PCAP™ – Certified Associate in Python Programming (Exam PCAP-31-02) – EXAM SYLLABUS

PCAP-31-02 Exam

Status: RETIRED
(December 31, 2021)

The exam consists of four sections:

Section 1 → 10 items	Max Raw Score: 25 (25%)
Section 2 → 10 items	Max Raw Score: 25 (25%)
Section 3 → 10 items	Max Raw Score: 25 (25%)
Section 4 → 10 items	Max Raw Score: 25 (25%)

Last updated: September 17, 2020

Aligned with Exam PCAP-31-02

Section 1: Control and Evaluations (25%)

Objectives covered by the block (10 exam items)

- basic concepts: interpreting and the interpreter, compilation and the compiler, language elements, lexis, syntax and semantics, Python keywords, instructions, indenting
- literals: Boolean, integer, floating-point numbers, scientific notation, strings
- operators: unary and binary, priorities and binding
- numeric operators: ***, *, /, %, //, +, –*
- bitwise operators: *~, &, ^, |, <<, >>*
- string operators: **, +*
- Boolean operators: *not*, *and*, *or*
- relational operators: *==, != >, >=, <, <=*
- building complex Boolean expressions
- assignment and shortcut operators
- accuracy of floating-point numbers
- basic input and output: *input()*, *print()*, *int()*, *float()*, *str()* functions
- formatting the *print()* output with the *end=* and *sep=* arguments
- conditional statements: *if*, *if-else*, *if-elif*, *if-elif-else*
- the *pass* instruction
- simple lists: constructing vectors, indexing and slicing,
  the *len()* function
- simple strings: constructing, assigning, indexing, slicing comparing, immutability
- building loops: *while*, *for*, *range()*, *in*, iterating through sequences
- expanding loops: *while-else*, *for-else*, nesting loops and conditional statements
- controlling loop execution: *break*, *continue*

Section 2: Data Aggregates (25%)

Objectives covered by the block (10 exam items)

- strings in detail: ASCII, UNICODE, UTF-8, immutability, escape characters and escaping using the \ character, single and double quotes inside strings, multiline strings, copying vs. cloning, advanced slicing, string vs. string, string vs. non-string
- basic string methods: *upper()*, *lower()*, *isxxx()*, *capitalize()*, *split()*, *join()*, etc.
- basic string functions: *len()*, *chr()*, *ord()*

- lists in detail: indexing, slicing, the *del* instruction, iterating lists with the *for* loop, initializing lists, the *in* and *not in* operators, list comprehensions, copying and cloning lists
- basic list methods: *append()*, *insert()*, *index()*, etc.
- basic list functions: *len()*, *sorted()*, etc.
- lists in lists: matrices and cubes
- tuples: indexing, slicing, building, immutability
- tuples vs. lists: similarities and differences, lists inside tuples and tuples inside lists
- dictionaries: building and indexing dictionaries, adding and removing keys, iterating through dictionaries as well as their keys and values, checking a key's existence
- basic dictionary methods: *keys()*, *items()*, *values()*, etc.

Section 3: Functions and Modules (25%)

Objectives covered by the block (10 exam items)

- defining and invoking user-created functions
- generators
- the *return* and *yield* keywords, returning results, the *None* keyword, recursion
- parameters vs. arguments, positional keyword and mixed argument passing, default parameter values
- converting generator objects into lists using the *list()* function
- name scopes, name hiding (shadowing), the *global* keyword
- lambda functions, defining and using lambdas
- functions: *map()*, *filter()*, *reduce()*, *reversed()*, *sorted()*
- methods: *sort()*
- the *if* operator
- import directives, qualifying entities with module names, initializing modules
- writing and using modules, the *__name__* variable
- creating and using *pyc* files
- constructing and distributing packages, packages vs. directories, the role of the *__init__.py* file
- hiding module entities
- Python hashbangs, using multiline strings as module documentation

Section 4: Classes, Objects, and Exceptions (25%)

Objectives covered by the block (10 exam items)

- defining user-created classes, superclasses, subclasses, inheritance, searching for missing class components, creating objects
- class attributes: class variables and instance variables, defining, adding and removing attributes, explicit constructor invocation
- class methods: defining and using class methods, the *self* parameter: meaning and usage
- inheritance and overriding, finding class/object components
- single inheritance vs. multiple inheritance
- name mangling
- invoking methods, passing and using the *self* argument/parameter
- the *__init__* method
- the role of the *__str__* method
- introspection – properties: *__dict__*, *__name__*, *__module__*, *__bases__*, examining class/object structure
- writing and using constructors
- functions: *hasattr()*, *type()*, *issubclass()*, *isinstance()*, *super()*
- using predefined exceptions and defining user-created exceptions
- the *try-except-else-finally* block, the *raise* statement, the *except-as* variant
- exceptions hierarchy, assigning more than one exception to one except branch
- adding user-created exceptions to an existing hierarchy
- assertions
- the anatomy of exception objects
- input/output essentials: opening files with the *open()* function, stream objects, binary vs. text files, newline character translation, reading and writing files, bytearray objects
- methods: *read()*, *readinto()*, *readline()*, *write()*, *close()*