# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Communication Medium and Network
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Communication Mediums: Students will learn about different types of communication media, including wired and wireless, and their roles in transmitting data.<br>2.Types of Transmission Media: Students will explore the distinctions between guided (wired) and unguided (wireless) media, such as fiber optics, coaxial cables, and radio waves.<br>3.Bandwidth and Data Transmission: Students will understand the | 1.Introduction to Communication Mediums: Begin with an overview of why communication mediums are essential in networking, discussing wired and wireless technologies in daily life.<br>2.Explaining Types of Transmission Media: Present an overview of guided vs. | 1.Media Identification Task: Provide samples or images of different media types, asking students to identify each and explain its advantages and disadvantages.<br>2.Bandwidth Calculation Exercise: Assign a simple exercise to | 1.Link to Physics: Relate the concepts of frequency, wavelength, and transmission in wireless communication to physics.<br>2.Life Skills: Enhance critical thinking, decision-making, and analytical skills as students compare and select communication media based on | 1.Physical Media Samples: Use samples or images of cables (like twisted pair and fiber optic) to give students a tangible understanding of wired media.<br>2.Projector for Demonstrations: Show diagrams and animations of data transmission processes in wired and wireless media. | 1.Oral Questioning: Ask questions like, “What is the difference between guided and unguided media?” and “How does bandwidth affect data transmission?”<br>2.Worksheet Exercises: Provide exercises on identifying media types, calculating | 1.Positive Reinforcement: Recognize students who accurately understand and apply knowledge of communication mediums, reinforcing correct selection and usage.<br>2.Additional Support: Offer simpler examples and step-by-step guidance for | 1.Encouraging Participation: Ensure all students engage in media identification and selection tasks, promoting inclusivity and collaboration.<br>2.Visual Support and Incremental Guidance: Use visuals, step-by-step examples, and hands-on |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| concept of bandwidth and how it impacts data transmission speed and quality.<br>4.Wired Communication Mediums: Students will examine various wired media types, including twisted pair cables, coaxial cables, and fiber optic cables, and their applications.<br>5.Wireless Communication Mediums: Students will study wireless media, such as radio waves, microwaves, and infrared, understanding where each is best applied.<br>6.Evaluating Mediums for Different Network Setups: Students will evaluate the pros and cons of each medium, learning to select the appropriate medium based on network requirements. | unguided media, giving examples of each and discussing their typical applications.<br>3.Hands-On Demonstration of Wired Mediums: Display examples or images of twisted pair cables, coaxial cables, and fiber optic cables, explaining their structures and uses.<br>4.Demonstrating Wireless Communication Media: Explain the working of wireless media using real-life examples like Wi-Fi and mobile networks, illustrating the principles behind radio and microwaves.<br>5.Bandwidth and Data Transmission | help students calculate and compare bandwidth requirements for different data transmission scenarios.<br>3.Wired vs. Wireless Evaluation Activity: Provide a scenario (e.g., setting up a network for a small business) and ask students to choose between wired and wireless options, justifying their choice.<br>4.Network Setup Project: Ask students to design a network layout for a specific environment (like a school or office), selecting appropriate communicatio | needs and limitations.<br>3.Values: Encourage responsibility and attention to detail, as proper selection of communication media is essential for secure and efficient networking. | 3.Worksheets: Provide worksheets with exercises on identifying media types, calculating bandwidth, and evaluating media based on network scenarios.<br>4.Reference Sheets: Create a quick reference sheet listing characteristics of various communication media with their advantages and disadvantages. | bandwidth, and choosing suitable media for given network scenarios.<br>3.Scenario-Based Assessment: Observe students as they choose and justify media types for hypothetical network setups, assessing their decision-making skills.<br>4.Quiz: Conduct a quiz covering the characteristics, applications, and advantages of various communicatio n media. | students who need help understanding media properties or bandwidth.<br>3.Clarification Sessions: Address common issues, such as confusion between media types or their applications, with further examples and explanations. | materials to assist students needing extra support with network media concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from basic media identification to complex media selection scenarios, accommodating diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Exercise: Use practical examples (e.g., comparing internet speeds) to explain bandwidth, discussing how different media types affect data rates and reliability.<br>6.Project-Based Learning with Network Design: Assign a project where students design a network setup, selecting communication mediums based on specific requirements like speed, distance, and cost. | n media and explaining their choices. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Computer Network and Devices
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Computer Networks: Students will learn the definition, purpose, and importance of computer networks in facilitating data communication.<br>2.Types of Networks: Students will explore various types of networks, including LAN (Local Area Network), MAN (Metropolitan Area Network), and WAN (Wide Area Network).<br>3.Network Topologies: Students will understand network topologies such as bus, star, ring, and mesh, along with their | 1.Introduction to Computer Networks: Begin with examples of everyday networks (like Wi-Fi and LAN in schools) to explain how networks connect computers and devices.<br>2.Explaining Types of Networks with Diagrams: Draw or display diagrams of LAN, MAN, and | 1.Network Diagram Creation Activity: Have students draw network diagrams for different setups (e.g., a school LAN, an office building network), choosing topologies and devices.<br>2.Device Function Identification Task: Assign tasks where | 1.Link to Communication and Electronics: Relate networking to the study of communication systems in electronics, emphasizing the role of data transmission in modern technology.<br>2.Life Skills: Develop analytical thinking, problem-solving, and | 1.Network Simulation Software: Use simulation software like Cisco Packet Tracer to demonstrate network setups and device roles in virtual environments.<br>2.Projector for Demonstrations: Display network topologies, device images, and diagrams to provide visual | 1.Oral Questioning: Ask questions like, “What is the difference between LAN and WAN?” and “What role does a router play in a network?”<br>2.Worksheet Exercises: Provide exercises on identifying types of networks, choosing appropriate | 1.Positive Reinforcement: Recognize students who accurately demonstrate network concepts, reinforcing correct understanding and application of networking.