ASSIGNMENT-6.4

Task-1:

#write a python program to create a class named student with attributes name of stud ,roll_number and marks and display details and is_passed methods to the student class and then add code to input details for multiple students

Code:

```python
class student:
    def __init__(self, name_of_stud, roll_number, marks):
        self.name_of_stud = name_of_stud
        self.roll_number = roll_number
        self.marks = marks

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

    def is_passed(self, passing_marks):
        return self.marks >= passing_marks

students_list = []
num_students = int(input("Enter the number of students: "))

for i in range(num_students):
    print(f"\nEnter details for student {i+1}:")
    name = input("Enter student name: ")
    roll = input("Enter roll number: ")
    marks = int(input("Enter marks: "))

    new_student = student(name, roll, marks)
    students_list.append(new_student)

passing_marks = int(input("\nEnter the passing marks: "))

print("\n--- Student Details and Pass Status ---")
for student_obj in students_list:
    student_obj.display_details()
    print(f"Passed: {student_obj.is_passed(passing_marks)}")
    print("-" * 20)
```

Output:

```
Enter the number of students: 2

Enter details for student 1:
Enter student name: Bhavitha
Enter roll number: 099
Enter marks: 98

Enter details for student 2:
Enter student name: Nandhu
Enter roll number: 086
Enter marks: 99

Enter the passing marks: 45

--- Student Details and Pass Status ---
Student Name: Bhavitha
Roll Number: 099
Marks: 98
Passed: True
--------------------
Student Name: Nandhu
Roll Number: 086
Marks: 99
Passed: True
--------------------
```

Explanation:

1. `class student:` : Defines a `student` class with a constructor (`__init__`) taking name, roll number, and marks.
2. `def display_details(self):` : A method to print the student's name, roll number, and marks.
3. `def is_passed(self, passing_marks):` : A method returning `True` if marks are >= `passing_marks`, `False` otherwise.
4. `students_list = []` : Initializes an empty list to store student objects.
5. `num_students = int(input(...))` : Gets the number of students to enter.
6. `for i in range(num_students):` : Loops to get details for each student.
7. `name = input(...)`, `roll = input(...)`, `marks = int(input(...))` : Gets student's name, roll number, and marks.
8. `new_student = student(name, roll, marks)` : Creates a new `student` object.
9. `students_list.append(new_student)` : Adds the new student to the list.
10. `passing_marks = int(input(...))` : Gets the passing marks.
11. `for student_obj in students_list:` : Loops through the list to display details and pass status for each student.

Task-2:

Write a python program to create a list and write first 2 lines in loop and calculate the square of even numbers only .

Code and Output:

```python
[1] # Create a list
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Print the first 2 lines in a loop
print("Printing the first 2 elements:")
for i in range(min(2, len(my_list))):
    print(my_list[i])

# Calculate the square of even numbers
print("\nCalculating the square of even numbers:")
even_squares = []
for number in my_list:
    if number % 2 == 0:
        even_squares.append(number ** 2)

print("Original list:", my_list)
print("Squares of even numbers:", even_squares)
```

```
Printing the first 2 elements:
1
2

Calculating the square of even numbers:
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Squares of even numbers: [4, 16, 36, 64, 100]
```

Explanation:

- `my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` : This line creates a list named `my_list` containing integers from 1 to 10.
- `print("Printing the first 2 elements:")` : This line prints a descriptive header before the loop output.
- `for i in range(min(2, len(my_list))):` : This is a `for` loop that iterates a specific number of times. `len(my_list)` gets the length of the list (which is 10). `min(2, len(my_list))` ensures that the loop runs a maximum of 2 times, or fewer if the list has less than 2 elements. The `range()` function generates numbers from 0 up to (but not including) the value returned by `min()`.
- `print(my_list[i])` : Inside the loop, this line prints the element of `my_list` at the index `i`. Since the loop runs for `i` values of 0 and 1 (at most), it prints the first two elements of the list.
- `print("\nCalculating the square of even numbers:")` : This line prints another descriptive header. The `\n` creates a new line for better formatting.
- `even_squares = []` : This line initializes an empty list called `even_squares`. This list will store the squares of the even numbers found in `my_list`.
- `for number in my_list:` : This is another `for` loop that iterates through each element in `my_list`, assigning the current element to the variable `number` in each iteration.
- `if number % 2 == 0:` : Inside this loop, this is an `if` statement that checks if the current `number` is even. The modulo operator (`%`) gives the remainder of a division. If a number divided by 2 has a remainder of 0, it's an even number.
- `even_squares.append(number ** 2)` : If the `if` condition is true (the number is even), this line calculates the square of the `number` (`number ** 2`) and adds it to the `even_squares` list using the `append()` method.
- `print("Original list:", my_list)` : This line prints the original `my_list`.
- `print("Squares of even numbers:", even_squares)` : This line prints the `even_squares` list, which contains the squares of only the even numbers from the original list.

