PCEP: Certified Entry-Level Python Programmer

This course is the first in a 2-course series that will prepare you for the PCEP - Certified Entry-Level Python Programmer and PCAP: Certified Associate in Python Programming certification exams.

Duration: 16 hours

Section 1: Computer Programming and Python Fundamentals

PCEP-30-02 1.1 – Understand fundamental terms and definitions

- interpreting and the interpreter, compilation and the compiler
- lexis, syntax, and semantics

PCEP-30-02 1.2 – Understand Python's logic and structure

- keywords
- instructions
- indentation
- comments

PCEP-30-02 1.3 – Introduce literals and variables into code and use different numeral systems

- Boolean, integers, floating-point numbers
- scientific notation
- strings
- binary, octal, decimal, and hexadecimal numeral systems
- variables
- naming conventions
- implementing PEP-8 recommendations

PCEP-30-02 1.4 – Choose operators and data types adequate to the problem

- numeric operators: ** * / % // + –
- string operators: * +
- assignment and shortcut operators
- unary and binary operators
- priorities and binding
- bitwise operators: ~ & ^ | << >>
- Boolean operators: not, and, or
- Boolean expressions
- relational operators (== != > >= < <=)
- the accuracy of floating-point numbers
- type casting

PCEP-30-02 1.5 – Perform Input/Output console operations

- the print() and input() functions
- the sep= and end= keyword parameters
- the int() and float() functions

Section 2: Control Flow – Conditional Blocks and Loops

PCEP-30-02 2.1 – Make decisions and branch the flow with the if instruction

- conditional statements: if, if-else, if-elif, if-elif-else
- multiple conditional statements
- nesting conditional statements
- PCEP-30-02 2.2 – Perform different types of iterations
- the pass instruction
- building loops with while, for, range(), and in
- iterating through sequences
- expanding loops with while-else and for-else
- nesting loops and conditional statements
- controlling loop execution with break and continue

Section 3: Data Collections – Tuples, Dictionaries, Lists, and Strings

PCEP-30-02 3.1 – Collect and process data using lists

- constructing vectors
- indexing and slicing
- the len() function
- list methods: append(), insert(), index(), etc.
- functions: len(), sorted()
- the del instruction
- iterating through lists with the for loop
- initializing loops
- the in and not in operators
- list comprehensions
- copying and cloning
- lists in lists: matrices and cubes

PCEP-30-02 3.2 – Collect and process data using tuples

- tuples: indexing, slicing, building, immutability
- tuples vs. lists: similarities and differences
- lists inside tuples and tuples inside lists

PCEP-30-02 3.3 – Collect and process data using dictionaries

- dictionaries: building, indexing, adding and removing keys
- iterating through dictionaries and their keys and values
- checking the existence of keys
- methods: keys(), items(), and values()

PCEP-30-02 3.4 – Operate with strings

- constructing strings
- indexing, slicing, immutability
- escaping using the \ character
- quotes and apostrophes inside strings

- multi-line strings
- basic string functions and methods

Section 4: Functions and Exceptions

PCEP-30-02 4.1 – Decompose the code using functions

- defining and invoking user-defined functions and generators
- the return keyword, returning results
- the None keyword
- recursion

PCEP-30-02 4.2 – Organize interaction between the function and its environment

- parameters vs. arguments
- positional, keyword, and mixed argument passing
- default parameter values
- name scopes, name hiding (shadowing), and the global keyword

PCEP-30-02 4.3 – Python Built-In Exceptions Hierarchy

- BaseException
- Exception
- SystemExit
- KeyboardInterrupt
- abstract exceptions
- ArithmeticError
- LookupError
- IndexError
- KeyError
- TypeError
- ValueError

PCEP-30-02 4.4 – Basics of Python Exception Handling

- try-except / the try-except Exception
- ordering the except branches
- propagating exceptions through function boundaries
- delegating responsibility for handling exceptions