Python Coding Challenge – Scenario-Based Questions

Day 37 – Introduction to Programming Languages and Python

Project Title: *Student Profile Creator*

Scenario:
You are helping to build a basic student information system for a local school. The school wants to store some basic details about each student and let them update their interests.

Tasks:

1. Create variables to store the student's name, age, and their favourite subjects they are enrolled in.
2. Display the student's information in a user-friendly format.
3. Display the updated list of subjects.

Day 38 – String Methods and Slicing

Project Title: *Feedback Analyzer*

Scenario:
A content review team wants to analyse user feedback and comments. They want to check how frequently certain words appear and extract parts of the feedback for reporting.

Tasks:

1. Ask the user to input a sentence (representing feedback).
2. Count how many words are in the sentence.
3. Check how many times the word "data" appears in the sentence, regardless of case.
4. Extract and print the first 5 and last 5 characters of the feedback separately.
5. Reverse the sentence and print it.

Day 39 – Tuple, Immutable vs Mutable, List Creation

Project Title: *Retail Inventory Manager*

Scenario:
You are managing an inventory system for a retail shop. Certain product categories are fixed, but the actual products can be added or removed by the staff.

Tasks:

1. Create a tuple with fixed product categories like Electronics, Stationery, and Clothing.
2. Create a list of products that can be updated dynamically.
3. Ask the user to enter a new product and add it to the list.
4. Ask the user to input the index of a product to remove it from the list.
5. Display the final list of products and the product categories.

Day 40 – List Methods and Slicing

Project Title: *Contact Directory*

Scenario:
You have been assigned to build a contact management system where employees can store and retrieve contact details efficiently.

Tasks:

1. Create a 2D list where each row holds a contact's name, phone number, and email address.
2. Allow the user to add a new contact.
3. Remove the last item
4. Get new items from the users using input() and add those items to the list
5. Sort the list in ascending order based on any values

Day 41 – Dictionary, Set, Frozenset

Project Title: *Student Grades & Hobbies Tracker*

Scenario:
A student activity coordinator is collecting information about student grades and hobbies. They want to ensure grades are updated properly and hobby lists are stored efficiently.

Tasks:

1. Create a dictionary with student names as keys and their grades as values.
2. Allow the user to add or update grades for at least 3 students.
3. Create a set of hobbies and allow the user to input at least 3 hobbies.
4. Convert the set of hobbies into a frozenset and display it.

Day 42 – Operators, Conditional Statements

Project Title: *BMI Calculator*

Scenario:
A health organization wants a tool to help users calculate their Body Mass Index (BMI) and understand their health category based on it.

Tasks:

1. Ask the user to enter their weight (in kilograms) and height (in meters).
2. Calculate their BMI using the formula BMI = weight / (height × height).
3. Classify the BMI as underweight, normal, overweight, or obese using conditional statements.
4. Display the result clearly.
5. Ensure the weight and height entered are integer numbers.

Day 43 – While Loop, else, break, continue

Project Title: Shopping Cart System

Scenario:

You are creating a simple billing system where customers can enter the prices of items they want to purchase. The program should keep accepting prices until the user decides to stop, then apply appropriate discounts before generating the final bill.

Tasks:

- Continuously accept item prices using a **while loop**.
- Stop taking input when the user enters **0** (sentinel value).
- Keep track of the **total number of items** and the **subtotal amount**.
- Ensure no negative prices are allowed (ask again without counting as an item).
- Apply discounts based on the subtotal:
    - 20% for totals ≥ 1000
    - 10% for totals ≥ 500
    - 5% for totals ≥ 200
    - 0% for totals < 200
- Display a clear bill summary including subtotal, discount amount, and final total.

Day 44 – range, for loop, enumerate, list comprehension

Project Title: *Employee Hours Report*

Scenario:
You are analyzing student exam scores to generate performance reports.

Tasks:

1. Create a list of 10 random exam scores ranging from 0 to 100 using list comprehension.
2. Display each score with its index using the enumerate function.
3. Find all scores greater than 75 and display them.
4. Calculate and display the average score.

Day 45 – Built-in Functions and User-Defined Functions

Project Title: *Simple Calculator App*

Scenario:
A company wants to provide users with a simple calculator to perform basic mathematical operations.

Tasks:

1. Create functions for addition, subtraction, multiplication, and division.
2. Ask the user to input two numbers and the operation they want to perform.
3. Call the appropriate function and display the result.
4. Use built-in functions like abs(), round(), and pow() within your solution.
5. Handle division by zero gracefully and inform the user.

Day 46 – Function Arguments, Local and Global Namespace, Introduction to OOP

Project Title: *Bank Account Manager*

Scenario:
A bank wants to track account balances and transactions for each customer in an object-orientated way.

Tasks:

1. Create a class BankAccount with account_holder and balance attributes.
2. Implement methods for deposit(), withdraw(), and display_balance().
3. Use both local and global variables appropriately, such as a global transaction counter.
4. Demonstrate calling methods using both positional and keyword arguments.

5. Prevent withdrawals that would result in a negative balance.

Day 47 – Revision + Coding Challenge

Project Title: *Student Management System*

Scenario:

You are tasked with building a **Student Management System** that integrates multiple Python concepts to help staff manage students' data, performance, and reporting.

Tasks:

1. Use dictionaries and lists to store and retrieve student information such as name, age, and grades.

2. Allow users to add, delete, update, and search for students.

3. Track grades and categorize performance using functions and conditionals.

4. Use tuples or frozensets to store constant data like available classes.

5. Optionally implement file handling to save and load student data between program runs.
6. Use exception handling to manage invalid inputs.
7. Implement sorting and filtering of students for reports.