<br>2.Additional Support: Offer simpler explanations and guided examples for | 1.Encouraging Participation: Ensure all students engage in network design and analysis activities, promoting inclusivity and collaboration.<br>2.Visual Support and Incremental Guidance: Use visuals, step-by-step examples, and hands-on tasks |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| advantages and disadvantages.<br>4Network Devices: Students will learn about essential network devices, including routers, switches, hubs, repeaters, and gateways, and their roles in a network.<br>5.Data Transmission Modes: Students will explore modes of data transmission (simplex, half-duplex, and full-duplex) and how they affect communication.<br>6.Practical Applications of Networking: Students will understand the practical uses of computer networks in various fields, such as banking, education, and e-commerce. | WAN to visually represent their differences in size, scale, and application.<br>3.Network Topologies Demonstration: Use physical objects or visual aids to represent different topologies, explaining how data flows in each and discussing their pros and cons.<br>4.Demonstrating Network Devices: Show images or physical examples of network devices like routers, switches, and hubs, explaining their functions within a network.<br>5.Practical Examples of Data Transmission: Use simple examples (e.g., walkie-talkies for simplex, | students identify the role of each network device in a provided setup, reinforcing understanding of device functions.<br>3.Transmission Mode Activity: Provide scenarios for students to classify as simplex, half-duplex, or full-duplex, allowing them to apply transmission mode concepts.<br>4.Network Design Project: Ask students to design a network for a small business, specifying devices, topology, and transmission modes, practicing practical | organizational skills as students design and analyze network structures.<br>3.Values: Encourage teamwork, responsibility, and ethical use of networks, as networked communication requires careful management of data security and privacy. | support for complex concepts.<br>3.Worksheets: Provide worksheets with exercises on types of networks, device functions, and transmission modes.<br>4.Reference Sheets: Create a quick reference sheet listing network devices, topologies, and transmission modes with examples. | topologies, and determining the roles of various network devices.<br>3.Diagram Creation and Analysis: Observe students as they create network diagrams, assessing their understanding of topology and device selection.<br>4.Quiz: Conduct a quiz covering network types, topologies, device functions, and data transmission modes. | students needing help with complex concepts like network devices or topologies.<br>3.Clarification Sessions: Address common misunderstandings, such as confusion between device functions or topology selection, with extra examples and practice. | to assist students needing extra support with networking concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from basic network diagrams to complex network designs, accommodating diverse learning need |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | telephone for full-duplex) to explain data transmission modes in a relatable way.<br>6.Project-Based Learning with Networks: Assign a project where students create a network diagram for a hypothetical organization, selecting appropriate topology and devices. | application of concepts. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Computer Network and Protocols
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Network Protocols: Students will learn what network protocols are, why they are crucial for data transmission, and how they enable different devices to communicate effectively.<br>2.OSI Model and Protocol Layers: Students will understand the OSI (Open Systems Interconnection) model's seven layers, learning the role of each layer in managing data flow.<br>3.Common Network Protocols: Students will explore widely-used | 1.Introduction to Protocols with Analogies: Begin with simple analogies, such as following traffic rules, to explain how protocols standardize and manage data flow in networks.<br>2.Layered OSI Model Explanation: Use a diagram to visually explain the OSI | 1.OSI Model Layer Identification Activity: Provide a scenario where students identify each layer's role, such as sending an email, to reinforce the OSI model structure.<br>2.Protocol Application Task: Assign tasks where | 1.Link to Communication and Cybersecurity: Relate protocols to concepts in cybersecurity and data privacy, emphasizing the role of security protocols in protecting online data.<br>2.Life Skills: Develop analytical thinking, critical reasoning, and decision-making | 1.Projector for OSI Model and Protocol Diagrams: Use diagrams of the OSI model and flowcharts showing data transmission across protocols.<br>2.Simulation Software: Use software like Cisco Packet Tracer or Wireshark to simulate protocol functions and | 1.Oral Questioning: Ask questions like, "What is the function of the TCP/IP protocol?" and "How does the OSI model organize data transmission?"<br>2.Worksheet Exercises: Provide exercises on matching protocols with their functions, identifying OSI | 1.Positive Reinforcement: Recognize students who demonstrate strong understanding of network protocols and OSI layers, reinforcing correct protocol usage and knowledge.<br>2.Additional Support: Provide further examples and | 1.Encouraging Participation: Ensure all students engage in OSI model exercises, protocol matching, and network simulations, promoting inclusivity.<br>2.Visual Support and Incremental Guidance: Use step-by-step examples, visuals, and |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| network protocols, such as TCP/IP, HTTP, HTTPS, FTP, and SMTP, and understand their specific functions in network communication.<br>4.Security Protocols: Students will examine protocols like SSL/TLS and SSH, learning how they help secure data in transit.<br>5Application of Protocols in Real-Life Scenarios: Students will apply their knowledge by identifying and discussing protocols used in everyday activities, such as sending emails and browsing the internet. | model, discussing each layer's function and relating it to real-world scenarios like data encapsulation and transmission.<br>3.Demonstrating Common Protocols: Show examples of protocols in action, such as HTTP/HTTPS in web browsing and FTP for file transfers, to illustrate practical applications.<br>4.Security Protocols Discussion: Explain how security protocols like SSL and TLS protect data, using examples of secure websites (HTTPS) vs. non-secure sites (HTTP).<br>5.Project-Based Learning with Protocol | students match specific protocols with applications (e.g., HTTP with web browsing, SMTP with email), understanding each protocol's role.<br>3.Network Simulation Exercise: Use a network simulation tool like Cisco Packet Tracer to demonstrate how data travels through layers and protocols within a network setup.<br>4.Security Protocol Exploration: Assign groups to explore and present on specific security protocols (e.g., SSL/TLS), discussing how | skills as students examine protocols and their applications.<br>3.Values: Encourage responsible use of network resources, highlighting ethical considerations in secure data transmission. | data flow across networks.