`	No concept of "lines" / newline translation
Modes: r, w, a, r+ ...	Modes: rb, wb, ab, rb+ ...

3 Binary File Opening Modes

Mode	Meaning	Description
`rb`	Read Binary	Opens binary file for reading (default position: start).
`wb`	Write Binary	Creates / overwrites file for writing.
`ab`	Append Binary	Opens for writing at end of the file.
`rb+`	Read+Write Binary	File must exist; allows both operations.
`wb+`	Write+Read Binary	Overwrites file; allows read after write.

4 The pickle Module

Function	Purpose
`pickle.dump(obj, f)`	Serializes **obj** and writes it to binary file **f**.
`obj = pickle.load(f)`	Reads & deserializes the next object from file **f**.

```
# must import first
import pickle
```

5 Write & Read a Binary File

```
# Writing records
import pickle
student = {"roll":1,"name":"Aman"}
with open("stu.dat","wb") as f:
    pickle.dump(student, f)

# Reading records (loop until EOF)
with open("stu.dat","rb") as f:
    try:
        while True:
            rec = pickle.load(f)
            print(rec)
    except EOFError:
        pass
```

Exam Tip: `pickle.load()` raises `EOFError` when no more data is left — always read inside a `try / except EOFError` block!

6 CRUD Operations on Binary Files

- **Search:** read records one by one; compare key field (e.g. roll no.).
- **Update:** read all records into a list → modify matching record → rewrite entire file (mode 'wb').
- **Delete:** read all records → skip the one to delete → write remaining records back.

7 Search Example

```
def search(rno):
    with open("stu.dat","rb") as f:
        try:
            while True:
                r = pickle.load(f)
                if r["roll"]==rno:
                    return r
        except EOFError:
            return None
```

Memory Tip → Pickle Dumps & Loads = dump() writes, load() reads

Why Use Binary Files?

- ✔ Stores complex Python objects directly — no manual conversion to strings needed.
- ✔ Faster read/write and smaller file size than equivalent text storage.
- ✔ Preserves data type (int stays int) — no re-parsing required on reading.

8 Viva Voce Questions

1. Which module handles binary files in Python? → **pickle**
2. Error raised at end of binary file while reading? → **EOFError**
3. Function to write an object to a binary file? → **pickle.dump()**
4. Can text editors correctly display binary file content? → **No, appears garbled**
5. Mode to open a binary file for reading? → **'rb'**

9 Quick Recall

import pickle | dump() | load() | rb / wb / ab | EOFError | serialize | deserialize | CRUD

Binary File Handling

Pickle & Unpickle

Write, Read, Update, Delete records — List & Dictionary objects using the pickle module

what.md

What is a binary file?

A file that stores data exactly as it is held in memory (0s & 1s) — not as readable text. Python's pickle module is used to convert (serialize) Python objects like lists, tuples and dictionaries into a byte stream and write them, and to reverse the process (deserialize) while reading.

why.md

Why use it?

Text files store everything as strings, so numbers/objects need repeated type conversion and searching is slow. Binary files store objects directly in their native form — faster access, smaller size, no data loss, and whole records (list/dict) are saved & fetched in one go.

How Pickling & Unpickling Work

PICKLING (write) — serialize: a Python object (list / dict / tuple) is passed to pickle.dump(); it becomes a serialized byte stream, written to the binary file opened in "wb" or "ab" mode.

UNPICKLING (read) — deserialize: pickle.load() reads bytes from the binary file (opened in "rb" mode) and reconstructs the exact Python object — same type restored, no manual conversion needed.

Note: *read stops when EOFError is raised (no more bytes left) — unlike text files, a binary file does not store how many records it holds.*

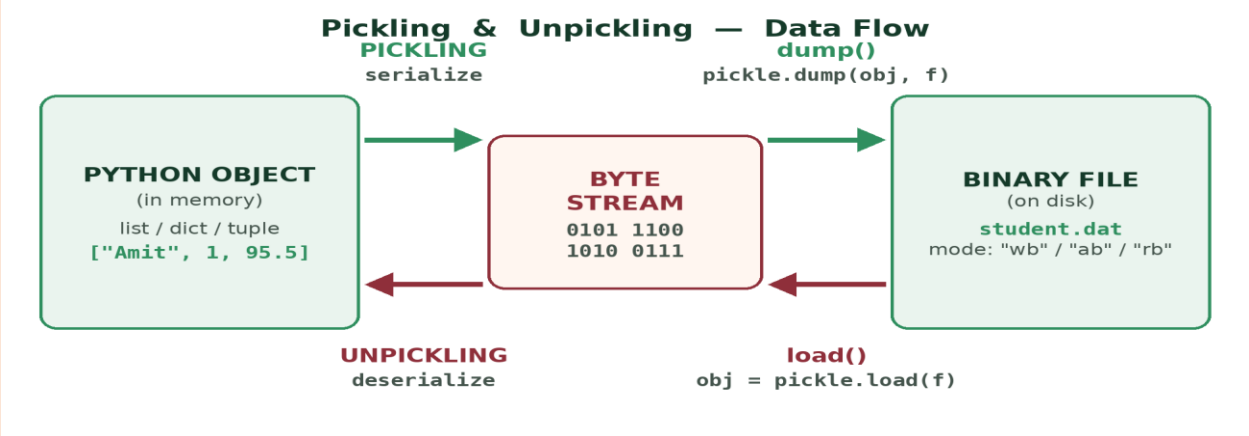


Fig: dump() serializes the object into a byte stream saved on disk; load() deserializes it back into the same Python object.

write_list.py **WRITE**

Write — list record (dump)

```
import pickle
stu = ["Amit", 1, 95.5]   # [Name, Roll, Marks]
f = open("student.dat", "wb")
pickle.dump(stu, f)
f.close()
```

read_list.py **READ**

Read — loop with EOFError (load)

```
f = open("student.dat", "rb")
while True:
   try:
      rec = pickle.load(f)
      print(rec)
   except EOFError:
      break
f.close()
```

1	2	3	4	5	6
import pickle	open(mode)	dump() / load()	process / edit	rewrite file (update/delete)	close()

Operation Summary

Operation	Method	Python example
WRITE	pickle.dump(obj, f)	pickle.dump(stu, f) # f opened in 'wb'
APPEND	pickle.dump(obj, f)	f = open("student.dat","ab"); pickle.dump(new,f)
READ	pickle.load(f)	rec = pickle.load(f) # inside try/except EOFError
UPDATE	read all → modify → rewrite	rec[2] = newmarks, then dump every record back in 'wb'
DELETE	read all except target → rewrite	if rec[1] != roll: pickle.dump(rec, fout)

Update, Delete & Dictionary Records

update_list.py UPDATE

Update — list [Roll, Name, Marks]

```
def update(r, nm):
    lst = []
    f = open("student.dat","rb")
    try:
        while True:
            lst.append(pickle.load(f))
    except EOFError:
        pass
    f.close()
    for rec in lst:
        if rec[0]==r:
            rec[2]=nm
    f = open("student.dat","wb")
    for rec in lst:
        pickle.dump(rec,f)
    f.close()
```

delete_list.py DELETE

Delete — by Roll No

```
r = int(input("Roll: "))
lst=[]
f=open("student.dat","rb")
try:
    while True:
        lst.append(pickle.load(f))
except EOFError:
    pass
f.close()
f=open("student.dat","wb")
for rec in lst:
    if rec[0]!=r:
        pickle.dump(rec,f)
f.close()
```

dict_ops.py **DICTIONARY**

Dictionary record — write / update

```
emp = {"Ecode":"E1","Name":"Raj","Sal":50000}
f=open("emp.dat","ab")
pickle.dump(emp,f); f.close()

# update Sal where Ecode matches
for rec in lst:
   if rec["Ecode"]=="E1":
      rec["Sal"]=55000
```

★ Golden rule — import pickle → open(mode) → dump()/load() → (update/delete: read-all, modify, rewrite) → close() ★

Board-Pattern Questions

Binary file handling · pickle-based · CBSE 2023–2026

WRITE / READ / UPDATE / DELETE for list & dictionary records, plus viva-style conceptual questions.

2023_board.py WRITE

Write records from user input

Binary file item.dat stores [Icode, Iname, Qty] as a list. Accept values from the user and write one record.

```
import pickle
f = open("item.dat","ab")
ic = int(input("Code: "))
nm = input("Name: ")
qt = int(input("Qty: "))
pickle.dump([ic,nm,qt], f)
f.close()
```

2023_comptt.py READ

Display all records

Read every record of item.dat and display, catching EOFError to stop the loop.

```
f = open("item.dat","rb")
while True:
    try:
        print(pickle.load(f))
    except EOFError:
        break
f.close()
```

2024_board.py UPDATE

Update Qty by Item code

Table product.dat holds [Item_code, Product_name, Qty]. Update quantity for a code entered by the user.

```
code=int(input("Item code: "))
nq=int(input("New qty: "))
lst=[]
```

```
f=open("product.dat","rb")
try:
   while True: lst.append(pickle.load(f))
except EOFError: pass
f.close()
for r in lst:
   if r[0]==code: r[2]=nq
f=open("product.dat","wb")
for r in lst: pickle.dump(r,f)
f.close()
```

2024_comptt.py DELETE

Delete by key entered by user

Table member.dat holds [MemNo, Name, Books]. Delete the record whose MemNo is entered.

```
mno=int(input("MemNo: "))
lst=[]
f=open("member.dat","rb")
try:
   while True: lst.append(pickle.load(f))
except EOFError: pass
f.close()
f=open("member.dat","wb")
for r in lst:
   if r[0]!=mno: pickle.dump(r,f)
f.close()
```

2025_pattern.py WRITE

Write dictionary records

Store book issue details in lib.dat as a dictionary {BookNo, Title, Fine} entered by the user.

```
bk = {}
bk["BookNo"]=input("Book No: ")
bk["Title"]=input("Title: ")
bk["Fine"]=float(input("Fine: "))
f=open("lib.dat","ab")
pickle.dump(bk,f)
f.close()
```

2026_sample.py **UPDATE**

Update fine in dictionary record

In lib.dat, update the Fine value for the BookNo entered; confirm how many records changed.

```
bno=input("Book No: ")
nf=float(input("New fine: "))
lst=[]; cnt=0
f=open("lib.dat","rb")
try:
   while True: lst.append(pickle.load(f))
except EOFError: pass
f.close()
for r in lst:
   if r["BookNo"]==bno:
      r["Fine"]=nf; cnt+=1
f=open("lib.dat","wb")
for r in lst: pickle.dump(r,f)
f.close()
print("Updated:",cnt)
```

Conceptual Questions

CONCEPTUAL · Q1

Why can't we use normal read()/write() text mode to store a list or dictionary directly?

Ans: A text file can only store strings. A list/dictionary written directly gets saved as its string representation, which cannot be read back as a usable Python object — pickle preserves the actual object type.

CONCEPTUAL · Q2

Why must pickle.load() be enclosed in a try/except EOFError block?

Ans: A binary file does not store how many records it holds, so load() has no way to signal "no more records" — it simply raises EOFError once the end of file is reached, which the except block catches to stop the loop.

Board favourites — write/append with pickle.dump(), read loop with EOFError, update & delete via read-all → modify → rewrite, list vs dictionary records.

Chapter Name: Binary File and CSV File

Question Type: MCQ

Q1	In binary mode, how do you move the file pointer to the beginning of the file? A) file.goto(0) B) file.move(0) C) file.seek(0) D) file.reset()	1
Q2	What happens if you try to open a binary file in 'rb' mode that does not exist? A) It creates a new file B) It raises a FileNotFoundError C) It returns None D) It opens an empty file	1
Q3	What is the correct way to load an object from a binary file using pickle? A) pickle.open(file) B) pickle.read(file) C) pickle.load(file) D) pickle.extract(file)	1
Q4	When we open a Binary file in append mode, by default new data can be added at the? (A) Beginning of the file (B) End of the file (C) At the middle of the file (D) All of the above	1
Q5	In python binary file is treated as a : (A)Sequence of bits (B)Sequence of Bytes (C)Sequence of numbers (D)Sequence of characters	1
Q6	The statement to prepare the buffer in memory to examine the contents of the binary file (‘data.dat’) (A) f=open(“data.dat” , “r”) (B) f=open(data.dat , “rb”). (C) f=open(data.dat, “r”) (D) f=open(“data.dat” , “rb”)	1

Q7	The method used to unpickling the data from a binary file (A) unpickle( ) (B) load( ) (C) deserialise( ) (D) read( )	1
Q8	What's the role of the pickle module in the context of binary files? A) Compress binary data B) Encrypt data for binary files C) Serialize/deserialize Python objects to/from binary form D) Manipulate file paths in binary file systems	1
Q9	The syntax of seek() is: file_object.seek(offset [,reference_point]) What is reference_point indicate? A. reference_point indicates the starting position of the file object B. reference_point indicates the ending position of the file object C. reference_point indicates the current position of the file object D. None of the above.	1
Q10	f.seek (10,0) will move 10 bytes forward from beginning of file. (T/F) A) True B) False	1

Answers:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
C	B	C	B	B	D	B	C	A	A

Question Type: Writing Functions

Q1	Consider a binary file emp.dat having records in the form of dictionary. E.g {eno:1, name:"Rahul", sal: 5000} write a python function to display the records of above file forthose employees who get salary between 25000 and 30000	3

Ans	import pickle
	def search():
	  f = open("emp.dat", "rb") # use normal double quotes
	  while True:
	  try:
	  d = pickle.load(f)
	  if d['sal'] >= 25000 and d['sal'] <= 30000:
	  print(d)
	  except EOFError:
	  break
	  f.close()	

Q2	A Binary file, CINEMA.DAT has the following structure: {MNO:[MNAME, MTYPE]} Where MNO – Movie Number MNAME – Movie Name MTYPE is Movie Type Write a user defined function, findType(mtype), that accepts mtype as parameter and displays all the records from the binary file CINEMA.DAT, that have the value of Movie Type as mtype.	3

Ans	import pickle
	def findType(mtype):
	  f = open("cinema.dat", "rb")
	  while True:
	  try:
	  record = pickle.load(f)
	  # assuming record is a dictionary
	  for y in record:
	  if record[y][1] == mtype:
	  print(record)
	  except EOFError:
	  break
	  f.close()	

Q3	A binary file “student.dat” has structure [rollno, name, marks]. Write a function searchRollNo(r) in python which accepts the student’s rollno as parameter and searches the record in the file “student.dat” and shows the details of student i.e. rollno, name and marks (if found) otherwise shows the message as ‘No record found’.	3

Ans	import pickle
	def searchRollNo(r):
	  flag = False
	  try:
	  with open("student.dat", "rb") as f:
	  while True:
	  try:
	  rec = pickle.load(f)
	  if rec["Rollno"] == r:
	  print("Roll No:", rec["Rollno"])
	  print("Name:", rec["Name"])
	  print("Marks:", rec["Marks"])
	  flag = True
	  except EOFError:
	  break
	  except FileNotFoundError:
	  print("The file 'student.dat' was not found.")
	  return
	  except Exception as e:
	  print("An error occurred:", e)
	  return
	  if not flag:
	  print("No record Found")	

Q4	A binary file “STUDENT.DAT” has structure (admission_number, Name, Percentage). Write a function countrec() in Python that would read contents of the file “STUDENT.DAT” and display the details and total number of those students whose percentage is above **75.**	3

Ans	import pickle
	def CountRec():
	  num = 0
	  with open("STUDENT.DAT", "rb") as f:
	  try:
	  while True:
	  rec = pickle.load(f)
	  if rec[2] > 75: # Marks at index 2
	  print(rec[0], rec[1], rec[2])
	  num += 1
	  except EOFError:
	  pass
	  return num	

Q5	A binary file named "EMP.dat" has some records of the structure [EmpNo, EName, Post, Salary]. Create a binary file "EMP.dat" that stores the records of employees and display the records of all those employees who are getting salaries between 25000 to 30000.	3

Ans	import pickle
	def add_and_save_employees():
	with open("EMP.dat", "wb") as f:
	while input("Add record? (y/n): ").lower() == 'y':
	emp = { "EmpNo": int(input("Emp No: ")), "EName":
	input("Name: "),
	"Post": input("Post: "), "Salary": float(input("Salary: ")) }
	pickle.dump(emp, f)
	print("Records saved to EMP.dat.\n")
	def show_employees():
	print("\nEmployees with Salary Between 25000 and 30000:")
	found = False
	try:
	with open("EMP.dat", "rb") as f:
	while True:
	try:
	emp = pickle.load(f)
	if 25000 <= emp["Salary"] <= 30000:
	print(emp)
	found = True
	except EOFError:
	break
	except FileNotFoundError:
	print("EMP.dat not found.")
	return	

Q6	A binary file "Book.dat" has structure [BookNo, Book_Name, Author, Price]. Write a function CountRec(Author) in Python which accepts the Author name as parameter and count and return number of books by the given Author are stored in the binary file "Book.dat".	3

Ans	import pickle
	def CountRec(Author):
	num = 0
	with open("Book.dat", "rb") as f:
	try:
	while True:
	rec = pickle.load(f)
	if rec[2] == Author: # Author is at index 2
	num += 1
	except EOFError:
	pass
	return num	

Q7	Consider a binary file "data.dat" which stores the record of" Hotel" in the form of list containing Room_no, Price, Room_type. Write the following functions: 1. addrec() to add a record in a file. 2. specific_disp(room_no) which takes room number as argument and display its details.	4

Ans	import pickle
	# 1. Function to add a record
	def addrec():
	try:
	room_no = int(input("Enter Room Number: "))
	price = float(input("Enter Price: "))
	room_type = input("Enter Room Type: ")
	rec = [room_no, price, room_type]
	with open("data.dat", "ab") as f: # append in binary mode
	pickle.dump(rec, f)
	print(" Record added successfully!\n")
	except Exception as e:
	print("Error adding record:", e)
	
	# 2. Function to display details of a specific room
	def specific_disp(room_no):
	found = False
	try:
	with open("data.dat", "rb") as f:
	while True:
	try:
	rec = pickle.load(f)
	if rec[0] == room_no:
	print("\nRoom Found:")
	print("Room Number:", rec[0])
	print("Price:", rec[1])
	print("Room Type:", rec[2])
	found = True
	break
	except EOFError:
	break	

Q8	Consider a file, SPORT.DAT, containing records of the following structure:[SportName, TeamName, No_Players] • Write a function, CountRecord(),that count the number of records in the file. • Write a function, copyData(), that reads contents from the file SPORT.DAT and copies the records with Sport name as “Cricket”to the file named CRICKET.DAT, The function should return the total number of records copied to the file CRICKET.DAT.	4

Ans	import pickle
	# Function to count number of records in sport.dat
	def countRecord():
	  count = 0
	  with open('sport.dat', 'rb') as f:
	  try:
	  while True:
	  record = pickle.load(f)
	  count += 1
	  except EOFError:
	  pass
	  return count
	
	# Function to copy cricket records to cricket.dat
	def copyData():
	  with open('sport.dat', 'rb') as f1, open('cricket.dat', 'wb') as f2:
	  try:
	  while True:
	  record = pickle.load(f1)
	  if record[0] == 'Cricket': # corrected quotes
	  pickle.dump(record, f2)
	  except EOFError:
	  pass	

Q9	A binary file named “EMP.dat” has some records of the structure [EmpNo, EName, Post, Salary] (a) Write a user-defined function named NewEmp() to input the details of a new employee from the user and store it in EMP.dat. (b) Write a user-defined function named SumSalary(Post) that will accept an argument the post of employees & read the contents of EMP.dat and calculate the SUM of salary of all employees of that Post	4

Ans	import pickle
	# (a) Function to add employee record
	def NewEmp():
	  with open("EMP.dat", "ab") as f: # append mode to keep all records
	  EmpNo = int(input("Enter employee number: "))
	  EName = input("Enter name: ")
	  Post = input("Enter post: ")
	  Salary = int(input("Enter salary: "))
	  rec = [EmpNo, EName, Post, Salary]
	  pickle.dump(rec, f)
	  print("Record added successfully!")	

