

Minimum Level of Learning Class XII Computer Science



“SESSION 2025-26

KENDRIYA VIDYALAYA SANGATHAN

Zonal Institute of Education & Training, Gwalior

OUR MENTOR:

MR. B.L. MORODIA

Deputy Commissioner & Director

KVS Zonal Institute of Education and Training, Gwalior

Course Coordinator

MRS. ANITA KANAJIA, T.A. ECONOMICS

RESOURCE PERSONS

1. **MRS SANGEETA M CHAUHAN ,
PGT CS , PM SHRI K V NO3 GWALIOR**
2. **MR. ALOK GUPTA
PGT COMP, PM SHRI K V ETAWAH**

VETTED BY:

1. **MR RAJU DIXIT ,
PGT CS, PM SHRI K V ALIGARH**
2. **MR. RAKESH KUMAR SINGH YADAV
PGT CS, PM SHRI K V DOGRA LINES MEERUT CANTT**
3. **MR. SUNIL KUMAR BHELE
PGT CS, PM SHRI K V PUNJAB LINES MEERUT CANTT**
4. **MR. MANISH GUPTA
PGT CS, PM SHRI K V HATHRAS**
5. **MR PANKAJ SINGH
PGT(CS) , PM SHRI KV TALBEHAT**

Identifiers/Data types/Conversions/Evaluation of Expression

Features in Python

1. Free and Open Source. ...
2. Object-Oriented Language. ...
3. GUI Programming Support. ...
4. Cross-platform....
5. Portable language and case sensitive ...
6. Interpreted Language....

Tokens :

The Smallest individual unit of the program is called Token.

Difference between keyword and identifier:

Keyword	Identifier
Reserved words for special purpose	User-defined names for program
Cannot be used as identifiers	Used as names for variables, functions, etc.
Exam: if, else, for, while, def, class	Exam :my_variable, calculate_area

Rules for identifier names:

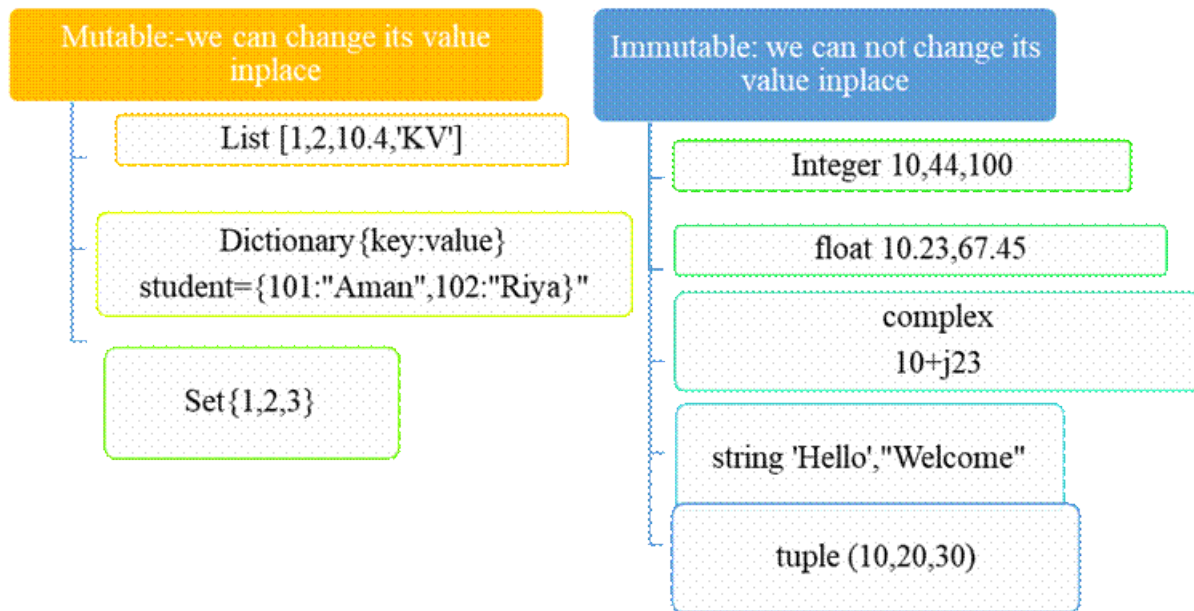
1. Keywords and operators are not to be used
2. Start with an alphabet or an underscore.
3. Special characters other than underscores are not allowed.
4. Space is not allowed.
5. Can not start with a number.
6. Note: Python is case sensitive and hence uppercase and lowercase are treated differently.

Examples: mark, _For , WHILE, mark123 ,true

Identify valid and invalid identifiers:

- a) _123
- b) 1_23
- c) 123_
- d) a.123
- e) true
- f) break
- g) data.file
- h) file@record

Data Types: to identify the type of data and associated operation of handling it.



Sequence Data Type : list,tuple,string

Mapping: Dictionary

Q. Write the data type of the given data:

100, 234.89, True, true, (10), (10,), [1,2,3], {(1,2):"KVS"}

Literals are fixed data values

Python supports several types of literals:

- **Numeric Literals:** Represent numerical values.
 - Integer literals: Whole numbers, e.g., 11, -5, 0. They can also be represented in binary (prefix 0b), octal (prefix 0o), or hexadecimal (prefix 0x).
 - Float literals: Decimal numbers, e.g., 3.14, -0.001, 1.0e8.
 - Complex literals: Numbers with a real and imaginary part, e.g., 2 + 3j.
- **String Literals:** Represent sequences of characters.
 - Enclosed in single quotes ('), double quotes ("), or triple quotes (""" or """) for multiline strings. e.g., 'hello', "world", """welcome to kvs""".
- **Boolean Literals:** Represent either True or False value.
- **Special Literal:** None represents the absence of a value.
- **Collection Literals:** Represent collections of data.
 - List literals: Ordered, mutable sequences, e.g., [1, 2, 3].
 - Tuple literals: Ordered, immutable sequences, e.g., (1, 2, 3).
 - Dictionary literals: Unordered collections of key-value pairs, e.g., {'a': 1, 'b': 2}.
 - Set literals: Unordered collections of unique elements, e.g., {1, 2, 3}

Operators:

Arithmetic operators	+, -, *, //(floor), /, %(modulus), ** (exponent)	a=10, b=3 c=10//3 print(c) output 3	a=10, b=3 d=a%b print(d) output 1
Relational Operators	>, >=, <, <=, ==, !=		
Logical Operators	AND, OR, NOT		
Identity operators	is, is not		
Membership operators	in, not in		
Argument operators	+=, -=, *=, /=, //=, %=, **=	a=a+10 or a+=10	

Precedence of Operators: Order of execution of the operators.

