Python for Beginners - Complete 15-Session Course Syllabus

Powered by CodeYAA Network

Course Overview

Duration: 15 Sessions (1 hour each, weekends only)
Target Audience: Complete beginners with no programming experience
Goal: Build job-ready Python skills and confidence to create independent projects
Format: Theory + Hands-on coding in every session

Session 1: Python Fundamentals & Setup

Topics Covered:

- What is Python and why learn it?
- Installing Python and IDE setup (VS Code/PyCharm)
- Understanding the Python interpreter
- Running your first Python program
- Python syntax basics and indentation
- Comments and documentation

Hands-on Exercises:

- Install Python and set up development environment
- Write "Hello, World!" program
- Create a simple calculator for addition
- Practice proper commenting

Learning Outcomes: Students can run Python programs and understand basic syntax

---

Session 2: Variables, Data Types & Input/Output

Topics Covered:

- Variables and naming conventions (PEP 8)

- Data types: int, float, str, bool
- Type conversion and type checking
- User input with input() function
- String formatting (f-strings)
- Basic debugging techniques

Hands-on Exercises:

- Create a personal information collector
- Build a unit converter (celsius to fahrenheit)
- Practice variable naming conventions
- Debug common beginner errors

Learning Outcomes: Students can handle different data types and user interaction

---

Session 3: Strings & String Methods

Topics Covered:

- String creation and manipulation
- String indexing and slicing
- Common string methods (upper, lower, strip, split, etc.)
- Escape characters and raw strings
- String concatenation vs f-strings

Hands-on Exercises:

- Build a name formatter program
- Create a word counter tool
- Email validator (basic version)
- Text case converter

Mini-Project: Password Strength Checker - Analyze password length, character types, and provide feedback

Learning Outcomes: Students can manipulate strings effectively for real-world applications

---

Session 4: Conditional Statements & Logic

Topics Covered:

- Boolean expressions and logical operators
- if, elif, else statements
- Nested conditions
- Comparison operators
- Boolean logic best practices

Hands-on Exercises:

- Age category classifier
- Grade calculator with letter grades
- Simple login system
- Number comparison tool

Mini-Project: Interactive Quiz Game - Multiple choice questions with scoring system

Learning Outcomes: Students can implement decision-making logic in programs

---

Session 5: Loops & Iteration

Topics Covered:

- for loops with range()
- while loops
- Loop control: break, continue, pass
- Nested loops
- Loop optimization tips

Hands-on Exercises:

- Multiplication table generator
- Number guessing game
- Pattern printing exercises
- Sum calculator for user inputs

Mini-Project: ATM Simulator - Menu-driven program with balance checking, withdrawal, deposit

Learning Outcomes: Students can use loops for repetitive tasks and user interaction

---

Session 6: Lists & List Operations

Topics Covered:

- Creating and accessing lists
- List methods (append, remove, pop, insert, etc.)
- List slicing and indexing
- List comprehensions (introduction)
- Iterating through lists

Hands-on Exercises:

- Shopping cart manager
- Student grade tracker
- List sorting and searching
- Remove duplicates from lists

Mini-Project: To-Do List Manager - Add, remove, view, and mark tasks as complete

Learning Outcomes: Students can work with dynamic data collections

---

Session 7: Tuples, Sets & Dictionaries

Topics Covered:

- Tuples: creation, accessing, immutability
- Sets: unique elements, set operations
- Dictionaries: key-value pairs, methods
- When to use each data structure
- Nested data structures basics

Hands-on Exercises:

- Coordinate storage with tuples
- Unique visitor tracker with sets

- Student database with dictionaries
- Menu pricing system

Mini-Project: Contact Management System - Store, search, update, delete contacts using dictionaries

Learning Outcomes: Students can choose appropriate data structures for different scenarios

---

Session 8: Functions & Modular Programming

Topics Covered:

- Defining and calling functions
- Parameters vs arguments
- Return statements
- Local vs global scope
- Default parameters and keyword arguments
- Lambda functions (introduction)

Hands-on Exercises:

- Temperature conversion functions
- Area calculator for different shapes
- Password generator function
- Data validation functions

Mini-Project: Simple Banking System - Modular functions for different banking operations

Learning Outcomes: Students can write reusable, modular code

Session 9: File Handling & Data Persistence

Topics Covered:

- Reading from and writing to files
- File modes and context managers (with statement)
- Working with CSV files
- Text file processing
- Error handling with file operations

Hands-on Exercises:

- Text file reader/writer
- Log file analyzer
- CSV data processor
- Configuration file handler