	# (b) Function to calculate sum of salaries for a given Post
	def SumSalary(Post):
	c = 0
	with open("EMP.dat", "rb") as f:
	try:
	while True:
	g = pickle.load(f)
	if g[2] == Post: # Post is at index 2
	c += g[3] # Salary is at index 3
	except EOFError:
	pass
	print("Sum of salary for", Post, ":", c)	

Q10	A binary file "Items.dat" has structure as [Code,Description, Price]. i. Write a user defined function MakeFile( ) to input multiple items from the user and add to Items.dat ii. Write a function SearchRec(Code) in Python which will accept the code as parameter and search and display the details of the corresponding code on screen from Items.dat.	4

Ans	import pickle
	# (i) Function to create and add items
	def MakeFile():
	while True:
	code = input("Enter Item Code: ")
	desc = input("Enter description: ")
	price = float(input("Enter price: "))
	d = [code, desc, price]
	with open("Items.dat", "ab") as f: # open once per record, append mode
	pickle.dump(d, f)
	ch = input("Add more record? (y/n): ")
	if ch.lower() == 'n':
	break
	print("Records saved successfully!")
	# (ii) Function to search item by code	

	def SearchRec(code):
	found = False
	with open("Items.dat", "rb") as f:
	try:
	while True:
	g = pickle.load(f)
	if g[0] == code:
	print("Item Code:", g[0])
	print("Description:", g[1])
	print("Price:", g[2])
	found = True
	break
	except EOFError:
	pass
	if not found:
	print("No such record")	

Q11	A binary file "Bank.dat" has structure as [account_no, cust_name, balance]. i. Write a user-defined function addfile( ) and add a record to Bank.dat. ii. Create a user-defined function CountRec( ) to count and return the number of customers whose balance amount is more than 100000.	4

Ans	import pickle
	# (i) Add a record to bank.dat
	def addfile():
	with open("bank.dat", "ab") as f: # append mode so old records are not lost
	acc_no = int(input("Enter account number: "))
	cust_name = input("Enter name: ")
	bal = int(input("Enter balance: "))
	rec = [acc_no, cust_name, bal]
	pickle.dump(rec, f)
	print("Record added successfully!")
	# (ii) Count accounts with balance > 100000
	def CountRec():
	c = 0	

	with open("bank.dat", "rb") as f:
	try:
	while True:
	rec = pickle.load(f)
	if rec[2] > 100000: # balance is at index 2
	c += 1
	except EOFError:
	pass
	return c	

Q12	A binary file “student.dat” has structure [rollno,name, marks]. (i)Write a user defined function insertRec() to input data for a student and add to student.dat. (ii)Write a function searchRollNo(r) in Python which accepts the student’s rollno as parameter and searches the record in the file “student.dat” and shows the details of student i.e. rollno, name and marks (if found) otherwise shows the message as ‘No record found’.	4

Ans	import pickle
	# (i) Insert a student record
	def insertRec():
	with open("student.dat", "ab") as f:
	rollno = int(input("Enter Roll Number: "))
	name = input("Enter Name: ")
	marks = int(input("Enter Marks: "))
	rec = [rollno, name, marks]
	pickle.dump(rec, f)
	print("Record added successfully!")
	# (ii) Search a student by roll number
	def searchRollNo(r):
	flag = False
	with open("student.dat", "rb") as f:
	try:
	while True:
	rec = pickle.load(f)
	if rec[0] == r: # rollno is at index 0	

	print("Roll No:", rec[0]) print("Name:", rec[1]) print("Marks:", rec[2]) flag = True break except EOFError: pass if not flag: print("No record Found")	
Q13	A binary file TEST.DAT stores records of tests in the form of a list: [TestID, Subject, MaxMarks, ScoredMarks] Write Python functions to do the following using pickle: NewTest() → To accept details of a test (Test ID, Subject, Maximum Marks, Marks Scored) from the user and store them in TEST.DAT. UpdateMM(Sub) → To increase the Maximum Marks by 10 for all records of a given subject (Sub). DisplayAvgMarks(Sub) → To display the average of Marks Scored (ScoredMarks) for a given subject (Sub).	5

Ans	import pickle
	# (I) Function to add new test record
	def NewTest():
	Tid = int(input("Enter Test ID: "))
	Sub = input("Enter Subject Name of Test: ")
	MM = int(input("Enter Max. Marks of Test: "))
	SM = float(input("Enter Marks Scored in the Test: "))
	REC = [Tid, Sub, MM, SM]
	with open("TEST.DAT", "ab") as F:
	pickle.dump(REC, F)
	print("Record added successfully!")
	# (II) Function to update Max Marks for a given subject
	def UpdateMM(Sub):
	records = []
	updated = False
	try:	

```
            with open("TEST.DAT", "rb") as file:
                while True:
                    records.append(pickle.load(file))
        except EOFError:
            pass
        for rec in records:
            if rec[1] == Sub:
                rec[2] += 10  # Increase Max Marks by 10
                updated = True
        with open("TEST.DAT", "wb") as file:
            for rec in records:
                pickle.dump(rec, file)
        if updated:
            print("Max. Marks updated for subject:", Sub)
        else:
            print("Subject not found!")
# (III) Function to display average marks for a given subject
    def DisplayAvgMarks(Sub):
        total = 0
        count = 0
        try:
            with open("TEST.DAT", "rb") as file:
                try:
                    while True:
                        rec = pickle.load(file)
                        if rec[1] == Sub:
                            total += rec[3]   # Marks scored
                            count += 1
                except EOFError:
                    pass
            if count > 0:
                print("Average marks of", Sub, "are:", total / count)
            else:
                print("No records found for subject:", Sub)
```

	except FileNotFoundError: print("No Test data found. Please add Test data first.")	
Q14	Amit is a manager working in a recruitment agency. He needs to manage the records of various candidates. For this, he wants the following information of each candidate to be stored: -[Candidate_ID ,Candidate_Name ,Designation, Experience] You as a Programmer of the company, have been assigned to do this job for Amit. (i)Write a function to input the data of a candidate and append it in a binary file "candidates.dat" (ii) Write a function to update the data of candidates whose experience is more than 12 years and change their designation to "Sr. Manager". (iii) Write a function to read the data from the binary file and display the data of all those candidates who are not "Sr. Manager".	5

Ans	import pickle
	# (I) Function to input candidate details
	def input_candidates():
	candidate_id = int(input("Enter Candidate ID: "))
	candidate_name = input("Enter Candidate Name: "))
	designation = input("Enter Designation: "))
	experience = float(input("Enter Experience (in years): "))
	candidate = [candidate_id, candidate_name, designation, experience]
	with open("candidates.bin", "ab") as file:
	pickle.dump(candidate, file)
	print("Candidate data appended successfully.")
	# (II) Function to update designation to 'Senior Manager' if experience > 10
	def update_senior_manager():
	updated_candidates = []
	try:
	with open("candidates.bin", "rb") as file:
	while True:
	try:
	candidate = pickle.load(file)
	if candidate[3] > 10: # experience > 10
	candidate[2] = "Senior Manager"	

	updated_candidates.append(candidate)
	except EOFError:
	break
	except FileNotFoundError:
	print("No candidate data found. Please add candidates first.")
	return
	with open("candidates.bin", "wb") as file:
	for candidate in updated_candidates:
	pickle.dump(candidate, file)
	print("Candidates updated to Senior Manager where applicable.")
	# (III) Function to display non-Senior Managers
	def display_non_senior_managers():
	try:
	with open("candidates.bin", "rb") as file:
	while True:
	try:
	candidate = pickle.load(file)
	if candidate[2] != "Senior Manager":
	print(candidate)
	except EOFError:
	break
	except FileNotFoundError:
	print("No candidate data found. Please add candidates first.")	

Q15	Raj is a supervisor at a software development company. He needs to manage the records of various employees. For this, he wants the following information of each employee to be stored:[Employee_ID ,Employee_Name ,Position ,Salary] You, as a programmer of the company, have been assigned to do this job for Raj. (I) Write a function to input the data of an employee and append it to a binary file “empdata.dat”. (II) Write a function to update the data of employees whose salary is greater than 50000 and change their position to "Team Lead". (III) Write a function to read the data from the binary file and display the data of all those employees who are not "Team Lead".	5

Ans	import pickle
	# (I) Function to add employee record
	def add_employee():
	employee_id = int(input("Enter Employee ID: "))
	employee_name = input("Enter Employee Name: ")
	position = input("Enter Position: ")
	salary = float(input("Enter Salary: "))
	new_employee = (employee_id, employee_name, position, salary)
	with open("empdata.dat", "ab") as file:
	pickle.dump(new_employee, file)
	print("Employee added successfully.")
	# (II) Function to update employees with salary > 50000 to "Team Lead"
	def update_employee():
	employees = []
	try:
	with open("empdata.dat", "rb") as file:
	while True:
	try:
	employees.append(pickle.load(file))
	except EOFError:
	break
	except FileNotFoundError:
	print("No employee data found.")
	return
	for i in range(len(employees)):
	if employees[i][3] > 50000: # Salary > 50,000
	employees[i] = (employees[i][0], employees[i][1], "Team Lead",
	employees[i][3])
	with open("empdata.dat", "wb") as file:
	for employee in employees:
	pickle.dump(employee, file)
	print("Employees updated successfully.")
	# (III) Function to display all non-Team Lead employees
	def display_non_team_leads():	

	print("\nEmployees who are not Team Leads:")
	try:
	with open("empdata.dat", "rb") as file:
	while True:
	try:
	employee = pickle.load(file)
	if employee[2] != "Team Lead":
	print(employee)
	except EOFError:
	break
	except FileNotFoundError:
	print("No employee data found.")	

Q16	(i) Explain the difference between the 'a' and 'x' file opening modes in Python. (ii) Consider a file Company.dat containing records of the following structure: [CompanyName, Type, No_members] Write a function copyData() that reads contents form the file Company.dat and copies the records with Type "EduTech" to the file name EduTechs.dat. The function should return the total number of records copied to the file EduTechs.dat.	2+3

Ans	(i) 'a' (append) mode: Opens the file for appending. If the file exists, it appends new data at the end. If the file doesn't exist, it creates a new file.
	'x' (exclusive creation) mode: Opens the file for exclusive creation. If the file already exists, it raises a FileExistsError. If the file doesn't exist, it creates a new file.
	(ii)import pickle
	def copyData():
	fr = open("Company.dat", "rb")
	fw = open("EduTechs.dat", "wb")
	c = 0
	try:
	while True:	

	r = pickle.load(fr) # Read record
	if r[1] == "EduTech": # Check company name
	pickle.dump(r, fw) # Write to new file
	c += 1
	except EOFError:
	pass # End of file reached
	fr.close()
	fw.close()
	return c	

Q17	i) What is the difference between seek() and tell()? (ii) A binary file "STUD.DAT" has structure (admission_number, Name, total_mark) with student details. •Write a user defined function WriteFile() to create a file input data for a record and write to "stud.dat" file. The number of records to be entered until the user chooses 'Y' / 'Yes'. •Write a function ReadFile() in Python that would read the contents of the file "STUD.DAT" and display the details of those students whose total mark is less than or equal to 250 under the printed heading "REMEDIAL STUDENT LIST ". Also display and the number of students scoring above 250 marks as Bright Student.	2+3
Ans	(i) seek(): this method is used to position the file object at a particular position in a file. The syntax of seek() is: file_object.seek(offset [, reference_point]) * offset is the number of bytes by which the file object is to be moved. * reference_point indicates the starting position of the file object. That is, with reference to which position, the offset has to be counted. It can have any of the following values: 0- beginning of the file 1-current position of the file 2- end of fil tell(): This function returns an integer that specifies the current position of the file object in the file. That is the byte position from the beginning of the file till the current position of the file object. The syntax of using tell() is: file_object.tell() (ii)	

```
import pickle
def WriteFile():
    fobj=open("stud.dat","wb")
    ch='y'
    while ch.upper()=='Y':
        admission_number=int(input("Enter Admission Number : "))
        name=input("Enter Student Name :")
        total_mark = int(input("Enter total mark : "))
        rec=[admission_number, name, total_mark]
        pickle.dump(rec,fobj)
        print("\n Have you any more record to enter: ", end='')
        ch=input()
fobj.close()
def ReadFile():
    fobj=open("stud.dat","rb")
    bright = 0
    print("***************** REMEDIAL STUDENT LIST **************")
    print("Admission No\tName of the Student\tTotal_mark")
    try:
      while True:
          rec=pickle.load(fobj)
          if rec[2]<=250:
              print("\t",rec[0],"\t",rec[1], "\t\t", rec[2])
          else:
              bright = bright + 1
     except:
              fobj.close()
     return bright
WriteFile()
print("No. of Bright students: ",ReadFile())
```

Q18	(i) What are the similarities and difference between a+ and w file modes in Python. (ii) A binary file "product.dat" has structure [PNO, PNAME, BRAND_NAME, PRICE] to maintain and manipulate Product details. a. Write a user defined function CreateFile() to input data for a record and add to "product.dat" file. The number of records to be entered until the user chooses	2+3

	'Y' / 'Yes'. b. Write a function CountData(BRAND_NAME) in Python which accepts the brand name as parameter and count and return number of products by the given brand are stored in the binary file "product.dat".	

Ans	(i) Similarities:
	In a+ and w mode If the file doesn't exist, then a new file will be created.
	Differences:
	a+: Opens the file in append and read mode. And Previous contents will be there.
	w-Opens the file in write mode. But here all the contents will be overwritten.
	(ii)
	`import pickle`
	`def CreateFile():`
	`  fobj=open("product.dat","ab")`
	`  ch='y'`
	`  while ch.upper()=='Y':`
	`  pno=int(input("Enter Product Number : "))`
	`  pname=input("Enter Product Name :")`
	`  brand_name = input('Enter Product Brand Name: ')`
	`  Price = int(input("Enter Price of the Product : "))`
	`  rec=[pno,pname,brand_name,Price]`
	`  pickle.dump(rec,fobj)`
	`  print("\n Have you any more record to enter: ", end='')`
	`  ch=input()`
	`  fobj.close()`
	`def CountData(brand_name):`
	`  fobj=open("product.dat","rb")`
	`  num = 0`
	`  try:`
	`  while True:`
	`  rec=pickle.load(fobj)`
	`  if brand_name==rec[2]:`
	`  num = num + 1`
	`  except:`
	`  fobj.close()`
	`  return num`
	`CreateFile()`
	`print(CountData("LG"))`	

CSV FILE HANDLING		
Q1	Which of the following is the default character for the newline parameter for a csv file object opened in write mode in Python IDLE ? (A) \n (B) \t (C) , (D) ;	1
Q2	Which of the following function is used with the csv modules in Python to read the content of a csv file into an object ? (A) readrow() (B) readrows() (C) reader() (D) load()	1
Q3	Assertion (A): The csv.reader object in Python returns each row as a list of	1

	strings. Reason (R): The csv module automatically converts all numeric values in the CSV to integers. Options: A) Both A and R are true, and R is the correct explanation of A b) Both A and R are true, but R is not the correct explanation of A C) A is true, but R is false D) A is false, but R is true	
Q4	Assertion (A): The newline='' parameter should be used when opening a CSV file in Python. Reason (R): It prevents insertion of extra blank lines while writing to a CSV file on windows. Options: A) Both A and R are true, and R is the correct explanation of A b) Both A and R are true, but R is not the correct explanation of A C) A is true, but R is false D) A is false, but R is true	1
Q5	What will be the output of the following code? import csv with open("sample.csv", "w", newline="") as f: writer = csv.writer(f) writer.writerow("Hello") a) The file will contain: Hello b) The file will contain: H,e,l,l,o c) Error: writerow takes a list d) Nothing is written to the file	1
Q6	Assertion (A): While writing data into a CSV file using csv.writer(), each row must be passed as a list or tuple. Reason (R): The writerow() method of csv.writer class takes a single string as input and writes it directly to the CSV file. Options: A) Both A and R are true, and R is the correct explanation of A b) Both A and R are true, but R is not the correct explanation of A C) A is true, but R is false D) A is false, but R is true	1

Q7	Assertion (A): The csv.reader object in Python returns each row of a CSV file as a list. Reason (R): Each row read by csv.reader can be accessed using a for loop. Options: A) Both A and R are true, and R is the correct explanation of A b) Both A and R are true, but R is not the correct explanation of A C) A is true, but R is false D) A is false, but R is true	1
Q8	Assertion (A): csv.writer.writerow() can write a list of strings to a CSV file. Reason (R): writerow() automatically adds a newline character after each row. Options: A) Both A and R are true, and R is the correct explanation of A b) Both A and R are true, but R is not the correct explanation of A C) A is true, but R is false D) A is false, but R is true	1
Q9	What happens if you open a CSV file in text mode without specifying newline='' on Windows? A) Lines will merge B) Extra blank lines may be added C) File won't be saved D) It automatically adds fieldnames	1
Q10	How can you change the delimiter from comma to semicolon when writing a CSV file? A) Use delimiter=';' in open() B) Use csv.writer(f, sep=';') C) Use csv.writer(f, delimiter=';') D) You cannot change the delimiter	1

Answers

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
A	A	C	A	B	C	A	B	B	C

Q11	Write a program to count the number of students in students.csv (excluding header).	3

Ans	import csv
	filename = 'students.csv'
	count = 0
	with open(filename, mode='r', newline='') as file:
	  reader = csv.reader(file)
	  next(reader)
	  for row in reader:
	  count += 1
	print("Total number of students: “,count)
		3

Q12	Write a program to read students.csv and display students older than 17. The structure of records are [sname,class,age]	3

Ans	import csv
	with open('students.csv', 'r', newline='') as file:
	  reader = csv.reader(file)
	  for row in reader:
	  if int(row[2]) > 17:
	  print(row)	

Q13	Write a user-defined function High_Price() in Python that reads data from a CSV file devices.csv, which contains device details in the format: [id, name, price]. The function should display the names of all devices with a price greater than 5000.	3

Ans	import csv
	def High_Price():
	  try:
	  with open("devices.csv", "r") as file:
	  reader = csv.reader(file)
	  print("Devices with price > 5000:")
	  for row in reader:
	  if row: # skip empty rows
	  if int(row[2]) > 5000:
	  print("-", row[1])
	  except FileNotFoundError:
	  print("The file 'devices.csv' was not found.")	

