Here’s a comprehensive course outline for a Python programming course that covers the fundamentals, intermediate topics, and some advanced concepts. This course can be adapted for beginners with no programming experience or intermediate learners looking to deepen their Python knowledge.

---

Course Title: Python Programming: From Fundamentals to Advanced

Course Duration: 12 Weeks (2-3 hours per week)

---

Course Objectives:

By the end of this course, students will:

1. Understand the basics of Python programming, including syntax, data types, and control structures.

2. Learn to write efficient, organized, and error-free code.

3. Gain hands-on experience with functions, modules, and libraries.

4. Work with real-world data using libraries like Pandas and Matplotlib.

5. Develop problem-solving skills and apply Python to data analysis and automation tasks.

6. Build mini-projects to apply Python programming skills.

---

Course Outline

---

Module 1: Introduction to Python and Setup

- Overview of Python and its applications

- Installing Python and setting up a coding environment (Anaconda, Jupyter, or IDEs like PyCharm)

- Introduction to Jupyter Notebook

- Writing your first Python program

- Basic syntax, indentation, and comments

Module 2: Basic Python Syntax and Data Types

- Variables and data types: integers, floats, strings, booleans

- Type conversion and casting

- Basic operators (arithmetic, comparison, logical)

- Strings in-depth (concatenation, slicing, formatting)

- Exercise: Creating a simple calculator

Module 3: Control Structures

- Conditional statements: `if`, `elif`, `else`

- Loops: `for` loop, `while` loop

- List comprehensions and dictionary comprehensions

- Break and continue statements

- Exercise: Creating a number guessing game

Module 4: Data Structures in Python

- Lists, tuples, sets, and dictionaries

- Adding, updating, and deleting elements in data structures

- Nested data structures

- Common operations on data structures

- Exercise: Building a contact list app

Module 5: Functions and Modules

- Defining functions and understanding scope

- Parameters, arguments, and return values

- Lambda functions and higher-order functions

- Importing modules and creating custom modules

- Exercise: Developing a function library

Module 6: File Handling

- Reading and writing files (text, CSV)

- Working with JSON data

- Error handling and exceptions

- File handling in real-world scenarios (e.g., data logs)

- Exercise: Creating a text-based data storage system

Module 7: Object-Oriented Programming (OOP) in Python

- Introduction to classes and objects

- Attributes, methods, and constructors

- Inheritance and polymorphism

- Encapsulation and abstraction

- Magic methods and operator overloading

- Exercise: Building a simple inventory system

Module 8: Working with Libraries

- Overview of popular Python libraries (Pandas, NumPy, Matplotlib)

- Introduction to data manipulation with Pandas

- Basic data visualization with Matplotlib

- Numerical operations with NumPy

- Exercise: Analyzing and visualizing a dataset

Module 9: Error Handling and Debugging

- Common errors and exceptions

- Using `try`, `except`, `finally`, and `else` blocks

- Debugging techniques and best practices

- Exercise: Identifying and fixing errors in a pre-written script

Module 10: Introduction to Web Scraping

- Overview of web scraping and its applications

- Using libraries like BeautifulSoup and requests

- Parsing HTML to extract data

- Ethics and legal aspects of web scraping

- Exercise: Creating a basic web scraper to gather data

Module 11: Working with APIs

- Introduction to APIs and RESTful services

- Sending requests and handling responses

- Working with JSON data from APIs

- Exercise: Building a weather app using a public API

Module 12: Final Project

- Choose and design a final project (e.g., data analysis, mini-game, web scraper, or automation script)

- Project development with guidance on code structure, readability, and optimization

- Presentation of the final project and feedback

---

Assessment and Evaluation

- Weekly quizzes and exercises to reinforce learning

- Midterm project (e.g., data processing or automation script)

- Final project (presented in the last week)

- Participation in discussions and code reviews

Resources

- Recommended IDEs: Jupyter Notebook, VS Code, PyCharm

- Textbooks and online resources: "Automate the Boring Stuff with Python" by Al Sweigart, Python documentation

---

This course outline provides a structured approach to building proficiency in Python and helps students apply Python programming in a variety of practical scenarios.