Operators	Description	Associativity
()	parenthesis	Left to Right
**	Exponentiation	Right to Left
+x, -x	Positive, negative (Unary)	Left to Right
*, /, //, %	Arithmetic	
+, -		
in, not in, is, is not , <, <=, >, >=, ==, !=	Membership, Identity, Relational	
not	Boolean not	
and	Boolean and	
or	Boolean or	

Evaluate the following expression and gives the output

- 6 < 12 and not (20 > 15) or (10 > 5)
- print(4 + 2 ** 3 ** 2 - 55 // 11 % 3)

Comments in python: Comments are non-executable statements of python. It increases the readability and understandability of code.

Types of comment:

- i. Single line comment (#) – comments only on a single line.
e.g., `a=7 # 7 is assigned to variable 'a'`
- ii. multi-line comment (""".....""") – Comments multiple line.
- iii inline comment (#) if `a==b: # compare value of a and b`

Type conversion : Type conversion is the process of changing the data type of a value in Python.

1. implicit (done by system(automatically))
2. explicit (type casting or user defined conversion)

Implicit type conversion happens automatically when Python encounters mixed data types in an expression. It converts the data type to a higher-order data type to prevent data loss. For example, when adding an integer to a float, Python implicitly converts the integer to a float before performing the addition.

Example

```
a=10+20.7  
print(a) #30.7
```

Explicit type conversion(type casting) is when the programmer manually converts the data type of a value using built-in functions like `int()`, `float()`, `str()`, `list()`, `tuple()`, `set()`, and `dict()`.

Example:

```
print(int(234.78)) #234
```

Conditional Statements/Errors

Python if else Statements – Conditional Statements

In Python, if-else is a fundamental conditional statement used for decision-making in programming. if...else statement allows the execution of specific blocks of code depending on the condition is True or False.

if Statement:-

if statement is the most simple decision-making statement. If the condition evaluates to True, the block of code inside the if statement is executed.

Example of if Statement:

```
i = 10  
# Checking if i is greater than 15  
if (i > 15):  
    print("10 is less than 15")  
  
print("I am Not in if")
```

if...else Statement :

if...else statement is a control statement that helps in decision-making based on specific conditions. When the if condition is False. If the condition in the if statement is not true, the else block will be executed.

Let's look at some examples of if-else statements.

- Simple if-else

```
i = 20
```

```
# Checking if i is greater than 0
```

```
if (i > 0):
```

```
    print("i is positive")
```

```
else:
```

```
    print("i is 0 or Negative")
```

- if else in One-line

If we need to execute a single statement inside the if or else block then one-line shorthand can be used.

```
a = -2
```

```
# Ternary conditional to check if number is positive or negative
```

```
res = "Positive" if a >= 0 else "Negative"
```

```
print(res)
```

Output

Negative

Logical Operators with if..else :

We can combine multiple conditions using logical operators such as and, or, and not.

```
age = 25
```

```
exp = 10
```

```
# Using '>' operator & 'and' with if-else
```

```
if age > 23 and exp > 8:
```

```
    print("Eligible.")
```

```
else:
```

```
    print("Not eligible.")
```

Output

Eligible.

Nested if else Statement :

Nested if...else statement occurs when if...else structure is placed inside another if or else block.

Nested If..else allows the execution of specific code blocks based on a series of conditional checks.

Nested-statement

Nested if Statement

Example of Nested If Else Statement:

```
i = 10
```

```
if (i == 10):
```

```
    # First if statement
```

```
    if (i < 15):
```

```
        print("i is smaller than 15")
```

if...elif...else Statement :

if-elif-else statement in Python is used for multi-way decision-making. This allows us to check multiple conditions sequentially and execute a specific block of code when a condition is True. If none of the conditions are true, the else block is executed.

If-elif-else-Statement:-

Example:

```
i = 25
# Checking if i is equal to 10
if (i == 10):
    print("i is 10")
# Checking if i is equal to 15
elif (i == 15):
    print("i is 15")
# Checking if i is equal to 20
elif (i == 20):
    print("i is 20")
# If none of the above conditions are true
else:
    print("i is not present")
```

Iterative Statement

Iterative Statement/Repetition of a set of statements in a program is made possible using looping constructs.

The 'for' Loop

The for statement is used to iterate over a range of values or a sequence. The for loop is executed for each of the items in the range. These values can be either numeric, or they can be elements of a data type like a string, list, tuple or even dictionary.

Syntax of the for Loop :

```
for <control-variable> in <sequence/ items in range>:
    <statements inside body of the loop>
```

The 'while' Loop

The while statement executes a block of code repeatedly as long as the control condition of the loop is true. The control condition of the while loop is executed before any statement inside the loop is executed. After each iteration, the control condition is tested again and the loop continues as long as the condition remains true. When this condition becomes false, the statements in the body of loop are not executed and the control is transferred to the statement immediately following the body of the while loop. If the condition of the while loop is initially false, the body is not executed even once.

Syntax of while loop:-

```
while test_condition:
    body of while
```

Break and Continue Statement

In certain situations, when some particular condition occurs, we may want to exit from a loop (come out of the loop forever) or skip some statements of the loop before continuing further in the loop. These requirements can be achieved by using break and continue statements, respectively.

Lists and Tuples

Lists in Python :

Definition:

- A list is an ordered, mutable (changeable) collection of items.
- Defined using square brackets [].

Example:

```
my_list = [1, 2, 3, "apple", True]
```

Key Features:

- Ordered: Items have a defined index.
- Mutable: You can change, add, or remove elements.

Common Operations:

```
my_list.append("banana")    # Add item
my_list[0] = 100            # Change item at index 0
del my_list[1]              # Delete item at index 1
len(my_list)                # Get length
```

Tuples in Python :

Definition:

- A tuple is an ordered, immutable (unchangeable) collection of items.
- Defined using parentheses ().

Example:

```
my_tuple = (1, 2, 3, "apple", True)
```

Key Features:

- Ordered: Items have a fixed position.
- Immutable: Cannot be changed after creation.

Common Operations:

```
len(my_tuple)              # Get length
my_tuple[1]                 # Access item at index 1
```

Single-item Tuple:

```
one_item = (5,)            # Note the comma!
```


List vs Tuple :

Feature	List	Tuple
Brackets	[]	()
Mutable	Yes	No
Methods	Many (e.g., append(), remove())	Few (e.g., count(), index())
Use Case	When data may change	When data should stay constant
Performance	Slower	Faster
Memory Usage	Higher	Lower
Hashable	No	Yes (if elements are hashable)

Dictionaries :

- In python a dictionary is a python data type which can store mappings in form of key-value pairs.
- A dictionary is an ordered sequence(in recent python versions) of key-value pairs.
- A Key and value in a key-value pair in a dictionary are separated by a colon. Multiple key-value pairs in a dictionary are separated by comma(s) and are enclosed within curly braces.
- Keys of the dictionaries are immutable types such as Integers or Strings etc. but since values of dictionaries are mutable hence dictionaries are considered as mutable data type.