Q14	Write a user-defined function List_Unique_Prizes() that reads data from a CSV file named sports.csv, which contains records in the format [sport_id,	3

	competition, prize_won]. The function should display a list of unique prize types (case-insensitive), sorted alphabetically, that appear in the file.	

Ans	import csv
	def List_Unique_Prizes():
	  unique_prizes = [ ]
	  try:
	  with open("sports.csv", "r") as file:
	  reader = csv.reader(file)
	  for row in reader:
	  if row and len(row) == 3:
	  prize = row[2].strip().lower()
	  unique_prizes.append(prize)
	  if unique_prizes:
	  print("Unique Prizes:")
	  for prize in sorted(unique_prizes):
	  print("-", prize.capitalize())
	  else:
	  print("No prizes found in the file.")
	  except FileNotFoundError:
	  print("The file 'sports.csv' does not exist.")	

Q15	A librarian is managing book inventory using a CSV file named `Inventory.csv`. The file structure is: `[BookID, Title, Author, Available]` where `BookID` is an integer, `Title` and `Author` are strings, and `Available` is an integer representing the number of copies available. The librarian needs to write the following functions: - add_book(): This function accepts new book details from the user and adds them to `Inventory.csv`. The file should be created with column headers if it doesn't exist. - check_availability(book_id): This function takes a `book_id` as input and returns the number of copies available for that book. If the book is not found, it should return -1.	4

Ans	import csv
	import os
	def add_book():
	file_exists = os.path.isfile('Inventory.csv')
	with open('Inventory.csv', 'a', newline='') as file:
	writer = csv.writer(file)
	if not file_exists:
	writer.writerow(['BookID', 'Title', 'Author', 'Available'])
	book_id = input("Enter BookID: ")
	title = input("Enter Title: ")
	author = input("Enter Author: ")
	available = input("Enter number of copies available: ")
	writer.writerow([book_id, title, author, available])
	print("Book added successfully!")
	def check_availability(book_id):
	try:
	with open('Inventory.csv', 'r') as file:
	reader = csv.DictReader(file)
	for row in reader:
	if row['BookID'] == str(book_id):
	return int(row['Available'])
	return -1 # Book not found
	except FileNotFoundError:
	print("Inventory file not found.")
	return -1	

Q16	A CSV file "HealthData.csv" contains the data of a health survey. Each record of the file contains the following data: • Name of a country • Life Expectancy (average number of years a person is expected to live) • GDP per capita (Gross Domestic Product per person) • Percentage of population with access to healthcare For example, a sample record of the file may be: ['Wonderland', 82.5, 40000, 95].	4

	Write the following Python functions to perform the specified operations on this file: (I) Read all the data from the file in the form of a list and display all those records for which the life expectancy is greater than 75. (II) Count the number of records in the file.	

Ans	(I)
	import csv
	def read_health_data(filename):
	records = []
	with open(filename, mode='r') as file:
	reader = csv.reader(file)
	next(reader) # Skip the header row if present
	for row in reader:
	country = row[0]
	life_expectancy = float(row[1])
	gdp_per_capita = float(row[2])
	access_to_healthcare = float(row[3])
	if life_expectancy > 75:
	records.append([country, life_expectancy, gdp_per_capita, access_to_healthcare])
	return records
	(II)
	def count_records():
	records = read_health_data("HealthData.csv")
	return len(records)	

Q17	Raj is the manager of a medical store. To keep track of sales records, he has created a CSV file named Sales.csv, which stores the details of each sale. The columns of the CSV file are: Product_ID, Product_Name, Quantity_Sold and Price_Per_Unit. Help him to efficiently maintain the data by creating the following user-defined functions: (i) Accept() – to accept a sales record from the user and add it to the file Sales.csv.	4

	(ii) CalculateTotalSales() – to calculate and return the total sales based on the Quantity_Sold and Price_Per_Unit.	

Ans	(i)
	import csv
	def Accept():
	product_id = input("Enter Product ID: ")
	product_name = input("Enter Product Name: ")
	quantity_sold = int(input("Enter Quantity Sold: "))
	price_per_unit = float(input("Enter Price Per Unit: "))
	with open('Sales.csv', 'a', newline='') as file:
	writer = csv.writer(file)
	writer.writerow([product_id, product_name, quantity_sold, price_per_unit])
	print("Sales record added successfully.")
	(ii)
	def CalculateTotalSales():
	total_sales = 0.0
	with open('Sales.csv', 'r') as file:
	reader = csv.reader(file)
	for row in reader:
	total_sales += int(row[2]) * float(row[3])
	print("Total Sales is:", total_sales)	

Q18	Write a program in Python that defines and calls the following user defined functions: (i) Add_Teacher( ) : It accepts the values from the user and inserts record of a teacher to a csv file 'Teacher.csv'. Each record consists of a list with field elements as T_id,Tname and desig to store teacher ID, teacher name and designation respectively. (ii) Search_Teacher( ) : To display the records of all the PGT (designation) teachers.	4
Ans	import csv def Add_Teacher(): with open("Teacher.csv", "a", newline='') as fout: T_id = int(input("Enter Teacher ID: ")) Tname = input("Enter Teacher Name: ")	

	desig = input("Enter Designation: ")
	rec = [T_id, Tname, desig]
	csvw = csv.writer(fout)
	csvw.writerow(rec)
	print("Teacher record added successfully.")
	import csv
	
	def Search_Teacher():
	with open("Teacher.csv", "r") as fin:
	csvr = csv.reader(fin)
	found = False
	for record in csvr:
	if len(record) >= 3 and record[2] == "PGT":
	print(record)
	found = True
	if not found:
	print("No teacher with designation 'PGT' found.")
	Add_Teacher( )
	Search_Teacher( )	

Q19	A csv file "Happiness.csv" contains the data of a survey. Each record of the file contains the following data: •Name of a country •Population of the country •Sample Size (Number of persons who participated in the survey in that country) •Happy (Number of persons who accepted that they were Happy) For example, a sample record of the file may be: ['Signiland', 5673000, 5000, 3426] Write the following Python functions to perform the specified operations on this file: (I) Read all the data from the file in the form of a list and display all those records for which the population is more than 5000000. (II) Count the number of records in the file.	5

Ans	(i)
	import csv
	def show():
	  with open("happiness.csv", 'r', newline='') as f:
	  records = csv.reader(f)
	  next(records, None) # Skip the header row
	  for row in records:
	  try:
	  if int(row[1]) > 5000000:
	  print(row)
	  except (ValueError, IndexError):
	  # Skip rows with invalid data or missing columns
	  Continue
	(ii) import csv
	def Count_records():
	  count = 0
	  with open("happiness.csv", 'r', newline='') as f:
	  records = csv.reader(f)
	  next(records, None) # Skip the header row
	  for row in records:
	  count += 1
	  print(count)	

Q20	(i) Differentiate between Binary file and csv file. (ii)Vedansh is a Python programmer working in a school. For the Annual Sports Event, he has created a csv file named Result.csv, to store the results of students in different sports events. The structure of Result.csv is :[St_Id, St_Name, Game_Name, Result] Where St_Id is Student ID (integer) ST_name is Student Name (string) Game_Name is name of game in which student is participating(string) Result is result of the game whose value can be either 'Won', 'Lost' or 'Tie' For efficiently maintaining data of the event, Vedansh wants to write the following user defined functions:	2+3

	Accept( ) – to accept a record from the user and add it to the file Result.csv. The column headings should also be added on top of the csv file. wonCount( ) – to count the number of students who have won any event. As a Python expert, help him complete the task.	

Ans	(i) A binary file stores data in a non-readable binary format and requires specific programs or logic to read or write its contents. It is typically used for storing complex data like images, videos, or serialized objects. On the other hand, a CSV (Comma-Separated Values) file stores data in a plain text format where values are separated by commas. It can be easily read using text editors or spreadsheet programs like Microsoft Excel.
	(ii)
	import csv
	import os
	def Accept():
	  sid = int(input("Enter Student ID: "))
	  sname = input("Enter Student Name: ")
	  game = input("Enter name of game: ")
	  res = input("Enter Result (WON/LOST): ")
	  headings = ["Student ID", "Student Name", "Game Name", "Result"]
	  data = [sid, sname, game, res]
	  with open('Result.csv', 'a', newline='') as f:
	  csvwriter = csv.writer(f)
	  csvwriter.writerow(headings) # Write header only once
	  csvwriter.writerow(data)
	def wonCount():
	  with open('Result.csv', 'r') as f:
	  csvreader = csv.reader(f)
	  header = next(csvreader) # Read and skip header
	  print(header) # Optionally print header
	  for row in csvreader:
	  if len(row) > 3 and row[3]. strip().upper() == "WON":
	  print(row)	

Q21	(i) Write one difference between CSV and text files. (ii) Write a program in Python that defines and calls the following user	2+3

	defined functions: COURIER ADD (): It takes the values from the user and adds the details to a csv file 'courier.csv'. Each record consists of a list with field elements as cid, s name, Source, destination to store Courier ID, Sender name, Source and destination address respectively. COURIER_SEARCH () : Takes the destination as the input and displays all the courier records going to that destination.	

Ans	(i) A CSV (Comma-Separated Values) file stores data in a tabular (row/column) format, where each row is a record and columns are separated by commas. Easier to import into spreadsheets or databases.
	A text file stores data as plain text and does not follow any specific structure or format. It may contain paragraphs, code, lists, etc. Meant for general textual information.
	(ii) import csv
	# Function to add courier records to 'courier.csv'
	def COURIER_ADD():
	with open("courier.csv", "a", newline="") as f:
	writer = csv.writer(f)
	while input("Add a courier record? (y/n): ").lower() == 'y':
	cid = input("Enter Courier ID: ")
	s_name = input("Enter Sender Name: ")
	source = input("Enter Source Address: ")
	destination = input("Enter Destination Address: ")
	writer.writerow([cid, s_name, source, destination])
	print("Records added successfully!\n")
	# Function to search couriers by destination
	def COURIER_SEARCH():
	dest = input("Enter destination to search: ")
	found = False
	try:
	with open("courier.csv", "r") as f:
	reader = csv.reader(f)
	print(f"\nCouriers going to '{dest}':")	

	for row in reader:
	if len(row) == 4 and row[3].strip().lower() == dest.strip().lower():
	print(row)
	found = True
	except FileNotFoundError:
	print(" 'courier.csv' not found.")	

Q22	(i) What is the purpose of the csv module in Python? Mention any two characteristics of a CSV file. (ii) Write a Program in Python that defines and calls the following user defined functions: (i) ADD( ) – To accept and add data of an employee to a CSV file 'record.csv'. Each record consists of a list with field elements as empid, name and salary to store employee id, employee name and employee salary respectively. (ii) COUNTR( ) – To count the number of records present in the CSV file named 'record.csv'.	2+3
Ans	(i) The csv module in Python provides functionality to read from and write to CSV files easily. It simplifies working with comma-separated data using functions like csv.reader() and csv.writer(). And 1. CSV files store data in a **row-column format** using commas to separate values. 2. CSV files are **plain text**, making them easy to read and edit with any text editor or spreadsheet software. (ii) import csv # Function to add employee record def ADD(): with open('record.csv', 'a', newline='') as f: wr = csv.writer(f) eid = int(input("Enter Employee ID: ")) name = input("Enter name of employee: ") sal = float(input("Enter salary: ")) data = [eid, name, sal] wr.writerow(data) print(" Record added successfully.\n") # Function to count and print the number of records	

	def COUNTR():
	  try:
	  with open('record.csv', 'r') as f:
	  data = csv.reader(f)
	  count = sum(1 for _ in data)
	  print(f"Total number of records: {count}")
	  except FileNotFoundError:
	  print(" 'record.csv' not found.")	

Q23	(i) Differentiate between a CSV file and a database table. (ii) Mr. Mahesh is a Python Programmer working in a school. He has to maintain the records of the sports students. He has created a csv file named sports.csv, to store the details. The structure of sports.csv is : [sport_id, competition, prize_won] where sport_id, is Sport id (integer) competition is competition name (string) prize_won is ("Gold", "Silver", "Bronze") Mr. Mahesh wants to write the following user-defined functions: Add_detail( ): to accept the detail of a student and add to a csv file, "sports.csv". Count_Medal( ): to display the name of competitions in which students have won "Gold" medal. Help him in writing the code of both the functions.	2+3

Ans	(i) A CSV file is a simple text file that stores data in rows and columns, separated by commas. It has no schema, constraints, or data types, and is mainly used for lightweight data storage or transfer.
	A database table, on the other hand, is part of a structured system that supports data types, relationships, constraints, and efficient querying. It's designed for handling larger, more complex, and secure datasets with multiple users.
	(ii)
	import csv
	# Function to add a new student sport record
	def Add_detail():
	  sport_id = int(input("Enter Sport ID: "))
	  competition = input("Enter Competition Name: ")
	  prize_won = input("Enter Prize Won (Gold/Silver/Bronze): ").capitalize()
	  with open('sports.csv', 'a', newline='') as file:	

	writer = csv.writer(file)
	writer.writerow([sport_id, competition, prize_won])
	print("Record added successfully!")
	def Count_Medal():
	competitions_with_gold = set()
	try:
	with open('sports.csv', 'r') as file:
	reader = csv.reader(file)
	for row in reader:
	if len(row) == 3 and row[2].strip().lower() == "gold":
	competitions_with_gold.add(row[1])
	if competitions_with_gold:
	print("Competitions where Gold medals were won:")
	for comp in competitions_with_gold:
	print("-", comp)
	else:
	print("No Gold medals recorded.")
	except FileNotFoundError:
	print("The file 'sports.csv' does not exist.")	

Q24	(i) Can you directly open and read a CSV file using a text editor? Will the structure remain readable? (ii)Write a program in Python that defines and calls the following user defined functions: (i) Add_Device( ) : The function accepts and adds records of the peripheral devices to a csv file 'peripheral.csv'. Each record consists of a list with field elements as P_id, P_name and Price to store peripheral device ID, device name, and price respectively. (ii) Count_Device( ) : To count and display number of peripheral devices, whose price is less than ₹ 1000.	2+3
Ans	(i) Yes, a CSV file is a plain text file and can be opened in any text editor. The structure remains readable, though formatting may not be as clean as in spreadsheet applications. (ii) import csv def Add_Device():	

```
    with open("peripheral.csv", "a", newline='') as fout:
        P_id = int(input("Enter Device Id: "))
        P_name = input("Enter Device name: ")
        Price = int(input("Enter Price: "))
        rec = [P_id, P_name, Price]
        csvw = csv.writer(fout)
        csvw.writerow(rec)
    print("Device record added successfully!")
def Count_Device():
    try:
        with open("peripheral.csv", "r") as fin:
            csvr = csv.reader(fin)
            ctr = 0
            for record in csvr:
                if record:  # Ensure row is not empty
                    if int(record[2]) < 1000:
                        ctr += 1
        print("Count of devices with Price < 1000 is:", ctr)
    except FileNotFoundError:
        print("File 'peripheral.csv' not found.")
# Function calls
Add_Device()
Count_Device()
```

STACK — DATA STRUCTURE

Python • Class 12 Computer Science (083) • Revision Sheet

1 What is a Stack? Linear data structure — LIFO order

- A stack is a linear data structure that follows **LIFO — Last In, First Out**: the last element added is the first one removed.
- Insertion & deletion happen only from **one end**, called the **TOP** of the stack.
- Real-life examples: stack of plates, bangles on a wrist, browser back button, Undo (Ctrl+Z) in editors.

2 Basic Terminology

Term	Meaning
PUSH	Operation to insert / add an element onto the top of the stack.
POP	Operation to remove / delete the topmost element of the stack.
PEEK / TOP	View the topmost element **without** removing it.
Overflow	Trying to PUSH into a completely full stack.
Underflow	Trying to POP from an empty stack.

3 Stack Using a Python List

Operation	List Equivalent
PUSH(x)	`stack.append(x)`
POP()	`stack.pop()` — removes & returns last element
PEEK()	`stack[-1]` — last element, unchanged
isEmpty()	`len(stack)==0`

4 Implementing Stack Operations

```
stack = []

def push(x):
    stack.append(x)

def pop():
    if not stack:
        print("Underflow!")
    else:
        return stack.pop()

def peek():
    if stack:
        return stack[-1]
```

5 Visualising PUSH & POP

Action	Stack State (top → right)
Start	`[ ]`
push(10)	`[10]`
push(20)	`[10, 20]`
push(30)	`[10, 20, 30]`
pop() → 30	`[10, 20]`
peek() → 20	`[10, 20]` (unchanged)

⭐ **Exam Tip:** In a Python list-based stack, `append()` & `pop()` both act on the **end of list** — this makes them O(1) fast, ideal for stack operations.

6 Applications of Stack

- ✔ Reversing a string / list of elements.
- ✔ Checking balanced parentheses in an expression.
- ✔ Undo/Redo functionality in text & graphic editors.
- ✔ Function call management — the **call stack** in recursion.
- ✔ Converting infix expressions to postfix / prefix.

