# Certified Entry-Level Python Programmer (PCEP)

**Duration: 2 Days**

**Target Audience:**

The Certified Entry-Level Python Programmer (PCEP) course is tailored for newcomers to programming, seeking foundational Python skills.

- Absolute beginners in programming
- Students and recent graduates aiming for tech careers
- Career switchers entering the tech industry
- IT professionals needing a basic understanding of Python
- Hobbyists wanting to learn Python for personal projects
- Data analysis enthusiasts and researchers
- Educators and academic instructors looking to update their technical skills
- Automation professionals seeking scripting knowledge
- Technical support staff aiming to enhance their troubleshooting tools
- Quality assurance testers interested in automating test scripts

**Prerequisites:**

For students looking to undertake training in the Certified Entry-Level Python Programmer (PCEP) course, the following minimum prerequisites are recommended to ensure a successful learning experience:

- Basic understanding of computer operations, such as file management and using a text editor.
- Familiarity with elementary mathematical concepts, such as addition, subtraction, multiplication, and division.
- Ability to think logically and problem-solve, which will be beneficial when learning programming concepts and debugging code.
- No prior programming experience is required, as the course is designed for entry-level learners.
- A willingness to learn and dedicate time to practice coding outside of the lessons to reinforce the topics covered in the course modules.

These prerequisites are intended to provide a foundation upon which the course will build. The PCEP course is tailored to be accessible and understandable for beginners, and our instructors are committed to supporting students every step of the way.

**Course Objectives:**

- Understand the basics of Python and its place in the modern computing environment.
- Differentiate between interpreters and compilers and understand their roles in Python programming.
- Successfully install Python and distinguish between Python 2 and Python 3.
- Learn about the history and evolution of Python as a programming language.
- Comprehend and utilize different data types in Python for variable management.
- Perform basic input/output operations and apply basic operators for data manipulation.
- Make decisions in code using Boolean values and conditional execution structures.
- Control program flow with loops and perform logical and bitwise operations.
- Define and use functions for reusable code and understand tuples, dictionaries, lists, and sets.
- Master indexing and slicing operations to manipulate strings and collections in Python.

**Course Outlines:**

**Day 1 – Python Basics, Flow Control, and Data Structures**

Module 1: Python Overview & Setup
- Introduction to Python (history, features, applications)
- Python Execution Model (script vs interactive mode, Jupyter Notebook)
- Python syntax & indentation rules
- Comments (single-line, multi-line, docstrings)

Module 2: Variables, Data Types, and Operators
- Variables & assignment rules
- Primitive data types: int, float, bool, str
- Type conversion (explicit vs implicit)
- Arithmetic, comparison, logical, bitwise operators
- Operator precedence & associativity
- Boolean expressions & truth tables

Module 3: Input/Output and String Handling
- Input with input()
- Formatted output (print, f-strings, .format())
- String indexing, slicing, immutability
- Common string methods: upper(), lower(), strip(), split(), replace(), find()

Module 4: Control Flow Structures
- if, elif, else
- Nested conditions
- Loops: for, while
- Loop control statements: break, continue, pass
- range() function in loops

Module 5: Data Structures – Lists, Tuples, Dictionaries, and Sets
- Lists: creation, indexing, CRUD operations, slicing, built-in methods (append, extend, sort)
- Tuples: immutability, use cases
- Dictionaries: key-value pairs, CRUD operations, methods (keys, values, items)
- Sets: uniqueness, set operations (union, intersection, difference)

**Day 2 – Functions, Exceptions, Modules, and Certification Prep**

Module 6: Functions in Python
- Defining and calling functions
- Parameters: positional, keyword, default arguments
- Return values
- Variable scope: local vs global
- Recursion basics

Module 7: Exception Handling
- Errors vs exceptions
- Built-in exceptions (ZeroDivisionError, ValueError, etc.)
- try, except, else, finally blocks
- Raising exceptions with raise
- Program to handle division by zero gracefully
- Custom input validation with exceptions

Module 8: Modules & Packages
- Importing standard modules (math, random, datetime, os, sys)
- Using dir() and help()
- Writing custom modules
- Concept of packages & __init__.py
- Installing external packages with pip

**Capstone Lab (End-to-End Mini Project)**

Student Grade Management System
- Input student names and marks
- Store them in a dictionary
- Compute average, highest, and lowest marks
- Assign grades (A/B/C) using if-else
- Handle invalid inputs using exception handling
- Export results as a formatted string