Creating an Empty Dictionary:

```
Mydictionary = { }          # Empty Dictionary named Mydictionary created
print(Mydictionary)
```

Output:

```
{ }
```

Creating a Dictionary with multiple key value pairs:

```
Mydictionary2 = {"SName" : "Ramesh", "Class" : 12, "City" : "Prayagraj"}
print(Mydictionary2)
```

Output:

```
{"SName" : "Ramesh", "Class" : 12, "City" : "Prayagraj"}
```

Creating empty Dictionary using dict() function:

dict() function is used to create an empty dictionary as follows:

```
Days = dict()          # Creates an empty dictionary
print(Days) # Prints an empty dictionary
```

Output:

```
{ }
```

Creating Dictionary(non empty) with key-value pairs using dict() function:

```
DaysMonth= dict(Jan = 31, Feb = 29, March = 31)
```

```
print(DaysMonth)
```

Output:

```
{ 'Jan': 31, 'Feb': 29, 'March': 31 }
```

Adding key-value pairs in an existing Dictionary:

We can use Square Brackets([]) with keys for adding or updating values in a dictionary. For

Example:

```
Months={ }
```

```
Months[1] = "Jan"
```

```
Months[2] = "Feb"
```

```
Months[3] = "Mar"
```

```
Months[4] = "Apr"
```

```
Months[5] = "May"
```

```
Months[6] = "Jun"
```

```
print(Months)
```

Output:

```
{ 1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun' }
```

Accessing Elements of a Dictionary:

Elements of Dictionary may be accessed by writing the Dictionary name and key within square brackets ([]) as given below:

```
SeqDay = {0 : "Sunday", 1 : "Monday", 2: "Tuesday"}
```

```
print(SeqDay [1])
```

Output:

Monday

Note: Attempting to access a key that does not exist, produces an error. Consider the following statement

that is trying to access a non – existent key (7) from the dictionary SeqDay. it raises KeyError.

```
print(SeqDay[7])
```

Output:

KeyError : 7

Dictionary Methods/Functions:

Dictionary Methods/Functions	Description
keys()	Returns a view object that displays a list of all the keys in the dictionary.
values()	Returns a view object containing all dictionary values.
items()	Return the list with all dictionary keys along with their values
get()	Returns the value for the given key
clear()	Deletes all items from the dictionary
copy()	Returns a shallow copy of the dictionary
fromkeys()	Creates a dictionary from the given sequence
pop()	Returns and removes the element with the given key
popitem()	Returns and removes the item that was last inserted into the dictionary.
setdefault()	Returns the value of a key if the key is in the dictionary, otherwise it inserts the key with a value to the dictionary
update()	Updates the dictionary with the elements from another dictionary or an iterable of key-value pairs. By using this method we can include new data or merge it with existing dictionaries.

Applying The Methods / Functions on Dictionary :

```
AMonths={ 1:"Jan",2:"Feb",3:"Mar",4:"Apr",5:"May",6:"Jun",  
7:"July",8:"Aug",9:"Sep",10:"Oct",11:"Nov",12:"Dec"}
```

```
print(AMonths)                    # Prints the entire dictionary  
print("All the Keys :", AMonths.keys())    # Prints all the keys of the dictionary  
print("All the Values :",AMonths.values())  # Prints all the values of the dictionary  
print("All the Key-value Pairs :",AMonths.items())  # Prints all the keys and values of the  
dictionary  
print("The Value of Key 5 :",AMonths.get(5))    # Accesses the value of the given key  
AMonths.clear()                    # Deleting all the key-value pairs  
print("Printing after applying clear() Method :", AMonths) # Printing Dictionary after applying  
clear()  
AMonths.update({ 1:"Jan",2:"Feb",3:"Mar",4:"Apr"})    # Adding key-value pairs if not  
already exists  
print("Printing Dictionary after Adding key-value pairs using update() Method :", AMonths)  
AMonths.update({ 3:"March- The Third Month"})    # Modifying key-value pairs as it  
already exists  
print("Printing Dictionary after Modifying a key-value pair using update() Method :", AMonths)  
AMonths.pop(4)                    # deleting key-value pair having key as 4  
print("Printing Dictionary after deleting a key-value pair having key as 4 :",AMonths)
```

String: Represent sequences of characters.

It is enclosed in single quotes ('), double quotes ("), or triple quotes (''' or ''')

e.g., 'hello', "world", '''This is a multiline string'''.

Accessing characters in Python String

Str='Python'

0	1	2	3	4	5	Forward index
P	y	t	h	o	n	
-6	-5	-4	-3	-2	-1	Backward index

STRING SLICING

Slicing is a way to extract portion of a string by specifying the start and end indexes. The syntax for slicing is string[start:end], where start starting index and end is stopping index (excluded).

<pre>>>> s="welcome" >>> s[1:5] 'elco' >>> s[1:8:2] 'ecm' >>> s[-5:-1] 'com' >>> s[-1:8] '' >>> s[-1:8:-1] 'emoclew'</pre>	<pre>>>> s[2:] 'come' >>> s[:2] 'we' >>> s[2::2] 'loe'</pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------

STRING OPERATORS:

Two basic operators + and * are allowed

+ is used to combine two String (Concatenation)

*is used for replication(repetition)

Functions in String:

Function	Description
mystring[:N]	Extract N number of characters from the start of the string.

<code>mystring[-N:]</code>	Extract N number of characters from end of string
<code>mystring[X:Y]</code>	Extract characters from middle of string, starting from X position and ends with Y
<code>str.split(sep=' ')</code>	Split Strings
<code>str.replace(old_substring, new_substring)</code>	Replace a part of text with different substring
<code>str.lower()</code>	Convert characters to lowercase
<code>str.upper()</code>	Convert characters to uppercase
<code>str.contains('pattern', case=False)</code>	Check if pattern matches (Pandas Function)
<code>str.extract(regular_expression)</code>	Return matched values (Pandas Function)
<code>str.count('sub_string')</code>	Count occurrence of pattern in string
<code>str.find()</code>	Return position of substring or pattern
<code>str.isalnum()</code>	Check whether string consists of only alphanumeric characters

Question 1 : Give reason why in string `s='Welcome'`, `s[0]='w'` is generating error.

Question 2 : Consider the following string `s="Green India,New India"`

What will be the output of the following string operation

(1) `print(s.lower())` (2) `print(s.upper())` (3) `print(s.count('a'))` (4) `print(s.find('New'))`
 (5) `print(len(str1)-3)` (6) `print(s.split(","))` (7) `print(s.split())` (8) `print(s.replace('New', "Old"))`
 (9) `print(s.isdigit())` (10) `print(s.title())`

Answer:

1. green revolution 2. GREEN REVOLUTION
 3. 0 4. 12 5. 18 6. ['Green India', 'New India'] 7. ['Green', 'India,New', 'India']
 8. Green India,Old India 9. False 10. Green India,New India

Python Module and Function Basics

A Python module is a file that contains a collection of related functions, classes, and variables that can be used in other Python programs. Modules are a way to organize and reuse code, making it easier to write and maintain large programs.