7 Sample Program — Menu Driven Stack

```
stack = []
while True:
    ch = int(input("1-Push 2-Pop 3-Exit: "))
    if ch==1:
        stack.append(input("Value: "))
    elif ch==2:
        if stack: print("Popped:",stack.pop())
        else: print("Underflow")
    else:
        break
```

Memory Tip → LIFO = Last In First Out · Stack works like a pile of plates

8 Viva Voce Questions

1. Which order does a stack follow? → LIFO (Last In First Out)
2. Which list method implements PUSH? → append()
3. What is trying to POP an empty stack called? → Underflow
4. Which end of the list acts as TOP of stack? → the last / rightmost element
5. Name one real-life application of stack. → Undo feature / call stack

9 Quick Recall

LIFO · PUSH · POP · PEEK · TOP · append() · pop() · Overflow · Underflow

CBSE Class 12 CS (083) | Unit: Data Structures — Stack — 2026 Syllabus — Kendriya Vidyalaya Sangathan · Regional Office Agra

Chapter Name: Stack (List & Dictionary Based)

1. What is a Stack?

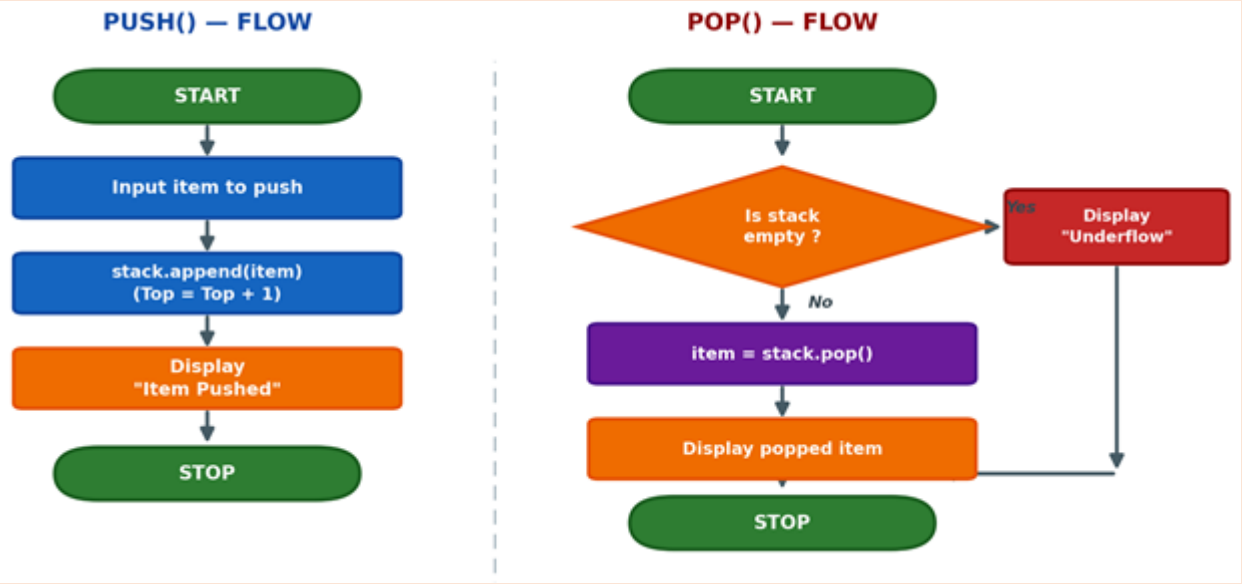
A stack is a linear data structure in which insertion and deletion of elements take place from only one end, called the TOP. It follows the LIFO (Last In, First Out) principle — the last element pushed onto the stack is the first one to be popped out. In Python, a stack is most commonly implemented using a list, where append() is used to push an element and pop() is used to remove one.

Real-life examples: Pile of plates, bangles on a wrist, browser back button, undo/redo in editors, pile of books.

New elements are always added to and removed from the same end (TOP). An empty stack condition is called Underflow, and is checked using the condition: if stack == [ ] or len(stack) == 0.

2. Flow of the Program

The flowcharts below trace the logic of the two core stack operations — pushing a new item onto the stack, and popping (removing) the top item while guarding against Underflow.



3. Key Terms

Term	Meaning
PUSH	Insert a new element at the TOP of the stack.
POP	Remove and return the element from the TOP of the stack.

Term	Meaning
PEEK / PEEP	View the TOP element only, without removing it.
LIFO	Last In, First Out — the order in which elements are processed.
List-based	Stack implemented using a Python list; uses append() and pop().

4. Function Implementations

4.1 push() — Insert at TOP

append() adds the item to the end of the list, which represents the TOP of the stack.

```
stack = []
def push(stack, item):
   stack.append(item)      # adds item to the END of list
                           # which represents the TOP of stack

push(stack, 10)
push(stack, 20)
push(stack, 30)
print(stack)               # [10, 20, 30]
```

4.2 pop() — Remove from TOP

pop() removes and returns the last item of the list (the TOP). Always check for an empty stack first to avoid Underflow.

```
def pop(stack):
   if stack == []:
      print("Stack Empty / Underflow")
      return None
   else:
      return stack.pop()   # removes & returns LAST item

print(pop(stack))          # 30 (last pushed, first out)
```

4.3 display() — Show TOP to BOTTOM

display() only reads the elements from TOP to BOTTOM; unlike pop(), it does NOT delete any element.

```
def display(stack):
   if stack == []:
      print("Stack Empty")
   else:
```

```
    for i in range(len(stack)-1, -1, -1):
        print(stack[i])   # does NOT delete elements
```

4.4 peek() / peep() — View TOP only

peek() returns the last element (current TOP) without removing it from the stack.

```
def peek(stack):
    if stack == []:
        print("Stack Empty")
        return None
    else:
        return stack[-1]     # last element = current TOP
                             # does NOT remove it
```

5. Operations Summary

Operation	List Equivalent	Effect
push(item)	stack.append(item)	Adds item at the end (TOP) — O(1) operation.
pop()	stack.pop()	Removes & returns the last item (TOP); check empty first.
peek() / peep()	stack[-1]	Reads the TOP element without deleting it.
isEmpty()	len(stack)==0 or stack==[]	Used before pop() to avoid Underflow error.

Quick recap: Stack = LIFO • push → append() at end • pop → removes from end • display → reads top-to-bottom without deleting.

6. CBSE Board-Pattern Questions (2023 – 2026 Style)

S.No	Question
1	***[Board 2024 Pattern]*** A list Nums has random integers. Write PushBig() to push every number having 5 or more digits into the stack BigNums, and PopBig() to pop and display all numbers, showing "Stack Empty" when done.

Ans	Nums = [213, 10025, 167, 254923, 14, 1297653, 31498, 386, 92765]
	BigNums = []
	def PushBig():
	  for N in Nums:
	  if len(str(N)) >= 5:
	  BigNums.append(N)
	
	def PopBig():
	  while BigNums:
	  print(BigNums.pop())
	  print("Stack Empty")

2	***[Board 2025 Pattern]*** A stack ClrStack stores colour records as tuples (Name, R, G, B). Write push_Clr() to push a new record, pop_Clr() to pop & return the top record (display "Underflow" if empty), and isEmpty() to return True/False.

Ans	def push_Clr(ClrStack, new_Clr):
	  ClrStack.append(new_Clr)
	
	def pop_Clr(ClrStack):
	  if len(ClrStack) == 0:
	  print("Underflow")
	  return None
	  return ClrStack.pop()
	
	def isEmpty(ClrStack):
	  return len(ClrStack) == 0

3	***[Sample Paper Pattern]*** A list VALUES has integers. Write push_even(N) to push all even numbers into stack EvenNumbers; pop_even() to pop & return the top (display "Empty" if none); Disp_even() to display all elements without deleting (display 'None' if empty).

Ans	VALUES = [10, 5, 8, 3, 12]
	EvenNumbers = []
	
	def push_even(N):
	  for i in N:
	  if i % 2 == 0:
	  EvenNumbers.append(i)
	
	def pop_even():
	  if EvenNumbers == []:
	  print("Empty"); return None
	  return EvenNumbers.pop()

S.No	Question

	def Disp_even():
	  if EvenNumbers == []:
	  print("None")
	  else:
	  for i in range(len(EvenNumbers)-1, -1, -1):
	  print(EvenNumbers[i])

4	***[Dictionary-Based Pattern]*** A dictionary d_city stores {city: state}. Write push_city(d_city) that pushes into stack CITY all cities whose state name has more than 4 characters.

Ans	d_city = {"Agra":"UP", "Patna":"Bihar", "Shimla":"HP", "Jaipur":"Rajasthan"}
	CITY = []
	
	def push_city(d_city):
	  for city, state in d_city.items():
	  if len(state) > 4:
	  CITY.append(city)
	# CITY -> ['Patna', 'Jaipur']

5	***[Dictionary-Based Pattern]*** A dictionary R stores {name: marks}. Write a function to push into a stack all names whose marks are above 80, then pop and display them (highest-pushed name shown first).

Ans	R = {"OM":76, "JAI":45, "BOB":89, "ALI":65, "ANU":90, "TOM":82}
	Toppers = []
	
	def pushTop(R):
	  for name, marks in R.items():
	  if marks > 80:
	  Toppers.append(name)
	
	def popTop():
	  while Toppers:
	  print(Toppers.pop())
	# Output: TOM, ANU, BOB (LIFO order)

6	***[Conceptual]*** Why is Stack called a LIFO structure? Which end is used for both insertion and deletion?
Ans	The element added last is removed first; both push and pop act only on the TOP end — never the bottom.
7	***[Conceptual]*** What is Underflow? How is it different from an empty list check?
Ans	Underflow is the error condition of trying to pop from an empty stack. It is always checked with if stack == [] (or len(stack) == 0) before calling pop(), to avoid an IndexError.
8	***[Application]*** A list Books stores book-name strings. Push into stack BooksStack only those names starting with a vowel; then use peep() to show the topmost book without removing it.

S.No	Question

Ans	Books = ["Atlas","Encyclopedia","Novel","Idioms","Biology"]
	BooksStack = []
	
	def push_book(BooksStack, b):
	  if b[0].upper() in "AEIOU":
	  BooksStack.append(b)
	
	for b in Books:
	  push_book(BooksStack, b)
	
	def peep(BooksStack):
	  if BooksStack == []:
	  print("None")
	  else:
	  print(BooksStack[-1])
	
	peep(BooksStack) # -> Idioms

7. Board Favourites — Quick Tips

- Filter-then-push from a list or dictionary using a condition.
- Pop-till-empty pattern, printed with an "Empty / Underflow" message.
- Peek / display without deleting any element from the stack.

CHAPTER NAME: STACK

Question Type: MCQ AND DESCRIPTIVE

1.	What is the process of inserting data into a stack called? A. Create B. Insert C. Push D. Evaluate	
2.	Which pointer is associated with a stack? A. First B. Front C. Rear D. Top	
3.	Assume a stack has size 10. If a user tries to push a 11^{th} element to a stack, which of the mentioned condition will arise? A. Underflow B. Overflow C. Crash D. Successful Insertion	
4.	Which of these is not an application of stack? A. Parenthesis Balancing program B. Evaluating Arithmetic Expressions C. Reversing Data D. Data Transfer between Process	
5.	The peek operation refers to accessing/inspecting the top element in the stack A. True B. False	
6.	Stack implementation can be performed using a list in Python. A. True B. False	
7.	The top operation does not modify the contents of a stack. A. True B. False	
8.	len() method used to find the size of stack. A. True B. False	
9.	**Assertion (A):** A stack is used to reverse a string. **Reason (R):** Stack follows Last-In, First-Out (LIFO) order. **Options:** A. Both A and R are true, and R is the correct explanation of A	

	B. Both A and R are true, but R is not the correct explanation of A C. A is true, but R is false D. A is false, but R is true	
10.	**Assertion (A):** Stack allows element addition at one end only. **Reason (R):** Stack operations are performed at both end. **Options:** A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, but R is not the correct explanation of A C. A is true, but R is false D. A is false, but R is true	
11.	You have a stack named **BooksStack** that contains records of books. Each book record is represented as a list containing **book_title**, **author_name**, and **publication_year**. Write the following user-defined functions in Python to perform the specified operations on the stack **BooksStack**: ● push_book(BooksStack, new_book): This function takes the stack BooksStack and a new book record new_book as arguments and pushes the new book record onto the stack. ● pop_book(BooksStack): This function pops the topmost book record from the stack and returns it. If the stack is already empty, the function should display "Underflow". ● peep(BookStack): This function displays the topmost element of the stack without deleting it. If the stack is empty, the function should display 'None'.	
12.	Tushar received a message(string) that has upper case and lower-case alphabet. He want to extract all the upper case letters separately .Help him to do his task by performing the following user defined function in Python: a) Push the upper case alphabets in the string into a STACK b) Pop and display the content of the stack. For example: **If the message is "All the Best for your Pre-board Examination"** **The output should be : E P B A**	

13.	Write a function in Python, Push(EventDetails) where , EventDetails is a dictionary containing the number of persons attending the events– {EventName : NumberOfPersons}. The function should push the names of those events in the stack named 'BigEvents' which have number of persons greater than 200. Also display the count of elements pushed o n to the stack. For example: If the dictionary contains the following data: EventDetails ={"Marriage":300, "Graduation Party":1500, "Birthday Party":80, "Get together" :150} The stack should contain : Marriage Graduation Party The output should be: The count of elements in the stack is 2	
14.	A list of numbers is used to populate the contents of a stack using a function push(stack, data) where stack is an empty list and data is the list of numbers. The function should push all the numbers that are even to the stack. Also write the function pop(stack) that removes and returns the top element of the stack on its each call.	
15.	A list contains following record of a student: [student_name, age, hostel] Write the following user defined functions to perform given operations on the stack named '**stud_details'**: (i) Push_element() - To Push an object containing name and age of students who live in hostel "Ganga" to the stack (ii) Pop_element() - To Pop the objects from the stack and display them. Also, display "Stack Empty" when there are no elements in the stack. For example: If the lists of customer details are: ["Barsat",17,"Ganga"] ["Ruben", 16,"Kaveri"] ["Rupesh",19,"Yamuna"]	

	The output should be: [“Barsat”,17,”Ganga”] Stack Empty	
16.	Write a function in Python PUSH_IN(L), where L is a list of numbers. From this list, push all numbers which are multiple of 3 into a stack which is implemented by using another list.	
17.	Write a function in Python, Push(KItem),where KItem is a dictionary containing the details of Kitchen items– {Item:price}. The function should push the names of those items in a stack which have price less than 100. Also display the average price of elements pushed into the stack. For example: If the dictionary contains the following data: **{"Spoons":116,"Knife":50,"Plates":180,"Glass":60}** The stack should contain **Glass** **Knife** The output should be: **The average price of an item is 55.0**	
18.	Given a Dictionary Stu_dict containing marks of students for three test-series in the form Stu_ID:(TS1, TS2, TS3) as key-value pairs. Write a Python program with the following user-defined functions to perform the specified operations on a stack named Stu_Stk (i) Push_elements(Stu_Stk, Stu_dict) : It allows pushing IDs of those students, from the dictionary Stu_dict into the stack Stu_Stk, who have scored more than or equal to 80 marks in the TS3 Test. (ii) Pop_elements(Stu_Stk): It removes all elements present inside the stack in LIFO order and prints them. Also, the function displays 'Stack Empty' when there are no elements in the stack. Call both functions to execute queries. For example: If the dictionary Stu_dict contains the following data: **Stu_dict ={5:(87,68,89), 10:(57,54,61), 12:(71,67,90), 14:(66,81,80), 18:(80,48,91)}** After executing Push_elements(), Stk_ID should contain **[5,12,14,18]**	

	After executing Pop_elements(), The output should be: **18** **14** **12** **5** **Stack Empty**	
19.	Consider a list named Nums which contains random integers. Write the following user defined functions in Python and perform the specified operations on a stack named BigNums. (1)PushBig(): It checks every number from the list Nums and pushes all such numbers which have 5 or more digits into the stack, BigNums. (ii) PopBig(): It pops the numbers from the stack, BigNums and displays them. The function should also display "Stack Empty" when there are no more numbers left in the stack. For example: If the list Nums contains the following data: Nums [213,10025,167,254923,14,1297653,31498,386,92765) Then on execution of PushBig(), the stack BigNums should store: [10025, 254923, 1297653, 31498, 92765] And on execution of PopBig (), the following output should be displayed: 92765 31498 1297653 254923 10025 Stack Empty	
20.	A dictionary, d_city contains the records in the following format: (state:city) Define the following functions with the given specifications: (i) push_city (d_city): It takes the dictionary as an argument and pushes all the cities in the stack CITY whose states are of more than 4 characters. (ii) pop_city(): This function pops the cities and displays "Stack empty" when there are no more cities in the stack.	

