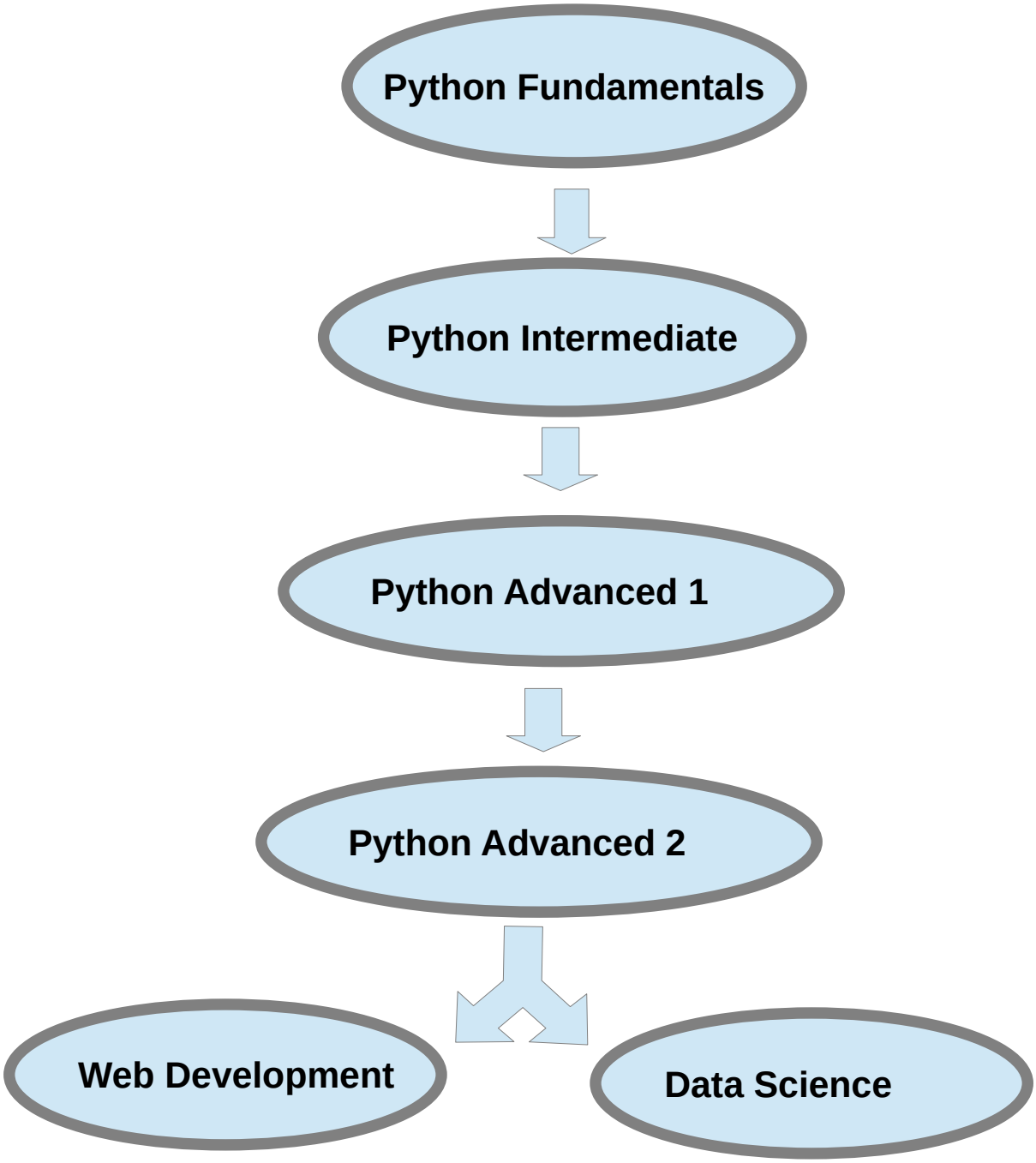
Python Programming Professional career path guide:



PYTHON FUNDAMENTALS

The Python Certified Entry Level Programmer qualification is a professional credential that measures ones ability to accomplish coding tasks related to the essentials of programming in the Python language. It teaches the universal concepts of computer programming, the syntax and semantics of the Python language as well as the skills in resolving typical implementation challenges with the help of the Python Standard Library.

