

KOENIG SOLUTIONS • COURSE OUTLINE

Certified Associate in Python Programming (PCAP)

Learn to use Python modules, exceptions, strings and object-oriented programming to build and troubleshoot Python applications.

4 Days

32 HOURS

5

MODULES

66

TOPICS

COURSE OVERVIEW

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.

WHO THIS COURSE IS FOR

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

What You Will Learn

- Import and use standard modules such as math, random and platform
- Create and use user-defined modules and packages
- Handle errors using Python-defined and self-defined exceptions
- Operate on strings using indexing, slicing and built-in methods
- Apply list comprehensions and lambda functions in code
- Build class hierarchies using inheritance and polymorphism
- Construct and initialise objects using constructors
- Perform input/output operations using streams and buffers

COURSE SYLLABUS

5 Modules, 66 Topics

01

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

02

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

03

Strings

UNDERSTAND MACHINE REPRESENTATION OF CHARACTERS

- **encoding standards:** ASCII, UNICODE, UTF-8, code points, escape sequences

OPERATE ON STRINGS

- **functions:** ord(), chr()
- indexing, slicing, immutability
- iterating through strings, concatenating, multiplying, comparing (against strings and numbers)
- **operators:** in, not in

EMPLOY BUILT-IN STRING METHODS

- **methods:** .isxxx(), .join(), .split(), .sort(), sorted(), .index(), .find(), .rfind()

04

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

Object-Oriented Programming (continued)

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

05

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

NOTES

- This course is the second in a 2-course series that will prepare you for the PCAP: Certified Associate in Python Programming certification exam. The course picks up where the PCEP course leaves off.

WHAT IS INCLUDED

Everything You Need to Succeed

- Koenig course material for every module
- Delivery by certified instructors and subject matter experts
- Hands-on exercises throughout the course
- Course completion certificate from Koenig

ABOUT KOENIG

Who We Are

Established in 1993, Koenig Solutions is a globally recognised training provider with 35+ technology partnerships, 5,000+ training programmes and a network of certified instructors and subject matter experts, supporting individuals and enterprise teams worldwide.

FLEXIBLE DELIVERY

Learn the Way That Suits You

Classroom Training

Live Online Classes (ILO)

Fly-Me-A-Trainer (FMAT)

Flexi (Self-Paced)

1-on-1 Training

Ready to Get Started?

Talk to a learning advisor about schedules, group training and customised delivery.

Visit koenig-solutions.com

✉ info@koenig-solutions.com