Importing Modules

1. `import module_name`: Imports the entire module, and you can access its functions and variables using the module name.
2. `from module_name import function/variable`: Imports a specific function or variable from the module, and you can use it directly.
3. `from module_name import *`: Imports all functions and variables from the module, and you can use them directly.

The Python random module provides functionalities for generating random numbers. Here are some basic functions:

Random Number Generation

1. `random()`: Returns a random floating-point number between 0 and 1.
2. `randint(a, b)`: Returns a random integer between a and b (inclusive).

Random Sequence Operations

1. `choice(seq)`: Returns a random element from the sequence seq.
2. `shuffle(seq)`: Randomly rearranges the elements of the sequence seq.

Functions :

In Python, a function is a block of code that can be executed multiple times from different parts of your program. Functions are useful for:

Benefits of Functions

1. **Code Reusability**: Functions allow you to reuse code, reducing duplication and improving maintainability.
2. **Modularity**: Functions help to break down a large program into smaller, manageable modules.
3. **Readability**: Functions make your code more readable by providing a clear and concise way to perform a specific task.

Defining a Function

In Python, you define a function using the `def` keyword followed by the function name and parameters in parentheses. Here's a basic example:

```
def greet(name):  
    print(f'Hello, {name}!')
```

Function Components

1. **Function Name**: The name of the function.
2. **Parameters**: Variables that are passed to the function when it's called.
3. **Function Body**: The code that gets executed when the function is called.
4. **Return Value**: The value that the function returns to the caller.

Type of Functions and Calling Functions

Python Function Types :

Python functions can be broadly classified into three categories:

- Built-in Functions
- Library Functions (also called Functions defined in modules)
- User-defined Functions

1. Built-in Functions

Definition:

Built-in functions are pre-defined functions provided by Python. They are readily available and do not require importing any module. These functions are designed to perform common operations like mathematical calculations, data type conversions, string manipulations, and more.

Common Built-in Functions:

print(): Used to display messages or values on the screen.

len(): Returns the length of an object (e.g., string, list, or tuple).

sum(): Adds all elements in an iterable (e.g., list or tuple).

type(): Returns the type of the object (e.g., int, str).

max(): Returns the maximum element from an iterable or multiple arguments.

min(): Returns the minimum element from an iterable or multiple arguments.

Example of Built-in Functions

```
print("Welcome to Python!") # Output: Welcome to Python!
```

```
length_of_string = len("Class 12")
```

```
print("Length of string 'Class 12' is:", length_of_string) # Output: 8, as there are 8 characters
```

```
total_sum = sum([10, 20, 30, 40])
```

```
print("Sum of the list [10, 20, 30, 40] is:", total_sum) # Output: 100, sum of the list elements
```

```
data_type = type(25.5)
```

```
print("Data type of 25.5 is:", data_type) # Output: <class 'float'>
```

2. Library Functions (Module Functions)

Definition:

Library functions are predefined functions that are part of external libraries (also called modules). To use these functions, you need to import the respective module using the import statement. These functions are not included in Python's core and extend its functionality.

Common Library Modules:

math: Provides mathematical functions like square root, trigonometry, logarithms, and constants like pi.

random: Offers functions for generating random numbers and selections.

datetime: Handles date and time manipulations.

os: Interacts with the operating system (e.g., file handling, directory management).

Examples:

```
import math # Importing the math module
```

```
# Using math module functions
```

```
square_root = math.sqrt(16)
```

```
print("Square root of 16 is:", square_root) # Output: 4.0, square root of 16
```

```
cosine_value = math.cos(0)
```

```
print("Cosine of 0 radians is:", cosine_value) # Output: 1.0, cosine of 0 radians
```

```
pi_value = math.pi
```

```
print("Value of pi is:", pi_value) # Output: 3.141592653589793
```

3. User-defined Functions

Definition:

User-defined functions are functions that are created by the user using the def keyword. These functions allow programmers to group related code and logic into reusable blocks. They promote code reusability, modularity, and better organization.

Syntax:

```
def function_name(parameters):  
    # Function body  
    # Perform some operations  
    return value # Optional
```

Examples:

Function to calculate the factorial of a number

```
def factorial(n):
```

```
    if n == 0 or n == 1:  
        return 1
```

```
    else:
```

```
        return n * factorial(n - 1)
```

Calling the user-defined function

```
result = factorial(5)
```

```
print("Factorial of 5 is:", result) # Output: 120, as factorial(5) = 5 * 4 * 3 * 2 * 1
```

Advantages of User-defined Functions:

- Code Reusability: Allows the same code to be reused multiple times.
- Modularity: Breaks down complex code into smaller, manageable pieces.
- Readability: Improves code readability and understanding.
- Easy Maintenance: Functions make it easier to modify and maintain code.

Understanding Actual and Formal Parameters in Python Functions

When discussing functions in Python, these are terms used to differentiate between values passed to a function and the variables that receive these values within the function.

1. Formal Parameters (Parameters):

Definition: Formal parameters are the variables defined in the function's declaration. They act as placeholders for the values that will be passed to the function when it is called.

Location: Formal parameters appear in the function definition, inside the parentheses.

Example:

```
def greet(name, age):  
    print(f"Hello {name}, you are {age} years old.")
```

2. Actual Parameters (Arguments):

Definition: Actual parameters, often called arguments, are the real values or data that you pass to a function when calling it.

Location: Actual parameters appear in the function call, inside the parentheses.

Example:

```
greet("Sachin", 33)
```


Passing Parameters to Functions in Python

In Python, functions can accept parameters in various ways to allow flexibility when calling them. Understanding how parameters are passed to functions is essential for writing efficient and readable code.

Python supports three primary types of parameter passing:

- Positional Arguments (Required Arguments)
- Keyword Arguments (Named Arguments)
- Default Arguments

1. Positional Arguments (Required Arguments)

Definition: Positional arguments are passed to a function in the same order as the parameters are defined. The position (sequence) in which the arguments are passed during the function call determines which parameter gets which value.

Usage: These arguments are mandatory, meaning if you skip any required positional argument while calling the function, it will result in an error.

Example 1: Basic Positional Arguments

```
def add(a, b): # 'a' and 'b' are formal parameters.  
    return a + b  
result = add(10, 5) # 10 and 5 are positional arguments.  
print("Sum is:", result) # Output: Sum is: 15
```

Example 2: Swapping the Positions

```
def student_details(name, age):  
    print(f"Student Name: {name}")  
    print(f"Student Age: {age}")  
student_details("John", 18)  
student_details(18, "John") # Output: Student Name: John, Student Age: 18  
# Output: Student Name: 18, Student Age: John
```

2. Keyword Arguments (Named Arguments)

Definition: Keyword arguments allow you to pass values by explicitly specifying the parameter names, regardless of their position. This makes the function call more readable and avoids confusion.

Usage: When using keyword arguments, the order of the arguments does not matter because the parameter names are used to match the values.

