CERTIFIED ASSOCIATE IN PYTHON PROGRAMMING

BY: IMRAN

PCAP-31-03
(PVTCs, OnVUE)

- INTRODUCTION
- SYLLABUS
- LEARNING OUTCOMES
- WHY PYTHON?
- APPLICATIONS

Exam Information

- Exam Name: PCAP – Certified Associate in Python Programming
- Exam Level: Associate
- Exam Code: PCAP-31-03 (PVTCs, OnVUE)
- Duration: 65 minutes (exam) + 10 minutes (Non-Disclosure Agreement)
- No. of Questions: 40
- Format: Single and Multiple Choice
- Python Version: 3.x
- Passing Marks: 70%
- Exam Fee: USD 295
- Syllabus: Available at Python Institute Website
- Sample Test: Available at Python Institute Website

PCAP-31-03
(PVTCs, OnVUE)

- INTRODUCTION
- SYLLABUS
- LEARNING OUTCOMES
- WHY PYTHON?
- APPLICATIONS

- **Exam block #1: Modules and Packages (12%)**
  - **Objectives covered by the block (6 items)**
    - **import variants; advanced qualifying for nested modules**
    - **dir(); sys.path variable**
    - **math: ceil(), floor(), trunc(), factorial(), hypot(), sqrt(); random: random(), seed(), choice(), sample()**
    - **platform: platform(), machine(), processor(), system(), version(), python_implementation(), python_version_tuple()**
    - **idea, __pycache__, __name__, public variables, __init__.py**
    - **searching for modules/packages; nested packages vs directory tree**

PCAP-31-03
(PVTCs, OnVUE)

- INTRODUCTION
  - SYLLABUS
  - LEARNING OUTCOMES
- WHY PYTHON?
- APPLICATIONS

Exam block #2: Exceptions (14%)

- **Objectives covered by the block** (5 items)
  - except, except:-except; except:-else:, except (e1,e2)
  - the hierarchy of exceptions
  - raise, raise ex, assert
  - event classes, except E as e, arg property
  - self-defined exceptions, defining and using

PCAP-31-03
(PVTCs, OnVUE)

- INTRODUCTION
  - SYLLABUS
  - LEARNING OUTCOMES
- WHY PYTHON?
- APPLICATIONS

Exam block #3: Strings (18%)

- **Objectives covered by the block** (8 items)
  - ASCII, UNICODE, UTF-8, codepoints, escape sequences
  - ord(), chr(), literals
  - indexing, slicing, immutability
  - iterating through,
  - concatenating, multiplying, comparing (against strings and numbers)
  - in, not in
  - .isxxx(), .join(), .split()
  - .sort(), sorted(), .index(), .find(), .rfind()

PCAP-31-03
(PVTCs, OnVUE)

- INTRODUCTION
- SYLLABUS
- LEARNING OUTCOMES
- WHY PYTHON?
- APPLICATIONS

Exam block #4: Object-Oriented Programming (34%)

- **Objectives covered by the block** (12 items)
  - ideas: class, object, property, method, encapsulation, inheritance, grammar vs class, superclass, subclass
  - instance vs class variables: declaring, initializing
  - __dict__ property (objects vs classes)
  - private components (instance vs classes), name mangling
  - methods: declaring, using, self parameter
  - instrospection: hasattr() (objects vs classes), __name__, __module__, __bases__ properties
  - inheritance: single, multiple, isinstance(), overriding, not is and is operators
  - inheritance: single, multiple, isinstance(), overriding, not is and is operators
  - constructors: declaring and invoking
  - polymorphism
  - __name__, __module__, __bases__ properties, __str__() method
  - multiple inheritance, diamonds

Exam block #5: Miscellaneous (List Comprehensions, Lambdas, Closures, and I/O Operations) (22%)

- **Objectives covered by the block** (9 items)
  - list comprehension: if operator, using list comprehensions
  - lambdas: defining and using lambdas, self-defined functions taking lambda as as arguments; map(), filter();
  - closures: meaning, defining, and using closures
  - I/O Operations: I/O modes, predefined streams, handles; text/binary modes
    open(), errno and its values; close()
    .read(), .write(), .readline(); readlines() (along with bytearray())

- **Here are the important things you will learn:**
  - how to adopt general coding techniques and best practices in your projects;
  - how to process strings;
  - how to use object-oriented programming in Python;
  - how to import and use Python modules, including the *math*, *random*, *platform*, *os*, *time*, *datetime*, and *calendar* modules;
  - how to create and use your own Python modules and packages;
  - how to use the exception mechanism in Python;
  - how to use generators, iterators, and closures in Python;
  - how to process files.

There are also a couple of factors that make Python great for learning:

- It is easy to learn
  - the time needed to learn Python is shorter than for many other languages; this means that it's possible to start the actual programming faster;
- It is easy to use for writing new software
  - it's often possible to write code faster when using Python;
- It is easy to obtain, install and deploy
  - Python is free, open and multiplatform; not all languages can boast that.

- Where do we use Python?
  - Dropbox, Uber, Spotify, Pintrest, BuzzFeed – they are written, to a greater or lesser extent, in Python. Other examples are as follows:
    - Internet Applications, such as BitTorrent, Jogger Publishing Assistant, TheCircle, TwistedMatrix
    - 3D CAD/CAM (FreeCAD, Fandango, Blender, Vintech RCAM)
    - Enterprise Applications like Odoo, Tryton, Picalo, LinOTP 2, RESTx
  - Image Applications, such as Gnofract 4D, Gogh, imgSeek, MayaVi, VPython
  - There are numerous applications as well.