ANNEXURE - III

Python Programming

Introduction to Python

- Overview
- Python Features
- Environment Setup
- Running Python
- IDLE
- Commercial and Free Python IDEs

Basic Syntax

- Interactive Mode
- Script Mode
- Python Identifiers
- Reserved Words
- Lines and Indentation
- Multi-Line Statements
- Quotations in Python
- Comments in Python
- Using Blank Lines
- User inputs
- Command Line Arguments

Variable Types

- Creating variables
- Assigning Values to Variables
- Accessing variables
- Multiple Assignment

Downloading, Installation & Configure Various Python Tools

- Python 2.x or 3.x on various OS platforms like Windows, Linux, Unix & MacOs
- Pycharm
- Anaconda & Spyder

Standard Data Types

- Numbers
- Strings
- Lists
- Tuples
- Dictionaries
- Sets

Python Data Types – Numbers

- int
- float
- complex
- Number type conversions
- Mathematical Functions
- Random Number Functions
- Trigonometric Functions
- Mathematical Constants

Python Data Types – Strings

- Accessing values in strings
- updating strings
- Escape Character
- String Special Operator
- String Formatting Operator
- Triple Quotes
- Unicode String
- Built-in String Methods

Python Data Types – Lists

- Creating the lists
- Accessing the values in lists
- Updating Lists
- Deleting List Elements
- Basic List Operators
- Indexing, Slicing and Matrixes
- Built-in List Functions & Methods

Python Data Types – Tuples

- Creating the tuple
- Accessing values in Tuples
- Updating Tuples
- Delete Tuples Elements
- Basic Tuples Operations
- Indexing, Slicing and Matrixes
- Built-in Tuple Functions

Python Data Types – Dictionary

- Creating a Dictionary
- Accessing Values in Dictionary
- Updating Dictionary
- Deleting Dictionary Elements
- Properties of Dictionary Keys
- Built-in Dictionary Functions & Methods

Python Data Types – Sets

- Creating A Set
- Accessing A Set
- Modifying A Set
- Removing Items From A Set
- Common Set Operations
- Built-in Set Functions & Methods

Python Operators

- List of Operators
- Arithmetic Operators
- Comparison Operators
- Assignment Operators
- Bitwise Operators
- Logical Operators
- Membership Operators
- Identity Operators
- Operators Precedence

Decision Making

- Single Statement Suites
- if statements
- if else statements
- if elif statements
- Nested statements

Python Loops

- Various Types of Loops
- while loop
- for loop
- nested loop
- Loop Control Statements
- break statement
- continue statement
- pass statement
- Iterator and Generator

Python Functions

- Defining a Function
- Calling a Function
- Pass by reference vs value
- Function Arguments
- Required Arguments
- Keyword Arguments
- Default Arguments
- Variable-length arguments
- The anonymous Functions
- The return Statement
- Scope of Variables
- Global vs Local Variables

Working with PIP

- Download PIP
- Configure Environment Variables
- Search Modules using PIP

- Download & Install Modules using PIP
- Various examples of PIP
- Downloading, Installing of Various modules

Object-Oriented Programming

- The class statement
- Creating a Class
- Methods
- The self
- "self" the self reference
- Object Methods
- Using Object Methods
- The__init__method (The default constructor)
- Using the__init__method Private(__var) & Semi-private variables(_var)
- Class and Object Variables
- Using Class and Object Variables

Inheritance

- Using Inheritance
- Multiple-inheritance
- Operator Overloading
- operator — Standard operators as functions

collections.abc — Abstract Base Classes for Containers

- Errors and Exceptions
- User-defined Exceptions
- Defining Clean-up Actions
- Predefined Clean-up Actions

Functional Programming

- itertools — Functions creating iterators for efficient looping
- functools — Higher-order functions and operations on callable objects
- operator — Standard operators as functions
- Decorators
- List Comprehensions - Nested List Comprehensions
- Looping Techniques
- More on Conditions
- Comparing Sequences and Other Types
- Iterators
- Generators

The import statement

- The from import statement
- The from import * statement
- Executing modules as scripts
- Locating Modules

Reading and Writing Files

- The write() method
- The read() method
- File Positions
- Renaming and Deleting Files
- The rename() Method & remove() method

Directories in Python

- The mkdir() method
- The chdir() method
- The getcwd() method
- The rmdir() method
- File & Directory Related Methods

Error & Exceptions Handling in Python

- Assertions in Python
- The assert statement
- What is an exception?
- The except clause with no exceptions
- The except clause with multiple exceptions
- The try-finally clause
- Argument of an exception
- Raising an exception
- User-defined exceptions

Working with the Databases

- What is MySQL?
- Database Connection
- Creating Database Table
- Insert operation
- Read operation
- Update operation
- Delete operation
- Performing Transactions
- Commit Operation
- Rollback Operation
- Error Handling

Handling HTTP & HTTPS

- Introduction to HTTP Protocol
- Different HTTP Method GET, POST,PUT
- HTTP Requests
- HTTP Response
- HTTP Headers
- Custom HTTP Requests
- Request Status Codes
- HTTP Authentication
- HTTP Data Download
- Implement using requests library

Python GUI – PyQT and Tkinter

- Installing Python 3 and Tcl/Tk, PyQt for Mac or Windows
- Creating and configuring themed Tk widgets
- Decorating the GUI with text labels and images
- Capturing input from buttons, menus, and entry fields
- Presenting choices with check boxes and radio buttons
- Using geometry managers to lay out the GUI
- Organizing widgets inside of frames and windows
- Handling user actions with event-driven programming

Flask

- Introduction To Flask For Beginners
- Flask Template, Form, View, And Redirect

- Flask Database Handling
- Flask App And Flask Project Layout With Blueprint & Bootstrap
- Flask Design Patterns And Best Practices For Web Applications
- Flask API
- Extending Flask With APIs