Answers		
	1. Ans. C 2. Ans. D 3. Ans. B 4. Ans. D 5. Ans. A 6. Ans. A 7. Ans. A 8. Ans. A 9. Ans. A 10. Ans. C	

11.	def push_book(BooksStack, new_book):
	  BooksStack.append(new_book)
	
	def pop_book(BooksStack):
	  if len(BooksStack) == 0:
	  print("Underflow")
	
	  else:
	  return BooksStack.pop()
	
	def peep(BooksStack):
	
	  if len(BooksStack) == 0:
	  print("None")
	
	  else:
	  print(BooksStack[-1])
	  return BooksStack[-1]	

12.	def push_uppercase(stack, message):
	  for char in message:
	  if char.isupper():
	  stack.append(char)
	
	def pop_and_display(stack):
	  while stack:
	  print(stack.pop(), end=' ')
	  print() # For newline after output	

13.	def Push(EventDetails):
	  BigEvents = [] # Stack to hold big event names
	  for event, persons in EventDetails.items():
	  if persons > 200:
	  BigEvents.append(event)	

	# Display the stack
	for event in BigEvents:
	  print(event)
	
	# Display the count
	print("The count of elements in the stack is", len(BigEvents))	

14.	def push(stack, data):
	  for num in data:
	  if num % 2 == 0:
	  stack.append(num)
	
	def pop(stack):
	  if len(stack) == 0:
	  return None # Or raise an error if preferred
	  return stack.pop()	

15.	stud_details = [] # Stack initialized
	
	def Push_element(student_list):
	  if student_list[2] == "Ganga":
	  stud_details.append(student_list)
	
	def Pop_element():
	  if not stud_details:
	  print("Stack Empty")
	  else:
	  while stud_details:
	  print(stud_details.pop())
	  print("Stack Empty")	

16.	def PUSH_IN(L):
	  stack = []
	  for num in L:
	  if num % 3 == 0:
	  stack.append(num)
	  return stack	

17.	def Push(KItem):
	  stack = []
	  total = 0
	  count = 0
	
	  for item, price in KItem.items():
	  if price < 100:
	  stack.append(item)
	  total += price
	  count += 1
	
	  # Display stack content
	  for i in stack:
	  print(i)	

	# Display average if count > 0: average = total / count print(f"The average price of an item is {average}")	

18.	def Push_elements(Stu_Stk, Stu_dict):
	for Stu_ID, marks in Stu_dict.items():
	if marks[2] >= 80: # TS3 is the third test (index 2)
	Stu_Stk.append(Stu_ID)
	
	# Function to pop and print all elements from the stack
	def Pop_elements(Stu_Stk):
	while Stu_Stk:
	print(Stu_Stk.pop())
	print("Stack Empty")
	
	# Main program
	Stu_dict = {
	5: (87, 68, 89),
	10: (57, 54, 61),
	12: (71, 67, 90),
	14: (66, 81, 80),
	18: (80, 48, 91)
	}
	
	Stu_Stk = []
	
	# Perform operations
	Push_elements(Stu_Stk, Stu_dict)
	Pop_elements(Stu_Stk)	

19.	# Sample list of random integers
	Nums = [213, 10025, 167, 254923, 14, 1297653, 31498, 386, 92765]
	
	# Stack to hold numbers with 5 or more digits
	BigNums = []
	
	# Function to push big numbers (5 or more digits) to the stack
	def PushBig():
	for num in Nums:
	if len(str(num)) >= 5:
	BigNums.append(num)
	
	# Function to pop and display all elements from the stack
	def PopBig():
	while BigNums:
	print(BigNums.pop())
	print("Stack Empty")	

	# Example usage: PushBig() # Push 5 or more digit numbers into the stack PopBig() # Pop and display the numbers	

20.	CITY = []
	
	# Function to push cities where the state has more than 4 characters
	def push_city(d_city):
	for state, city in d_city.items():
	if len(state) > 4:
	CITY.append(city)
	
	# Function to pop cities from the stack
	def pop_city():
	while CITY:
	print(CITY.pop())
	print("Stack empty")	

COMPUTER NETWORKS
Revision Sheet

1. What is a Computer Network?

A Computer Network is a collection of interconnected computers and devices that can communicate and share resources such as data, printers, software and internet.

- Resource Sharing
- Fast Communication
- Data Sharing
- Centralized Management
- Cost Effective

2. Types of Networks

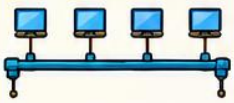
Type	Full Form	Coverage Area	Example
PAN	Personal Area Network	Few meters	Bluetooth between phone & earbuds
LAN	Local Area Network	Building/School	School computer lab
MAN	Metropolitan Area Network	City	City-wide cable network
WAN	Wide Area Network	Country/World	Internet

Memory Tip: PAN < LAN < MAN < WAN (Smallest to Largest)

3. Network Topologies

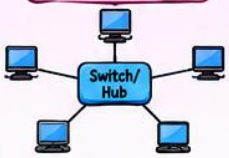
Topology → Physical or logical arrangement of computers in a network

Bus Topology



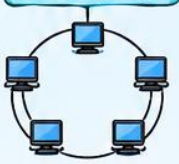
- Low cost
- Easy installation
- Main cable failure affects whole network
- Difficult troubleshooting

Star Topology



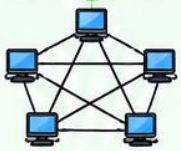
- Easy maintenance
- Failure of one node does not affect others
- ★ Most commonly used

Ring Topology



- Equal access to resources
- Failure of one node affects network

Mesh Topology



- Highly reliable
- Multiple paths available
- Expensive
- Complex installation

Tree Topology

- Scalable
- Easy management
- Backbone failure affects network

4. Transmission Media

Path through which data travels

A. Guided Media (Wired)

1. Twisted Pair Cable

- Cheap, Easy installation
- More noise

2. Coaxial Cable

- Better shielding
- Costlier than twisted pair

3. Optical Fiber Cable

- Very high speed
- Long distance, Secure
- Expensive installation

B. Unguided Media (Wireless)

Radio Waves (Wi-Fi, FM Radio)

Microwaves (Mobile communication, Satellite links)

Infrared (TV Remote, Short-range)

Satellite Communication (GPS, TV Broadcasting)

5. Networking Devices

Device		Function
	Modem	Converts digital signals to analog & vice versa. Used for Internet.
	Hub	Broadcasts data to all devices.
	Switch	Sends data only to intended device.
	Router	Connects different networks. Routes packets using IP addresses.
	Repeater	Regenerates weak signals. Increases network range.
	Bridge	Connects two similar LAN segments.
	Gateway	Connects networks using different protocols.
	Access Point (AP)	Provides wireless connectivity.

6. Common Protocols

Protocol	Full Form	Purpose
HTTP	Hyper Text Transfer Protocol	Web pages
HTTPS	Hyper Text Transfer Protocol Secure	Secure websites
FTP	File Transfer Protocol	File transfer
SMTP	Simple Mail Transfer Protocol	Sending emails
POP3	Post Office Protocol Version 3	Receiving emails
IMAP	Internet Message Access Protocol	Access emails on server
TCP	Transmission Control Protocol	Reliable transmission
IP	Internet Protocol	Addressing & routing
UDP	User Datagram Protocol	Fast transmission
DHCP	Dynamic Host Configuration Protocol	Automatic IP allocation
DNS	Domain Name System	Domain name to IP

7. Important Points

- Star Topology → Most popular
- Fiber Optic → Fastest medium
- Router → Connects networks
- Switch → Intelligent forwarding
- DNS → Name to IP conversion
- HTTPS → Secure web communication
- SMTP → Sending emails
- WAN → Largest network (Internet)

8. Viva Questions

1. Which topology is most commonly used? → Star
2. Which device connects different networks? → Router
3. Which medium uses light? → Optical Fiber
4. Protocol for secure websites? → HTTPS
5. Domain name to IP? → DNS
6. Largest network? → WAN
7. Full form of PAN? → Personal Area Network

One Page Quick Revision

Network Types: PAN, LAN, MAN, WAN

Topologies: Bus, Star, Ring, Mesh, Tree

Guided: Twisted Pair, Coaxial, Optical Fiber
Unguided: Radio Waves, Microwave, Infrared, Satellite

Devices: Hub, Switch, Router, Modem, Repeater, Bridge, Gateway, Access Point

Protocols: HTTP, HTTPS, FTP, SMTP, POP3, IMAP, TCP/IP, UDP, DNS, DHCP

Network connects the world...!

Chapter Name: Computer Network

Question Type: MCQ

Q1	Which of the following is not a unit for data transfer rate? A.MBPS B.KBPS C.SBPS D.GBPS	1
Q2	This was the first network. A.CSNET B.NSFNET C.ANSNET D.ARPANET	1
Q3	A_____is a data communication system within a building, plant, or campus, or between nearby buildings. A.MAN B.LAN C.WAN D. None of the above	1
Q4	_____ is a collection of many separate networks A. A MAN B. An internet C. A LAN D. None of the above	1
Q5	Computer Network is A. Collection of h/w components and computers B. Sharing of resources and Information C. Interconnected by communication channels D. All of the Above	1
Q6	What is a Firewall in a Computer Network? **A.** The physical boundary of Network **B.** An operating System of Computer Network **C.** A system designed to prevent unauthorized access **D.** A web browsing Software	1
Q7	DNS is the abbreviation of A. Dynamic Name System B. Dynamic Network System C. Domain Name System D. Domain Network Service	1
Q8	What is the meaning of Bandwidth in a Network? A. Transmission capacity of a communication channels B. Class of IP used in Network C. Connected Computers in the Network D. None of Above	1
Q9	ADSL is the abbreviation of A. Asymmetric Dual Subscriber Line B. Asymmetric Digital System Line C. Asymmetric Dual System Line D. Asymmetric Digital Subscriber Line	1

Q10	What is the use of Bridge in Network? A. to connect LANs B. to separate LANs C. to control Network Speed D. All of the above	1
Q11	Each IP packet must contain A. Only Source addresses B. Only Destination address C. Source and Destination address D. Source or Destination address	1
Q12	The last address of IP address represents A. Unicast address B. Network address C. Broadcast address D. None of above	1
Q13	The main computer in any network is called as A. Switch B. Hub C. Client D. Server	1
Q14	Which network topology requires a central controller or hub? A. Star B. Mesh C. Ring D. Bus	1
Q15	Which of the following networking devices forwards data packets between computer Networks? A. Router B. Gateway C. Switch D. Hub	1
Q16	A device that connects networks with different protocols – A. Switch B. Hub C. Gateway D. Proxy Server	1
Q17	Which of the following allows user to view a web page? A. Website B. Compiler C. Web browser D. Operating System	1
Q18	Protocols are set of rules to govern A. Communication B. Standard C. Metropolitan communication D. Bandwidth	1
Q19	The term HTTP stands for? A. Hyper terminal tracing program B. Hypertext tracing protocol C. Hypertext transfer protocol D. Hypertext transfer program	1
Q20	Sunil has purchased a new Smart TV and wants to cast a video from his mobile to his new Smart TV. Identify the type of network he is using and explain it. A. MAN B. LAN C. PAN D. None of the above	1

Answers

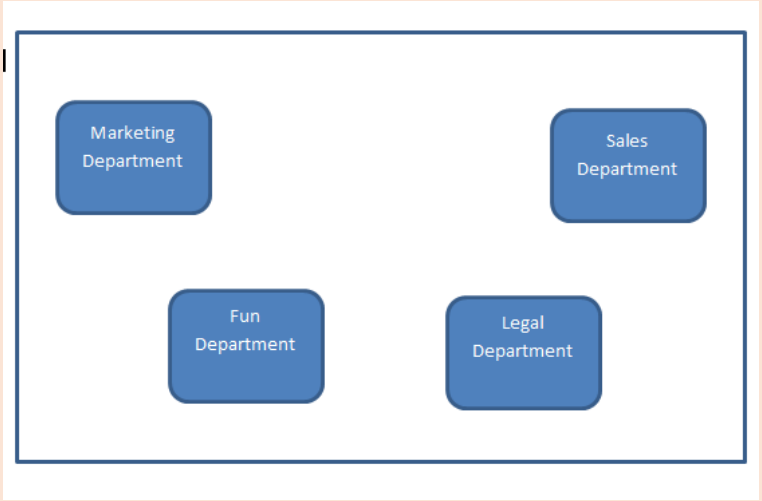
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
C	D	B	B	D	C	C	A	D	A
Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
C	C	D	A	A	C	C	A	B	C

Question Type: Very Short/Short Answer questions

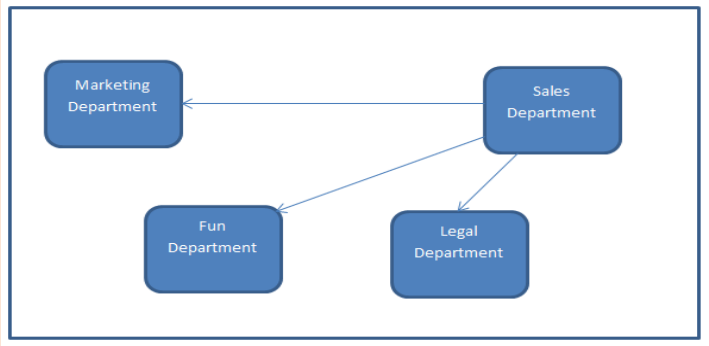
Q1	Give one example of each – Guided media and unguided media.	1
Ans	1- Guided – Twisted pair, Coaxial Cable, Optical Fiber (any one) Unguided – Radio Waves, Satellite, Micro Waves (any one)	
Q2	Name the transmission media best suitable for connecting to desert areas.	1
Ans	Microwave.	
Q3	Rearrange the following terms in increasing order of speedy medium of data transfer: Telephone line, Fiber Optics, Coaxial Cable, Twisted Paired Cable.	1
Ans	Telephone line, Twisted Pair Cable, Coaxial Cable, Fiber Optics.	
Q4	Name the transmission media suitable to establish PAN.	1
Ans	Bluetooth, infra-red.	
Q5	Which type of network (out of LAN, PAN and MAN) is formed, when you connect two mobiles using Bluetooth to transfer a picture file?	1
Ans	When two mobiles are connected using Bluetooth to transfer a picture file, a PAN (Personal Area Network) is created.	
Q6	Name the protocol that is used to transfer file from one computer to another.	1
Ans	File Transfer Protocol.	
Q7	Name the protocol that is used to send emails.	1
Ans	Simple Mail Transfer Protocol (SMTP).	
Q8	Name the protocol that is used to receive emails.	1
Ans	IMAP (Internet Message Access Protocol) and POP3 (Post Office Protocol ver 3)	
Q9	What is the difference between star topology and bus topology of a network?	2
Ans	In star topology, nodes are connected to server individually whereas in bus topology all nodes are connected to server along a single length of cable.	
Q10	Differentiate between packet switching and message switching technique in network Communication.	2
Ans	In Circuit Switching a physical connection between the sender and the receiver is set up first. Then, through that physical link, the complete message is communicated continuously. After completion of the transmission, the physical connection is terminated. In Packet Switching, the message is split into small units called packets. Then these packets are passed from source to destination simultaneously through different routes in the network. As the flow of packets	

	are asynchronous, sequence numbers are assigned to each packet to facilitate re-ordering of packets	
Q11	Differentiate between packet switching and message switching technique in network Communication.	2
Ans	Message switching in message switching data is stored in buffer form. The message is sent to the nearest directly connected switching node. This process continues until data is delivered to the destination computer. Packet Switching In this form of switching data is transferring into packet form. A fixed size of packet that can be transmitted across the network is specified. All the packets are stored in the main memory instead of disk.	
Q12	What is the difference between HTTP and FTP?	2
Ans	FTP is a protocol used to upload files from a workstation to a FTP server or download files from a FTP server to a workstation. HTTP is a protocol used to transfer files from a web server onto a browser in order to view a web page that is on the Internet.	
Q13	What is the advantage of using SWITCH over HUB?	2
Ans	Switch provides a dedicated line at full bandwidth between two devices but hub doesn't provide a dedicated line. Hub shares the bandwidth.	
Q14	Define the term firewall.	2
Ans	Firewall is a feature used for Network Security. In a Network there is always danger of information leaking out or leaking in. Firewall is a feature which forces all information entering or leaving the network to pass through a check to make sure that there is no unauthorized usage of the network.	
Q15	What is the importance of URL in networking?	2
Ans	URL stands for Uniform Resource Locator. Each page that is created for Web browsing is assigned a URL that effectively serves as the page's worldwide name or address. URL's have three parts: the protocol, the DNS name of the machine on which the page is located and a local name uniquely indicating the specific page (generally the filename).	

Question Type: Case Study based questions

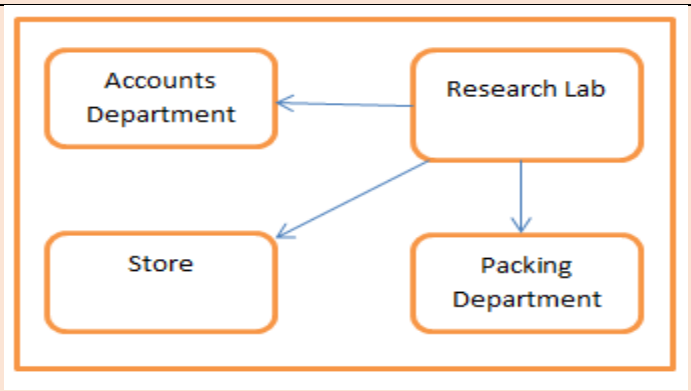
<table>
<tr>
<td>Q1</td>
<td>
Packers and movers Pvt. Ltd. is setting up the network in Mumbai. There are four departments named as MrktDept, FunDept, LegalDept, and SalesDept.


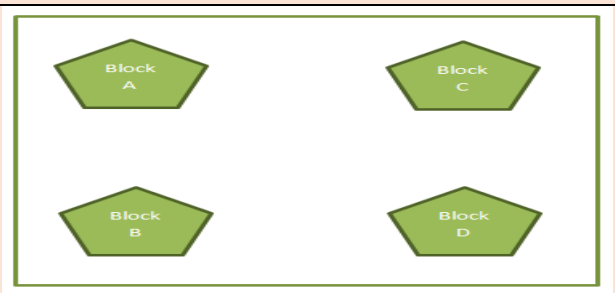
<table>
<tr>
<td>Distance between various buildings:
MrktDept to FunDept 80 m
MrktDept to LegalDept 180 m
MrktDept to SalesDept 100 m
LegalDept to SalesDept 150 m
LegalDept to FunDept 100 m
FunDept to SalesDept 50 m</td>
<td>Number of Computers in the buildings
MrktDept 20
LegalDept 10
FunDept 8
SalesDept 42</td>
</tr>
</table>
(i) Suggest a cable layout of connections between the Departments and specify topology.
(ii) Suggest the most suitable building to place the server with a suitable reason.
(iii) Suggest the placement of
a) Modem b) Hub /Switch in the network.
(iv) The organization is planning to link its sale counter situated in various part of the same
city/ which type of network out of LAN, WAN, MAN will be formed? Justify.
</td>
<td>4</td>
</tr>
</table>

<table>
<tr><td>Ans</td><td>i. Star Topology should be used.
 

ii) As per 80 – 20 rule, SalesDept because it has maximum no. of computers.
iii) Each building should have a hub/switch and Modem in case Internet connection is required.
iv) MAN (Metropolitan Area Network)</td><td></td></tr>
<tr><td>Q2</td><td>Techno Pvt. Ltd. is setting up the network in Chennai. There are four departments
 

<table>
<tr><td>Distances between various buildings :
Accounts to Research Lab 55 m
Accounts to Store 150 m
Store to Packaging Unit 160 m
Packaging Unit to Research Lab 60 m
Accounts to Packaging Unit 125 m
Store to Research Lab 180 m</td><td>Number of Computers
Accounts 25
Research Lab 100
Store 15
Packaging unit 60</td></tr>
</table></td><td>4</td></tr>
</table>

	i) Suggest a cable layout of connections between the buildings. ii) Suggest the most suitable place (i.e. buildings) to house the server of this organization. iii) Suggest the placement of the following device with justification: a) Repeater b) Hub/Switch iv) The organization is planning to link its sale counter situated in various part of the same city/ which type of network out of LAN, WAN, MAN will be formed? Justify your answer.	
Ans	(i)  (ii) The most suitable place/ building to house the server of this organization would be building Research Lab, as this building contains the maximum number of computers (iii). a)Repeater : distance between Store to Research Lab is quite large, so a repeater would ideally be placed. b)Hub/Switch : Each would be needed in all the buildings to interconnect the group of cables from the different computers in each building. iv) LAN	
Q3	**Knowledge Supplement Organization has set up its new center at Mangalore for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below:**	4



Center to center distances between various blocks	Number of Computers
Black A to Block B 50 m	Black A 25
Block B to Block C 150 m	Block B 50
Block C to Block D 25 m	Block C 125
Block A to Block D 170 m	Block D 10
Block B to Block D 125 m	
Block A to Block C 90 m	

a) Suggest a cable layout of connections between the blocks.

b) Suggest the most suitable block to house the server of this organization with a suitable reason.

