Certified Associate in Python Programming (PCAP)

Duration: 4 Days

Course Description:

The Certified Associate in Python Programming (PCAP) course is a comprehensive program designed to equip learners with a robust understanding of Python, one of the most popular programming languages. Aimed at both beginners and those looking to formalize their skills, this course offers a deep dive into Python's essential concepts and constructs through four detailed Modules. Module 1 focuses on the foundational elements like Functions, Modules, and Packages, including how to create your own, manage them using PIP, and work with File handling. Module 2 enhances your knowledge of Python's data structures, such as strings, lists, tuples, and sets, along with Error and exception handling, and introduces List comprehension. Module 3 delves into Object-Oriented Programming, covering Classes, Objects, Methods, Inheritance, and Polymorphism. Lastly, Module 4 touches upon advanced topics like Generators, Iterators, and various Modules for System operations and Time management, as well as best practices in Testing and code quality with Pylint. Earning the PCAP certification validates a candidate's proficiency in Python, which is aligned with the PCAP 31-03 exam objectives. This credential not only enhances a learner's resume but also bolsters their ability to tackle real-world programming challenges with Python's powerful capabilities.

Target Audience:

The Certified Associate in Python Programming course offers comprehensive training in Python essentials, suitable for beginners and intermediate programmers.

Target Audience for the Certified Associate in Python Programming Course:

- Aspiring software developers
- Computer science students
- Data analysis enthusiasts
- Entry-level programmers
- IT professionals looking to expand their skill set
- Automation engineers
- Quality assurance specialists
- System administrators
- Academic researchers
- Hobbyists interested in learning programming
- Technical product managers
- Professionals in tech roles seeking to learn a new scripting language

Prerequisites:

To successfully undertake the Certified Associate in Python Programming (PCAP) course offered by Koenig Solutions, students should meet the following minimum prerequisites:

- Basic understanding of computer operations, such as file management and the use of command-line interfaces.
- Familiarity with core programming concepts like variables, data types, control structures (e.g., loops, conditional statements), and basic input/output operations.
- Some exposure to foundational mathematical concepts, particularly those relevant to computer science (e.g., logic, sets, and functions).
- No prior experience with Python is strictly necessary, but a general understanding of coding in any programming language is beneficial.
- Eagerness to learn and the ability to think logically, which is crucial for problem-solving and debugging code.

Note: While these prerequisites provide a foundation for learning, the PCAP course is designed to introduce Python to beginners as well as to bolster the skills of those with some programming experience. Hence, motivation and a willingness to practice coding exercises are as important as any formal prerequisite.

Course Objectives:

- Understand and create Python functions, enhancing code reusability and modularity.
- Learn to handle arguments and build custom modules and packages for structured program development.
- Utilize PIP for package management and lambda functions for creating anonymous, in-line functions.
- Gain proficiency in file handling to read, write, and manage file operations within Python applications.
- Master string, list, tuple, and set methods for effective data manipulation and list comprehensions for concise code.
- Develop error handling techniques using exceptions to create robust and fault-tolerant Python programs.
- Comprehend the principles of Object-Oriented Programming, including classes, objects, inheritance, and polymorphism, to design scalable software.
- Explore and implement iterators, generators, and Python's os and datetime modules for advanced programming tasks.
- Learn testing methodologies in Python and use Pylint to ensure code quality and adherence to coding standards.
- Enhance problem-solving skills and prepare for the PCAP certification exam to validate programming expertise.

Course Outlines:

Module 1: Modules and Packages

Import and use modules and packages
- import variants: import, from import, import as, import *
- advanced qualifying for nested modules
- the dir() function
- the sys.path variable

Perform evaluations using the math module
- functions: ceil(), floor(), trunc(), factorial(), hypot(), sqrt()

Generate random values using the random module
- functions: random(), seed(), choice(), sample()

Discover host platform properties using the platform module
- functions: platform(), machine(), processor(), system(), version(),
- python_implementation(), python_version_tuple()

Create and use user-defined modules and packages
- idea and rationale;
- the __pycache__ directory
- the __name__ variable
- public and private variables
- the __init__.py file
- searching for/through modules/packages
- nested packages vs. directory trees

Module 2 Exceptions

Handle errors using Python-defined exceptions
- except, except:-except, except:-else:, except (e1, e2)
- the hierarchy of exceptions
- raise, raise ex
- assert
- event classes
- except E as e
- the arg property

Extend the Python exceptions hierarchy with self-defined
- exceptions
- self-defined exceptions
- defining and using self-defined exceptions

Discover host platform properties using the platform module
- functions: platform(), machine(), processor(), system(), version(),
- python_implementation(), python_version_tuple()

Create and use user-defined modules and packages
- idea and rationale;
- the __pycache__ directory
- the __name__ variable
- public and private variables
- the __init__.py file
- searching for/through modules/packages
- nested packages vs. directory trees

Module 2 Exceptions

Handle errors using Python-defined exceptions
- except, except:-except, except:-else:, except (e1, e2)
- the hierarchy of exceptions
- raise, raise ex
- assert
- event classes
- except E as e
- the arg property

Extend the Python exceptions hierarchy with self-defined
- exceptions
- self-defined exceptions
- defining and using self-defined exceptions

Module 4 : Object-Oriented Programming

Understand the Object-Oriented approach
- Class
- Object property, method
- Encapsulation
- Inheritance
- Superclass
- Subclass
- identifying class components

Employ class and object properties
- instance vs. class variables: declarations and initializations
- the __dict__ property (objects vs. classes)
- private components (instances vs. classes)
- name mangling

Equip a class with methods
- declaring and using methods
- the self parameter

Discover the class structure
- introspection and the hasattr() function (objects vs classes)
- properties: __name__, __module__ , __bases__

Build a class hierarchy using inheritance
- single and multiple inheritance
- the isinstance() function
- overriding
- operators:
- not is
- is
- polymorphism
- overriding the __str__() method
- Diamonds

Construct and initialize objects
- declaring and invoking constructors

Module 5 Miscellaneous

Build complex lists using list comprehension
- list comprehensions: the if operator, nested comprehensions

Embed lambda functions into the code
- lambdas: defining and using lambdas
- self-defined functions taking lambdas as arguments
- functions: map(), filter()

Define and use closures
- closures: meaning and rationale
- defining and using closures

Understand basic Input/Output terminology
- I/O modes
- predefined streams
- handles vs. streams
- text vs. binary modes

Perform Input/Output operations
- the open() function
- the errno variable and its values
- functions: close(), .read(), .write(), .readline(), readlines()
- using bytearray as input/output buffer