





Python  
[Project Requirement]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1 | 14/08/2025 | Ramakant D. |  |  | Finalised contents |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Document History**

**Course Title:**

# Course Summary:

The course aims at explaining the upgrading the participant’s understanding on Python. The course discusses the fundamental syntax and philosophy of python along with its core programming elements. It will additionally cover text file-io for various formats, using regular expressions and FastAPI for asynchronous API development. This course will culminate with a capstone project requirements that the participant can implement on his/her own, based on the topics covered.

1. **Pre-Requisite**

Basic understanding of programming would be preferrable.

1. **Audience**

Engineers who are looking forward to start programming.

1. **Hardware & Network Requirements**

* Any quad core CPU or above
* Windows or mac
* 8gb RAM or above
* Basic internet connection

1. **Software Requirements**
   1. OS: Windows 10/11 Pro or Enterprise
   2. IDE: Visual Studio Code (<https://code.visualstudio.com/>)
   3. Git
      1. [Git for windows](https://github.com/git-for-windows/git/releases/download/v2.43.0.windows.1/Git-2.43.0-64-bit.exe) (for Windows platform)
      2. git package for Linux/MacOs
   4. Installed software/modules:
      1. Python Python 3.11.9
         1. <https://www.python.org/downloads/release/python-3119/>
         2. <https://www.python.org/ftp/python/3.11.9/python-3.11.9-amd64.exe>
      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
      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

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\_\_)

* + 1. ***Program should run without error and list out the versions of the modules correctly.***

# Learning Outcomes:

* Understand the basic language structure of python
* Python fundamental system
* Python collection
* Python object-oriented programming