Example 1: Using Keyword Arguments

```
def introduce (name, city):  
    print(f"Hello, my name is {name} and I am from {city}.")  
introduce (name="Alice", city="New York") # Output: Hello, my name is Alice and I am from  
New York.
```

Example 2: Using Keyword Arguments

```
def display_employee_info (emp_name, emp_id, emp_dept):  
    print(f"Employee Name: {emp_name}")  
    print(f"Employee ID: {emp_id}")  
    print(f"Employee Department: {emp_dept}")  
name = input("Enter the employee's name: ")  
id = input("Enter the employee's ID: ")
```

```
department = input("Enter the employee's department: ")
```

```
display_employee_info (emp_dept=department, emp_id=id, emp_name=name)
```

3. Default Arguments

Definition: Default arguments are parameters that assume a default value if no argument is passed during the function call. They are defined by assigning a value to the parameter in the function definition.

Usage: Default arguments are useful when you want to provide flexibility and avoid errors from missing arguments.

Example 1: Basic Default Argument

```
def greet (name="Guest"):    # 'name' has a default value of "Guest".
```

```
    print(f'Hello, {name}!')
```

```
greet()
```

```
greet("Sachin") # Output: Hello, Guest!
```

```
# Output: Hello, Sachin!
```

Example 2: Multiple Default Arguments

```
def power(base, exponent=2): # 'exponent' has a default value of 2.
```

```
    return base ** exponent
```

```
print(power(5))
```

```
print(power(5, 3))
```

```
# Output: 25, as  $5^2 = 25$  (exponent uses default value)
```

```
# Output: 125, as  $5^3 = 125$  (default value is overridden)
```

Scope of Variable, Returning of Values, Types of Parameters in Functions

Scope of Variable

A variable is a name used to store data.

Local variable:

- Created inside a function. * Can be used only inside that function.

Example:

```
def show():
```

```
    x = 5                # local variable
```

```
    print(x)
```

Global variable:

- Created outside of all functions. * Can be used inside and outside any function.

Example:

```
x = 10                # global variable
```

```
def show():
```

```
    print(x)
```

Returning of Values

- A function can send back a result using the return statement.
- return ends the function and gives back a value.
- You can return one or more values.

Types of Parameters

Parameters are values given to a function when it is called.

a. Required Parameters

- Must be given in the correct order.

Example:

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

b. Default Parameters

- Have a value already set. * If no value is given, the default is used.

Example:

```
def greet(name="Guest"):  
    print("Hello", name)  
greet() # Output: Hello Guest
```

c. Keyword Parameters

- You give the name of the parameter while calling.

Example:

```
def greet(name, message):  
    print(message, name)  
greet(message="Hi", name="Ravi")
```

d. Variable-length Parameters

- When the number of arguments is not fixed.
- *args – for many values
- **kwargs – for many key=value pairs

Example:

```
def show(*names):  
    for name in names:  
        print(name)  
show("Ram", "Shyam")
```

MLL-TEXT FILE

Opening a text file - It is done using the open() function.

File_object = open(r"File_Name","Access_Mode")

The file should exist in the same directory as the python program file else, the full address of the file should be written in place of the filename.

Text File open modes –

Mode	Description
r	Open a file for reading.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
r+	opens for reading and writing (cannot truncate a file)
w+	For writing and reading (can truncate a file)
a	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.

Closing a text file – This is done by using close() function.

File_object.close()

Opening a file using with clause – the syntax is -

with open(file_path, mode) as file:

Here - file_path is the path to the file to open, and

Mode is the mode of operation on the file. Eg. read, write etc.

Writing data to a text file –

For writing data into a file, the file must be opened in write mode. There are 2 methods for writing data into a file –

write(string) : It writes the given string to the file and return the number of characters written.

Syntax –

file_object.write(string)

writelines(list) : Using this function we can give list of lines to write into the file. Syntax –

file_object.writelines(list)

Appending data to a text file –

If we want to add new contents to an already existing file, then the file must be opened in append mode. Both the functions write() and writelines() can be used to add contents to the file.

Reading from a text file –

For reading the contents of a file, it must be opened in read mode. There are three ways to read data from a text file –

read() : Returns the read bytes in form of a string. Reads n bytes, if no n specified, reads the entire file.

Syntax – File_object.read([n])

readline() : Reads a line of the file and returns in form of a string. For specified n, reads at most n bytes. However, does not reads more than one line, even if n exceeds the length of the line. Syntax

File_object.readline([n])

readlines() : Reads all the lines and return them as each line a string element in a list. Syntax –

File_object.readlines()

Seek and tell methods –

The tell() method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

The seek(offset[, from]) method changes the current file position. The offset argument indicates the number

of bytes to be moved. The from argument specifies the reference position from where the bytes are to be moved. If from is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

Binary Files

Binary files store data after converting it into binary language (Os and 1s), there is no EOL (End Of Line) character. This file type returns bytes. This is the file to be used when dealing with non-text files such as images or exe.

To write data into a binary file, we need to **import Pickle** module. Pickling means converting structure (data types) into byte stream before writing the data into file. Pickle module has two main functions:

1. pickle.dump(): To write the Object into the file

Syntax: pickle.dump(object_to_write,file_object)

2. pickle.load(): To read the Object from the file

Syntax: container_obj=pickle.load(file_object)

Program to write record in a binary file:

```
import pickle
f=open("myfile. dat ","wb")
for i in range(5):
    r=input("Enter roll:")
    n=input("Enter name:")
    m=input("Enter marks:")
    d={"roll":r,"name":n,"marks":m}
    pickle.dump(d,f)
    print("successfully done!!")
```

f.close()

Program to append record in a binary file:

```
import pickle
f=open("myfile.dat","ab")
for i in range(5):I
```

```
    r=input("Enter roll:")
    n=input("Enter name:")
    m=input("Enter marks:")
    d={"roll":r,"name":n,"marks":m}
    pickle.dump(d,f)
    print("successfully done!!")
```

f.close()

Program to read all records from a binary file:

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

f.close()

Binary File Mode	Description
'rb'	Read-only
'wb'	Write only
'ab'	Append
'r+b' or 'rb+'	Read and write
'w+b' or 'wb+'	Write and read
'a+b' or 'ab+'	Write and read

Text File	Binary File
Stores information in ASCII or UNICODE character.	Stores the data in the same way as stored in the memory.
Each line is terminated by a special EOL (End Of Line) character.	There is no delimiter for a new line.
Stores only plain text in a file.	Stores different types of data like audio, image, text, etc. in a file.
Can be directly read.	Cannot be directly read.
Sequential Access	Random access

CSV

CSV file: A CSV (Comma Separated Values) file is a plain text file that stores tabular data in a simple format. Each line in the file represents a row, and columns are separated by commas.