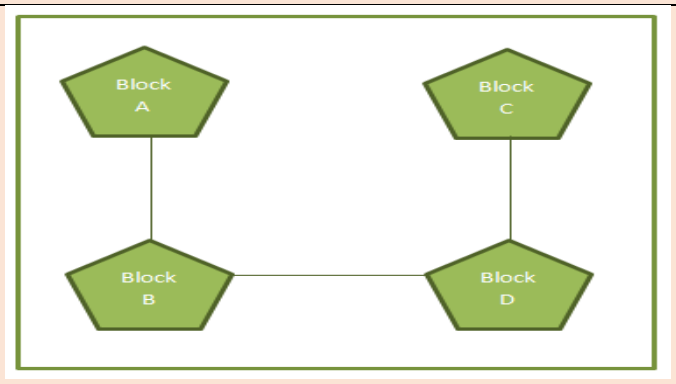
c) Suggest the placement of the following devices with justification

(i) Repeater (ii) Hub/Switch

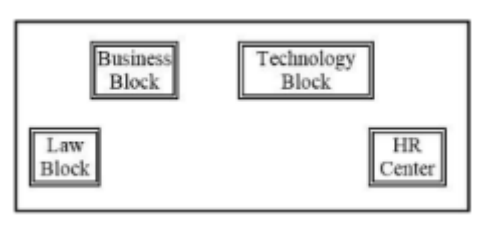
d) The organization is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

Ans

i)



	(ii) The most suitable place/ building to house the server of this organization would be block C, as this block contains the maximum number of computers. (iii) a) Repeater : distance between block B to clock C is quite large, so a repeater should be placed. b) Hub/Switch: Each would be needed in all the buildings to interconnect the computers in each building. iv) LAN	
Q4	**Agra University is setting up its academic blocks at Eidgah road and is planning to set up a network. The University has 3 academic blocks and one Human Resource Center as shown in the diagram below: Study the following structure and answer questions (i) to (v)**	5

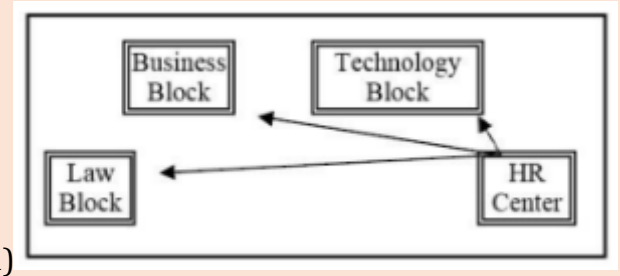


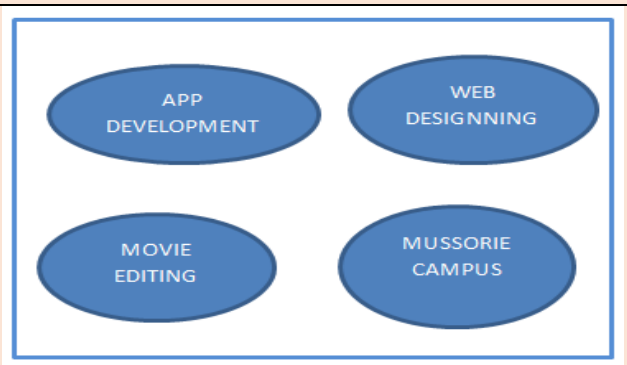
Center to Center distances between various blocks/center is as follows:	**Number of computers in each of the blocks/Center is as follows:**	
Law Block to business Block 40m	Law Block	15
Law block to Technology Block 80m	Technology Block	40
Law Block to HR center 105m	HR center	115
Business Block to technology Block 30m	Business Block	25
Business Block to HR Center 35m		
Technology block to HR center 15m		

i) Suggest the most suitable place (i.e., Block/Center) to install the server of this University with a suitable reason.

ii) Suggest an ideal layout for connecting these blocks/centers for a wired connection.

iii) Which device will you suggest to be placed/installed in each of these

	blocks/centers to efficiently connect all the computers within these blocks/centers? iv) Suggest the placement of a Repeater in the network with justification. v) The university is planning to connect its admission office in Delhi, which is more than 1250km from university. Which type of network out of LAN, MAN, or WAN will be formed? Justify your answer.	
Ans	i) The most suitable place to install the server is the HR center, as this center has a maximum number of computers. ii)  iii). Switch iv) Repeaters may be placed when the distance between 2 buildings is more than 70 meters. v) WAN, as the given distance is more than the range of LAN and MAN.	
Q5	**Make In India Corporation, an Uttarakhand based IT training company, is planning to set up training centres in various cities in the next 2 years. Their first campus is coming up in Kashipur district. At Kashipur campus, they are planning to have 3 different blocks for App development, Web designing and Movie editing. Each block has a number of computers, which are required to be connected in a network for communication, data and resource sharing. As a network consultant of this company, you have to suggest the best network related solutions for them for issues/problems raised in question nos. (i) to (v), keeping in mind the distances between various Distance between various blocks/locations:**	5



Block	Distance	Block	Number of Computers
App development to Web designing	28 m	App development	75
App development to Movie editing	55 m	Web designing	50
Web designing to Movie editing	32 m	Movie editing	30
Kashipur Campus to Mussoorie Campus	232 km	-	-

i) Suggest the type of network to connect the blocks.

ii) Suggest the most appropriate block/location to house the SERVER in the Kashipur campus (out of the 3 blocks) to get the best and effective connectivity. Justify your answer.

iii) Suggest a device/software to be installed in the Kashipur Campus to take care of data security.

iv) Suggest the placement of Switch / Hub with appropriate reasons.

v) Suggest the best wired medium and draw the cable layout (Block to Block) to economically connect various blocks within the Kashipur Campus.

Ans

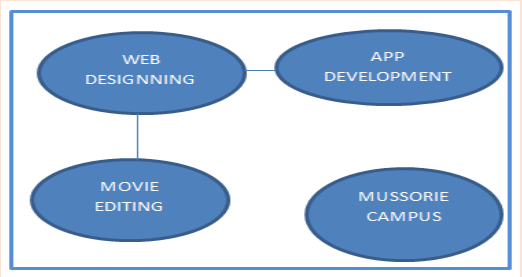
i) WAN

ii) The most suitable place/ building to house the server of this organization would be App development , as this block contains the maximum number of computers.

iii) Firewall

iv) Switch/Hub should be placed between all the blocks to connect several computers within the block.

v)



DATABASE MANAGEMENT & SQL QUERIES

COMPUTER SCIENCE – CLASS 12

1. DATABASE MANAGEMENT SYSTEM (DBMS)

A DBMS is a software that allows users to define, create, maintain and control access to a database. It acts as an interface between users and the data.

- **Data Security** — Protects data from unauthorized access.
- **Data Sharing** — Allows multiple users to access and share data.
- **Data Integrity** — Ensures accuracy and consistency of data.
- **Data Independence** — Changes in data structure do not affect applications.
- **Backup & Recovery** — Provides backup and restores data in case of failure.

2. DATABASE STRUCTURE

DATABASE (Collection of related data)
↓
TABLE (Collection of related records)
↓
RECORD / ROW (A single entry)
↓
FIELD / COLUMN (A specific attribute)

Example Table: STUDENT

RollNo	Name	Class	Marks
101	Aditi	12	88
102	Rahul	12	92
103	Priya	12	76

- **Table Name** : STUDENT
- **Fields/Columns** : RollNo, Name, Class, Marks
- **Records/Rows** : Each row represents a student

3. SQL (STRUCTURED QUERY LANGUAGE)

SQL is a standard language used to manage relational databases.

TYPES OF SQL COMMANDS

DDL	Data Definition Language (CREATE, ALTER, DROP, TRUNCATE)
DML	Data Manipulation Language (INSERT, UPDATE, DELETE)
DQL	Data Query Language (SELECT)
DCL	Data Control Language (GRANT, REVOKE)
TCL	Transaction Control Language (COMMIT, ROLLBACK, SAVEPOINT)

SYNTAX OF SELECT

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY column_name [ASC|DESC];
```

4. COMMON SQL QUERIES WITH EXAMPLES

1. CREATE TABLE — Creates a new table.

```
CREATE TABLE STUDENT (
  RollNo INT PRIMARY KEY,
  Name  VARCHAR(30),
  Class INT,
  Marks INT
);
```

2. INSERT DATA — Inserts new records into a table.

```
INSERT INTO STUDENT (RollNo, Name, Class, Marks)
VALUES (101, 'Aditi', 12, 88),
       (102, 'Rahul', 12, 92),
       (103, 'Priya', 12, 76);
```

3. SELECT ALL RECORDS — Retrieves all records from a table.

```
SELECT *
FROM STUDENT;
```

4. SELECT SPECIFIC COLUMNS — Retrieves selected columns.

```
SELECT Name, Marks
FROM STUDENT;
```

5. WHERE CLAUSE — Retrieves records that satisfy a condition.

```
SELECT *
FROM STUDENT
WHERE Marks > 80;
```

6. AND / OR OPERATORS — Combines multiple conditions.

```
SELECT *
FROM STUDENT
WHERE Class = 12 AND Marks > 80;
```

7. ORDER BY — Sorts the result set.

```
SELECT *
FROM STUDENT
ORDER BY Marks DESC;
```

8. LIKE OPERATOR — Searches for a pattern.

```
SELECT *
FROM STUDENT
WHERE Name LIKE 'A%';
-- Names starting with 'A'
```

9. BETWEEN OPERATOR — Selects values within a range.

```
SELECT *
FROM STUDENT
WHERE Marks BETWEEN 70 AND 90;
```

10. IN OPERATOR — Matches any value in a list.

```
SELECT *
FROM STUDENT
WHERE Class IN (11, 12);
```

11. AGGREGATE FUNCTIONS — Performs calculations on data.

```
SELECT COUNT(*) AS Total_Students,
       AVG(Marks) AS Avg_Marks,
       MAX(Marks) AS Highest_Marks,
       MIN(Marks) AS Lowest_Marks
FROM STUDENT;
```

12. GROUP BY & HAVING — Groups records and filters groups.

```
SELECT Class, AVG(Marks) AS Avg_Marks
FROM STUDENT
GROUP BY Class
HAVING AVG(Marks) > 75;
```

13. UPDATE DATA — Modifies existing records.

```
UPDATE STUDENT
SET Marks = 95
WHERE RollNo = 101;
```

14. DELETE DATA — Deletes records that satisfy a condition.

```
DELETE FROM STUDENT
WHERE RollNo = 103;
```

15. DROP TABLE — Deletes the entire table.

