Python

Candidates for this exam should be able to recognize and write syntactically correct well-documented Python 3 code that will logically solve a given problem, correctly use data types supported by Python, and use common libraries to write a program that solves a complex problem.

Candidates are expected to have had at least 150 hours of instruction and/or hands-on experience with the Python programming language, be familiar with its features and capabilities, and understand how to write, debug, and maintain well-formed, well-documented Python code.

To be successful on the test, the candidate is also expected to have the following prerequisite knowledge and skills:

- 8th grade reading skills
- Basic computer skills
- Algebra I

1. Operations using Data Types and Operators

1.1 Evaluate expressions to identify the data types Python assigns to variables

- str, int, float, and bool

1.2 Perform and analyze data and data type operations

- Data type conversion, indexing, slicing, construct data structures, lists, list operations (including sorting, merging, appending, inserting, removing, finding maximum and minimum, and reversing)

1.3 Determine the sequence of execution based on operator precedence

- Assignment (=, +=, -=, /=, %=, //=, **=), comparison (==, >=, <=, !=), logical (and, or, not), logical, arithmetic (+, -, /, //, %, **, unary + and -), identity (is), containment (in)

1.4 Select operators to achieve the intended results

- Assignment (=, +=, -=, /=, %=, //=, **=), comparison (==, >=, <=, !=), logical (and, or, not), logical, arithmetic (+, -, /, //, %, **, unary + and -), identity (is), containment (in)

2. Flow Control with Decisions and Loops

2.1 Construct and analyze code segments that use branching statements

- if, elif, else, nested and compound conditional expressions

2.2 Construct and analyze code segments that perform iteration

- while, for, break, continue, pass, nested loops, loops that include compound conditional expressions

3. Input and Output Operations

3.1 Construct and analyze code segments that perform file input and output operations

- open, close, read, write, append, check existence, delete, with statement

INFORMATION TECHNOLOGY
SPECIALIST

3.2 Construct and analyze code segments that perform console input and output operations

- Read input from console, print formatted text (string.format() method, f-String method), use command-line arguments

4. Code Documentation and Structure

4.1 Document code segments

- Use indentation, white space, comments, and docstrings; generate documentation by using pydoc

4.2 Construct and analyze code segments that include function definitions

- Call signatures, default values, return, def, pass

5. Troubleshooting and Error Handling

5.1 Analyze, detect, and fix code segments that have errors

- Syntax errors, logic errors, runtime errors

5.2 Analyze and construct code segments that handle exceptions

- try, except, else, finally, raise

5.3 Perform unit testing

- Unittest, functions, methods, and assert methods (assertIsInstance, assertEqual, assertTrue, assertIs, assertIn)

6. Operations using Modules and Tools

6.1 Perform basic file system and command-line operations by using built-in modules

- io, os, os.path, sys (importing modules, using modules to open, read, and check existence of files, command-line arguments)

6.2 Solve complex computing problems by using built-in modules

- Math (fabs, ceil, floor, trunc, fmod, frexp, nan, isnan, sqrt, isqrt, pow, pi) datetime (now, strftime, weekday), random (randrange, randint, random, shuffle, choice, sample)

INFORMATION TECHNOLOGY
SPECIALIST