Mini-Project: Expense Tracker - Save and load expenses from files, generate reports

Learning Outcomes: Students can persist data and work with external files

---

Session 10: Error Handling & Debugging

Topics Covered:

- Understanding common Python errors
- try, except, finally blocks
- Handling specific exceptions
- Debugging strategies and tools
- Logging basics
- Writing defensive code

Hands-on Exercises:

- Safe user input validator
- File operation error handler
- Division by zero protection
- Data type validation

Mini-Project: Robust Calculator - Handle all possible errors gracefully with user-friendly messages

Learning Outcomes: Students can write stable, error-resistant programs

---

Session 11: Object-Oriented Programming Basics

Topics Covered:

- Classes and objects concepts

- Creating classes and instances
- Attributes and methods
- Constructor (**init**) method
- Instance vs class variables
- Basic inheritance

Hands-on Exercises:

- Create a Student class
- Design a Book class for library
- Build a Car class with methods
- Practice object interaction

Mini-Project: Library Management System - Book and Member classes with borrowing functionality

Learning Outcomes: Students understand OOP principles and can design simple class hierarchies

---

Session 12: Advanced OOP & Best Practices

Topics Covered:

- Encapsulation and data hiding
- Method types: instance, class, static
- Property decorators
- Method overriding
- Polymorphism basics
- Code organization and PEP 8

Hands-on Exercises:

- Bank account with private attributes
- Shape hierarchy with area calculations
- Employee management system
- Practice refactoring code

Mini-Project: Online Store Simulation - Product, Customer, and Order classes with interactions

Learning Outcomes: Students can design professional, maintainable object-oriented code

Session 13: Working with Libraries & APIs

Topics Covered:

- Installing packages with pip
- Working with popular libraries (requests, datetime, random)
- JSON data handling
- Making HTTP requests
- Virtual environments introduction
- Reading documentation

Hands-on Exercises:

- Weather app using API
- Random quote generator
- Date and time utilities
- JSON data parser

Mini-Project: News Reader App - Fetch and display news from a public API

Learning Outcomes: Students can integrate external libraries and work with APIs

---

Session 14: Version Control & Project Organization

Topics Covered:

- Git basics: init, add, commit, push
- GitHub introduction
- Project structure best practices
- README files and documentation
- Code commenting standards
- Preparing code for collaboration

Hands-on Exercises:

- Initialize Git repository
- Create meaningful commit messages

- Structure a Python project properly
- Write project documentation

Mini-Project: Portfolio Setup - Organize previous projects in a professional GitHub repository

Learning Outcomes: Students can manage code professionally and collaborate effectively

---

Session 15: Final Project & Career Preparation

Topics Covered:

- Project planning and requirements gathering
- Code review best practices
- Testing your code (basic unit tests)
- Deployment considerations
- Building a developer portfolio
- Next steps in Python learning

Hands-on Exercises:

- Code review of peer projects
- Write basic tests for functions
- Optimize and refactor existing code
- Practice explaining code to others

Final Project: Personal Finance Manager

- Complete application combining all learned concepts
- File persistence, error handling, OOP design
- User-friendly interface
- Proper documentation and Git management
- 30-minute presentation of the project

Learning Outcomes: Students can independently plan, build, and present a complete Python application

---

Course Learning Objectives

By the end of this course, students will be able to:

✅ Write clean, readable Python code following industry standards
✅ Implement all major programming concepts (variables, loops, functions, OOP)
✅ Handle errors gracefully and debug code effectively
✅ Work with files, APIs, and external libraries
✅ Use version control (Git) for project management
✅ Design and build complete applications independently
✅ Present and explain their code professionally
✅ Continue learning advanced Python topics confidently

Required Tools & Setup

- **Python 3.8+** (latest stable version)
- **IDE:** VS Code or PyCharm Community Edition
- **Git** for version control
- **GitHub account** for project hosting
- **Text editor** for notes and documentation

Assessment Methods

- **Weekly Coding Exercises** (40%)
- **Mini-Projects** (40%)
- **Final Project & Presentation** (20%)

Resources Provided

- Session-by-session code examples
- Exercise solutions and explanations
- Project templates and starter code
- Recommended reading and practice sites
- Career guidance and next steps roadmap

Course Designed by CodeYAA Network

Empowering the next generation of Python developers

This syllabus is designed to be practical, progressive, and industry-relevant. Each session builds upon previous knowledge while introducing new concepts at a manageable pace. The combination of theory, hands-on practice, and real-world projects ensures students graduate with both knowledge and confidence.