<br>3.Worksheets: Provide worksheets with exercises on protocol functions, OSI layer matching, and network communication scenarios.<br>4.Reference Seets: Create a quick reference sheet listing common protocols, their associated OSI layers, and their primary uses. | layers, and recognizing protocol roles in network setups.<br>3.Simulation-Based Assessment: Observe students in simulations, assessing their understanding of data flow and protocol applications.<br>4.Quiz: Conduct a quiz covering the OSI model, common protocols, and security protocols. | guided practice for students needing assistance with protocol concepts or the OSI model.<br>3.Clarification Sessions: Address common misconceptions, such as confusing protocol functions, with additional real-world examples and guided explanations. | simulations to support students needing extra help with protocol concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from basic protocol identification to complex OSI model applications, accommodating diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Identification: Assign students to research and identify protocols used in different applications, such as social media or online banking, reinforcing protocol knowledge in a real-world context. | they protect data and enhance security. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Data Structure - Stack
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Stack Data Structure: Students will learn the basic concept of stacks as a data structure that follows the Last-In-First-Out (LIFO) principle.<br>2.Stack Operations: Students will learn essential stack operations, including push (inserting an element), pop (removing the top element), and peek (viewing the top element).<br>3.Implementation of Stack: Students will understand how to | 1.Introduction to LIFO with Real-Life Examples: Begin by explaining the concept of LIFO using relatable examples, such as stacking plates, where the last plate placed is the first one removed.<br>2.Demonstration of Stack Operations: Use a simple Python program to | 1. Stack Implementation Activity: Have students implement a basic stack in Python, allowing them to practice adding and removing elements (push and pop).<br>2.Operation Practice Task: Provide tasks where students perform a | 1.Link to Mathematics and Logic: Reinforce logical thinking and mathematical concepts through stack operations and expression evaluation.<br>2.Life Skills: Develop problem-solving skills, analytical thinking, and persistence, as students work through stack-based problems. | 1.Link to Mathematics and Logic: Reinforce logical thinking and mathematical concepts through stack operations and expression evaluation.<br>2.Life Skills: Develop problem-solving skills, analytical thinking, and persistence, as students work through stack-based problems. | 1.Python IDE: Use an IDE like IDLE or Jupyter Notebook for hands-on programming with stack operations.<br>2.Projector for Demonstrations: Display examples of stack operations and applications to visually reinforce the concept of LIFO. | 1.Oral Questioning: Ask questions like, “What does LIFO mean?” and “How is a stack different from other data structures?”<br>2.Worksheet Exercises: Provide exercises on stack implementation, push/pop operations, and tracking | .Positive Reinforcement : Acknowledge students who accurately implement stack operations, reinforcing correct stack management.<br>2.Additional upport: Provide extra guidance on understanding the LIFO principle and the order of operations for |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| implement stacks using lists or arrays in Python.<br>4.Applications of Stacks: Students will explore real-world applications of stacks, such as undo functions in text editors, evaluating expressions, and managing function calls.<br>5Problem Solving with Stacks: Students will apply their understanding of stacks to solve programming problems, using stack operations to manage data effectively. | demonstrate push, pop, and peek operations on a stack, explaining each operation's function.<br>3.Implementing Stacks in Python: Guide students in creating a stack in Python using lists, showing how Python's append() and pop() methods can simulate stack operations.<br>4.Exploring Applications of Stacks: Discuss applications where stacks are used, such as in expression evaluation (postfix, prefix), checking balanced parentheses, and managing browser history.<br>5.Problem-Solving with Stack Exercises: Assign exercises where students use stack | series of stack operations and track the stack's state after each operation.<br>3.Application Challenge - Balancing Parentheses: Assign a task where students use a stack to check if an expression has balanced parentheses, applying the LIFO principle.<br>4.Evaluating Postfix Expressions: Guide students to write a program that uses a stack to evaluate postfix expressions, applying stack operations to solve real-world problems. | 3.Values: Encourage accuracy and discipline in managing data structures, as stack operations require attention to the order and correctness of data handling. | 3.Values: Encourage accuracy and discipline in managing data structures, as stack operations require attention to the order and correctness of data handling. | 3.Worksheets: Provide worksheets with stack operation exercises, including push, pop, and peek tasks.<br>4.Reference Sheets: Create a quick reference sheet listing stack operations, their applications, and example scenarios. | the stack's state.<br>3.Coding Assignments: Observe students as they implement and manipulate stacks in Python, assessing their understanding of stack behavior.<br>4.Quiz: Conduct a quiz covering stack concepts, LIFO, basic operations, and applications. | students needing help.<br>3.Clarification Sessions: Address common issues, such as incorrect usage of pop or tracking the stack's state, with guided examples. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | operations to solve problems, reinforcing the utility and behavior of stacks in various scenarios. | | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Exception Handling
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Exceptions: Students will learn what exceptions are, why they occur, and the difference between syntax errors and runtime errors.<br>2.Basic Exception Handling with Try-Except: Students will understand and apply the try and except blocks to catch and handle exceptions in Python.<br>3.Using Finally and Else Blocks: Students will learn how finally and else blocks work with exception handling to ensure code reliability. | 1.Introduction to Exceptions with Real-Life Analogies: Begin by explaining exceptions as unexpected events using relatable analogies, such as a sudden power outage while watching a movie.<br>2.Demonstration of Basic Try-Except Block: Write a simple program that divides two | 1.Try-Except Exercise: Have students write a program that reads an integer from the user, catching errors if a non-integer is entered.<br>2.Common Exception Identification Activity: Provide tasks where students deliberately cause | 1.Link to Problem Solving and Debugging: Relate exception handling to debugging in programming and engineering, emphasizing the importance of identifying and managing unexpected situations.<br>2.Life Skills: Enhance resilience and patience, as | 1.Python IDE: Use an IDE like IDLE or Jupyter Notebook for hands-on practice with exception handling in various scenarios.