```
DROP TABLE STUDENT;
```

KEY POINTS

- A Database organizes data efficiently.
- SQL helps in storing, retrieving and managing data.
- Use the right query for the right task.

BEST PRACTICES

- Always validate data before inserting.
- Use appropriate data types.
- Keep regular backups of your database.

CHAPTER NAME: DATABASE MANAGEMENT SYSTEM

Question Type: **MCQ**

S.No	Question	Marks
Q1	What is the full form of DBMS? A) Data Backup Management System B) Database Management System C) Data Block Management System D) D. Digital Base Management System	1
Q2	Which of the following is NOT a function of DBMS? A) Data storage B) Data manipulation C) Data deletion D) Data transmission	1
Q3	Which of the following is NOT an advantage of DBMS? A) Redundancy control B) Data consistency C) More data duplication D) Data security	1
Q4	Which one of the following is an example of a DBMS? A) MS Excel B) Oracle C) Notepad D) Microsoft Word	1
Q5	A table in a database is also called a: A) Column B) Row C) File D) Relation	1
Q6	Which of the following is NOT a type of database model? A) Relational B) Network C) Hierarchical D) Tabular	1

Q7	Which language is used to access or manipulate data in a database? A) HTML B) SQL C) XML D) HTTP	1
Q8	What does the SQL command SELECT do? A) Modifies data B) Deletes data C) Retrieves data D) Inserts data	1
Q9	Which of the following is used to uniquely identify each record in a table? A) Foreign key B) Candidate key C) Primary key D) Alternate key	1
Q10	Which clause is used to filter records in SQL? A) Where B) Select C) Order By D) Group By	1
Q11	Which SQL command is used to remove a record from a table? A) Remove B) Delete C) Erase D) Drop	1
Q12	Which of the following is a valid SQL command to display all records from a table named Students? A) SHOW * FROM Students; B) DISPLAY ALL FROM Students; C) SELECT * FROM Students; D) VIEW * FROM Students;	1
Q13	In a relational database, a row is also known as a: A) Attribute B) Field	1

	C) Tuple D) Relation	
Q14	Which SQL keyword is used to sort the result-set? A) ORDER B) SORT C) GROUP D) ORDER BY	1
Q15	Which of the following is a DDL (Data Definition Language) command? A) SELECT B) INSERT C) DELETE D) CREATE	1
Q16	Which of the following can act as a foreign key? A) Any column in any table B) A column that references the primary key of another table C) A column that stores dates D) The same as a candidate key	1
Q17	Which of the following is used to remove a table from the database? A) Erase Table B) Remove Table C) Drop Table D) Delete Table	1
Q18	What does DML stand for? A) Data Management Language B) Data Modification Language C) Data Manipulation Language D) Data Mapping Language	1
Q19	Which of the following is not a valid SQL statement? A) SELECT MIN(pub_date) FROM books GROUP BY category where pub_id = 4; B) SELECT MIN(pub_date) FROM books WHERE category = 'COOKING'; C) SELECT COUNT(*) FROM orders WHERE customerid = 1005; D) SELECT MAX(COUNT(customerid)) FROM orders GROUP BY customerid;	1

Q20	Which SQL command can change the cardinality of an existing relation? A) Insert B) Delete C) Both a) & b) D) Drop	1
Q21	In SQL, a relation consists of 5 columns and 6 rows. If 2 columns and 3 rows are added to the existing relation, what will be the updated degree of a relation? A) Degree: 7 B) Degree: 8 C) Degree: 9 D) Degree: 6	1
Q22	Find Odd One Out? A) Group By B) Desc C) Asc D) Order By	1

Answers:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
B	D	C	B	D	D	B	C	C	A
Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
B	C	C	D	D	B	C	C	A	C
Q21	Q22								
A	A								

Question Type: Assertion Reasoning

S No	Question	Marks
Q1	**Assertion (A):** Every table in a relational database must have a primary key. **Reason (R):** Primary key helps to uniquely identify each record in the table. **Options:** A. Both A and R are true, and R is the correct explanation of A. B. Both A and R are true, but R is not the correct explanation of A. C. A is true but R is false. D. A is false but R is true.	1
Q2	**Assertion (A):** SQL is a programming language used to create software applications. **Reason (R):** SQL is used to perform operations like insert, update, delete, and retrieve data in databases. **Options:** A. Both A and R are true, and R is the correct explanation of A. B. Both A and R are true, but R is not the correct explanation of A. C. A is false but R is true. D. Both A and R are false.	1
Q3	**Assertion (A):** The DELETE command in SQL permanently removes a table from the database. **Reason (R):** The DELETE command removes rows from a table based on a condition. **Options:** A. Both A and R are true, and R is the correct explanation of A. B. Both A and R are true, but R is not the correct explanation of A. C. A is false but R is true. D. Both A and R are false.	1
Q4	**Assertion (A):** A foreign key can contain duplicate values. **Reason (R):** Foreign key is used to create a link between two tables. **Options:** A. Both A and R are true, and R is the correct explanation of A. B. Both A and R are true, but R is not the correct explanation of A. C. A is true but R is false. D. A is false but R is true.	1

Q5	**Assertion (A):** The DROP command is used to remove records from a table. **Reason (R):** The DROP command deletes the structure of the table along with all its data. **Options:** A. Both A and R are true, and R is the correct explanation of A. B. Both A and R are true, but R is not the correct explanation of A. C. A is true but R is false. D. A is false but R is true.	1
Q6	**Assertion (A)**: COUNT () function counts number of values in any column excluding the NULLs. **Reason (R):** COUNT (*) counts number of rows in query output including NULLs. **Options:** A. Both A and R are true and R is the correct explanation of A B. Both A and R are true but R is not the correct explanation of A C. A is true but R is false D. A is false but R is true	1
Q7	**Assertion (A)**: The group by clause combines all records from a particular field of a table. **Reason (**R): Group by displays the results in a group of particular values from the columns specified with group by clause in MySQL query. **Options:** A. Both A and R are true and R is the correct explanation of A B. Both A and R are true but R is not the correct explanation of A C. A is true but R is false D. A is false but R is true	1

Answers:

Q1	Q2	Q3	Q4	Q5	Q6	Q7
A	C	C	A	D	B	A

Question Type: Find Output / Short Answer / Long Answer Type

S.No.	Question	
Q1	What is a database?	2
Ans	A database is an organized collection of data that allows easy access, management, and updating. It stores data in a structured format, usually in tables.	
Q2	What is DBMS? List any two advantages.	2
Ans	DBMS (Database Management System) is software that helps store, manage, and retrieve data efficiently from database. Advantages: Reduces data redundancy, Ensures data consistency and integrity	
Q3	Define the terms- (i) Tuple (ii) Attribute (iii) Degree (iv) Cardinality (v) Relation	2
Ans	(i) Tuple- A row in the table (ii) Attribute-A column in the table (iii) DEGREE- Total number of Columns in the table (D-C) (iv) CARDINALITY- Total number of Rows in the table (C-R) (v) RELATION- Data arranged in the rows and columns is called RELATION or TABLE.	
Q4	What is the purpose of SQL?	2
Ans	SQL (Structured Query Language) is used to create, manage, and manipulate databases. It allows users to insert, retrieve, update, and delete data in relational databases.	
Q5	Define the following: a) Primary Key b) Foreign Key	2
Ans	a) Primary Key**:** A column or set of columns that have unique values and used to differentiate one row from other row in a table. b) Foreign Key: A column in a table, whose values have been taken from primary key of another table. It is used to establishing a relationship between two tables.	

Q6	What is meant by data redundancy? How does DBMS help reduce it?	2
Ans	Data redundancy means storing the same data at multiple places. A DBMS reduces redundancy by centralizing data storage and using relationships between tables.	
Q7	What is meant by data integrity in DBMS?	2
Ans	Data integrity refers to the accuracy, consistency, and reliability of data stored in a database. DBMS enforces integrity through constraints like Primary Key, Foreign Key, and Unique Key.	
Q8	What is the difference between char and varchar?	2

Ans.

Char	Varchar
Fixed length. If you define CHAR (10), it always uses 10 bytes, even if the value is shorter.	Variable length. If you define VARCHAR(10) and store 5 characters, it only uses 5 bytes (plus 1-2 bytes for length metadata).
Faster for fixed-length data since it doesn't need to calculate the size at runtime.	Slightly slower for large data as it calculates length dynamically.
Suitable for storing fixed-length data, like state codes (e.g., 'NY', 'CA').	Ideal for storing variable-length data, like names or addresses.
Pads shorter strings with spaces to match the defined length.	Does not pad strings; stores exactly what is entered.
Wastes storage space for shorter strings.	Efficient for data with varying lengths.

Q9	What is the difference between DDL and DML?	2
Ans		

DDL	DML
It stands for Data Definition Language.	It stands for Data Manipulation Language.
It is used to create database schema/structure.	It is used to add, retrieve or update the data.
DDL commands affect the database or table.	DML commands affect one or more records in a table.
DDL does not use WHERE clause in its statement.	While DML can use WHERE clause in its statement.
Example- CREATE, DROP, ALTER	Example -UPDATE, INSERT, DELETE

Q10	What is the difference between where and having clause?	2
Ans		

Where clause	Having clause
It specifies the condition which is to be satisfied by the records.	Having clause is used with the group by command.
Where clause selects rows before grouping.	Having clause selects rows after grouping.
Where clause cannot contain aggregate functions.	Having clause can contain aggregate functions.
Eg- select * from student where marks>40;	Eg-select stream,max(marks) from student group by stream having max(marks)>40;

Q11	What is the difference between count(col name) and count(*) ?	2
Ans		

Count(colname)	Count(*)
This is used to find the number of not null values in a column.	This is used to find the number of records/tuples in the table.
Eg- Select Count(Sal) from EMP;	Eg- Select count(*) from EMP;

<table>
<tr><td>Q12</td><td>What is the difference between Order by and Group by.</td><td>2</td></tr>
<tr><td>Ans</td><td>
<table>
<tr><th>Order by</th><th>Group by</th></tr>
<tr><td>Order by is used to display the data in ascending or descending order.</td><td>Group by is used to group the data on the basis of values of a particular column.</td></tr>
<tr><td>It is not mandatory to use aggregate functions in order to use Order by command.</td><td>It is mandatory to use aggregate functions in order to use group by command.</td></tr>
<tr><td>Eg- select * from student order by marks;</td><td>Eg- select city , count(*) from student group by city;</td></tr>
</table>
</td><td></td></tr>
<tr><td>Q13</td><td>What is the purpose of Like operator in SQL? What are wildcards?</td><td>2</td></tr>
<tr><td>Ans</td><td>The LIKE operator in SQL is used to search for character string with the specified pattern using wildcards in a column.
The wildcards are % (percentage) and _(underscore).
% is used when length is not known.
_ is used when length is known. (1 underscore represents 1 character).
For example-
(i) SELECT * FROM Student WHERE Name LIKE ‘A%’; (Display all names starting with ‘A’)
(ii) SELECT * FROM Student WHERE Name LIKE ‘_ _ _ _ _’;(Display all names having 5 characters)</td><td></td></tr>
<tr><td>Q14</td><td>What is the purpose of Distinct clause?</td><td>2</td></tr>
<tr><td>Ans</td><td>Distinct clause is used to displays the values of a column after excluding duplicate values.
For example- if a column color contains values as – Red, Blue, Red, Green, Yellow, Green
Then the query “ Select distinct color from t1; “ will give result as Red, Blue, Green, Yellow</td><td></td></tr>
<tr><td>Q15</td><td>Write suitable commands to do the following in MySQL.
I. View the table structure.
II. Create a database named SQP</td><td>2</td></tr>
<tr><td>Ans</td><td>I. Desc table_name; or describe table_name;
II. Create database SQP;</td><td></td></tr>
</table>

<table>
<tr><td>Q16</td><td>With the help of example, explain the concept of group by and having clause.</td><td>2</td></tr>
<tr><td>Ans</td><td>a) Display total number of records citywise from table emp.
Table: Emp
<table><tr><th>Empid</th><th>Name</th><th>City</th><th>Salary</th></tr><tr><td>101</td><td>Rohan</td><td>Delhi</td><td>10000</td></tr><tr><td>102</td><td>Shubham</td><td>Haryana</td><td>30000</td></tr><tr><td>103</td><td>Vishal</td><td>Delhi</td><td>25000</td></tr><tr><td>104</td><td>Vaibhav</td><td>Jaipur</td><td>30000</td></tr><tr><td>105</td><td>Shiv</td><td>Delhi</td><td>20000</td></tr><tr><td>106</td><td>Bhav</td><td>Haryana</td><td>30000</td></tr></table>SELECT CITY, COUNT(*) FROM EMP GROUP BY CITY;
<table><tr><th colspan="2">OUTPUT</th></tr><tr><td>CITY</td><td>COUNT(*)</td></tr><tr><td>Delhi</td><td>3</td></tr><tr><td>Haryana</td><td>2</td></tr><tr><td>Jaipur</td><td>1</td></tr></table>b) Display total number of records of Delhi city from table emp .
Table: Emp
<table><tr><th>Empid</th><th>Name</th><th>City</th><th>Salary</th></tr><tr><td>101</td><td>Rohan</td><td>Delhi</td><td>10000</td></tr><tr><td>102</td><td>Shubham</td><td>Haryana</td><td>30000</td></tr><tr><td>103</td><td>Vishal</td><td>Delhi</td><td>25000</td></tr><tr><td>104</td><td>Vaibhav</td><td>Jaipur</td><td>30000</td></tr><tr><td>105</td><td>Shiv</td><td>Delhi</td><td>20000</td></tr><tr><td>106</td><td>Bhav</td><td>Haryana</td><td>30000</td></tr></table>SELECT CITY, COUNT(*) FROM EMP GROUP BY CITY HAVING CITY=”Delhi”;
<table><tr><th colspan="2">OUTPUT</th></tr><tr><td>CITY</td><td>COUNT(*)</td></tr><tr><td>Delhi</td><td>3</td></tr></table></td><td></td></tr>
<tr><td>Q17</td><td>Write the purpose of ALTER command in SQL.</td><td>2</td></tr>
<tr><td>Ans</td><td>Alter command can be used for many purposes.
1. To add a new column-
ALTER table table_name ADD column_name column_type;
2. To delete a column-
ALTER table table_name DROP column_name;
3. To modify the type of column-
ALTER table table_name MODIFY column_name column_new_type;</td><td></td></tr>
<tr><td>Q18</td><td>(i) Write the SQL command to see the list of tables in a database.
(ii) Write the SQL command to insert a new record in the table.</td><td>2</td></tr>
<tr><td>Ans</td><td>(i) SHOW TABLES
(ii) INSERT INTO</td><td></td></tr>
<tr><td>Q19</td><td>(i) What constraint should be applied on a table column so that NULL is not allowed in that column, but duplicate values are allowed?
(ii) What constraint should be applied on a table column so that duplicate values are not allowed in that column, but NULL is allowed?</td><td>2</td></tr>
</table>

Ans	(i) NOT NULL constraint (ii) UNIQUE constraint	
Q20	Write the names of Aggregate functions.	2
Ans	SUM() - Returns the sum of the column given as parameter MIN() - Returns the minimum value in the given column or set of values MAX() - Returns the maximum in the given column or set of values AVG() - Returns the average value of data in the given column or set of values COUNT() - Returns the total number of values / records as per the given column	
Q21	Observe the following table and answer the question (i), (ii) and (iii) TABLE: VISITOR VisitiorID | VisitorName | ContactNumber V001 | ANAND | 9898989898 V002 | AMIT | 9797979797 V003 | SHYAM | 9696969696 V004 | MOHAN | 9595959595 (i) Write the name of most appropriate columns which can be considered as Candidate keys. (ii) Out of selected candidate keys, which one will be the best to choose as Primary Key? (iii) What is the degree and cardinality of the table?	3
Ans	(i) As per data all 3 columns can be candidate key, but practically VisitorID and ContactNumber are the best for candidate keys. (ii) VisitorID (iii) Degree(No. of columns)= 3 , Cardinality(No. of rows) = 4	
Q22	Write the SQL command to create a table demonstrating all the constraints.	2
Ans	CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, Name VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE, PhoneNumber VARCHAR(20) UNIQUE, HireDate DATE ,	

	Salary DECIMAL(10, 2) CHECK (Salary >= 0), DepartmentID INT DEFAULT 111, FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID));	
Q23	Ankur has written the following commands with respect to a table employee having fields empno, name, department, and commission. Command 1 – select count (*) from employee; Command 2 – select count(commission) from employee; He gets the output as 6 for the first command but gets 4 rows for the second command. Explain the output with justification.	2
Ans	This is due to the reason that column commission contains 2 records with NULL values. As count (*) includes NULL values also, so command 1 returns the total count as 6. Whereas count(commission) does not include NULL values, so command 2 returns total count as 4 ignoring NULL.	
Q24	What is the difference between Equi join and natural join?	2
Ans		

EQUI JOIN	NATURAL JOIN
Join condition Must be specified manually	Automatically uses common column names for joining
Can join on differently named columns	Only joins on columns with the same name
Duplicate Columns Appears in result	Duplicate Columns removed from the result
SELECT * FROM employees e , departments d where e.dept_id = d.dept_id;	SELECT * FROM employees NATURAL JOIN departments;

Q25	Consider two relations STUDENT (SNO, FNAME, LNAME) and DETAIL (ROLLNO, AGE) below: STUDENT DETAILS	2

SNO	FNAME	LNAME
1	Albert	Smith
2	Pearl	Weber

ROLLNO	AGE
5	18
9	21

What will be the Cartesian product of two tables?

Ans

On applying CROSS PRODUCT on STUDENT and DETAIL, we get:

STUDENT × DETAILS

SNO	FNAME	LNAME	ROLLNO	AGE
1	Albert	Smith	5	18
1	Albert	Smith	9	21
2	Pearl	Weber	5	18
2	Pearl	Weber	9	21

Question Type: Case Study Based Questions

S No	Question	Marks
Q1	Consider the table SALES as given below:	4

sales_id	customer_name	product	quantity_sold	price
S001	John Doe	Laptop	5	50000
S002	Jane Smith	Smartphone	10	30000
S003	Michael Lee	Tablet	3	15000
S004	Sarah Brown	Headphones	7	2000
S005	Emily Davis	Smartwatch	8	8000
S006	David	Smartwatch	3	16000
S007	Mark	Tablet	5	34000

	A.Write the following queries: (i) To display the total quantity sold for each product whose total quantity sold exceeds 12. (ii) To display the records of SALES table sorted by Product name in descending order. (iii) To display the distinct Product names from the SALES table. (iv) To display the records of customers whose names end with the letter 'e'. OR **B. Predict the output of the following:** (i) SELECT * FROM SALES WHERE PRODUCT='TABLET'; (ii) SELECT SALES_ID, CUSTOMER_NAME FROM SALES WHERE PRODUCT LIKE 'S%'; (iii) SELECT COUNT (*) FROM SALES WHERE PRODUCT IN ('LAPTOP', 'TABLET'); (iv) SELECT AVG(PRICE) FROM SALES WHERE PRODUCT='TABLET';	
Ans	A. (i) SELECT PRODUCT, SUM(QUANTITY_SOLD) FROM SALES GROUP BY PRODUCT HAVING SUM(QUANTITY_SOLD) > 12; (ii) SELECT * FROM SALES ORDER BY PRODUCT DESC; (iii) SELECT DISTINCT PRODUCT FROM SALES; (iv) SELECT * FROM SALES WHERE CUSTOMER_NAME LIKE "%E"; OR B. (i) (ii) (iii) (iv)	

(i)

sales_id	customer_name	product	quantity_sold	price
S003	Michael Lee	Tablet	3	15000
S007	Mark	Tablet	5	34000

(ii)

sales_id	customer_name
S002	Jane Smith
S005	Emily Davis
S006	David

(iii)

COUNT(*)
3

(iv)

AVG(price)
24500.0000

Q2	Madhav is managing a Travel Database and needs to access certain information from the Hotels and Bookings tables for an upcoming tourism survey. Help him extract the required information by writing the appropriate SQL queries as per the tasks mentioned below: **Table: Hotels** +------+-----------+----------+------------+ | H_ID | Hotel_Name | City | Star_Rating | +------+-----------+----------+------------+ | 1 | Hotel1 | Delhi | 5 | | 2 | Hotel2 | Mumbai | 5 | | 3 | Hotel3 | Hyderabad | 4 | | 4 | Hotel4 | Bengaluru | 5 | | 5 | Hotel5 | Chennai | 4 | | 6 | Hotel6 | Kolkata | 4 | +------+-----------+----------+------------+ **Table: Bookings** +------+------+---------------+------------+------------+ | B_ID | H_ID | Customer_Name | Check_In | Check_Out | +------+------+---------------+------------+------------+ | 1 | 1 | Jiya | 2024-12-01 | 2024-12-05 | | 2 | 2 | Priya | 2024-12-03 | 2024-12-07 | | 3 | 3 | Alicia | 2024-12-01 | 2024-12-06 | | 4 | 4 | Bhavik | 2024-12-02 | 2024-12-03 | | 5 | 5 | Charu | 2024-12-01 | 2024-12-02 | | 6 | 6 | Esha | 2024-12-04 | 2024-12-08 | | 7 | 6 | Dia | 2024-12-02 | 2024-12-06 | | 8 | 4 | Sonia | 2024-12-04 | 2024-12-08 | +------+------+---------------+------------+------------+ (i) To display a list of customer names who have bookings in any hotel of 'Delhi' city. (ii) To display the booking details for customers who have booked hotels in 'Mumbai', 'Chennai', or 'Kolkata'. (iii) To delete all bookings where the check-in date is before '2024-12-03'. (iv) To display the Cartesian Product of the two tables. OR To display the customer's name along with their booked hotel's name.	4
Ans	(i) SELECT CUSTOMER_NAME FROM HOTELS, BOOKINGS WHERE HOTELS.H_ID = BOOKINGS.H_ID AND CITY = 'DELHI'; (ii) SELECT BOOKINGS.* FROM HOTELS, BOOKINGS WHERE HOTELS.H_ID = BOOKINGS.H_ID AND CITY IN ('MUMBAI', 'CHENNAI', 'KOLKATA'); (iii) DELETE FROM BOOKINGS WHERE CHECK_IN < '2024-12-03'; (iv) SELECT * FROM HOTELS, BOOKINGS; OR SELECT CUSTOMER_NAME, HOTEL_NAME FROM HOTELS, BOOKINGS WHERE HOTELS.H_ID = BOOKINGS.H_ID;	

Q3	Consider the tables PRODUCT and BRAND given below:	4

Table: PRODUCT

PCode	PName	UPrice	Rating	BID
P01	Shampoo	120	6	M03
P02	Toothpaste	54	8	M02
P03	Soap	25	7	M03
P04	Toothpaste	65	4	M04
P05	Soap	38	5	M05
P06	Shampoo	245	6	M05

Table: BRAND

BID	BName
M02	Dant Kanti
M03	Medimix
M04	Pepsodent
M05	Dove

Write SQL queries for the following:

(i) Display product name and brand name from the tables PRODUCT and BRAND.

(ii) Display the structure of the table PRODUCT.

(iii) Display the average rating of Medimix and Dove brands

(iv) Display the name, price, and rating of products in descending order of rating.

Ans	(i) SELECT PNAME, BNAME FROM PRODUCT P, BRAND B WHERE P.BID=B.BID; (ii) DESC PRODUCT; (iii) SELECT BNAME, AVG(RATING) FROM PRODUCT P, BRAND B WHERE P.BID=B.BID GROUP BY BNAME HAVING BNAME='MEDIMIX' OR BNAME='DOVE'; (iv) SELECT PNAME, UPRICE, RATING FROM PRODUCT ORDER BY RATING DESC;	
Q4	Consider the tables given below. **Table: STOCK**	4

Itcode	Itname	Dcode	Qty	UnitPrc	StkDate
444	Drawing Copy	101	10	21	31-June-2009
445	Sharpener Camlin	102	25	13	21-Apr-2010
450	Eraser Natraj	101	40	6	11-Dec-2010
452	Gel Pen Montex	103	80	10	03-Jan-2010
457	Geometry Box	101	65	65	15-Nov-2009
467	Parker Premium	102	40	109	27-Oct-2009
469	Office File	103	27	34	13-Sep-2010

Table: DEALERS

Dcode	Dname	Location
101	Vikash Stationers	Lanka Varanasi
102	Bharat Drawing Emporium	Luxa Varanasi
103	Banaras Books Corporation	Bansphatak Varanasi

(i) To display all the information about items containing the word "pen" in the field Itname in the table **STOCK**

(ii) List all the itname sold by Vikash Stationers

(iii) List all the Itname and StkDate in descending order of StkDate

(iv) List all the Itname, Qty and Dname for all the items having quantity more than 40.

OR

List the details of all the items for which UnitPrc is in the range of 10 to 50(both inclusive)

Ans

(i) SELECT * FROM STOCK WHERE ITNAME LIKE "%PEN%";

(ii) SELECT DISTINCT(ITNAME) FROM STOCK, DEALERS WHERE STOCK.DCODE= DEALERS.DCODE AND DNAME=" VIKASH STATIONERS";

(iii) SELECT ITNAME, STKDATE FROM STOCK ORDER BY STKDATE DESC;

(iv) SELECT ITNAME, QTY, DNAME FROM STOCK, DEALERS WHERE STOCK.DCODE= DEALERS.DCODE AND QTY>40;

OR

SELECT * FROM STOCK WHERE UnitPrc BETWEEN 10 AND 50;

or

SELECT * FROM STOCK WHERE UnitPrc >= 10 AND UnitPrc <=50;

Q5 4

TABLE TRANSACT

TRNO	ANO	AMOUNT	TYPE	DOT
T001	101	2500	Withdraw	2017-12-21
T002	103	3000	Deposit	2017-06-01
T003	102	2000	Withdraw	2017-05-12
T004	103	1000	Deposit	2017-10-22
T005	102	12000	Deposit	2017-11-06

Write SQL queries for (i) to (iv) which are based on tables

TABLE : ACCOUNT

ANO	ANAME	ADDRESS
101	Nirja Singh	Bangalore
102	Rohan Gupta	Chennai
103	Ali Reza	Hyderabad
104	Rishabh Jain	Chennai
105	Simran Kaur	Chandigarh

(i) To display details of all transactions of TYPE Withdraw from TRANSACT table

(ii) To display ANO and AMOUNT of all Deposit and Withdrawals done in month of "May" 2017 from table TRANSACT.

(iii) To display first date of transaction (DOT) from table TRANSACT for Account having ANO as 102.

(iv) To display ANO, ANAME, AMOUNT and DOT of those persons from ACCOUNT and TRANSACT tables who have done transaction less than or equal to 3000.

OR

(iv) What will be the degree and cardinality of ACCOUNT table?

Ans

(i) SELECT * FROM TRNSACT WHERE TYPE='WITHDRAW';

(ii) SELECT ANO, AMOUNT FROM TRANSACT WHERE DOT = '2017-05-%';

(iii) SELECT MIN(DOT) FROM TRANSACT WHERE ANO=102;

(iv) SELECT A. ANO, ANAME, AMOUNT, DOT FROM ACCOUNT A, TRANSACT T WHERE A.ANO=T.ANO AND AMOUNT<=3000;

OR

(iv) Degree(No. of columns)= 3, Cardinality(No. of rows) = 5

Q6

4

Table: GARMENT

G CODE	G NAME	SIZE	COLOUR	PRICE
111	T Shirt	XL	Red	1400.00
112	Jeans	L	Blue	1600.00
113	Skirt	M	Black	1100.00
114	Ladies Jacket	XL	Blue	4000.00
115	Trousers	L	Brown	1500.00
116	Ladies Toop	L	Pink	1200.00

Consider the table Garment and write the query:

	(i) Display the Minimum price of the Garment. (ii) Count and display the number of GARMENT from each SIZE where number of GARMENTS are more than 1. (iii) Display the sum of price of each color garment. (iv) Display average price of XL size garments.	
Ans	(i) SELECT MIN(PRICE) FROM GARMENT; (ii) SELECT SIZE, COUNT (*) FROM GARMENT GROUP BY SIZE HAVING COUNT (*)>1; (iii) SELECT COLOUR, SUM(PRICE) FROM GARMENT GROUP BY COLOUR; (iv) SELECT AVG(PRICE) FROM GARMENT WHERE SIZE='XL';	
Q7	Consider the table TEACHER given below. **Write commands in SQL for (i) to (iii) and output for (iv) to (v)**	4

ID	Name	Dept	Hiredate	Category	Gender	Salary
1	Tanya Nanda	Social Studies	3/17/1994	TGT	F	25000
2	Saurabh Sharma	Art	2/12/1990	PRT	M	20000
3	Nandta Arora	English	5/16/1980	PGT	F	30000
4	James Jacob	English	10/16/1989	TGT	M	25000
5	Jaspreet Kaur	Hindi	8/1/1990	PRT	F	22000
6	Disha Sehgal	Math	3/17/1980	PRT	F	21000
7	Siddharth Kapoor	Science	9/2/1994	TGT	M	27000
8	Sonali Mukherjee	Math	11/17/1980	TGT	F	24500

(i) To display all information about teachers of PGT category.

(ii) To list the names of female teachers of Hindi department.

(iii) To list names, departments and date of hiring of all the teachers in ascending order of date of joining.

(iv) SELECT DISTINCT(GENDER) FROM TEACHER;

(v) SELECT AVG(SALARY) FROM TEACHER WHERE NAME LIKE 'J%';

Ans	(i) SELECT * FROM TEACHERS WHERE CATEGORY =" PGT"; (ii) SELECT NAME FROM TEACHERS WHERE GENDER='F' AND DEPT='HINDI'; (iii) SELECT NAME, DEPARTMENT, HIREDATE FROM TEACHER ORDER BY HIREDATE; (iv) F M	

	(v) 23500	
Q8	Write SQL queries for the following statements-	4

TABLE TRANSACT

TRNO	ANO	AMOUNT	TYPE	DOT
T001	101	2500	Withdraw	2017-12-21
T002	103	3000	Deposit	2017-06-01
T003	102	2000	Withdraw	2017-05-12
T004	103	1000	Deposit	2017-10-22
T005	102	12000	Deposit	2017-11-06

TABLE : ACCOUNT

ANO	ANAME	ADDRESS
101	Nirja Singh	Bangalore
102	Rohan Gupta	Chennai
103	Ali Reza	Hyderabad
104	Rishabh Jain	Chennai
105	Simran Kaur	Chandigarh

(i) To display the Aname, Amount and Type of account.

(ii) Command to add a new column named Benefits of 50 characters in ACCOUNT Table.

(iii) Display the names of all customers alphabetically.

(iv) Count the total no. of records and total amount Typewise in TRANSACT table.

OR

Identify the Foreign key from the given tables.

Ans	(i) SELECT ANAME, AMOUNT, TYPE FROM ACCOUNT, TRANSACT WHERE ACCOUNT.ANO=TRANSACT.ANO; (ii) ALTER TABLE ACCOUNT ADD BENEFITS VARCHAR (50); (iii) SELECT ANAME FROM ACCOUNTS ORDER BY ANAME; (iv) SELECT COUNT (*), SUM(AMOUNT) FROM TRANSACT GROUP BY TYPE; OR (iv) ANO in the TRANSACT table will act as Foreign key as it is acting as Primary key in ACCOUNT table and non-key attribute in TRANSACT table.	

Q9	Modern Public School is maintaining fees records of students. The database administrator Aman decided that- o Name of the database -School o Name of the table – Fees The attributes of Fees are as follows: o Rollno - numeric o Name – character of size 20 o Class - character of size 20 o Fees – Numeric o Qtr – Numeric **Answer the following questions:** (i) Identify the attribute best suitable to be declared as a primary key. (ii) Write the degree of the above table if table contains 4 rows. (iii) Write SQL query to insert one record into the Fees table. (iv) (A) Aman wants to remove the whole data from Fees table. Which command will he use from the following: a) DELETE FROM Fees; b) DROP TABLE Fees; c) DROP DATABASE Fees; d) DELETE Fees FROM Fees; OR (B) Now Aman wants to display the structure of the table Fees, i.e, name of the attributes and their respective data types that he has used in the table.	4
Ans	i)Primary Key – Rollno ii)Degree of table= 5 iii)Insert into fees values(101,'Aman','XII',5000,1); or Insert into fees(Rollno,Name,Class,Fees,Qtr) values(101,'Aman','XII',5000,1); iv)(A) DELETE FROM Fees; OR (B) Desc Fees;	

Database Connectivity — Cheat Sheet

Class 12 Computer Science | Python + MySQL

1. Import & connect

