ASSIGNMENT-6.4

Name: Sindhu Katakam

Batch-41

Enrollment number:2303A52481

Task 1: Student Performance Evaluation System

Prompt:

```
"""create a class named Student with attributes name,roll_number,marks.add a method that displays
 the student details use self attributes ,and add another method that checks whether the students
 marks are above the class average using conditional statements and returns an appropriate message.
output has to display the student details and performance status in neat and organised way.
take data of 10 students."""
```

Code:

```
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def display_details(self):
        print(f"Name: {self.name}, Roll Number: {self.roll_number}, Marks: {self.marks}")

    def check_performance(self, class_average):
        if self.marks > class_average:
            return f"{self.name} has scored above the class average."
        else:
            return f"{self.name} has scored below the class average."
# List to hold student objects
students = [
    Student("Alice", "101", 85.0),
    Student("Bob", "102", 78.5),
    Student("Charlie", "103", 92.0),
    Student("David", "104", 88.5),
    Student("Eva", "105", 79.0),
    Student("Frank", "106", 84.5),
    Student("Grace", "107", 90.0),
    Student("Hannah", "108", 87.5),
    Student("Ian", "109", 82.0),
    Student("Jane", "110", 91.0)
]
```

```python
# Calculate class average
total_marks = sum(student.marks for student in students)
class_average = total_marks / len(students)
print(f"\nClass Average Marks: {class_average}\n")
# Display details and performance of each student
for student in students:
    student.display_details()
    print(student.check_performance(class_average))
    print()
```

Output:

```
PS D:\AI_ASSIT_CODING> & C:/Users/MYSELF/AppData/Loc

Class Average Marks: 85.8

Name: Alice, Roll Number: 101, Marks: 85.0
Alice has scored below the class average.

Name: Bob, Roll Number: 102, Marks: 78.5
Bob has scored below the class average.

Name: Charlie, Roll Number: 103, Marks: 92.0
Charlie has scored above the class average.

Name: David, Roll Number: 104, Marks: 88.5
David has scored above the class average.

Name: Eva, Roll Number: 105, Marks: 79.0
Eva has scored below the class average.
```

Explanation:

This program creates a Student class to store a student's name, roll number, and marks. It makes a list of students using this class. Then it finds the class average by adding all marks and dividing by the number of students. After that, it shows each student's details. Finally, it tells whether each student scored above or below the class average.

Task 2: Data Processing in a Monitoring System

Prompt:

```
"""You are working on a basic data monitoring script where sensor readings are collected
as numbers.use for loop to iterate over a list identify the even numbers using modulus
operator then calculate their squares and print the result in a readable format"""
```

Code:

```
sensor_readings = [12, 7, 9, 20, 15, 8, 30, 25, 14, 6]
print("Even Sensor Readings and their Squares:")
for reading in sensor_readings:
    if reading % 2 == 0:
        square = reading ** 2
        print(f"Sensor Reading: {reading}, Square: {square}")
print()
```

Output:

```
PS D:\AI_ASSIT_CODING> & C:/Users/MYSELF/AppData/Local/
Even Sensor Readings and their Squares:
Sensor Reading: 12, Square: 144
Sensor Reading: 20, Square: 400
Sensor Reading: 8, Square: 64
Sensor Reading: 30, Square: 900
Sensor Reading: 14, Square: 196
Sensor Reading: 6, Square: 36
```

Explanation:

This program works with a list of sensor readings. It checks each reading one by one to see if it is an even number. If the reading is even, the program calculates its square. Then it prints the sensor reading along with its square. Odd numbers are skipped and not shown.

