**Assignment 6.4**

Prompt: Start a Python class named Student with attributes name, roll\_number, and marks.complete methods for displaying details and checking if marks are above average.

# 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"Student Name: {self.name}")

        print(f"Roll Number: {self.roll\_number}")

        print(f"Marks: {self.marks}")

    def is\_above\_average(self, average\_marks):

        return self.marks > average\_marks

# Create a student object

student1 = Student("Alice", "A123", 85)

# Display student details

student1.display\_details()

# Check if the student is above average

average\_marks = 75

if student1.is\_above\_average(average\_marks):

    print(f"{student1.name} is above the average marks of {average\_marks}.")

else:

    print(f"{student1.name} is not above the average marks of {average\_marks}.")

# output:

Student Name: Alice

Roll Number: A123

Marks: 85

Alice is above the average marks of 75.

Prompt 2:  
Write the first two lines of a for loop to iterate through a list of numbers how to calculate and print the square of even numbers only.

code:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:

    if number % 2 == 0:

        square = number \*\* 2

        print(f"The square of {number} is {square}")

# output:

The square of 2 is 4

The square of 4 is 16

The square of 6 is 36

The square of 8 is 64

The square of 10 is 100

# Prompt3:

• Create a class called BankAccount with attributes account\_holder and balance.complete methods for deposit(), withdraw(), and check for insufficient balance.

# Code:

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 > 0:

            if self.balance >= amount:

                self.balance -= amount

                print(f"Withdrew {amount}. New balance: {self.balance}")

            else:

                print("Insufficient balance.")

        else:

            print("Withdrawal amount must be positive.")

    def check\_balance(self):

        print(f"Current balance: {self.balance}")

# Create a bank account object

account1 = BankAccount("John Doe", 1000)

# Check initial balance

account1.check\_balance()

# Deposit money

account1.deposit(500)

# Withdraw money

account1.withdraw(200)

# Attempt to withdraw more than the balance

account1.withdraw(1500)

# Check final balance

account1.check\_balance()

# output:

Current balance: 1000

Deposited 500. New balance: 1500

Withdrew 200. New balance: 1300

Insufficient balance.

Current balance: 1300

# Prompt 4:

Define a list of student dictionaries with keys name and score. write a while loop to print the names of students who scored more than 75.

# Code:

students = [

    {"name": "Alice", "score": 85},

    {"name": "Bob", "score": 70},

    {"name": "Charlie", "score": 92},

    {"name": "David", "score": 78},

    {"name": "Eve", "score": 65}

]

index = 0

while index < len(students):

    student = students[index]

    if student["score"] > 75:

        print(student["name"])

    index += 1

# output:

Alice

Charlie

David

# Prompt 5:

Begin writing a class ShoppingCart with an empty items list. generate methods to add\_item, remove\_item, and use a loop to calculate the total bill using conditional discounts.

# Code:

class ShoppingCart:

    def \_\_init\_\_(self):

        self.items = []

    def add\_item(self, item\_name, price, quantity=1):

        """Adds an item to the shopping cart."""

        self.items.append({"name": item\_name, "price": price, "quantity": quantity})

        print(f"Added {quantity} x {item\_name} to the cart.")

    def remove\_item(self, item\_name):

        """Removes an item from the shopping cart."""

        initial\_item\_count = len(self.items)

        self.items = [item for item in self.items if item["name"] != item\_name]

        if len(self.items) < initial\_item\_count:

            print(f"Removed {item\_name} from the cart.")

        else:

            print(f"{item\_name} not found in the cart.")

    def calculate\_total(self):

        """Calculates the total bill with conditional discounts."""

        total = 0

        for item in self.items:

            total += item["price"] \* item["quantity"]

        # Apply conditional discounts (example: 10% off if total is over 100)

        if total > 100:

            discount = total \* 0.10

            total -= discount

            print(f"Applied a 10% discount. Discount amount: {discount:.2f}")

        return total

# Create a shopping cart instance

my\_cart = ShoppingCart()

# Add items to the cart

my\_cart.add\_item("Laptop", 1200, 1)

my\_cart.add\_item("Mouse", 25, 2)

my\_cart.add\_item("Keyboard", 75, 1)

my\_cart.add\_item("Laptop", 1200, 1) # Adding another laptop to trigger discount

# Remove an item

my\_cart.remove\_item("Mouse")

# Calculate and print the total

total\_bill = my\_cart.calculate\_total()

print(f"The final total bill is: {total\_bill:.2f}")

# output:

Added 1 x Laptop to the cart.

Added 2 x Mouse to the cart.

Added 1 x Keyboard to the cart.

Added 1 x Laptop to the cart.

Removed Mouse from the cart.

Applied a 10% discount. Discount amount: 247.50

The final total bill is: 2227.50