<br>2.Projector for Demonstrations: Display examples of exception handling blocks, including try, except, finally, and custom exceptions. | 1.Oral Questioning: Ask questions like, “What is the purpose of a try-except block?” and “When would you use a finally block?”<br>2.Worksheet Exercises: Provide exercises on catching exceptions, using finally blocks, and handling | .Positive Reinforcement: Recognize students who accurately apply exception handling techniques, reinforcing the importance of error management.<br>2.Additional Support: Offer additional guidance for students struggling with understanding | 1.Encouraging Participation: Ensure all students engage in exception handling exercises, promoting inclusivity in programming concepts.<br>2.Visual Support and Incremental Guidance: Use step-by-step examples and visuals to assist students |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4.Common Python Exceptions: Students will identify common Python exceptions like ZeroDivisionError, ValueError, and TypeError, learning how to handle them.<br>5.Creating Custom Exceptions: Students will explore how to create custom exceptions to handle unique error conditions in their programs.<br>6.Practical Application of Exception Handling: Students will use exception handling techniques in real-world scenarios, making their programs user-friendly and fault-tolerant. | numbers and use try-except to handle a ZeroDivisionError.<br>3.Exploring Finally and Else Blocks: Demonstrate the use of finally and else blocks by showing examples, such as closing a file after attempting to read it.<br>4.Identifying and Handling Common Exceptions: Guide students through examples involving common exceptions, explaining how each error occurs and can be handled effectively.<br>5.Custom Exception Handling: Show students how to create custom exceptions for specific error conditions in programs, | common exceptions (like ZeroDivisionError and TypeError) to understand their occurrence and how to handle them.<br>3.Using Finally Block: Assign an activity where students read from a file and ensure the file is closed with the finally block, even if an error occurs.<br>4.Custom Exception Creation: Ask students to create a program with a custom exception, such as checking for a negative number where only positive numbers are allowed. | managing errors requires a methodical approach and persistence.<br>3.Values: Foster responsibility and precision, as handling exceptions contributes to building reliable and safe programs. | 3.Worksheets: Provide worksheets with exercises on identifying, handling, and understanding exceptions.<br>4.Reference Sheets: Create a quick reference sheet listing common Python exceptions and handling techniques | specific exceptions.<br>3.Coding Assignments: Observe students as they implement exception handling in various programs, assessing their understanding of error management.<br>4.Quiz: Conduct a quiz covering exceptions, try-except structure, common Python exceptions, and custom exception handling. | the use of try-except or finally blocks, using simpler examples.<br>3.Clarification Sessions: Address common issues, such as incorrect exception handling syntax, with targeted examples and personalized guidance. | needing extra support with exception handling.<br>3.Differentiated Instruction: Provide a range of tasks, from basic try-except exercises to complex custom exception applications, to meet diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | explaining when and why custom exceptions are useful.<br>6.Project-Based Learning with Exception Handling: Assign a project that requires students to implement exception handling in a real-world application, such as a calculator or user input validation system. | | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Functions and Modules
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Functions: Students will review and deepen their understanding of functions, including function creation, calling, and parameter handling.<br>2.Scope and Lifetime of Variables: Students will explore variable scope (local and global) and understand how variables are managed in different parts of the program.<br>3.Returning Values from Functions: Students will practice writing functions with return values, enhancing the | 1.Introduction to Functions and Scope: Start by demonstrating a simple function and explaining parameter passing, local vs. global variables, and variable lifetime within a function.<br>2.Hands-On Practice with Function Return Values: Show students how to create functions that return | 1.Function Definition and Call Exercise: Assign exercises where students create functions for simple tasks, like converting temperatures or calculating the area of shapes.<br>2.Scope and Lifetime Practice: Provide tasks where | 1.Link to Mathematics: Reinforce mathematical concepts through functions and modules, such as calculating areas, using mathematical operations, or performing statistical calculations.<br>2.Life Skills: Develop logical thinking, problem-solving, and | 1.Python IDE: Use an IDE like IDLE or Jupyter Notebook for hands-on programming with functions and modules.<br>2.Projector for Demonstrations: Display examples of function definitions, imports, and module usage to visually reinforce modular | 1.Oral Questioning: Ask questions like, “What is the purpose of a module?” and “How do you pass arguments to a function?”<br>2.Worksheet Exercises: Provide exercises on defining functions, using variable scope, and creating modules. | 1.Positive Reinforcement: Recognize students who actively participate in function and module exercises, reinforcing correct use of modular programming concepts.<br>2.Additional Support: Offer one-on-one guidance for students needing help | 1.Encouraging Participation: Ensure all students engage in function and module activities, encouraging exploration of modular programming.<br>2.Visual Support and Incremental Guidance: Use step-by-step examples and visuals to assist students |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| functionality of their programs.<br>4.Modular Programming with Functions and Modules: Students will understand how to use modules to organize code into reusable units and import specific functionalities as needed.<br>5.Importing Modules and Using Built-In Libraries: Students will learn how to import custom and standard library modules, practicing the use of commonly used libraries like math, random, and datetime.<br>6.Real-Life Applications of Functions and Modules: Students will apply functions and modules to create organized and maintainable code for real-world scenarios. | values, guiding them through examples like calculating areas or summing numbers.<br>3.Creating Custom Modules: Demonstrate how to create a custom module with a few utility functions (e.g., basic arithmetic operations) and then import and use this module in another script.<br>4.Exploring Standard Library Modules: Introduce students to importing and using Python's standard library modules (e.g., math, random). Demonstrate practical applications such as generating random numbers or | students experiment with local and global variables, observing how they behave within and outside functions.<br>3.Module Creation Activity: Ask students to create a custom module, define a few functions within it, and import it into another script to use the functions.