

Certified Professional Python Programmer Level 1 (PCPP1)

Course Duration: 32 Hours (4 Days)

Overview

The PCPP1 – Certified Professional in Python Programming 1 development course is designed to deepen the learner's knowledge of Python and prepare them for professional programming practices. It covers a comprehensive range of topics, from Advanced object-oriented programming, including Classes, Inheritance, Polymorphism, and Metaprogramming, to understanding and applying Python Enhancement Proposals (PEPs) which are essential for writing clean and maintainable code. The course also delves into GUI programming with modules like Tkinter and Pygame, enabling learners to create graphical user interfaces. Additionally, participants will gain hands-on experience with Network programming, particularly working with RESTful APIs, and File processing, ensuring they can handle data and interact with the web efficiently. Through this development course, learners will become adept at using Python's advanced features, making them well-equipped as a Certified Professional in Python Programming 1. This accreditation will open doors to advanced Python-related job roles and opportunities.

Audience Profile

The PCPP1 – Certified Professional in Python Programming 1 course is designed for intermediate to advanced Python developers seeking specialization.

- Software Developers and Programmers who want to master Python
- Experienced Python Developers aiming for certification
- IT Professionals looking to enhance their OOP skills in Python
- Computer Science/Engineering Students with a focus on Python
- Data Scientists and Analysts wanting to deepen their Python expertise
- Backend Developers looking to adopt Python
- DevOps Engineers interested in automating with Python
- Technical Leads managing Python projects
- Test Engineers writing automation scripts in Python
- System Administrators automating tasks with Python scripts
- Anyone pursuing a career in Python development with a focus on OOP, GUI, or Network Programming

Course Syllabus

Section 1: Advanced Object-Oriented Programming

PCPP-32-101 1.1 – Understand and explain the basic terms and programming concepts used in the OOP paradigm

- essential terminology: class, instance, object, attribute, method, type, instance and class variables, superclasses and subclasses
- reflexion: `isinstance()`, `issubclass()`
- the `__init__()` method
- creating classes, methods, and class and instance variables; calling methods; accessing class and instance variables

PCPP-32-101 1.2 – Perform Python core syntax operations

- Python core syntax expressions – magic methods: comparison methods (e.g. `__eq__(self, other)`), numeric methods (e.g. `__abs__(self)`), type conversion methods (e.g. `__init__(self)`), object intro- and retrospection (e.g. `__str__(self)`, `__instancecheck__(self, object)`), object attribute access (e.g. `__getattr__(self, attribute)`), accessing containers (e.g. `__getitem__(self, key)`)
- operating with special methods
- extending class implementations to support additional core syntax operations

PCPP-32-101 1.3 Understand and use the concepts of inheritance, polymorphism, and composition

- class hierarchies
- single vs. multiple inheritance
- Method Resolution Order (MRO)
- duck typing
- inheritance vs. composition
- modelling real-life problems using the "is a" and "has a" relations

PCPP-32-101 1.4 Understand the concept of extended function argument syntax and demonstrate

- proficiency in using decorators
- special identifiers: `*args`, `**kwargs`
- forwarding arguments to other functions
- function parameter handling
- closures
- function and class decorators
- decorating functions with classes

- creating decorators and operating with them: implementing decorator patterns, decorator arguments, wrappers
- decorator stacking
- syntactic sugar
- special methods: `__call__`, `__init__`

PCPP-32-101 1.5 Design, build, and use Python static and class methods

- implementing class and static methods
- class vs. static methods
- the `cls` parameter
- the `@classmethod` and `@staticmethod` decorators
- class methods: accessing and modifying the state/methods of a class, creating objects

PCPP-32-101 1.6 Understand and use Python abstract classes and methods

- abstract classes and abstract methods: defining, creating, and implementing abstract classes and abstract methods
- overriding abstract methods
- implementing a multiple inheritance from abstract classes
- delivering multiple child classes

PCPP-32-101 1.7 Understand and use the concept of attribute encapsulation

- definition, meaning, usage
- operating with the getter, setter, and deleter methods

PCPP-32-101 1.8 Understand and apply the concept of subclassing builtin classes

- inheriting properties from built-in classes
- using the concept of subclassing the built-ins to extend class features and modify class methods and attributes

PCPP-32-101 1.9 Demonstrate proficiency in the advanced techniques for creating and serving exceptions

- exceptions as objects, named attributes of exception objects, basic terms and concepts
- chained exceptions, the `__context__` and `__cause__` attributes, implicitly and explicitly chained exceptions
- analyzing exception traceback objects, the `__traceback__` attribute
- operating with different kinds of exceptions

PCPP-32-101 1.10 Demonstrate proficiency in performing shallow and deep copy operations

- shallow and deep copies of objects
- object: label vs. identity vs. value
- the id() function and the is operand
- operating with the copy() and deepcopy() methods

PCPP-32-101 1.11 Understand and perform (de)serialization of Python objects

- object persistence, serialization and deserialization: meaning, purpose, usage
- serializing objects as a single byte stream: the pickle module, pickling various data types the dumps() and loads functions
- serializing objects by implementing a serialization dictionary: the shelve module, file modes, creating chelve objects

