Module 1: Introduction to Object-Oriented Programming and GUI (16 Hrs)

Prerequisites for Module 1:

Introduction to Object-Oriented Programming, Programming Fundamentals: Basic understanding of programming concepts (variables, loops, conditionals). Familiarity with a programming language such as Python, Java etc.

Object-Oriented Concepts: Understanding of basic concepts like functions, data types, and control structures. Awareness of the importance of code organization and reusability.

Recommended Reading/Preparation: Introduction to Python programming language (if Python will be used in the course).

Online tutorials or introductory books on Object-Oriented Programming.

Lecture Number	Topic	Pre-Reading Material	Subtopics	Post-Reading Material	Learning Methodology/Activity	Learning Outcomes
Lecture 1	Basics of OOP	Intro to Programming Paradigms	- OOP Overview - Key Concepts: Classes, Objects	OOP Principles Review	Lecture, Discussions	- Understand the concept of OOP and its significance in software development
Lecture 2	Classes and Objects	Basic Python Concepts	- Class Definition - Creating Objects	Code Samples, Exercises	Define classes and create objects, grasp instance variables and methods	- Define classes and create objects, grasp instance variables and methods
Lecture 3	Instance Variables and Methods	Understanding Instances	- Instance Variables - Instance Methods	Practice Problems	Implement instance variables and methods in classes, differentiate their usage	- Implement instance variables and methods in classes, differentiate their usage
Lecture 4	Constructors and Destructors	Initializing Objects	- Constructor Methods - Destructor Methods	Coding Tasks, Case Studies	Define constructors and destructors in Python classes, manage object initialization and deletion	- Define constructors and destructors in Python classes, manage object initialization and deletion
Lecture 5	Class Attributes and Methods	Exploring Class-level Elements	- Class Attributes - Class Methods	Problem-Solving Tasks, Code Development	Implement class attributes and methods, understand their scope and usage	- Implement class attributes and methods, understand their scope and usage
Lecture 6	Inheritance: Creating Subclasses	Extending Class Functionality	- Subclasses and Superclasses	Coding Exercises, Hands-on Activities	Implement inheritance, create subclasses, and use superclass functionalities	- Implement inheritance, create subclasses, and use superclass functionalities
Lecture 7	Overriding Methods in Subclasses	Modifying Inherited Behavior	- Method Overriding	Code Samples, Practice Problems	Understand method overriding in subclasses, modify inherited methods	- Understand method overriding in subclasses, modify inherited methods
Lecture 8	Method Resolution Order (MRO)	Resolving Inherited Methods	- Understanding MRO	Case Studies, Discussions	Explore Method Resolution Order (MRO) in Python, comprehend inheritance hierarchy	- Explore Method Resolution Order (MRO) in Python, comprehend inheritance hierarchy
Lecture 9	Polymorphism and Its Use Cases	Utilizing Polymorphic Behavior	- Polymorphism Examples	Coding Tasks, Scenario Analysis	Apply polymorphism in various scenarios, understand its practical use cases	- Apply polymorphism in various scenarios, understand its practical use cases

Lecture 10	Encapsulation: Data Hiding	Protecting Data	- Access Modifiers - Data Hiding Techniques	Code Development, Problem-Solving Tasks	Implement data hiding and access control in Python classes, use access modifiers	- Implement data hiding and access control in Python classes, use access modifiers
Lecture 11	Encapsulation: Getter and Setter Methods	Controlling Attribute Access	- Getter and Setter Methods	Hands-on Exercises, Code Implementation	Create getter and setter methods to access and modify class attributes	- Create getter and setter methods to access and modify class attributes
Lecture 12	Abstraction: Abstract Classes	Defining Abstract Structures	- Abstract Class Definition	Coding Tasks, Conceptual Exercises	Define and use abstract classes in Python, understand their role in abstraction	- Define and use abstract classes in Python, understand their role in abstraction
Lecture 13	Abstraction: Interfaces and Implementation	Defining Interface Contracts	- Implementing Interfaces and Abstract Classes	Problem-Solving Scenarios, Code Development	Implement interfaces and abstract classes, adhere to interface contracts in Python	- Implement interfaces and abstract classes, adhere to interface contracts in Python
Lecture 14	Introduction to GUI Programming	Importance of GUIs in Software Development	- Overview of GUI Frameworks	Setting up GUI Development Environment	Recognize the importance of GUIs, understand available GUI frameworks in Python	- Recognize the importance of GUIs, understand available GUI frameworks in Python
Lecture 15	GUI Frameworks in Python	Exploring Tkinter, PyQt, wxPython	- Features and Setup	GUI Development Projects, Hands-on Sessions	Set up and develop GUI applications using popular frameworks in Python	- Set up and develop GUI applications using popular frameworks in Python
Lecture 16	Project and Review	Integration of Learned Concepts	- Project Work - Review Session	Final Project Presentation, Q&A Sessions	Apply OOP concepts to a comprehensive project, Review and consolidate learning outcomes	- Apply OOP concepts to a comprehensive project, Review and consolidate learning outcomes

Module 2: Advanced Python Concepts (16 Hrs)

Prerequisites for Module 2: Advanced Python Concepts

Intermediate Python Knowledge: Proficiency in fundamental Python concepts (data structures, functions, modules). Familiarity with libraries like NumPy and Pandas is beneficial.

Basic Understanding of Concurrency: Awareness of concurrent programming concepts (threads, processes).