<br>4.Using Standard Library Modules: Assign tasks that require the use of standard libraries, like generating random numbers (random), finding maximum and minimum | organizational skills through modular programming.<br>3.Values: Encourage discipline, precision, and collaboration, as modular programming promotes organized code and teamwork. | programming concepts.<br>3.Worksheets: Provide worksheets with exercises on defining functions, understanding scope, and creating and using modules.<br>4.Reference Sheets: Create a quick reference sheet listing standard library modules, their functions, and examples. | 3.Coding Assignments: Observe students as they complete programming tasks with functions and modules, assessing their ability to create organized code.<br>4.Quiz: Conduct a quiz covering concepts of functions, scope, return values, modules, and standard library usage. | with syntax or scope concepts, using simpler examples for clarification.<br>3.Clarification Sessions: Address common errors, such as incorrect imports or variable scope issues, with further examples and guided practice. | needing extra support with modular concepts.<br>3.Differentiated Instruction: Provide a range of activities, from basic function definition to complex module applications, to meet diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | calculating square roots.<br>5.Project-Based Learning with Functions and Modules: Assign a project that requires students to break down tasks into functions and organize related functions into modules, emphasizing modular programming principles. | values (max, min), or performing mathematical calculations (math).<br>5.Real-World Problem Solving with Modular Design: Have students work on a project where they use functions and modules to solve a problem, such as organizing a student database or creating a simple calculator. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Interface of Python and MySQL
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Python-MySQL Connectivity: Students will learn how Python can connect to MySQL, providing an interface to manage databases directly from a Python script.<br>2.Installing MySQL Connector: Students will understand the process of installing and importing MySQL Connector in Python for database connectivity.<br>3.Executing SQL Commands in Python: Students will learn to execute SQL commands (such as SELECT, INSERT, | 1.Introduction to Database Connectivity: Begin with an explanation of why Python-MySQL integration is useful, discussing real-world applications like data analysis, web applications, and report generation.<br>2.Installation of MySQL Connector: | 1Database Connection Activity: Have students write a Python script to connect to a MySQL database and print a success message upon connection.<br>2.Executing Queries with Python: Assign tasks where students use Python to create a table, insert sample data, and | 1.Link to Data Science and Analytics: Emphasize the importance of database connectivity in data science, where Python retrieves and processes data for analysis and visualization.<br>2.Life Skills: Promote logical thinking, problem-solving, and attention to detail, as | 1.Python IDE: Use an IDE like IDLE or Jupyter Notebook to write and execute Python scripts with MySQL database connectivity.<br>2.MySQL Server: Ensure MySQL Server is installed and accessible for hands-on database operations and queries. | 1.Oral Questioning: Ask questions like, “What is the purpose of a MySQL connector in Python?” and “How do you execute an SQL query in Python?”<br>2.Worksheet Exercises: Provide exercises on setting up database connections, performing | 1.Positive Reinforcement: Recognize students who effectively implement Python-MySQL connectivity, reinforcing correct practices in database management.<br>2.Additional Support: Provide simpler examples and guidance for students | 1.Encouraging Participation: Ensure all students engage in database connectivity tasks, promoting inclusivity in data management and programming skills<br>2.Visual Support and Incremental Guidance: Use flowcharts and |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| UPDATE, and DELETE) within Python using cursor objects.<br>4.Fetching Data from MySQL into Python: Students will practice retrieving data from MySQL tables and displaying it in Python, making use of functions like fetchone() and fetchall().<br>5.Error Handling in Python-MySQL Interface: Students will learn to manage errors during database operations, making their scripts more reliable.<br>6.Applications of Python-MySQL Interface: Students will explore practical uses of Python-MySQL connectivity, such as creating and managing a simple data-driven application. | Demonstrate the installation of MySQL Connector for Python and explain the setup process, including basic troubleshooting.<br>3.Connecting Python to MySQL: Guide students through writing Python code to connect to a MySQL database, demonstrating how to handle connection success and failure.<br>4.Executing SQL Queries in Python: Use examples to show how to create tables, insert data, and update records using Python's execute() function within a database connection.<br>5.Data Retrieval and Display: Demonstrate | update specific records in the database.<br>3.Data Retrieval Exercise: Provide tasks where students retrieve data from a MySQL table and display it in Python, practicing with fetchone() and fetchall() methods.<br>4.Error Handling Practice: Assign students to handle errors (such as connection errors or SQL syntax errors) in their Python scripts, using try-except blocks.<br>5.Mini Project: Encourage students to develop a small application (e.g., a library | database interaction requires a structured approach to managing data.<br>3.Values: Encourage data accuracy, responsibility, and ethical use of data as students interact with real data storage systems. | 3.Projector for Demonstrations: Display live demonstrations of Python-MySQL interface setup, SQL execution, and data retrieval in Python.<br>4.Worksheets: Provide worksheets with structured activities on database connection, query execution, and data fetching.<br>5.Reference Sheets: Create a quick reference sheet listing common commands for Python-MySQL connectivity, SQL operations, and cursor methods | SQL operations in Python, and retrieving data from MySQL tables.<br>3.Coding Assignments: Observe students as they complete tasks involving Python-MySQL integration, assessing their understanding of connectivity and SQL command execution.<br>4.Quiz: Conduct a quiz covering key concepts of Python-MySQL connectivity, including connector setup, executing queries, and handling errors. | needing help with the connector setup or SQL execution syntax.<br>3.Clarification Sessions: Address common issues, such as connection failures or query errors, with further explanations and examples. | step-by-step examples to assist students needing extra support with Python-MySQL integration.<br>3.Differentiated Instruction: Provide a range of tasks, from basic connectivity setup to complex data manipulation, accommodating diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | retrieving data using SQL queries, and explain how to display results in Python using cursor functions like fetchone() and fetchall().<br>6.Project-Based Learning with Database Interface: Assign a project where students create a simple Python program that connects to a MySQL database to perform CRUD (Create, Read, Update, Delete) operations. | management system) that uses Python to interact with a MySQL database, performing various operations. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Introduction to Database