Advantages of using CSV files:

- ❑ **Simple and Easy to Use**
 - CSV files are plain text files, which makes them easy to create, read, and edit using simple text editors like Notepad or tools like Excel.
- ❑ **Lightweight and Compact**
 - Since they are text-based, CSV files are smaller in size compared to formats like Excel (.xlsx), making them ideal for transferring and storing data.
- ❑ **Widely Supported**
 - Almost all programming languages (like Python, Java, C++) and applications (Excel, Google Sheets, databases) support reading and writing CSV files.
- ❑ **Simple and Easy to Use**
 - CSV files are plain text files, which makes them easy to create, read, and edit using simple text editors like Notepad or tools like Excel.
- ❑ **Lightweight and Compact**
 - Since they are text-based, CSV files are smaller in size compared to formats like Excel (.xlsx), making them ideal for transferring and storing data.
- ❑ **Widely Supported**
 - Almost all programming languages (like Python, Java, C++) and applications (Excel, Google Sheets, databases) support reading and writing CSV files.

Difference between CSV file and Text file:

A text file stores plain text with no structure, while a CSV file stores data in a tabular format using commas (or other delimiters) to separate values.

csv.reader() is used to read the file

Difference between **writerow()** and **writerows()**:

writerow() writes a single row to the file, while **writerows()** writes multiple rows (a list of lists).

Importing CSV Module:

Import csv

Reading from a CSV File:

```
with open("data.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

Writing to a CSV File:

```
with open("data.csv", "w", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(['RollNo', 'Name', 'Class', 'Marks']) # header
    writer.writerow(['104', 'Neha', '12', '85'])          # data
```

Python Exception Handling

What is Exception Handling?

Exception handling is a method used in programming to manage and respond to **runtime errors** (exceptions) without crashing the program.

Purpose:

- Prevent program crashes. Handle unexpected situations gracefully.
- Improve code robustness and user experience.

Common Errors Handled:

- Division by zero , File not found
- Invalid input , Out-of-range index

Basic Structure: try-except

Syntax:

```
try:
    # Code that may raise an exception
except ExceptionType:
    # Code to handle the exception
```

Explanation:

- **try block:** Code that *might* raise an error.
- **except block:** Code that runs *only if* an error of a specific type occurs.

Example 1: Handling Division by Zero

```
try:
    numerator = 10
    denominator = 0
    result = numerator / denominator # Will raise ZeroDivisionError
except ZeroDivisionError:
    print("Error: Division by zero.")
```

Output:

Error: Division by zero.

Example 2: Handling File Not Found

```
try:
    file = open("non_existent_file.txt", "r") # Will raise FileNotFoundError
except FileNotFoundError:
    print("Error: File not found.")
```

Output:

Error: File not found.

4. The finally Block

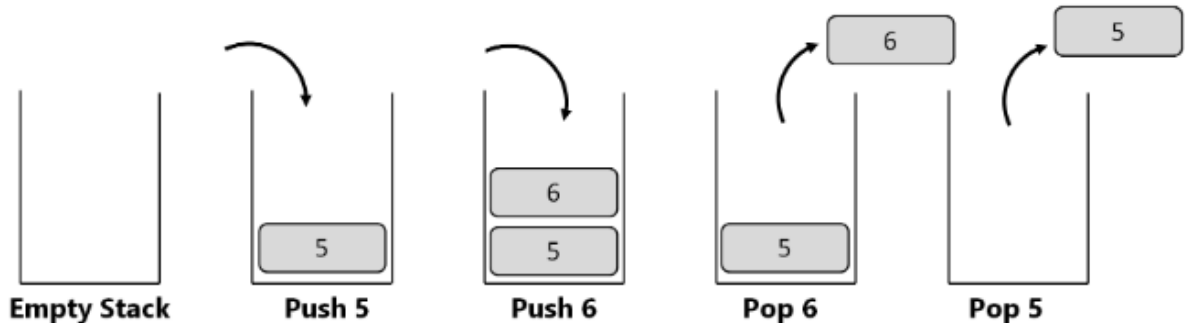
The finally block contains code that **always executes**, regardless of whether an exception occurred or not.

```
try:
    # Code that may raise exceptions
except ExceptionType:
    # Handle the exception
finally:
    # Code that always executes (e.g., cleanup)
```

Stack

A **stack** is a linear data structure that follows the **Last In First Out (LIFO)** principle, meaning the last element added is the first one to be removed. The primary operations for a stack are **push** (to add an element) and **pop** (to remove an element).

PUSH AND POP OPERATIONS IN STACK



Important Points about Stack:

- A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner.
- In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.
- Python's built-in data structure list can be used as a stack. List's `append()` function is used to add elements to the top of the stack while `pop()` removes the element in LIFO order.
- Also, underflow happens when we try to pop an item from an empty stack. Overflow happens when we try to push more items on a stack than it can hold.

Program to Demonstrate Push() and Pop() operations on Stack:

```
def ADDCustomer(Cust):
```

```
    n = int(input("Enter Customer no: "))
    nm = input("Enter Customer name: ")
    s = int(input("Enter salary: "))
    a = (n, nm, s)
    Cust.append(a)
    print("Customer added:", a)
    print("Current Stack:", Cust)
```

```
def DeleteCustomer(Cust):
```

```
    if not Cust:
        print("Stack Empty")
    else:
        v = Cust.pop()
        print("Element being deleted:", v)
        print("Current Stack:", Cust)
```

```
# Main program
Cust = []
while True:
    print("\n1. PUSH")
    print("2. POP")
    print("3. EXIT")
    ch = input("Choose any option: ")
    if ch == '1':
        ADDCustomer(Cust)
    elif ch == '2':
        DeleteCustomer(Cust)
    elif ch == '3':
        break
    else:
        print("Choose correct option")
```


Interface Python with MySQL

Interfacing Python with MySQL involves connecting a Python application to a MySQL database to perform operations such as data retrieval, insertion, updating, and deletion. This note will guide you through the process of setting up the interface, executing SQL queries, and handling results. We will use the `mysql-connector-python` library, which is a popular choice for interfacing Python with MySQL.

To install the `mysql-connector-python` library, run the following command:

```
pip install mysql-connector-python
```

Connecting to MySQL Database

The first step in interfacing Python with MySQL is to establish a connection to the database.

Establishing a Connection

To establish a connection, you need to import the `mysql.connector` module and use the `connect` method. Here's an example:

```
import mysql.connector
# Establishing the connection
conn = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="yourdatabase")
# Checking if the connection was successful
if conn.is_connected():
    print("Connected to MySQL database")
```

Closing the Connection

It is essential to close the connection after completing the database operations to free up resources.

```
# Closing the connection    conn.close()
```

Executing SQL Queries

Creating a Cursor Object

A cursor object allows you to execute SQL queries and fetch results.

```
cursor = conn.cursor()
```

Executing a Query

You can use the `execute` method to run SQL queries.

```
# Executing a query
cursor.execute("SELECT * FROM tablename")
```

Fetching Results

After executing a `SELECT` query, you can fetch the results using methods like `fetchall`, `fetchone`, or `fetchmany`.

