Assignment 6-4

Task1:

Prompt:

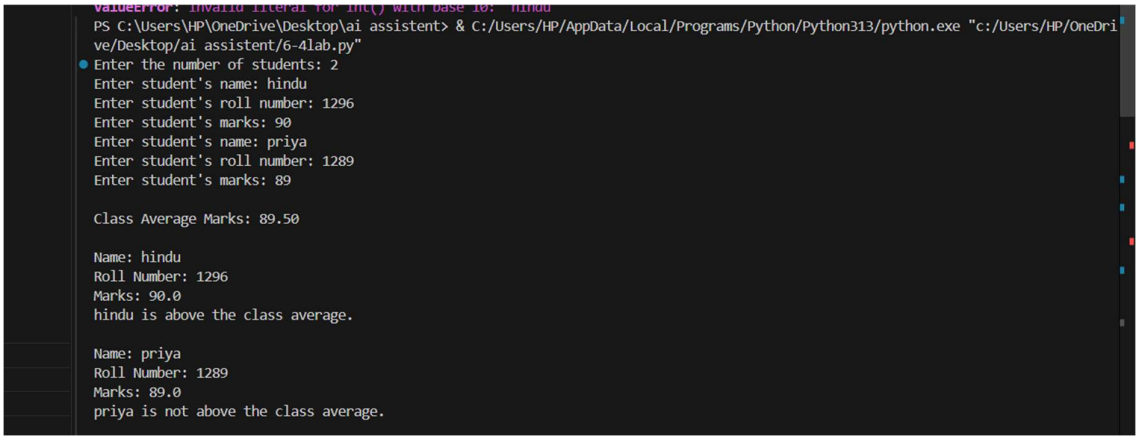
generate a Python code class Student with name, roll number, and marks. Add a method to display details and another method to check if marks are above the class average using if-else. ask input from user.

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}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

    def is_above_average(self, average_marks):
        if self.marks > average_marks:
            return f"{self.name} is above the class average."
        else:
            return f"{self.name} is not above the class average."
# user input for student details
students = []
num_students = int(input("Enter the number of students: "))
for _ in range(num_students):
    name = input("Enter student's name: ")
```

```
    roll_number = input("Enter student's roll number: ")
    marks = float(input("Enter student's marks: "))
    students.append(Student(name, roll_number, marks))
# Calculate class average
total_marks = sum(student.marks for student in students)
average_marks = total_marks / num_students
print(f"\nClass Average Marks: {average_marks:.2f}\n")
# Display details and check if above average
for student in students:
    student.display_details()
    print(student.is_above_average(average_marks))
    print()  # Add a newline for better readability
```

Output:



Analysis:

The code defines a Student class with attributes for name, roll number, and marks. It includes methods to display student details and check if the student's marks are above the class average. The program takes input for multiple students, calculates the class average, and then displays each student's details along with whether they are above the average or not.

Task2:

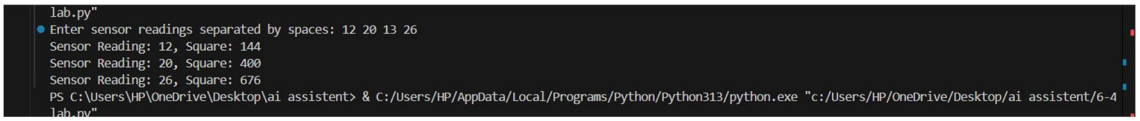
Prompt:

Write a for loop to iterate through sensor readings, identify even numbers using modulus operator, calculate their square, and print the result clearly and user input.

Code:

```
sensor_readings = list(map(int, input("Enter sensor readings separated by spaces: ").split()))

for reading in sensor_readings:

    if reading % 2 == 0:

        square = reading ** 2

        print(f"Sensor Reading: {reading}, Square: {square}")
```

Output:



Analysis:

In this code, we have a list of sensor readings. We use a for loop to iterate through each reading. Inside the loop, we use the modulus operator (%) to check if the reading is even. If it is, we calculate its square by raising it to the power of 2 and then print both the original sensor reading and its square in a clear format.

Task3:

Prompt:

generate a Python class Bank Account with account holder and balance. Add methods to deposit money, withdraw money, and prevent withdrawals when balance is insufficient using if-else user input.

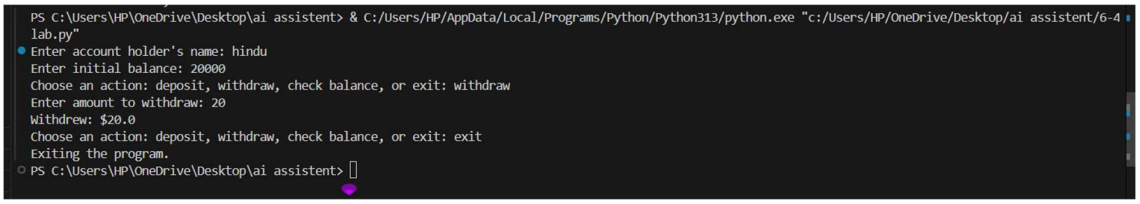
Code:

```
class BankAccount:

    def __init__(self, account_holder, initial_balance=0):

        self.account_holder = account_holder

        self.balance = initial_balance
```

```
    def deposit(self, amount):
        """Deposit a specified amount into the account."""
        if amount > 0:
            self.balance += amount
            print(f"Deposited: ${amount}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        """Withdraw a specified amount from the account if sufficient balance exists."""
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: ${amount}")
        else:
            print("Insufficient balance or invalid withdrawal amount.")

    def check_balance(self):
        """Check and return the current balance of the account."""
        print(f"Current balance: ${self.balance}")
# Demonstration of the BankAccount class
account_holder = input("Enter account holder's name: ")
initial_balance = float(input("Enter initial balance: "))
account = BankAccount(account_holder, initial_balance)
while True:
    action = input("Choose an action: deposit, withdraw, check balance, or exit: ").lower()
    if action == "deposit":
        amount = float(input("Enter amount to deposit: "))
        account.deposit(amount)
```

```
    elif action == "withdraw":
        amount = float(input("Enter amount to withdraw: "))
        account.withdraw(amount)
    elif action == "check balance":
        account.check_balance()
    elif action == "exit":
        print("Exiting the program.")
        break
    else:
        print("Invalid action. Please choose again.")
```

Output:



Analysis:

The Bank Account class has an initializer that sets the account holder's name and initial balance. The deposit method allows adding money to the account, while the withdraw method checks if there is enough balance before allowing a withdrawal. The check_balance method displays the current balance. The demonstration part of the code allows the user to interact with the Bank Account object by choosing actions in a loop until they decide to exit.

Task4:

Prompt:

Using a list of student dictionaries with name and score, write a while loop to print names of students who scored more than 75 user input.

Code:

```
students = []
num_students = int(input("Enter the number of students: "))
for _ in range(num_students):
```

```
    name = input("Enter student's name: ")
    score = float(input("Enter student's score: "))
    students.append({"name": name, "score": score})
index = 0
while index < len(students):
    if students[index]["score"] > 75:
        print(f"Student Name: {students[index]['name']}, Score: {students[index]['score']}")
    index += 1
```

Output:

Analysis:

The code first collects student names and scores from user input and stores them in a list of dictionaries. It then uses a while loop to iterate through the list, checking each student's score. If a student's score is greater than 75, it prints the student's name and score. The loop continues until all students have been checked.

Task5:

Prompt:

Create a Python class ShoppingCart that stores items. Add methods to add items, remove items, calculate total using a loop, and apply discount if total exceeds a limit user input.

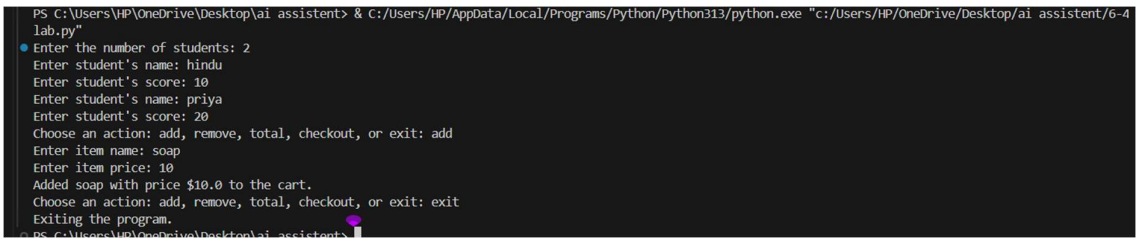
Code:

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

    def add_item(self, item_name, price):
        """Add an item with its price to the shopping cart."""
```

```
        self.items.append({"name": item_name, "price": price})
        print(f"Added {item_name} with price ${price} to the cart.")

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

    def calculate_total(self):
        """Calculate the total price of items in the cart."""
        total = 0
        for item in self.items:
            total += item["price"]
        return total

    def apply_discount(self, discount_threshold, discount_rate):
        """Apply a discount if the total exceeds a certain threshold."""
        total = self.calculate_total()
        if total > discount_threshold:
            discount = total * discount_rate
            total -= discount
            print(f"Discount of ${discount:.2f} applied.")
        return total
# Demonstration of the ShoppingCart class
```

```
cart = ShoppingCart()
while True:
    action = input("Choose an action: add, remove, total, checkout, or exit: ").lower()
    if action == "add":
        item_name = input("Enter item name: ")
        price = float(input("Enter item price: "))
        cart.add_item(item_name, price)
    elif action == "remove":
        item_name = input("Enter item name to remove: ")
        cart.remove_item(item_name)
    elif action == "total":
        total = cart.calculate_total()
        print(f"Current total: ${total:.2f}")
    elif action == "checkout":
        discount_threshold = float(input("Enter discount threshold: "))
        discount_rate = float(input("Enter discount rate (as a decimal): "))
        final_total = cart.apply_discount(discount_threshold, discount_rate)
        print(f"Final total after discount (if applicable): ${final_total:.2f}")
    elif action == "exit":
        print("Exiting the program.")
        break
    else:
        print("Invalid action. Please choose again.")
```

Output:



Analysis:

The Shopping Cart class allows users to manage a shopping cart by adding and removing items, calculating the total price, and applying discounts based on a threshold. The class uses a list to store items as dictionaries containing their names and prices. The demonstration part of the code provides an interactive loop for users to perform various actions on the shopping cart until they choose to exit.