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding Database Basics: Students will learn the definition and purpose of a database and why it is important for organizing large amounts of data.<br>2.Familiarity with DBMS and RDBMS: Students will understand the concept of a Database Management System (DBMS) and how a Relational Database Management System (RDBMS) organizes data in tables.<br>3.Data Models and Schemas: Students will explore different data | 1.Introduction to Database Concepts: Begin with real-life examples of databases (such as a library or inventory) to explain how databases store and organize data systematically.<br>2.DBMS and RDBMS Demonstration: Use a simple database management software like | 1.Database Schema Creation: Have students design a schema for a simple database, such as a book library, defining tables and fields.<br>2.SQL Command Practice: Assign exercises where students practice basic | Link to Mathematics and Data Analysis: Show how relational databases organize numerical and textual data, making it easier to analyze and interpret information.<br>2.Life Skills: Foster critical thinking, problem-solving, and organizational skills as | 1.Database Software: Use database software like MySQL or SQLite for practical demonstrations of database and SQL usage.<br>2.Projector for Demonstrations: Display examples of database schemas, SQL commands, and relational structures to | 1.Oral Questioning: Ask questions like, “What is a primary key?” and “Why are foreign keys important in relational databases?”<br>2.Worksheet Exercises: Provide exercises on creating simple database schemas, identifying keys, and | 1.Positive Reinforcement: Recognize students who accurately create database structures and use SQL commands, reinforcing correct database management practices.<br>2.Additional Support: Offer additional guidance for students who | 1.Encouraging Participation: Ensure all students engage in schema creation, SQL commands, and database structure tasks, fostering inclusivity in learning database management.<br>2.Visual Support and Incremental Guidance: Use |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| models, with a focus on relational databases and schemas, which define how data is structured.<br>4.Introduction to SQL: Students will be introduced to SQL (Structured Query Language) as a tool to interact with databases, learning basic commands like SELECT, INSERT, UPDATE, and DELETE.<br>5.Keys and Constraints: Students will learn about primary keys, foreign keys, and constraints, which ensure data integrity and establish relationships between tables.<br>6.Practical Applications of Databases: Students will explore practical uses of databases in real-world applications, such as managing school records, e-commerce data, and library systems. | MySQL or SQLite to show students how a DBMS and RDBMS work.<br>3.Explaining Data Models and Schemas: Draw a basic relational schema on the board, such as a student database with tables for Students, Courses, and Enrollments, to illustrate relational structure.<br>4.Hands-On SQL Commands: Guide students through basic SQL commands, such as SELECT to retrieve data, INSERT to add new data, and DELETE to remove data from tables.<br>5.Keys and Constraints Practice: Show students how primary keys uniquely identify rows in | SQL commands to manipulate data within a table.<br>3.Identifying Keys and Constraints: Provide tasks where students identify primary keys and set up foreign key relationships between tables.<br>4.Database Project: Ask students to work on a project that requires designing a database and applying SQL commands to manage data, reinforcing the application of database concepts. | students learn to design and interact with structured data systems.<br>3.Values: Encourage responsibility and ethical use of data, emphasizing the importance of data integrity, privacy, and secure management. | visually support the lesson.<br>3.Worksheets: Provide worksheets with exercises on database terminology, SQL commands, and schema design.<br>4.Reference Sheets: Create a quick reference sheet listing basic SQL commands and database terminology. | using basic SQL commands.<br>3.Coding Assignments: Observe students as they create tables and perform SQL operations, assessing their understanding of database management.<br>4.Quiz: Conduct a quiz covering database basics, SQL commands, keys, constraints, and database applications | need help with relational schemas or SQL syntax, using simpler examples to clarify concepts.<br>3.Clarification Sessions: Address common issues, such as understanding relationships between tables, with further examples and personalized guidance. | diagrams and step-by-step instructions to assist students needing extra support with database concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from basic SQL queries to designing relational schemas, accommodating diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | a table and how foreign keys establish relationships between tables. 6.Project-Based Learning with Databases: Assign a project where students design a simple database (e.g., a school management system) and use SQL commands to add and retrieve data. | | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : MySQL - Single Table and Double Table
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding MySQL Basics: Students will review MySQL fundamentals and learn the purpose of using a database to store and retrieve data systematically.<br>2.Creating and Managing Single Tables: Students will learn how to create, modify, and delete single tables in MySQL, understanding data types, primary keys, and constraints.<br>3.Querying Single Tables: Students will practice basic SQL commands for retrieving data from a | 1.Introduction to Single and Double Tables with Real-World Examples: Start by discussing real-life scenarios (such as managing a library) to explain the need for relational tables in databases.<br>2.Single Table Creation and Management Demonstration: Guide students through | 1.Single Table Creation Activity: Have students create a single table with different fields (such as a Books table) and practice adding, updating, and deleting records.<br>2.Single Table Querying Exercise: Assign tasks where students | 1.Link to Mathematics and Data Analysis: Emphasize relational concepts like one-to-many relationships and data organization, relevant to both math and data analysis.<br>2.Life Skills: Promote analytical thinking, organization, and problem- | 1.MySQL Database Software: Use MySQL or an equivalent database tool for practical exercises on creating, managing, and querying tables.<br>2.Projector for Demonstrations: Display live demonstrations of SQL queries and join operations for better comprehension. | 1.Oral Questioning: Ask questions like, “What is a primary key?” and “How do you retrieve data from two related tables?”<br>2.Worksheet Exercises: Provide exercises on creating tables, setting up relationships, and using SQL commands for | 1.Positive Reinforcement: Recognize students who accurately create relational tables and use SQL commands, reinforcing best practices in database management.