Python Programming

Day 1:

Module 1: Introduction to Python (9:00 AM to 11:00 AM)

- Introduction to Python
- Overview Python Based Applications
- Environment Setup
- Data Types
- Variables and its Scope
- Data Structure
- Operations on Data Structure
- Input and Output Operation
- Writing a Python Module

Module 2: Functions

- Define and use custom functions within a Python program
- Types of function
- Types of Arguments

**********20 Minutes Tea Break**********

Module 2: Functions (continued) (11:20 AM to 1:10 PM)

- Naming conventions
- Using Imports
- Documentation
- Executing Modules as Scripts
- Map Function
- Filter Function
- Reduce Function
- Extended Keyword Arguments (*args, **kwargs)
- Lambda Functions
- Decorators

**********50 Minutes **********

Module 3: Iterables and Conditional Statement (2:00 PM to 3:45 PM)

- Definitions
- Sequences
- Unpacking Sequences
- Dictionaries
- The len() function
- Sets

**********15 Minutes Break**********

4:00 PM to 5:00 PM (Lab Session)

Employee Data Processing System

The HR department at your company has asked you to create a data processing system to help manage employee records.

The system needs to process employee data, including names, departments, salaries, and quarterly performance scores.

This data will be used to determine salary adjustments, filter employees based on performance, and generate departmental reports.

Your task is to develop a Python program that:

1. Combines and filters employee records: Takes separate lists for employee names, departments, salaries, and performance scores,

and combines them into a list of dictionaries. Each dictionary should represent an employee and include only those with a performance

score above a certain threshold.

2. Applies salary adjustments: Increases the salary for employees based on their performance score. Employees with a score above 90

receive a 10% raise, and those with scores between 75 and 90 receive a 5% raise.

3. Demonstrates deep and shallow copying: Creates both shallow and deep copies of the list of employee dictionaries. You'll modify

the original list and observe how the changes affect each copy to understand the difference between deep and shallow copies.

4. Aggregates salaries by department: Calculates the total salary expenditure per department after adjustments, returning a summary

report with department names and total salaries.

Day 2:

Conditional Statements(9:00 AM to 11:00 AM)

- Loops in Python
- break and continue
- The enumerate() Function
- Generators
- List Comprehensions
- Advanced List Comprehensions
- Collections Module
- Mapping and Filtering
- Mutable and Immutable Built-in Objects
- Sorting
- Unpacking Sequences in Function Calls

Module 4: Modules

- What are modules
- General Format
- Importing Modules
- Executing functions from other modules
- The __name__ variable

**********20 Minutes Tea Break**********

Module 5: Python Strings(11:20 AM to 1:10 PM)

- Quotation Marks and Special Characters
- String Indexing
- Slicing Strings
- Concatenation and Repetition

**********50 minutes Lunch Break **********

Module 5: Python Strings (contd.) (2:00 PM to 3:45 PM)

- Common String Methods
- String Formatting
- Namespaces

- Formatted String Literals (f-strings and .format() method)
- Built-in String Functions

**********15 Minutes Break**********

4:00 PM to 5:00 PM (Lab Session)

Problem Statement: Imagine you are working for a retail company that operates multiple stores in different cities.

The company records daily sales data for each store, including the date, city, product name, quantity sold, and total revenue generated. Your task is to perform data analysis on this sales data using Python's map(), filter(), and reduce() functions.

The sales data is provided in the form of a list of dictionaries, where each dictionary represents the sales record for a specific day. The dictionary contains the following key-value pairs:

sales_data = [{"date": "2023-07-15", "city": "New York", "product": "Laptop", "quantity": 10, "revenue": 5000},{"date": "2023-07-15", "city": "Chicago", "product": "Phone", "quantity": 20, "revenue": 3000},{"date": "2023-07-16", "city": "New York", "product": "Tablet", "quantity": 5, "revenue": 2000},]

Tasks:

1. Calculate the total revenue generated for each city on a specific date.

2. Find the total quantity sold for a particular product across all cities and dates.

3. Filter the sales data to include only records for a particular city.'''

Day 3:

Module 6: Python Dates and Times (9:00 AM to 9:45 AM)

- Understanding Time
- The time Module
- The datetime Module

Module 7: Math (9:45 AM to 11:00 AM)

- Arithmetic Operators
- Assignment Operators
- Built-in Math Functions
- The math Module
- The random Module

**********20 Minutes Coffee Break**********

Module 8: File Processing (11:20 AM to 12:30 PM)

- Opening Files

- The os and os.path modules
- Reading files
- Writing into a file
- Appending data into a file

Module 9: Exception Handling(12:30 PM to1:10 PM)

- except Clauses
- The else clause
- The finally clause
- Using Exceptions For Flow Control

**********50 minutes Lunch Break**********

Module 11: Programmer tools(2:00 PM to 2:30 PM)

- Analyzing programs
- Using pylint
- Testing code
- Using unittest
- Debugging

Module 10: OOPS in Python (2:30 PM to 4:00 PM)