A certified Python entry level programmer is an individual who is familiar with universal computer programming concepts like data types, containers, functions, conditions, loops, as well as Python programming language syntax, semantics, and the runtime environment.

Topics covered:

- Basic Concepts
  - fundamental 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
  - comments
  - the print() function
  - the input() function
  - numeral systems (binary, octal, decimal, hexadecimal)
  - numeric operators: ** * / % // + –
  - string operators: * +
  - assignments and shortcut operators
- Data Types, Evaluations, and Basic I/O Operations
  - operators: unary and binary, priorities and binding
  - bitwise operators: ~ & ^ | << >>
  - Boolean operators: not and or
  - Boolean expressions

- relational operators (== != > >= < <=), building complex Boolean expressions
- accuracy of floating-point numbers
- basic input and output operations using the input(), print(), int(), float(), str(), len() functions
- formatting print() output with end= and sep= arguments
- type casting
- basic calculations
- simple strings: constructing, assigning, indexing, slicing comparing, immutability

- Flow Control – loops and conditional blocks (20%)
  - conditional statements: if, if-else, if-elif, if-elif-else
  - multiple conditional statements
  - the pass instruction
  - 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
- Data Collections – Lists, Tuples, and Dictionaries
  - simple lists: constructing vectors, indexing and slicing, the len() function
  - lists in detail: indexing, slicing, basic methods (append(), insert(), index()) and functions (len(), sorted(), etc.), del instruction, iterating lists with the for loop, initializing, in and not in operators, list comprehension, copying and cloning
  - 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, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods

- strings in detail: ASCII, UNICODE, UTF-8, immutability, escaping using the \ character, quotes and apostrophes 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.) and functions (len(), chr(), ord()), escape characters

- Functions
  - defining and invoking your own functions and generators
  - 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

Course duration : 3 Days
Price : R7,900 (ex. vat)

PYTHON INTERMEDIATE

The Certified Associate in Python programming qualification is a professional credential that measures one's ability to accomplish coding tasks related to the basics of programming in the Python language and the fundamental notions and techniques used in object-oriented programming.

The certification shows that the individual is familiar with general computer programming concepts like conditional execution, loops, Python programming language syntax, semantics, and the runtime environment, as well as with general coding techniques and object-oriented programming.

Becoming PCAP certified ensures that the individual is fully acquainted with all the primary means provided by Python 3 to enable one to start their own studies, and to open a path to the developer's career.

Topics covered:

- Control and Evaluations
  - 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
  - assignments and shortcut operators
  - accuracy of floating-point numbers
  - basic input and output: input(), print(), int(), float(), str() functions
  - formatting print() output with 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
- Data Aggregates
  - strings in detail: ASCII, UNICODE, UTF-8, immutability, escaping using the \ character, quotes and apostrophes 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.) and functions (len(), chr(), ord()), escape characters

- lists in detail: indexing, slicing, basic methods (append(), insert(), index()) and functions (len(), sorted(), etc.), del instruction, iterating lists with the for loop, initializing, in and not in operators, list comprehension, copying and cloning
- 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, indexing, adding and removing keys, iterating through dictionaries as well as their keys and values, checking key existence, keys(), items() and values() methods

- Functions and Modules (25%)
  - defining and invoking your own functions and generators
  - 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
  - map(), filter(), reduce(), reversed(), sorted() functions and the sort() method
  - the if operator
  - import directives, qualifying entities with module names, initializing modules
  - writing and using modules, the __name__ variable
  - pyc file creation and usage
  - 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
- Classes, Objects, and Exceptions

- defining your own 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, 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: __dict__, __name__, __module__, __bases__ properties, examining class/object structure
- writing and using constructors
- hasattr(), type(), issubclass(), isinstance(), super() functions
- using predefined exceptions and defining your own ones
- 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 your own exceptions to an existing hierarchy
- assertions
- the anatomy of an exception object
- input/output basics: opening files with the open() function, stream objects, binary vs. text files, newline character translation, reading and writing files, bytearray objects
- read(), readinto(), readline(), write(), close() methods

Python Advanced 1

The Certified Professional in Python Programming 1 certification shows that the one is familiar with the more advanced perspective of classes and features of object-oriented programming. The scope of certification also includes graphical user interface programming, as well as working with selected library modules allowing to process

different kinds of files, communicate with a program's environment, and utilize tools and resources for the purposes of carrying out math-, science-, and engineering-related tasks.