<br>2.Additional Support: Provide simpler examples and guidance for | 1.Encouraging Participation: Ensure all students engage in table creation, SQL operations, and join exercises, promoting inclusivity.<br>2.Visual Support and Incremental Guidance: Use step-by-step examples and diagrams to assist students |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| single table, including SELECT, WHERE, ORDER BY, and GROUP BY.<br>4.Double (Multiple) Table Operations: Students will explore how to create and manage relationships between tables using foreign keys and how to perform operations across two related tables.<br>5.Joining Tables in MySQL: Students will learn about different types of joins (INNER JOIN, LEFT JOIN, RIGHT JOIN) and how to use them to retrieve data from multiple tables.<br>6.Practical Applications of Relational Databases: Students will understand the practical use of single and double tables for storing complex data, such as in an inventory or school management system. | creating a single table in MySQL, explaining columns, data types, and constraints.<br>3.Querying Single Tables: Demonstrate how to retrieve specific data using SQL commands (SELECT, WHERE), filtering results based on conditions, and sorting data.<br>4.Double Table and Foreign Key Demonstration: Explain the concept of primary and foreign keys by creating two related tables (e.g., Students and Courses) and showing how data in one table connects to another.<br>5.Joining Tables in MySQL: Show examples of joining tables using different join types, | retrieve specific data using SQL queries, such as finding all books published in a certain year or sorting records alphabetically.<br>3.Double Table Relationship Setup: Ask students to create two related tables (e.g., Students and Subjects), establishing a relationship using foreign keys.<br>4.Join Operation Practice: Provide exercises that require using INNER JOIN, LEFT JOIN, and RIGHT JOIN to retrieve combined data from two tables.<br>5.Database Project: Encourage | solving skills as students design and manage relational data structures.<br>3.Values: Encourage responsibility and accuracy in data management, emphasizing the importance of reliable data handling in real-world applications. | 3.Worksheets: Provide worksheets with SQL exercises on single table queries, foreign key relationships, and join operations.<br>4.Reference Sheets: Create a quick reference sheet listing SQL commands for managing single and double tables, with examples. | querying single and double tables.<br>3.Coding Assignments: Observe students as they perform SQL operations on single and double tables, assessing their understanding of relational concepts.<br>4.Quiz: Conduct a quiz covering concepts of primary and foreign keys, join types, and basic SQL operations for managing tables. | students who struggle with foreign key concepts or SQL syntax.<br>3.Clarification Sessions: Address common issues, such as incorrect joins or data retrieval errors, with further examples and guided practice. | needing additional support with relational database concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from basic table creation to complex join operations, to accommodate diverse learning needs |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | explaining when to use each type for retrieving relevant data from multiple tables.<br>6.Project-Based Learning with Relational Databases: Assign a project where students create and manage a small relational database, applying both single and double table concepts. | students to design a small database with at least two tables, applying what they've learned about single and double tables to manage complex data relationships. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Review of Class 11 Python
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Recall Python Basics: Students will recall core Python concepts, including syntax, data types, and basic operations.<br>2.Mastering Control Structures: Students will reinforce their understanding of control structures (if-else, for and while loops) and apply them to solve programming problems.<br>3.Functions and Scope: Students will review functions, including function definition, parameter passing, | 1.Concept Recap with Examples: Begin each topic recap with an example program to remind students of syntax and basic usage (e.g., variables, operators).<br>2.Control Structure Exercises: Present programs with if, if-else, and loop constructs, demonstrating practical | 1.Quick Recap Exercises: Assign short exercises on core concepts, such as variable assignments, arithmetic operations, and control structure syntax.<br>2.Looping and Decision-Making Tasks: Have students solve problems involving for | 1.Link to Mathematics and Logic: Reinforce logical thinking and problem-solving skills relevant to mathematics through loops and decision-making structures.<br>2.Life Skills: Enhance analytical thinking, patience, and precision, as programming requires logical | 1 Python IDE: Use an IDE like IDLE or Jupyter Notebook for hands-on practice with all concepts, including control structures, functions, data structures, and file handling.<br>2.Projector for Demonstrations: Display code examples and outputs, showing how | 1.Oral Questioning: Ask questions like, “What is the difference between a list and a tuple?” and “How do you read a file in Python?”<br>2.Worksheet Exercises: Provide exercises covering all review topics, including control structures, functions, and | 1.Positive Reinforcement: Recognize students actively participating in review exercises, reinforcing accurate application of Python concepts.<br>2.Additional Support: Offer extra examples for students needing help with specific | 1.Encouraging Participation: Ensure all students engage in review tasks, fostering an inclusive environment for reinforcing foundational skills.<br>2.Visual Support and Incremental Guidance: Use visuals and step-by-step examples for students |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| return values, and scope of variables.<br>4.Data Structures in Python: Students will revisit data structures like lists, tuples, and dictionaries and practice operations on these structures.<br>5.File Handling Basics: Students will refresh their knowledge of file handling, including reading from and writing to files in Python.<br>6.Practical Application: Students will use review concepts to solve real-world problems, applying combined knowledge of functions, control structures, and data structures. | applications, like validating input or iterating over a range.<br>3.Functions Refresher with Hands-On Practice: Guide students in defining and calling functions, using return statements, and understanding scope through examples.<br>4.Data Structure Manipulation: Review lists, tuples, and dictionaries by performing common operations, like adding, removing, and accessing elements.<br>5.File Handling Demonstration: Demonstrate basic file operations, such as opening, reading, writing, and closing files, to refresh students on | and while loops and conditional statements to build decision-making skills.<br>3.Function Definition and Usage: Provide tasks where students define functions to perform tasks (e.g., calculating factorials or checking prime numbers) and call these functions within their programs.<br>4.Data Structure Challenges: Ask students to create and manipulate lists, tuples, and dictionaries with exercises like organizing a list of names, creating key-value pairs, | flow and attention to detail.