```
# Fetching all rows
rows = cursor.fetchall()
```

```
for row in rows:
    print(row)
```

Inserting Data

To insert data into a table, you can use the **INSERT INTO** SQL statement.

Inserting data

```
cursor.execute("INSERT INTO students (name, age) VALUES ('John Doe', 22)")
```

Committing the transaction

```
conn.commit()
```

Updating Data

To update existing records, use the **UPDATE** SQL statement.

Updating data

```
cursor.execute("UPDATE students SET age = 23 WHERE name = 'John Doe'")
```

Committing the transaction

```
conn.commit()
```

Deleting Data

To delete records, use the **DELETE** SQL statement.

Deleting data

```
cursor.execute("DELETE FROM students WHERE name = 'John Doe'")
```

Committing the transaction

```
conn.commit()
```

COMPUTER NETWORK

Q1. Explain different types of computer network

Ans.

Type	Full Form	Distance	Media Used	Devices Used
PAN	Personal Area Network	30-40 ft (A Room)	Bluetooth, Infrared, Data Cable etc.	
LAN	Local Area Network	0-1 Km	Wifi, Twisted Wire Pair, Ethernet Cable	Switch/Hub
MAN	Metropolitan Area Network	1-15Km	Coaxial Cable, Microwaves	Repeaters
WAN	Wide Area Network	∞	Radio Waves, Optical Fiber, Satellite Communication	Gateways, Routers

Q2. What do you mean by a hub or a switch?

Ans. Hub: Act as a Central Device in Star Topology. It is a passive device

Switch: is networking hardware that connects devices on a computer network by using packet switching to receive and forward data to the destination device. Also Known as Intelligent Hub.

Q3. What do you mean by a router?

Ans. Router: A router is a networking device that forwards data packets between computer networks. i.e a router connects networks. Routers are intelligent devices, and they store information about the networks they're connected.

Q4. What do you mean by Network Topology?

Ans. Network Topology: The physical way in which computers are connected to each other in a network is called Network Topology.

Bus: A bus topology is a topology for a Local Area Network (LAN) in which all the nodes are connected to a single cable. The cable to which the nodes connect is called a "backbone". If the backbone is broken, the entire segment fails.

Ring: A ring topology is a network configuration where device connections create a circular data path. Each networked device is connected to two others, like points on a circle.

Star: A star topology is a topology for a Local Area Network (LAN) in which all nodes are individually connected to a central connection point, like a hub or a switch.

Q5. What is URL?

Ans. URL: Uniform Resource Locator : address of a given unique resource on the Web.

e.g. <http://www.cbse.nic.in/index.html>

Q5. What is VoIP?

Ans. **VoIP**: Voice over Internet Protocol: is a technology that allows you to make voice calls using an Internet connection instead of a regular (or analog) phone line.

Q6. What is the difference between a webpage and a website?

Ans.

Webpage	Website
Webpage is a single document on the Internet	Website is a collection of multiple webpages with information on a related topic
Each webpage has a unique URL.	Each website has a unique Domain Name

Q7. Write difference between static and dynamic webpage.

Ans. **Static Web Page**: A static web page (sometimes called a flat page or a stationary page) is a web page that is delivered to the user's web browser exactly as stored. i.e. static Web pages contain the same prebuilt content each time the page is loaded

Dynamic web page: The contents of Dynamic web page are constructed with the help of a program. They may change each time a user visit the page. Example a webpage showing score of a Live Cricket Match.

Q8. What do you mean by a web browser? Give Example.

Ans. **Web Browser**: A web browser (commonly referred to as a browser) is a software application for accessing information on the World Wide Web. e.g. Internet Explorer, Google Chrome, Mozilla Firefox, MS Edge, Brave, and Apple Safari.

Q9. Write Full forms of the following:

Ans. ARPANET	Advanced Research Project Agency Network
TCP/IP	Transmission Control Protocol / Internet Protocol
PPP	Point To Point Protocol
VoIP	Voice Over Internet Protocol
SLIP	Serial Link Internet Protocol
IMAP	Internet Message Access Protocol
POP	Post Office Protocol
PAN	Personal Area Network
LAN	Local Area Network
MAN	Metropolitan Area Network

WAN	Wide Area Network
MODEM	MODulator DEModulator
SIM	Subscriber Identification Module
Wi-Fi	Wireless Fidelity

Q10. Explain different types of networks.

Ans.

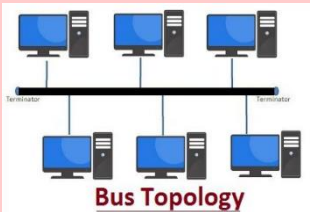
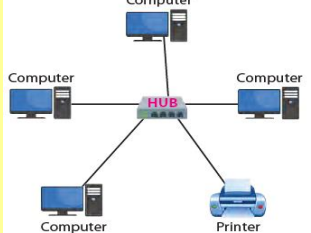
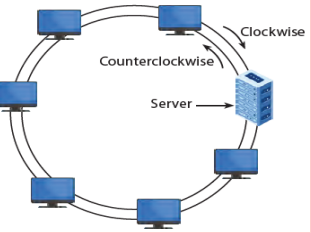
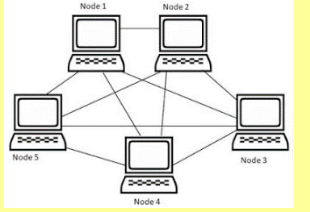
TYPES OF NETWORKS – Based on geographical area and data transfer rate

PAN (Personal Area Network) Interconnecting few personal devices like laptop, mobile etc. Area – 10 meters Bluetooth / USB	LAN (Local Area Network) Connects devices in limited area, say office, university campus etc. Area – upto 1 Km Ethernet Cable, Fibre Optics, Wi-Fi etc	MAN (Metropolitan Area Network) Extended form of LAN, within the city. Example – CableTV network in a town. Area – 30-40 Km	WAN (Wide Area Network) Connects devices, LANs and WANs across different parts of country or different countries or continents. Example – Internet
-----------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

Q11. Explain different kind of topologies.

Ans.

NETWORK TOPOLOGIES - pattern of layout or inter-connection between devices (computer nodes , printers etc.) in a network.

 <p>BUS topology Easy to setup ; Less cable length ; Fault diagnosis difficult; Not suitable for large networks</p>	 <p>STAR topology Centrally controlled ; Fault diagnosis easy; Expensive to setup; If central hub fails, network disrupts.</p>	 <p>RING topology Easy to setup ; Higher rate of data transmission; Troubleshooting difficult ;</p>	 <p>MESH topology Network can be expanded without affecting existing LAN ; Robust topology; Complex setup</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Q12. Explain different types of Transmission Media.

Ans.

TRANSMISSION MEDIA	
WIRED (Guided) Twisted Pair Cable (Ethernet Cable) Economical and Easy to use stp (shielded twisted pair) , utp (un- shielded twisted pair)	WIRELESS (Unguided) Infrared – Are electromagnetic radiation for line-of-sight; Frequency 300 GHz - 400 THz; Range 10-30 meters Bluetooth - standard wireless (radio wave) communication protocol uses 2.4 GHz frequency; max range 100 meter

Co-axial Cable Example = cable TV wire	Radio wave (frequency range 30 Hz – 300 GHz)
Optical Fiber Cable Most reliable, fast transmission, expensive	Satellite (downlink frequency 1.5 – 20 GHz) (Uplink frequency 1.6 GHz – 30 GHz) VERY FAST, EXPENSIVE
