





Python Advance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1 | 28/01/2024 |  |  |  | Initial Release |
| 2 | 03/02/2024 |  | Prateek Bajaj (MS/EVS-SDS-PJ-CST) | Prateek Bajaj (MS/EVS-SDS-PJ-CST) | Custom Revision Based on Need |
| 3 | 04/04/2024 | Ramakant Debata |  |  | Finalised contents; S/w and N/w requirements added |
|  |  |  |  |  |  |

**Document History**

**Advanced Python**

# 

**Course Title: ADVANCED PYTHON**

# Course Summary:

The course discusses advanced use cases of Python for developers with hands-on experience between

1. **Essential Pre-Requisite**

* **Hands-on experience with Python**
  + Understanding of Python fundamentals: statements, list, dict etc
  + [EXTREMELY IMPORTANT] Robust understanding of Object-Oriented Programming
  + Database Operations needed for RESTful services.

1. **Audience:** Existing Python developers looking forward to understanding advanced Python language features and use cases.
2. **Hardware & Network Requirements**

* Any quad core CPU or above
* Windows or mac
* 8gb RAM or above
* Basic internet connection
* Access to
  + <https://github.com/>
  + <https://www.python.org/>
  + <https://docs.python.org>
  + [https://peps.python.org](https://peps.python.org/pep-0000/)
  + <https://pypi.org/>
  + <https://www.python.org/downloads/>

1. **Software Requirements**

* IDE: Visual Studio Code (<https://code.visualstudio.com/>)
* Git for Windows (<https://gitforwindows.org/>)
* Installed software/modules:
  1. Deploy using installer: Python 3.12.3 (<https://www.python.org/downloads/release/python-3123/>)
     1. Windows Installer (<https://www.python.org/downloads/release/python-3123/>)
  2. Pip
     1. curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
     2. python get-pip.py # Install pip
     3. pip --version # Verify if pip installed
  3. Python3 Modules (On command line, where python is installed)
     1. py -m pip install --upgrade pip
     2. **pip install virtualenv numpy pandas scipy urllib3 multipledispatch requests PyQt6**
  4. Should be able to run below program (SetupTest.py) from participant login

# SetupTest.py

import platform

import sys

import virtualenv

import numpy

import pandas

import scipy

import urllib3

import multipledispatch

import requests

import PyQt6

print("Python version:", platform.python\_version())

print("virtualenv version:", virtualenv.\_\_version\_\_)

print("numpy version:", numpy.\_\_version\_\_)

print("pandas version:", pandas.\_\_version\_\_)

print("scipy version:", scipy.\_\_version\_\_)

print("urllib3 version:", urllib3.\_\_version\_\_)

print("multipledispatch version:", multipledispatch.\_\_version\_\_)

print("requests version:", requests.\_\_version\_\_)

print("PyQt version:", PyQt6.\_\_spec\_\_.name)

* + 1. ***Program should run without error and list out the versions of the modules correctly.***
* Jupyter
* No Firewall or firewall configured to allow access to PyPI repositories

# Learning Outcomes:

* Object Oriented Programming
* Exception Handling
* Operator Overloading
* Generators
* Decorators
* Multi-threading
* Networking
* Coroutines
* Files I/O — blocking and non-blocking
* Working with different types of files and database
* Understanding Data management with Numpy and Pandas
* Developing Restful services
* Developing a GUI with PyQT

# Course Content (day-wise):

# Duration: 5 Days (Highly Recommended 6Days)

## Day 1

### Python Special Functions

* \_\_init\_\_ and \_\_delete\_\_
* Operator overloading \_\_\_add\_\_, \_\_sub\_\_ etc
* More special object members \_\_dict\_\_, \_\_len\_\_
* Need to Iterators
* Implementing Iterators
* Python Generators.

### Functional Programming In python

* Diving Deeper in Python
* Function and Scopes
* Callback Functions
* Lambdas
* List Function Use Cases
* Closure
* Decorator as special case for closure
* Decorator accepting parameters
* More Decorator use Cases
* Decorators
  + - Function decorator
    - Class decorator
    - ((Instance decorator))

## **Day 2**

### Asynchronous Programming Using Threads

* Working with Threading module
* Creating Threads
* Passing parameters
* Controlling threads
* Thread sleep
* Join
* Synchronization
* Locks
* Pools
* Pipes
* Semaphores
* Events and timers

## **Day 3**

### Asynchronous Programming Using coroutines

* What is coroutine.
* How is it different from threads
* Piping
* Closing
* Getting results
* Synchronization

### Nonblocking I/O

* Synchronous and Asyncrhonous I/O
* Events and loops for network events

## **Day 4**

### Overview of working with GUI using PyQT

* Creating UI
* Understanding UI Component
* Composing a UI
* Handling an Event
* Responding to the event
* Common controls
* Avoiding blocking calls

## **Day 5**

### Regular Expressions

* What is Regular Expressions
* Use cases
* Metacharacters and patterns
* Search
* Match
* Findall
* Split

# Course Structure:

|  |  |
| --- | --- |
| **Activity** | **Indicative Number of Hours** |
| Pre-Read Hours | n/a |
| Teaching Hours | 30 |
| Hands on Sessions Hours | 2 |
| Assignments & Tutorial Hours | 3 |
| Mock Project Hours | n/a |

# Course Structure:

|  |  |  |
| --- | --- | --- |
| **Method of Assessment** | **Yes/No** | **Weightage** |
|  |  |  |
| Pre-Assessment | Yes | N/A |
| Mid-Assessment | No |  |
| Post-Assessment | Yes | 100% |
| Project Work | n/a |  |

# Course Resources:

* 1. **Code Samples: Sample code snippets and solutions for better understanding.**
  2. **Assignments: Practical assignments to reinforce learning and build real-world skills.**

# Recommended Reading Links:

# Course Owner (s):

|  |  |  |
| --- | --- | --- |
| **Employee Name** | **Employee Mail ID** | **Business Unit** |
|  |  |  |
|  |  |  |
|  |  |  |