<br>3.Values: Encourage self-discipline and persistence, as students revisit foundational skills, aiming to solidify their understanding and problem-solving abilities. | each topic builds upon Class 11 Python foundations.<br>3.Worksheets: Provide worksheets with review exercises on key topics, like control structures, functions, and file handling.<br>4.Reference Sheets: Create a quick reference sheet summarizing syntax for loops, functions, and file operations. | data structures.<br>3.Coding Assignments: Observe students as they complete coding tasks, assessing their ability to integrate multiple topics.<br>4.Quiz: Conduct a quiz that covers Python basics, control structures, functions, data structures, and file handling. | topics, such as nested loops or file handling syntax.<br>3.Clarification Sessions: Address common errors, such as syntax mistakes or scope issues, with further examples and personalized guidance. | needing additional support on complex concepts.<br>3.Differentiated Instruction: Provide a range of tasks, from simple to complex, to cater to diverse learning levels within the class. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | data storage and retrieval.<br>6.Practical Coding Tasks: Assign a project that combines multiple concepts from the review, such as building a simple calculator, managing data with lists and dictionaries, or creating a student record system with file storage. | and retrieving specific data.<br>5.File Handling Practice: Assign a task where students create a file, write data to it, and read back the data, applying file handling skills in practical scenarios. | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal

# LESSON PLAN

General Information: Date:

1. Name and Designation of the Teacher :
2. Class & Section : 12
3. Subject : COMPUTER SCIENCE
4. Number of Enrolled Students :
5. Name of the Lesson : Text, Binary, and CSV Files
6. No. of Periods required :
7. Date of Commencement :
8. Estimated Time Period from : to
9. Actual date of completion :

| Specific Learning Outcomes | Pedagogical Strategies for Experiential Learning | Individual/ Group Activities / Experiments / Hands-On Learning | Interdisciplinary Linkages and Infusion of Life Skills, Values | Resources (Including ICT) | Competency-Based Assessment Items for Measuring the Attainment of Learning Outcomes | Feedback and Remedial Teaching Plan | Inclusive Practices / Gender Sensitivity |
|---|---|---|---|---|---|---|---|
| 1.Understanding File Handling Concepts: Students will understand what file handling is and why it is important for storing, retrieving, and processing data.<br>2.Text Files in Python: Students will learn how to read, write, and manipulate text files, understanding the use of different file modes (r, w, a, etc.).<br>3.Binary Files in Python: Students will explore binary file handling, learning how binary files differ from text files and how they store data. | 1Introduction to File Types: Begin with an overview of file types (text, binary, and CSV) and discuss real-life applications, like text files for storing logs and CSV files for tabular data.<br>2.Text File Handling Demonstration: Show examples of opening, reading, writing, and appending text files, and | 1.Text File Operations Exercise: Have students create a text file, write data to it, append more data, and then read the content to practice text file handling.<br>2.Binary File Task: Assign a task where students read and write binary data, such as saving numbers or | 1.Link to Data Management and Analysis: Demonstrate how CSV files help in organizing and analyzing tabular data, making it easier to interpret.<br>2.Life Skills: Develop organizational and analytical thinking, as file handling requires structured data management | 1.Python IDE: Use an IDE like IDLE or Jupyter Notebook for hands-on programming practice with text, binary, and CSV file handling.<br>2.Projector for Demonstrations: Display file handling examples, including file operations in text, binary, and CSV formats. | 1.Oral Questioning: Ask questions like, “What is the difference between text and binary files?” and “Why do we use CSV files?”<br>2.Worksheet Exercises: Provide exercises on basic file operations, such as creating, reading, | 1.Positive Reinforcement: Recognize students who accurately implement file handling techniques, reinforcing best practices in data management.<br>2.Additional Support: Offer extra examples and guidance for students who need help understanding | 1.Encouraging Participation: Ensure all students engage in file handling exercises, promoting inclusivity in learning data management skills.<br>2.Visual Support and Incremental Guidance: Use visuals and step-by-step examples to assist students |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4.CSV Files in Python: Students will learn about CSV (Comma-Separated Values) files, understanding their structure and use for storing tabular data.<br>5.Applying File Handling in Real-World Scenarios: Students will practice using file handling in practical applications, such as storing records, exporting data, and managing files. | explain the importance of closing files after operations.<br>3.Binary File Handling Demonstration: Explain binary files by showing how data like images and executables are stored, demonstrating reading and writing binary data using Python.<br>4.CSV File Handling with Python's CSV Module: Use Python's csv module to demonstrate reading from and writing to CSV files, emphasizing CSV's utility for data exchange in spreadsheets.<br>5.Practical File Handling Exercises: Assign exercises where students use text, binary, and CSV files, | object data, to experience binary file handling.<br>3.CSV File Handling Activity: Provide a sample CSV file and ask students to read its contents, modify data, and save the updated file, practicing the use of the csv module.<br>4.Real-World Application Project: Ask students to create a small project that stores and retrieves data, such as an inventory management system or a contact book using file handling. | and attention to file operations.<br>3.Values: Encourage responsibility and discipline, as proper file handling is crucial for managing data securely and preventing data loss. | 3.Worksheets: Provide worksheets with exercises on reading, writing, and managing different types of files.<br>4.Reference Sheets: Create a quick reference sheet listing file operations, modes, and commands for each file type. | writing, and closing files.<br>3.Coding Assignments: Observe students as they complete file handling tasks, assessing their understanding of text, binary, and CSV operations.<br>4.Quiz: Conduct a quiz covering concepts of file types, file modes, and common file operations in Python. | file operations or CSV handling.<br>3.Clarification Sessions: Address common issues, such as file mode confusion or CSV formatting errors, with further examples and practice. | needing extra support with file handling concepts.<br>3.Differentiated Instruction: Provide a range of activities, from basic file operations to complex data handling applications, to accommodate diverse learning needs. |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | applying file handling techniques in simple applications, like managing user data or logs. | | | | | | |

Comments / Suggestions on Lesson Plan

Comments / Suggestions on Lesson Plan

Signature of the Teacher

VP/HM

Signature of the Principal