PCPP-32-101 1.12 Understand and explain the concept of metaprogramming

- metaclasses: meaning, purpose, usage
- the type metaclass and the type() function
- special attributes: __name__, __class__, __bases__, __dict__
- class variables, and class methods
- operating with metaclasses,

Section 2: Coding Conventions, Best Practices, and Standardization

PCPP-32-101 2.1 – Understand and explain the concept of Python Enhancement Proposals and Python philosophy

- the PEP concept and selected PEPs: PEP 1, PEP 8, PEP 20, PEP 257
- PEP 1: different types of PEPs, formats, purpose, guidelines
- PEP 20: Python philosophy, its guiding principles, and design; the import this instruction and
- PEP 20 aphorisms

PCPP-32-101 2.2 – Employ the PEP 8 guidelines, coding conventions, and best practices

- PEP 8 compliant checkers
- recommendations for code layout: indentation, continuation lines, maximum line length, line
- breaks, blank lines (vertical whitespaces) default encodings
- module imports
- recommendations for string quotes, whitespace, and trailing commas: single-quoted vs. double
- quoted strings, whitespace in expressions and statements, whitespace and trailing commas

- recommendations for using comments: block comments, inline comments
- documentation strings
- naming conventions: naming styles, recommendations
- programming recommendations

PCPP-32-101 2.3 – Employ the PEP 257 guidelines, conventions, and best practices

- docstrings: rationale, usage
- comments vs. docstrings
- PEP 484 and type hints
- one-line vs. multi-line docstrings
- creating, using, and accessing docstrings
- documentation standards, linters, fixers

Section 3: GUI Programming

PCPP-32-101 3.1 – Understand and explain the basic concepts and terminology related to GUI programming

- GUI: meaning, rationale, basic terms and definitions
- visual programming: examples, basic features
- widgets/controls – basic terms: windows, title and title bars, buttons, icons, labels, etc.
- classical vs. event-driven programming
- events – basic terms
- widget toolkits/GUI toolkits

PCPP-32-101 3.2 – Use GUI toolkits, basic blocks, and conventions to design and build simple GUI

- applications
- importing tkinter components
- creating an application's main window: the Tk(), mainloop(), and title methods
- adding widgets to the window: buttons, labels, frames, the place() method, widget
- constructors, location, screen coordinates, size, etc.
- launching the event controller: event handlers, defining and using callbacks, the destroy()
- method, dialog boxes
- shaping the main window and interacting with the user
- checking the validity of user input and handling errors
- working with Canvas and its methods
- using the Entry, Radiobutton, and Button widgets

- managing widgets with the grid and place managers
- binding events using the bind() method

PCPP-32-101 3.3 – Demonstrate proficiency in using widgets and handling events

- settling widgets in the window's interior, geometry managers
- coloring widgets, color modes: RGB, HEX
- event handling: writing event handlers and assigning them to widgets
- event-driven programming: implementing interfaces using events and callbacks
- widget properties and methods
- variables: observable variables and adding observers to variables
- using selected clickable and non-clickable widgets
- identifying and servicing GUI events

Section 4: Network Programming

PCPP-32-101 4.1 – Understand and explain the basic concepts of network programming REST

- network sockets
- Domains, addresses, ports, protocols, and services
- Network communication: connection-oriented vs. connectionless communication, clients and servers

PCPP-32-101 4.2 – Demonstrate proficiency in working with sockets in Python

- the socket module: importing and creating sockets
- connecting sockets to HTTP servers, closing connections with servers
- sending requests to servers, the send() method
- receiving responses from servers, the recv() method
- exception handling mechanisms and exception types

PCPP-32-101 4.3 – Employ data transfer mechanisms for network communication

- JSON: syntax, structure, data types (numbers, strings, Boolean values, null), compound data (arrays and objects), sample JSON documents and their anatomies
- the json module: serialization and deserialization, serializing Python data/deserializing JSON (the dumps() and loads methods), serializing and deserializing Python objects
- XML: syntax, structure, sample xml documents and their anatomies, DTD, XML as a tree processing xml files

PCPP-32-101 4.4 – Design, develop, and improve a simple REST client

- the request module
- designing, building, and using testing environments
- HTTP methods: GET, POST, PUT, DELETE
- CRUD
- adding and updating data
- fetching and removing data from servers
- analyzing the server's response
- response status codes

Section 5: File Processing and Communicating with a Program's Environment

PCPP-32-101 5.1 – Demonstrate proficiency in database programming in Python

- the sqlite module
- creating and closing database connection using the connect and close methods
- creating tables
- inserting, reading, updating, and deleting data
- transaction demarcation
- cursor methods: execute, executemany, fetchone, fetchall
- creating basic SQL statements (SELECT, INSERT INTO, UPDATE, DELETE, etc.)

PCPP-32-101 5.2 – Demonstrate proficiency in processing different file formats in Python

- parsing XML documents
- searching data in XML documents using the find and findall methods
- building XML documents using the Element class and the SubElement function
- reading and writing CSV data using functions and classes: reader, writer, DictReader, DictWriter
- logging events in applications
- working with different levels of logging
- using LogRecord attributes to create log formats
- creating custom handlers and formatters
- parsing and creating configuration files using the ConfigParser object
- interpolating values in .ini files