Task 3: Banking Transaction Simulation

Prompt:

```
"""create a class named BankAccount with the attributes account_holder and balance.add methods
for depositing money,withdrawing money and preventing withdrawals when the balance is
insufficient by user friendly messages.use self attributes and if else conditions """
```

Code:

```python
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: {amount}. New Balance: {self.balance}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance for this withdrawal.")
        elif amount <= 0:
            print("Withdrawal amount must be positive.")
        else:
            self.balance -= amount
            print(f"Withdrew: {amount}. New Balance: {self.balance}")
# Example usage
account = BankAccount("John Doe", 1000)
account.deposit(500)
account.withdraw(200)
account.withdraw(2000)
account.withdraw(-50)
account.deposit(-100)
```

Output:

```
PS D:\AI_ASSIT_CODING> & C:/Users/MYSELF/AppData/Local/
Deposited: 500. New Balance: 1500
Withdrew: 200. New Balance: 1300
Insufficient balance for this withdrawal.
Withdrawal amount must be positive.
Deposit amount must be positive.
Students scoring more than 75:
Alice
Charlie
Eva
Frank
Grace
Ian
Jane
```

Explanation:

This program creates a `BankAccount` class to manage a person's bank account. It stores the account holder's name and balance. The deposit method adds money if the amount is positive. The

withdraw method removes money only if there is enough balance and the amount is valid. It also shows messages for successful actions or errors like low balance or invalid amounts.

Task 4: Student Scholarship Eligibility Check

Prompt

```
"""create alist of dictionaries where each dictionary represents a students
with their name and score. use while loop to iterate through the list and then
list out the names of students who scores more than 75. """
```

Code:

```
students_scores = [
    {"name": "Alice", "score": 82},
    {"name": "Bob", "score": 74},
    {"name": "Charlie", "score": 91},
    {"name": "David", "score": 68},
    {"name": "Eva", "score": 77},
    {"name": "Frank", "score": 85},
    {"name": "Grace", "score": 79},
    {"name": "Hannah", "score": 73},
    {"name": "Ian", "score": 88},
    {"name": "Jane", "score": 90}
]

index = 0
print("Students scoring more than 75:")
while index < len(students_scores):
    if students_scores[index]["score"] > 75:
        print(students_scores[index]["name"])
    index += 1
print()
```

Output:

```
PS D:\AI_ASSIT_CODING> & C:/Users/MYSELF/AppData/Local/Programs/I
Students scoring more than 75:
Alice
Charlie
Eva
Frank
Grace
Ian
Jane
```

Explanation:

This program stores students' names and scores in a list of dictionaries. It uses a while loop to go through each student one by one. For every student, it checks if the score is greater than 75. If the score is above 75, the student's name is printed. The loop continues until all students are checked.

Task 5: Online Shopping Cart Module

Prompt

```
"""create a python class named shoppingcart with an empty list to store items
like name,price and quantity. add methods to add items to the cart,removing items
from the cart, calculating the total bill using a loop.if total exceeds a certain
amount apply discounts. give proper output demonstrating cart functionality"""
```

Code

```
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price, quantity):
        self.items.append({"name": name, "price": price, "quantity": quantity})
        print(f"Added {quantity} of {name} at ${price} each to the cart.")

    def remove_item(self, name):
        for item in self.items:
            if item["name"] == name:
                self.items.remove(item)
                print(f"Removed {name} from the cart.")
                return
        print(f"{name} not found in the cart.")

    def calculate_total(self):
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]
        if total > 100:
            discount = total * 0.1
            total -= discount
            print(f"Applied a discount of ${discount:.2f} for exceeding $100")
        return total
```

```python
# Example usage
cart = ShoppingCart()
cart.add_item("Apple", 2.0, 5)
cart.add_item("Banana", 1.5, 10)
cart.add_item("Orange", 3.0, 4)
cart.add_item("Grapes", 43.0, 3)
cart.remove_item("Banana")
total_bill = cart.calculate_total()
print(f"Total Bill: ${total_bill:.2f}")
print()
```

Output:

```
PS D:\AI_ASSIT_CODING> & C:/Users/MYSELF/AppData/Local/Programs,
Added 5 of Apple at $2.0 each to the cart.
Added 10 of Banana at $1.5 each to the cart.
Added 4 of Orange at $3.0 each to the cart.
Added 3 of Grapes at $43.0 each to the cart.
Removed Banana from the cart.
Applied a discount of $15.10 for exceeding $100
Total Bill: $135.90
```

Explanation:

This program creates a ShoppingCart class to manage items in a cart. It allows adding items with their price and quantity and removing items by name. The program calculates the total cost by multiplying price and quantity for each item. If the total exceeds $100, it applies a 10% discount. Finally, it prints the total bill after applying any discount.