Recommended Reading/Preparation: Intermediate-level Python programming practice.

Tutorials or guides on Python libraries for data visualization (Matplotlib, Pandas) and multithreading.

Lecture Number	Topic	Pre-Reading Material	Subtopics	Post-Reading Material	Learning Methodology/Activity	Learning Outcomes
Lecture 1	Multithreading	Basics of Concurrency	- Threads in Python - Threading Module	Coding Exercises, Case Studies	Implement threading in Python, manage concurrent tasks	- Implement threading in Python, manage concurrent tasks
Lecture 2	Data Visualization in Python	Statistics, NumPy	- Matplotlib for Visualization - Pandas for Data Manipulation	Data Visualization Projects, Practice Sessions	Perform statistical analysis using Python, visualize data using Matplotlib and Pandas	- Perform statistical analysis using Python, visualize data using Matplotlib and Pandas
Lecture 3	Socket Programming Basics	Networking Basics	- Socket Programming Introduction - Client-Server Communication	Networking Project, Socket Programming Tasks	Create client-server communication using Python sockets, send and receive data over sockets	- Create client-server communication using Python sockets, send and receive data over sockets
Lecture 4	Introduction to Selenium with Python	Web Automation Fundamentals	- Chrome WebDriver Setup - Locating Elements	Web Automation Tasks, Hands-on Practice	Automate web interactions using Selenium with Python, locate web elements for automation	- Automate web interactions using Selenium with Python, locate web elements for automation
Lecture 5	Web Element Interactions	Manipulating Web Elements	- Interacting with Web Elements	Code Samples, Practical Exercises	Interact with web elements, perform various actions on web pages using Selenium	- Interact with web elements, perform various actions on web pages using Selenium
Lecture 6	Locating Elements by Id	Finding Elements by Attribute	- Locating Elements by Id	Hands-on Activities, Code Implementation	Locate and interact with HTML elements using their 'id' attribute in Selenium	- Locate and interact with HTML elements using their 'id' attribute in Selenium
Lecture 7	Locating Elements by Name	Finding Elements by Attribute	- Locating Elements by Name	Problem-Solving Tasks, Coding Exercises	Locate and interact with HTML elements using their 'name' attribute in Selenium	- Locate and interact with HTML elements using their 'name' attribute in Selenium
Lecture 8	Locating Elements by XPath	Finding Elements by Path	- Locating Elements by XPath	Code Development, Practical Scenarios	Locate and interact with HTML elements using XPath expressions in Selenium	- Locate and interact with HTML elements using XPath expressions in Selenium

Lecture 9	Locating Hyperlinks by Link Text	Finding Elements by Text	- Locating Hyperlinks by Link Text	Hands-on Exercises, Scenario Analysis	Locate and interact with hyperlinks based on their text in Selenium	- Locate and interact with hyperlinks based on their text in Selenium
Lecture 10	Locating Elements by Tag Name	Finding Elements by Tag	- Locating Elements by Tag Name	Code Implementation, Conceptual Exercises	Locate and interact with HTML elements based on their tag name in Selenium	- Locate and interact with HTML elements based on their tag name in Selenium
Lecture 11	Locating Elements by Class Name	Finding Elements by Class	- Locating Elements by Class Name	Practical Sessions, Code Tasks	Locate and interact with HTML elements using their class name attribute in Selenium	- Locate and interact with HTML elements using their class name attribute in Selenium
Lecture 12	CSS Selectors	Advanced Element Locating	- Using CSS Selectors	Code Samples, Hands-on Practice	Locate and interact with HTML elements using CSS selectors in Selenium	- Locate and interact with HTML elements using CSS selectors in Selenium
Lecture 13	Advanced Web Element Interactions	Complex Element Manipulation	- Advanced Interactions	Real-world Scenarios, Project Development	Perform complex interactions with web elements, automate various user actions on web pages	- Perform complex interactions with web elements, automate various user actions on web pages
Lecture 14	Project and Review	Integration of Learned Concepts	- Project Work - Review Session	Final Project Presentation, Q&A Sessions	Apply Selenium concepts to a comprehensive project, Review and consolidate learning outcomes	- Apply Selenium concepts to a comprehensive project, Review and consolidate learning outcomes
Lecture 15	Revision and Practice	Reinforcing Key Concepts	- Revision Exercises - Practice Sessions	Problem-Solving Tasks, Coding Drills	Reinforce understanding through revision and practice of Selenium concepts	- Reinforce understanding through revision and practice of Selenium concepts
Lecture 16	Assessment and Conclusion	Evaluation and Recap	- Assessment Tasks - Recap Session	Assessment, Recap of Module Learning Outcomes	Evaluate understanding through assessment, recap and summarize Module 2 learnings	- Evaluate understanding through assessment, recap and summarize Module 2 learnings

Overall Course Objectives:

Comprehensive Understanding: Gain a comprehensive understanding of Object-Oriented Programming and advanced Python concepts.

Practical Application: Apply learned concepts to solve real-world problems through hands-on projects and tasks.

Proficiency in Python: Attain proficiency in using Python for various tasks including GUI development, data visualization, concurrency, networking, and web automation.

Evaluation and Assessment: Evaluate understanding through assessments, projects, and review sessions.

These outcomes aim to equip students with a solid foundation in Object-Oriented Programming principles and advanced Python concepts, enabling them to develop practical skills and apply Python for various tasks in software development and automation.