Mathematical Computation by numpy

- NumPy Overview
- Properties, Purpose, and Types of ndarray
- Class and Attributes of ndarray Object
- Basic Operations: Concept and Examples
- Accessing Array Elements: Indexing, Slicing, Iteration, Indexing with Boolean Arrays
- Copy and Views
- Universal Functions
- Shape Manipulation & Broadcasting
- Linear Algebra using numpy
- Stacking and resizing the array

Data Analysis with Pandas

- Introduction to Pandas
- Data Structures
- Series & DataFrame
- DataFrame basic properties
- Importing excel sheets, csv files, loading data from html
- Importing and exporting json files
- Selection of columns
- Filtering Dataframes
- Descriptive Analysis with pandas
- Data Cleaning Handling Missing Values
- Handling unwanted columns
- Handling outliers
- Handling duplicated entries
- Finding unique values
- Creating new categorical features from continuous variable
- Grouby operations
- Grouby statistical Analysis
- Apply method

Data Visualization

- Introduction to Data Visualization
- Python Libraries
- Data Visualization Best practices
- Matplotlib Features
- Line Properties Plot with (x, y)
- Controlling Line Patterns and Colors
- Set Axis, Labels, and Legend Properties
- Alpha and Annotation
- Multiple Plots
- Subplots
- Scatterplots
- Pie Charts
- barplots
- Types of Plots and Seaborn
- Boxplots
- Distribution Plots

- Heatmaps
- Swarmplots and countplots
- Pointplots

Case Study – using NumPy, Pandas, matplotlib and seaborn – EDA

- Customer Chun Dataset
- Insurance Dataset
- Drug Dataset
- CCPP Dataset

How to design Analytic Report

Exploratory Analytics with Data Visualization and Statistics

- Univariate Analytics
- Bivariate Analytics
- Multivariate Analytics

Regular Expression

- Regular Expression Syntax
- Example of w+ and ^ Expression
- Example of \s expression in re.split function
- Using regular expression methods
- Using re.match()
- Finding Pattern in Text (re.search())
- Using re.findall for text
- Python Flags
- Example of re.M or Multiline Flags

Advanced Python (Part-2 Prerequisite for Parallel Computing)

Data Serialization

- Serialization using JSON
- Serialization using Pickle

Multiprocessing

- Running Two Simple Processes
- Using Pool and Map

Multithreading

- Basics of multithreading
- Communicating between threads
- Creating a worker pool
- Advanced use of multithreads
- Stoppable Thread with a while Loop

Processes and Threads

- Global Interpreter Lock
- Running in Multiple Threads
- Running in Multiple Processes
- Sharing State Between Threads
- Sharing State Between Processes

Getting Started with Parallel Computing and Python

The parallel computing memory architecture

- SISD
- MISD
- SIMD
- MIMD

Memory organization

- Shared memory
- Distributed memory

    - Massively parallel processing
    - A cluster of workstations
    - The heterogeneous architecture

Parallel programming models

- The shared memory model
- The multithread model
- The message passing model
- The data parallel model

How to design a parallel program

- Task decomposition
- Task assignment
- Agglomeration
- Mapping
    - Dynamic mapping
        - Manager/worker
        - Hierarchical manager/worker
        - Decentralize

How to evaluate the performance of a parallel program

- Speedup
- Efficiency
- Scaling
- Amdahl's law
- Gustafson's law

Python in a parallel world

Introducing processes and threads

Start working with processes in Python

Start working with threads in Python

Thread-based Parallelism

- Using the Python threading module
- Define a thread
- Determine the current thread
- Use a thread in a subclass
- Thread synchronization with Lock and RLock
- Thread synchronization with RLock
- Thread synchronization with semaphores
- Thread synchronization with a condition
- Thread synchronization with an event
- Using the with statement
- Thread communication using a queue
- Evaluating the performance of multithread applications

Process-based Parallelism

- Spawn a process
- Name a process
- Run a process in the background
- Kill a process
- Use a process in a subclass
- Exchange objects between processes
    - Using queue to exchange objects
    - Using pipes to exchange objects
- Synchronize processes
- Manage a state between processes
- Use a process pool
- Using the mpi4py Python module
- Point-to-point communication
- Avoiding deadlock problems

- Collective communication using broadcast
- Collective communication using scatter
- Collective communication using gather
- Collective communication using Alltoall
- The reduction operation
- How to optimize communication

Asynchronous Programming

- Using the concurrent.futures Python modules
- Dealing with the process and thread pool
- Event loop management with Asyncio
- Handling coroutines with Asyncio
- Task manipulation with Asyncio
- Dealing with Asyncio and Futures

Distributed Python

- Using Celery to distribute tasks
- Create a task with Celery
- Scientific computing with SCOOP
- Handling map functions with SCOOP
- Remote Method Invocation with Pyro4
- Chaining objects with Pyro4
- Developing a client-server application with Pyro4
- Communicating sequential processes with PyCSP
- Using MapReduce with Disco
- .Remote procedure call with RPyC

GPU Programming with Python

- Using the PyCUDA module
- A hybrid programming model
- The kernel and thread hierarchy
- Build a PyCUDA application
- Understanding the PyCUDA memory model with matrix manipulation
- Kernel invocations with GPUArray
- Evaluating element-wise expressions with PyCUDA
- The MapReduce operation with PyCUDA
- GPU programming with NumbaPro
- Using GPU-accelerated libraries with NumbaPro
- Using the PyOpenCL module
- Build a PyOpenCL application
- Evaluating element-wise expressions with PyOpenCl
- Testing your GPU application with PyOpenCL