```
import mysql.connector as mycon
con = mycon.connect(host="localhost", user="root", passwd="pwd", database="school")
# host, user, passwd, database are keyword arguments passed to connect()
```

2. Cursor & executing SQL

```
cursor = con.cursor()
cursor.execute("SELECT * FROM student WHERE marks > 80")
cursor.execute("INSERT INTO student VALUES (%s,%s,%s)", (1,"Amit",92))
```

Fetch methods

`fetchone()`
Returns next single row (tuple), or None

`fetchmany(n)`
Returns next n rows as a list

`fetchall()`
Returns all remaining rows as a list

`cursor.rowcount`
Number of rows affected / fetched

Transaction control

`con.commit()`
Permanently saves INSERT/UPDATE/DELETE

`con.rollback()`
Undoes changes since last commit

`cursor.close()`
Closes the cursor

`con.close()`
Ends the connection to the server

3. SQL statement categories

SELECT	INSERT	UPDATE	DELETE
Retrieve data	Add rows	Modify rows	Remove rows
needs fetch()	*needs commit()*	*needs commit()*	*needs commit()*

4. Complete example

```
import mysql.connector as mycon

con = mycon.connect(host="localhost", user="root", passwd="pwd", database="school")
cursor = con.cursor()

cursor.execute("INSERT INTO student VALUES (%s, %s, %s)", (2, "Riya", 88))
con.commit()                    # save the new row

cursor.execute("SELECT * FROM student")
rows = cursor.fetchall()
for row in rows:
    print(row)

cursor.close()
con.close()
```

Exam tip

SELECT needs a fetch method to see results. INSERT / UPDATE / DELETE need con.commit() to save changes.
%s placeholders in execute() prevent SQL injection and let you pass Python variables safely.

Class 12 CS • Database Connectivity with Python

CHAPTER NAME: DATABASE CONNECTIVITY

Question Type: MCQ

S No	Question	Marks
Q1	Which Python module is commonly used to connect with a MySQL database? A) sqlite3 B) cx_Oracle C) mysql.connector D) pydbms	1
Q2	What does the connect () function in mysql.connector do? A) Creates a table in the database B) Executes SQL queries C) Establishes a connection to the database D) Disconnects from the server	1
Q3	Which method is used to execute SQL queries in Python after connecting to MySQL? A) run() B) execute() C) process() D) start()	1
Q4	What is the correct order of steps to connect Python with a MySQL database? A) Execute → Connect → Cursor → Commit B) Connect → Cursor → Execute → Commit C) Connect → Execute → Cursor → Commit D) Cursor → Connect → Execute → Commit	1
Q5	What is the use of commit () in database connectivity? A) To rollback the transaction B) To close the database C) To permanently save changes in the database D) To terminate the connection	1
Q6	Which method fetches all rows of a query result? A) fetch() B) fetchrws() C) fetchall() D) getall()	1

Q7	What does the cursor () method do? A) Executes SQL statements B) Fetches rows C) Creates a cursor object to interact with the database D) Commits changes	1
Q8	Which exception is raised for errors in MySQL connector? A) ValueError B) MySQLError C) DatabaseError D) Error (from mysql.connector)	1
Q9	Which of the following lines correctly connects to a MySQL database? A) mysql.connect(user="root", password="1234") B) mysql.connector.connect(host="localhost", user="root", password="1234", database="school") C) connect.mysql(user="root", db="school") D) open.mysql("localhost", "root", "school"	1
Q10	What is the purpose of close () in database connectivity? A) To delete all records B) To close the Python program C) To close the connection and free resources D) To remove the database	1

Answers

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
C	C	B	B	C	C	C	D	B	C

Question Type: Assertion Reasoning

S No	Question	Marks
Q1	**Assertion (A):** The mysql.connector module is used in Python to connect with a MySQL database. **Reason (R):** The mysql.connector.connect() function returns a cursor object. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1
Q2	**Assertion (A):** The cursor() method in Python creates an object used to execute SQL queries. **Reason (R):** A cursor object acts as a temporary workspace for executing and fetching queries. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1
Q3	**Assertion (A):** The commit() method is used to undo changes in a database. **Reason (R):** commit() saves all changes made during the transaction. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1
Q4	**Assertion (A):** The fetchall() method retrieves all rows returned by a SELECT query. **Reason (R):** fetchall() is used after executing a non-SELECT query like INSERT or UPDATE. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1

Q5	**Assertion (A):** Closing a database connection in Python is optional and has no impact on performance. **Reason (R):** The close() method releases resources held by the connection. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1
Q6	**Assertion (A):** The function mysql.connector.connect() requires parameters like host, user, and database. **Reason (R):** These parameters define which MySQL server and database the Python program should connect to. A. Both A and R are true, and R is the correct explanation of A B. Both A and R are true, and R is not the correct explanation of A C. Assertion is true but Reason is false D. Assertion is false but Reason is true	1

Answers

Q1	Q2	Q3	Q4	Q5	Q6
C	A	D	C	D	A

Question Type: **Find Output / Short Answer / Long Answer type**

S No	Question	Marks
Q1	Differentiate between fetchone() and fetchall() methods.	2
Ans	fetchone()- (i) retrieves the next row of a query result set, returning a single tuple. (ii) It returns None when no data is there in the result set. fetchall()- (i) It retrieves all the remaining rows of the query result set and returns them as a list of tuples. (ii) It returns Empty list when no data is there in the result set.	
Q2	What will be the output of the following code? import mysql.connector con = mysql.connector.connect(host='localhost', user='root', password='1234', database='school') cursor = con.cursor() cursor.execute("SELECT name FROM students WHERE grade='A'") result = cursor.fetchall() print(result)	2
Ans	The output will be a list of tuples containing names of all students who have grade 'A'. For example: [('Rahul',), ('Anita',), ('John',)]	
Q3	Write the steps involved in connecting Python with MySQL database.	2
Ans	1. Import the mysql.connector module 2. Establish connection using mysql.connector.connect() 3. Create a cursor object using cursor() 4. Execute SQL commands using execute() 5. Fetch results using fetchone(), fetchall() or fetchmany() 6. Commit changes using commit() if required 7. Close the connection using close()	

Question Type: Case Study Based Questions

Sno	Question	Marks
Q1	MySQL database named WarehouseDB has a product_inventory table in MySQL which contains the following attributes: • Item_code: Item code (Integer) • Product_name: Name of product (String) • Quantity: Quantity of product (Integer) • Cost: Cost of product (Integer) Consider the following details to establish Python-MySQL connectivity: • Username: admin_user • Password: warehouse2024 • Host: localhost Write a Python program to change the Quantity of the product to 100 whose cost is more than 500 in the product_inventory table.	4
Ans	import mysql.connector connection = mysql.connector.connect(host='localhost',user='admin_user',password='warehouse2024',database='WarehouseDB') cursor = connection.cursor() update_query = "UPDATE product_inventory SET Quantity = 100 WHERE cost>500" cursor.execute(update_query) connection.commit() print("Data updated successfully.") cursor.close() connection.close()	
Q2	Kabir wants to write a program in Python to insert the following record in the table named Student in MYSQL database, SCHOOL: ● rno(Roll number)- integer ● name(Name) - string ● DOB (Date of birth) – Date ● Fee – float	4

	Note the following to establish connectivity between Python and MySQL: ● Username - root ● Password - tiger ● Host - localhost The values of fields rno, name, DOB and fee has to be accepted from the user. Help Kabir to write the program in Python.	
Ans	import mysql.connector as mysql con1 = mysql.connect(host="localhost", user="root", password="tiger", database= "SCHOOL") mycursor=con1.cursor() rno = int(input("Enter Roll Number:: ")) name = input("Enter the name:: ") DOB= input("Enter date of birth:: ") fee= float(input("Enter Fee:: ")) query = "INSERT into student values({}, '{}', '{}',{})".format(rno,name, DOB, fee) mycursor.execute(query) con1.commit() print("Data added successfully") con1.close()	
Q3	Rishabh has created a table named Student in MYSQL database, SCHOOL: ● rno(Roll number)- integer ● name(Name) - string ● DOB (Date of birth) – Date ● Fee – float Note the following to establish connectivity between Python and MySQL: ● Username - root ● Password - tiger ● Host – localhost Rishabh, now wants to display the records of students whose fee is more than 5000. Help himj to write the program in Python.	4
Ans	import mysql.connector as mysql con1=mysql.connect(host="localhost",user="root", password="tiger", database= "SCHOOL") mycursor=con1.cursor() query = "SELECT * FROM student where fee>{}".format(5000) mycursor.execute(query) data=mycursor.fetchall()	

	for rec in data: print (rec) con1.close()	

Q4	Vishal wants to interface python with mysql and write some code. Help him to write the code -
	import_____.connector #Line1
	mydb=mysql.connector._____(host="localhost",user="root",
	passwd="tiger",database="school") #Line2
	cursor=mydb._____() #Line3
	cursor._____("select * from stud") #Line4
	data=cursor._____() # Line5 To retrieve all records
	count=cursor._____ #Line6 To count total rows
		4

Ans	Line1:-mysql Line2:-connect Line3:cursor Line4: execute Line5: fetchall Line6: rowcount	
Q5	Consider the Table 'ATTENDANCE' structure in the SCHOOL database below. **Field Name** / **Data Type** Rollno / Int(11) Studentname / Varchar(30) Classname / Varchar(3) Attenddate / Date Attendance / Varchar(1) Assume the following for Python-MySQL connectivity. Host: localhost User: root Password: tiger Write the Python function to perform the specified operations given below: A) RecordAttendance(): To accept details of the student and store them in the Table	

	ATTENDANCE. B) DisplayAttendance(): to accept the Attendance date and display all the records from Table ATTENDANCE as per the attendance date entered.	

Ans	import mysql.connector as sqlcon
	mycon=sqlcon.connect(host="localhost",user="root",passwd="tiger",database="SCHOOL")
	A) def RecordAttendance():
	rollno=input("Enter Roll Number:=")
	sname=input("Enter Student Name=")
	cname=input("Class=")
	adate=input ("Date(in yyyy-mm-dd Format):")
	att=input('Attendace:(P or A')
	record=(rollno, sname,cname,adate,att)
	if mycon.is_connected():
	cursor=mycon.cursor()
	sqlcmd="""insert into attendance values(%s,%s,%s,%s,%s)"""
	cursor.execute(sqlcmd,record)
	mycon.commit()
	print("Record added Successfully.!!")
	cursor.close()
	mycon.close()
	else:
	print("Unable to setup Connection to Database Successful:")
	
	B) def DisplayAttendance():
	adate=input ("Date(in yyyy-mm-dd Format):")
	data=(adate)
	if mycon.is_connected():
	cursor=mycon.cursor()
	sqlcmd="""select * from attendance where attendDate = %s"""
	cursor.execute(sqlcmd,data)
	resultset=cursor.fetchall()
	for row in resultset:
	print(row)	

	cursor.close() mycon.close() else: print("Unable to Fetch Attendance!")	

Reviewed By: **Mrs. Nausheen Khan, PGT(CS)**

KV CRPF Rampur

"Code is not just about writing instructions;

it is about transforming ideas into solutions."