Becoming PCPP1 certified ensures that the individual is fully acquainted with all the advanced means provided by Python 3 and related technologies to enable her/him to advance her/his career as a professional Python developer.

Topics covered:

- File Processing and Communicating with a Program's Environment
  - Processing different kinds of files
    - sqlite3 – interacting with SQLite databases
    - xml – creating and processing XML files
    - csv – CSV file reading and writing
    - logging – basics logging facility for Python
    - configparser – configuration file parser
  - Communicating with a program's environment:
    - os – interacting with the operating system,
    - datetime – manipulating with dates and time
    - io – working with streams,
    - time – time access and conversions

- Math, Science, and Engineering Tools
  - math – a basic tool for elementary evaluations
  - NumPy – fundamental package for scientific computing
  - SciPy – an ecosystem for mathematics, science, and engineering
  - Matplotlib – 2D plotting library producing publication quality figures
  - Pandas – a library providing high-performance and data analysis tools
  - SciKit-image – a collection of algorithms for image processing

- GUI Programming
  - What is GUI and where it comes from
  - Constructing a GUI – basic blocks and conventions
  - Event-driven programming
  - Currently used GUI environments and toolkits
  - tkinter — Python interface to Tcl/Tk
    - tkinter's application life cycle
    - Widgets, windows and events

- Sample applications
- pygame – a simple way of developing multimedia applications

- Python Enhancement Proposals
  - What is PEP?
  - Coding conventions – not only style and naming
  - PEP 20 – The Zen of Python: a collection of principles that influences the design of Python code
  - PEP 8 – Style Guide for Python Code: coding conventions for code comprising the standard library in the main Python distribution
  - PEP 257 – Docstring Conventions: what is docstring and some semantics as well as conventions associated with them
  - A tour of important PEPs

- Advanced Perspective of Classes and Object-Oriented Programming in Python
  - Classes, Instances, Attributes, Methods
  - Working with class and instance data
  - Copying object data using shallow and deep operations
  - Inheritance and Polymorphism
  - Different faces of Python methods: static and class methods
  - Abstract classes vs. method overloading
  - Composition vs. Inheritance – two ways to the same destination
  - Implementing Core Syntax
  - Subclassing built-ins
  - Attribute Encapsulation
  - Advanced techniques of creating and serving exceptions
  - Serialization of Python objects using the pickle module
  - Making Python object persistent using the shelve module
  - Metaprograming
    - Function decorators
    - Class decorators
    - Metaclasses

Python Advanced 2

The Certified Professional in Python Programming 2 certification shows that the one is familiar with and proficient in automating processes with Python as well as creating Python and Python-related tools, frameworks and systems. The scope of certification includes: Creating and Distributing Packages, Testing Principles and Techniques, The

Fundamentals of Design Patterns and Interprocess Communication (IPC), The Basics of Python Network Programming, Python-MySQL Database Access.

Becoming PCPP2 certified ensures that the individual is fully acquainted with all the advanced means provided by Python 3 and related technologies to enable her/him to advance her/his career as a senior/expert-level Python developer.

Topics covered:

- Creating and Distributing Packages
  - Using pip
  - Basic directory structure
  - The setup.py file
  - Sharing, storing, and installing packages
  - Documentation
  - License
  - Testing principles and techniques
  - unittest – Unit testing framework
  - Pytest – framework to write tests

- Design Patterns
  - Object-oriented design principles and the concept of design patterns
  - The Singleton Design Pattern
  - The Factory Pattern
  - The Façade Pattern
  - The Proxy Pattern
  - The Observer Pattern
  - The Command Pattern
  - The Template Method Pattern
  - Model-View-Controller
  - The State Design Pattern

- Interprocess Communication
  - multiprocessing — Process-based parallelism
  - threading — Thread-based parallelism
  - subprocess — Subprocess management
  - Multiprocess synchronisation
  - queue — A synchronized queue class
  - socket — Low-level networking interface
  - mmap — Memory-mapped file support

- Python Network Programming
  - Python Socket Module
  - Introduction to sockets
  - Server Socket Methods
  - Client socket methods
  - General socket methods
  - Client-Server vs. Peer-to-peer
  - Other Internet nodules

- Python-MySQL Database Access
  - Relational databases – fundamental principles and how to work with them
  - MySQL vs. rest of the world
  - CRUD Application
    - db connection
    - db create
    - db insert
    - db read
    - db update
    - db delete