Complete Python Programming

Module 1: Introduction to Python

1. **Getting Started**
    - What is Python?
    - History and features of Python
    - Setting up the Python environment (installations, IDEs)
    - Writing your first Python program
    - Python syntax, keywords, and indentation
2. **Basic Data Types**
    - Numbers: integers, floats, complex numbers
    - Strings: definition, manipulation, and methods
    - Booleans and `None`
    - Type casting and type checking

Module 2: Variables, Operators, and Expressions

1. **Variables and Constants**
    - Variable naming conventions
    - Assigning values and dynamic typing
2. **Operators**
    - Arithmetic operators
    - Comparison (relational) operators
    - Logical and bitwise operators
    - Assignment operators
    - Membership and identity operators
3. **Expressions and Statements**
    - Combining operators and expressions
    - Evaluation of expressions

Module 3: Control Flow

1. **Conditional Statements**
    - `if`, `if-else`, and `if-elif-else` constructs
2. **Loops**
    - `for` and `while` loops

    - Nested loops
    - `break`, `continue`, and `pass` statements
3. **Comprehensions**
    - List comprehensions
    - Dictionary comprehensions
    - Set comprehensions

Module 4: Functions

1. **Introduction to Functions**
    - Defining and calling functions
    - Function arguments (positional, keyword, default)
    - Return values
    - `*args` and `**kwargs`
2. **Scope and Lifetime**
    - Global vs. local variables
    - `global` and `nonlocal` keywords
3. **Advanced Functions**
    - Recursive functions
    - Lambda functions
    - Function annotations

Module 5: Data Structures

1. **Lists**
    - Operations, slicing, and methods
    - Nested lists
2. **Tuples**
    - Properties and usage
3. **Dictionaries**
    - Key-value pairs
    - Methods and operations
4. **Sets**
    - Properties and methods
    - Mathematical operations on sets

Module 6: Object-Oriented Programming

1. **Basics of OOP**
    - Classes and objects

    - Constructors and destructors
2. **Attributes and Methods**
    - Instance vs. class variables
    - Instance, class, and static methods
3. **Inheritance**
    - Types of inheritance
    - Method overriding
    - The `super()` function
4. **Polymorphism**
    - Method overloading and overriding
    - Duck typing
5. **Special Methods**
    - Operator overloading (`__add__`, `__str__`, etc.)
    - Context managers (`__enter__` and `__exit__`)

Module 7: Modules and Packages

1. **Using Built-in Modules**
    - `math`, `random`, `datetime`, etc.
2. **Creating Custom Modules**
    - Writing and importing your own modules
3. **Packages**
    - Creating and using packages
    - The role of `__init__.py`

Module 8: File Handling

1. **File Operations**
    - Reading, writing, and appending to files
    - Working with text and binary files
2. **File Context Manager**
    - Using `with` for file handling
3. **Exception Handling in File Operations**
    - Handling file-related errors

Module 9: Advanced Python Concepts

1. **Decorators**
    - Function decorators
    - Class decorators

    - Chaining decorators
2. **Iterators**
    - Creating custom iterators
    - Using `iter()` and `next()`
3. **Generators**
    - Creating generators with `yield`
    - Generator expressions
4. **Context Managers**
    - Implementing custom context managers

Module 10: Error and Exception Handling

1. **Understanding Exceptions**
    - Built-in exceptions in Python
2. **Handling Exceptions**
    - `try`, `except`, `else`, and `finally`
    - Raising exceptions with `raise`
3. **Custom Exceptions**
    - Creating user-defined exception classes

Module 11: Python Libraries and Tools

1. **Popular Libraries**
    - `numpy`, `pandas` for data manipulation
    - `matplotlib`, `seaborn` for data visualization
    - `requests` for web requests
    - `os` and `sys` for system-level operations
2. **Working with Virtual Environments**
    - Creating and managing virtual environments using `venv`

Module 12: Working with Data

1. **JSON**
    - Reading and writing JSON
    - Parsing JSON objects
2. **CSV**
    - Reading and writing CSV files
3. **SQLite Database**
    - Using `sqlite3` for database operations

Module 13: Testing and Debugging

1. **Debugging**
   - Using the `pdb` module
   - Debugging techniques
2. **Unit Testing**
   - Writing test cases using `unittest`
   - Mocking in Python

Module 14: Advanced Topics

1. **Multithreading and Multiprocessing**
   - Basics of threads and processes
   - Using the `threading` and `multiprocessing` modules
2. **Asynchronous Programming**
   - Introduction to `asyncio`
   - Writing coroutines
3. **Regular Expressions**
   - Pattern matching with the `re` module

Module 15: Python in Real-World Applications

1. **Web Development**
   - Introduction to Flask or Django
2. **Data Science**
   - Overview of data analysis and visualization libraries
3. **Automation**
   - Using Python for scripting and automation tasks
4. **APIs**
   - Consuming and building APIs

FAQS

Q1: Why should I choose this Python programming course over others?

This course is meticulously designed to cover Python from basics to advanced concepts, ensuring a gradual and thorough learning curve. It includes:

- Core Python fundamentals like syntax, data types, and control structures.
- Advanced topics like **decorators**, **iterators**, and **generators**, which are crucial for writing clean and efficient Python code.
- Real-world applications like **web development**, **data analysis**, and **automation**, preparing you for practical scenarios.
- Capstone projects to consolidate your skills and enhance your portfolio.

Q2: Who is this course for?

This course is suitable for:

- **Beginners** with no prior programming experience.
- **Intermediate programmers** looking to deepen their Python knowledge and tackle advanced topics.
- Professionals transitioning to Python for fields like **web development**, **data science**, or **machine learning**.

Q3: What are the key highlights of this course?

- Covers Python comprehensively, from basics to advanced features.
- Focuses on practical applications with libraries like `numpy`, `pandas`, and `matplotlib`.
- Introduces advanced programming techniques like **asynchronous programming**, **multithreading**, and **custom context managers**.
- Prepares learners for professional environments with **testing**, **debugging**, and **real-world projects**.

Q4: How is the course structured?

The course is modular, starting with basics and gradually progressing to advanced topics. Each module builds on the previous one, ensuring a seamless learning experience. Modules also include hands-on coding exercises and quizzes to reinforce learning.

Q5: Will I learn advanced Python features like decorators and iterators?

Yes! Advanced concepts like **decorators**, **iterators**, and **generators** are covered in detail, complete with real-world examples and use cases.

Q6: Does this course cover practical applications?

Absolutely! The course includes:

- Using Python for **data science** (e.g., data manipulation with `pandas`).
- Building web applications with **Flask/Django**.
- Automation and scripting tasks.
- API consumption and development.

Q7: Are there any hands-on projects?

Yes, each module includes practical coding exercises, and the course concludes with a **capstone project**. This could be a web application, a data visualization dashboard, or an automation script.

Q8: Do I need prior programming knowledge?

No prior experience is needed. The course starts with the basics and progresses to advanced topics.

Q9: Will I get help with debugging and testing?

Yes, the course includes a dedicated module on **debugging techniques** and **unit testing** using Python's `unittest` framework.

Q10: Can this course help in professional development?

Yes, the course equips you with:

- Core Python programming skills.
- Knowledge of widely-used libraries and tools.
- The ability to develop real-world projects, boosting your resume and career prospects.

Resume making & Mock interview Sessions.