- Introduction to Object-Oriented Python
- Creating Classes, Methods, and Objects
- Using Constructor and Attributes
- Using Class Attributes and Static Methods
- Understanding Object Encapsulation
- Private Attributes and Methods
- Inheritance
- Properties in Classes
- Abstract Methods and Classes

**********15 minutes Break**********

4:15 PM to 5:00 PM (Lab Session)

Create a text file having feedback of all the clients

Tasks:

1. Open the file using python program and analyse the file to generate How client feels about the product.
2. Print on screen the good feedback provided by client
3. Print the most frequent feedback given by client

Create a class representing a luxury watch;

- The class should allow you to hold a number of watches created in the watches_created class variable. The number could be fetched using a class method name get_number_of_watches_created;

- the class may allow you to create a watch with a dedicated engraving (text). As this is an extra option, the watch with the engraving should be created using an alternative constructor (a class method), as a regular __init__ method should not allow ordering engravings;

- the regular __init__ method should only increase the value of the appropriate class variable;

The text intended to be engraved should follow some restrictions:

- it should not be longer than 40 characters;

- it should consist of alphanumerical characters, so no space characters are allowed;

- if the text does not comply with restrictions, an exception should be raised;

Before engraving the desired text, the text should be validated against restrictions using a dedicated static method.

- Create a watch with no engraving

- Create a watch with correct text for engraving

- Try to create a watch with incorrect text, like 'foo@baz.com'. Handle the exception

- After each watch is created, call class method to see if the counter variable was increased

Day 4:

Module 12: NumPy (9:00 AM to 11:00 AM)

•Introduction to NumPy

•Installation

•NumPy Arrays (numpy.array)

•Array Indexing and Slicing

•Array Shape and Reshaping

•Array Operations (Sum, Mean, etc.)

•numpy.arange and numpy.linspace

•Array Stacking and Splitting

•Data Type Promotion

•Formation of 1D, 2D and 3D arrays

•Exploring dimension type, shape and size of array

•Exploring dimension type, shape and size of array

•Indexing of NumPy Arrays

•Slicing of NumPy Arrays

**********20 minutes Coffee Break**********

Module 12: NumPy (contd.) (11:20 AM to 1:10 PM)

•One's Matrix and Zero's Matrix

•Identity Matrix

•min() and argmin()

•max() and argmax()

•np.sort()

•np.argsort()

•Addition of matrices

•Multiplication of matrices

**********50 Minutes Lunch Break**********

Module 13: Pandas(2:00 PM to 3:30 PM)

•Introduction to Pandas

•Series

•DataFrames

•Missing Data

•Merging Joining and Concatenating

•Operations

•Data Input and Output

•Groupby()

•Pivoting

•VLookup

**********15 minutes Break**********

Module 12: Web Scraping and Data Extraction(3:45 PM – 4:00 PM)

- Introduction to web scraping
- Using BeautifulSoup and Requests libraries for web scraping

4:00 PM to 5:00PM (Lab Session)

Final Project

Mini Project

Problem 1: Online Shopping Cart System

Create an Online Shopping Cart System. The system should allow users to add products to their cart, view the cart contents,

calculate the total cost, and complete the purchase.

Create the following classes to model the Online Shopping Cart System:

Product: A class representing a product with attributes such as name, price, and quantity.

ShoppingCart: A class representing the shopping cart with methods to add products, view cart contents, and calculate the total cost.

User: A class representing a user who can have a shopping cart.

The classes should interact as follows:

Each Product object should have a name, price, and quantity.

The ShoppingCart object should be associated with a specific User.

The ShoppingCart should have methods to add products (and quantities) to the cart, view the cart contents with product details,

and calculate the total cost of the items in the cart.

The User class should have a method to check out, which completes the purchase and empties the shopping cart.'''

Problem 2: Insurance Policy Management System

Develop an Insurance Policy Management System for an insurance company. The system should allow the company to manage different types of insurance policies, enroll new policyholders, process policy claims, and calculate premiums.

Create the following classes to model the Insurance Policy Management System:

- Policy: A base class representing a generic insurance policy with attributes such as policy number, policyholder name, and premium amount.

- LifeInsurance: A subclass of Policy representing a life insurance policy, which includes additional attributes like the insured person's age and the policy term.

- HealthInsurance: A subclass of Policy representing a health insurance policy, which includes additional attributes like the policyholder's pre-existing medical conditions and coverage type.

- InsuranceCompany: A class representing the insurance company, which contains a list of policies, methods to enroll new policyholders, process claims, and calculate premiums.

- The classes should interact as follows:

The Policy class should have a constructor to initialize the common attributes (policy number, policyholder name, and premium amount) and a method to display policy details.

The LifeInsurance and HealthInsurance classes should inherit from the Policy class and extend it with additional attributes specific to each insurance type. They should also override the display policy details method to include their specific attributes.

The InsuranceCompany class should have a list to store all the policies and methods to enroll new policyholders, process claims, and calculate premiums based on policy type and specific attributes.'''