# Course Content (day wise):

|  |  |
| --- | --- |
| Course Schedule | |
| Day 1 | |  | | --- | | **Foundations of Python** | | Programming Fundamentals | | Program to Executable (Translation) | | Executable to Process (Mapping) | | Process memory structure/segments | | The Data-Segment (call-stack, heap and data-area) | | Call Stack and Activation Records | | Scope and lifetime of variables | | Python as a programming language | | Low level vs. High level Languages | | Compiled vs. Interpreted Languages | | Programming paradigms | | Best practices / Idioms - Zen of Python | | Enhancements and Documentations | | Programming environment | | Installations and configurations | | Using iPython (REPL) and its nuances | | Running a basic Python program | | Basic Data Types | | Introduction to Bool and None | | Numbers: Integer, Floats, Decimals, Fractions, Complex | |
| Day 2 | |  | | --- | | **Control Structures, Objects and Collections** | | Control Structures | | Conditional Statements: if-elif-else, match-cases | | Loops: for, while | | Object based | | Characteristics of an Object | | Id, type, value | | Mutability | | Distinction in Python | | Handling, implications and parameters of choice | | Collections | | Mutables: Lists, Sets, Dictionaries | | Immutables: Numbers, Strings, Tuples, Named Tuples, Frozen Sets | | Comprehensions | | List, Set, Dictionary, Generator, and Tuple Comprehensions | |
| Day 3 | |  | | --- | | **Advanced Concepts and Functions** | | Virtual Environments (venv) | | Creating and Managing | | Activation/Deactivation | | Module Isolation and package management | | Functions | | Syntax and Docstrings | | Parameter Handling and Scopes | | Positional arguments | | Named arguments | | Packing and Unpacking of elements | | Variable arguments | | Keyworded variable arguments | | Advanced Functions | | Special Arguments | | Enforcing argument structure via syntax | | Method Overloading | | Mechanism in other languages | | Support in Python | | Modules | | Creating Modules and Packages | | \_\_init\_\_.py file | | Using \_\_all\_\_ and \_ Variables | | Using \_\_name\_\_ | | Import Syntax and Variations | |
| Day 4 | |  | | --- | | **File I/O [Text, XML, JSON] and Debugging** | | Standard modules used with File IO | | Using the ‘sys’ module | | sys.argv, sys.path, sys.version | | Using the ‘os’ module | | Filesystem/directory operations | | An overview on \_\_builtin\_\_ and \_\_future\_\_ modules | | File I/O (Text) | | Basic Operations: Open, Modes, Read/Write | | File as Iterators | | File I/O Constructs | | Using Try-Finally | | Using Context Manager | | Working with XML | | DOM and Sax | | Introducing ElementTree | | Parsing XML | | Navigating the document | | Creating a new XML document | | Working with JSON | | JSON and Python dictionaries | | Parsing JSON into Python | | Converting Python into JSON | | **Debugging** | | Debugging the code | | Breakpoints | | Watch | | Shortcuts to navigate function calls | |
| Day 5 | |  | | --- | | **Duck Typing and Object-Oriented Programming** | | Duck Typing | | Ideology and its interpretation | | Impact on Python as a language. | | Implications on the design for Python programmers | | Classes & OOP | | Object Oriented Paradigm | | Major Pillars of Object-oriented paradigm | | Minor Pillars | | Translating concepts to implementations | | Basic constructs | | Constructor and Initializer | | Variables, and Methods | | Instance members vs. Class members | | Static methods | | Inheritance (Single) | | Design considerations when using Inheritance. | |
| Day 6 | |  | | --- | | **Object-Oriented and Exception Handling** | | Improving coding design | | String Representations – str, repr | | Operator Overloading | | Method Overriding | | Advanced OOP Concepts | | Multiple Inheritance | | Bases list, Method Resolution Order | | Impact of MRO on Diamond Inheritance | | Polymorphism | | Object Serialization | | Pickle - Python object serialization | | Errors, Exceptions and Stack unwinding | | Syntax | | Stack unwinding | | Built-in exceptions hierarchy | | Handlers | | Single Exception | | Multiple Exceptions | | Multiple Handlers and sequencing | | User-defined exception classes | | Exception Management | | Re-throwing the exception | | The exception object (type, traceback) | | **Using the ‘with’ block for exception handling** | |
| Day 7 | |  | | --- | | **Regular Expressions** | | Introduction to Regular Expressions: | | Definition and purpose of regular expressions. | | Applications in text processing and pattern matching. | | Working with re Module: | | Importing the re module. | | Functions | | Objects | | Exceptions | | Basic Patterns: | | Literal characters and metacharacters. | | Character classes and ranges. | | Special characters like dot (.) and caret (^). | | Modifiers and Flags: | | Case-insensitive matching. | | Multiline matching. | | Dot-all mode. | | Anchors and Boundaries: | | Caret (^) and dollar sign ($) as anchors. | | Word boundaries (\b) and non-word boundaries (\B). | | Character Classes: | | Predefined character classes (\d, \w, \s). | | Negation and custom character classes. | | Quantifiers and Repetition: | | Asterisk (\*), plus (+), and question mark (?) operators. | | Using curly braces ({}) for specifying repetition. | | Grouping and Capturing: | | Parentheses for grouping. | | Capturing groups and non-capturing groups. | | Backreferences. | | Assertions: | | Positive lookahead (?=) and negative lookahead (?!). | | Positive lookbehind (?<=) and negative lookbehind (?<!). | | Advanced Patterns: | | Greedy vs. non-greedy quantifiers. | | Using alternation (|) for OR conditions. | | Escape sequences. | |
| Day 8 | |  | | --- | | **Introduction to Concurrency concepts (fast-paced discussion)** | | Types of work | | I/O bound | | CPU bound | | Threads and resource sharing | | GIL and its Limitations | | Multi-processing as a Workaround for GIL | | Abstracting through Executor API | | Introducing Asyncio (Single threaded Multi tasking) | | Event loops and presence | | **Introduction to FastAPI and Basics** | |  | | Introduction and Setup | | FastAPI Overview | | Project Overview and Requirements | | Setting Up the Environment | | Installing FastAPI and Uvicorn | | Starting a FastAPI Project | | Creating the First FastAPI App | | Adding Our First Operation | | Running the Project | | Troubleshooting Project Execution | |
| Day 9 | |  | | --- | | **Introduction to FastAPI and Basics** | |  | | Understanding FastAPI Basics | | Exploring Auto-generated Documentation | | How FastAPI Runs Your Code | | Async vs. Non-Async Functions | | Practical Examples and Hands-On Practice | | Serving Data with FastAPI | | Adding Parameters (Query and Path) | | Serving Static Data | | Handling Optional Query Parameters | | Typed Parameters and Path Parameters | | **Serving Data and Pydantic Models** | |  | | Advanced Data Serving | | Using Path Parameters to Get by ID | | Debugging with FastAPI code | | Handling Errors (404 Not Found) | | Review of Data Serving Techniques | | Pydantic Models | | Creating Data Models with Pydantic | | Loading Data from JSON | | Using Pydantic Models in Read-only Operations | | Recap of HTTP Methods | |
| Day 10 | |  | | --- | | **Serving Data and Pydantic Models** | |  | | CRUD Operations with Pydantic Models | | Adding New Objects with POST | | The Need for Separate Input and Output Models | | The Response Model | | Implementing PUT and DELETE | | Advanced Pydantic Features | | Using the OpenAPI Schema with Postman | | Adding Example Data | | Nested Models | | Review of Pydantic Models | | **Advanced Features, Authentication, and Deployment** | |  | | Using a Database with FastAPI | | Introducing SQLModel | | Creating a Model Class | | Setting Up the Database | | CRUD Operations with SQLModel | | Advanced Database Operations | | Injecting the Session | | Implementing GET, PUT, and DELETE | | Working with Relations | | Review of Database Integration | |
| Day 11 | |  | | --- | | **Advanced Features, Authentication, and Deployment** | |  | | Advanced FastAPI Features | | Reorganizing the Code with APIRouter | | Serving Web Pages | | Dynamic HTML with Jinja | | Processing Form Data | | Authentication, Testing, and Deployment | | Adding Authentication and User Model | | Password Hashing and Column Settings | | HTTP Basic Authentication and OAuth2 | | Unit Testing with mocking | | Q&A Session | | Post training test | |
| Capstone |  |
| Day 12 | |  | | --- | | Requirement discussion | | Design session | |
| Day 12-14 | Implementation |
| Day 15 | Ideal solution discussion |

# Course Structure:

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

# Course Structure:

|  |  |  |
| --- | --- | --- |
| **Method of Assessment** | **Yes/No** | **Weightage** |
|  |  |  |
| Pre-Assessment | Yes |  |
| 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** |
|  |  |  |
|  |  |  |
|  |  |  |