	Microwave (frequency range 300 MHz – 300 GHz)
All unguided media = transmitter, receiver and atmosphere	

Q13. Explain difference between Router and Bridge.

Ans. ROUTER: It connects multiple networks with different protocols and can handle multiple protocols and works using IP addresses

BRIDGE: connects local networks with same standard but having different types of cables and cannot manage multiple protocols and works using MAC addresses.

Q14. What is a repeater?

Ans. REPEATER is used to re-generate received signal and re-transmit towards destination.

TIP - When to suggest use of Repeater?

When distance between devices is more than 90 meter

Q15. Write difference between a switch and a hub.

Ans.

SWITCH	v/s	HUB
An intelligent device that connects several nodes for form a network.		An electronic device which connects several nodes to form a network.
Sends information only to intended nodes		Redirects the information to all the nodes in broadcast form.

Q16. Write tips for case study based QA.

Ans.

Tips for CASE STUDY BASED questions

Question	Hint for Answering
Layout	Draw block diagram interconnecting blocks, prefer the block or unit with maximum devices as main to connect other blocks
Topology	Write name of topology – Star / Bus / Ring etc.
Placement of Server	In the unit/block with maximum number of computers
Placement of Hub/Switch	In every block / unit
Placement of Repeater	As per layout diagram, if distance between two blocks is above 90 meter
Cost-effective medium for internet	Broadband / connection over telephone lines
Communication media for LAN	Ethernet (upto 100 meter) / Co-axial cable for high speed within LAN
Cost/Budget NOT an issue in LAN	Optical Fiber
Communication media for Hills	Radio wave / Microwave
Communication media for Desert	Radio wave
Very fast communication between two cities / countries	Satellite (avoid it in case economical / budget is mentioned)
Device / software to prevent unauthorized access	Firewall (Hardware and/or Software)

- Q17. Write difference between http and https.
 Ans. HTTP: Hyper Text Transfer Protocol- transfer data from one device to another on the world wide web. HTTP defines how the data is formatted and transmitted over the network.
 HTTPS: Hypertext Transfer Protocol Secure: advanced and secure version of HTTP.
- Q18. What is an email?
 Ans. e-Mail or email, short for "electronic mail," is the transmission of messages electronically over computer networks.
- Q19. What are cookies?
 Ans. Cookies are combination of data and short codes, which help in viewing a webpage properly in an easy and fast way. Cookies are downloaded into our system, when we first open a site using cookies and then they are stored in our computer only. Next time when we visit the website, instead of downloading the cookies, locally stored cookies are used. Though cookies are very helpful but they can be dangerous, if miss-utilized.
- Q20. What are Protocols?
 Ans. Protocols are set of rules, which governs a Network communication. Or set of rules that determine how data is transmitted between different devices in a network.

DATA BASE & SQL

Q.NO.	PARTICULARS
1.	What is Database?
2.	What is the full forms of SQL?
3.	Write names of two command of DDL & DML
4.	Find out DDL & DML Commands from the following: INSERT, DELETE, ALTER, DROP
5.	Write a query to display all records from the table.
6.	What is a primary key?
7.	Write one difference between DDL and DML.
8.	A _____ is a collection of records.
9.	In a database table, each column is called a _____.
10.	The clause used to filter rows in SQL is _____.
11.	The command used to remove duplicates in a SELECT query is _____.
12.	Which SQL command is used to retrieve data?
13.	Which clause is used to sort the records in SQL?
14.	What is the function of the PRIMARY KEY?
15.	The wildcard character % is used in SQL with which clause?
16.	Which command is used to show structure of the table named employee?
17.	Write the SQL command to display all records from a table named student

18.	Aman wants to remove the table Product from the database SHOP which command will he use from the following.																
19.	<p>Write MySQL statements for the following:</p> <p>i. To create a database named FOOD</p> <p>ii. To create a table named Nutrients based on the following specification:</p> <table><tr><th>Column Name</th><th>Data Type</th><th>Constraints</th></tr><tr><td>Food_Item</td><td>Varchar(20)</td><td>Primary Key</td></tr><tr><td>Calorie</td><td>Integer</td><td></td></tr></table>	Column Name	Data Type	Constraints	Food_Item	Varchar(20)	Primary Key	Calorie	Integer								
Column Name	Data Type	Constraints															
Food_Item	Varchar(20)	Primary Key															
Calorie	Integer																
20.	<p>. Consider the following table stored in a database SHOP:</p> <p style="text-align: center;">Table: Product</p> <table><tr><th>Pcode</th><th>Pname</th><th>Qty</th><th>Price</th></tr><tr><td>100</td><td>Tooth Paste</td><td>100</td><td>78.0</td></tr><tr><td>101</td><td>Soap</td><td>500</td><td>20</td></tr><tr><td>102</td><td>Talc Powder</td><td>50</td><td>45.0</td></tr></table> <p>(i) What is the degree and cardinality of the above table?</p> <p>(ii) Write a SQL command to add a new column supcode of char (20) size in the table.</p>	Pcode	Pname	Qty	Price	100	Tooth Paste	100	78.0	101	Soap	500	20	102	Talc Powder	50	45.0
Pcode	Pname	Qty	Price														
100	Tooth Paste	100	78.0														
101	Soap	500	20														
102	Talc Powder	50	45.0														

Q.No.	Answer
1.	A database is a collection of related data.
2.	Structured Query Language
3.	DDL Commands- CREATE, ALTER, DROP DML Commands: , INSERT, DELETE, UPDATE, SELECT
4.	DDL: ALTER, DROP DML: INSERT, DELETE
5.	SELECT * FROM table_name;
6.	A column or group of columns that uniquely identifies each row.
7.	DDL- Defines and modifies the structure of database objects like tables, schemas, indexes etc. i.e. create, alter, drop DML- it is used to manage data within existing tables. It Changes the content of the database. i.e. select, update, insert, delete
8.	Table
9.	Field / Column/ attribute/ data item
10.	Where
11.	Distinct
12.	Select
13.	Order by

14.	Primary key is used to identify each record uniquely in the table. or A primary key is a column (or a set of columns) in a database table that uniquely identifies each row in that table.
15.	LIKE
16.	Describe employee;
17.	SELECT * FROM student;
18.	b) Drop table Product
19.	i. create database FOOD ii. create table Nutrients (Food_Item varchar(20) primary key,
20.	(i) degree-4 Cardinality-3 (ii) alter table product add (supcode char(20));