Task-3:

Write a python program to create a class named bank account with attributes like account_holder,balance.

Write a python program to check deposit,withdrawand check insufficient balance.

Code and Output:

```
[2] class BankAccount:
        def __init__(self, account_holder, initial_balance=0):
            self.account_holder = account_holder
            self.balance = initial_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 for {self.account_holder}: {self.balance}")
```

```
# Example usage:
account1 = BankAccount("Alice", 1000)
account1.check_balance()
account1.deposit(500)
account1.withdraw(200)
account1.withdraw(1500) # Test insufficient balance
account1.check_balance()
```

```
Current balance for Alice: 1000
Deposited 500. New balance: 1500
Withdrew 200. New balance: 1300
Insufficient balance.
Current balance for Alice: 1300
```

Explanation:

The code defines a `BankAccount` class. It has an `__init__` method to create accounts with a holder name and optional initial balance. The `deposit` method adds a positive amount to the balance. It prints the new balance after a successful deposit. The `withdraw` method subtracts a positive amount if sufficient balance exists. It prints the new balance after a successful withdrawal. If the balance is insufficient for withdrawal, it prints an error message. The `check_balance` method prints the current balance for the account holder. Example usage demonstrates creating an account, checking balance, depositing, and withdrawing. It also shows the insufficient balance case.

Task-4:

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

Code and Output:

Assignment 6.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Start coding or generate with AI.

[1]
```python
students = [
    {"name": "Alice", "score": 85},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 92},
    {"name": "David", "score": 78},
    {"name": "Eve", "score": 65},
]

print("Students with scores greater than 75:")
i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

```
Students with scores greater than 75:
Alice
Charlie
David
```

Explanation:

The code defines a list called `students`, where each item in the list is a dictionary containing a student's name and their score. It then goes through each student in the list using a `while` loop. Inside the loop, it checks if the student's score is greater than 75. If it is, the code prints the name of that student.

Task-5:

Write a python code beginning with shopping cart with empty list and generate methods to add item,remove item,using loop to calculate the total bill using conditional discounts

Code:

Assignment 6.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[4]

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

    def add_item(self, name, price, quantity=1):
        for item in self.items:
            if item['name'].lower() == name.lower():
                item['quantity'] += quantity
                return
        self.items.append({'name': name, 'price': price, 'quantity': quantity})

    def remove_item(self, name):
        self.items = [item for item in self.items if item['name'].lower() != name.lower()]

    def calculate_total(self):
        total = 0
        print("\n🛒 Cart Summary:")
        for item in self.items:
            name = item['name']
            price = item['price']
            quantity = item['quantity']

            # Bulk discount
            if quantity > 5:
                print(f"Bulk discount applied to {name}")
                price -= 1  # ₹1 off per item

            item_total = price * quantity
            total += item_total
            print(f"{name} - {quantity} x ₹{price:.2f} = ₹{item_total:.2f}")
```

Assignment 6.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[4]

```
        # General discount
        if total > 1000:
            print("General discount applied: 10% off on total above ₹1000")
            total *= 0.9

        print(f"Total bill: ₹{total:.2f}")
        return total

    def main():
        cart = ShoppingCart()

        while True:
            print("\nChoose an action:")
            print("1. Add item")
            print("2. Remove item")
            print("3. View total bill")
            print("4. Exit")

            choice = input("Enter your choice (1-4): ").strip()

            if choice == '1':
                name = input("Enter item name: ").strip()
                try:
                    price = float(input("Enter price in ₹: ").strip())
                    quantity = int(input("Enter quantity: ").strip())
                    cart.add_item(name, price, quantity)
                    print(f"{quantity} x {name} added to cart.")
                except ValueError:
                    print("Invalid input. Please enter valid numbers for price and quantity.")

            elif choice == '2':
```

Assignment 6.4.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[4]

```
            elif choice == '2':
                name = input("Enter item name to remove: ").strip()
                cart.remove_item(name)
                print(f"{name} removed from cart (if it existed).")

            elif choice == '3':
                cart.calculate_total()

            elif choice == '4':
                print("Exiting. Thank you for shopping!")
                break

            else:
                print("Invalid choice. Please select a valid option.")

    if __name__ == "__main__":
        main()
```

Output:

```
Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 1
Enter item name: laptop
Enter price in ₹: 60000
Enter quantity: 5
5 x laptop added to cart.

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 1
Enter item name: mouse
Enter price in ₹: 2000
Enter quantity: 5
5 x mouse added to cart.

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 2
Enter item name to remove: mouse
mouse removed from cart (if it existed).

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 3

🛒 Cart Summary:
laptop - 5 x ₹60000.00 = ₹300000.00
General discount applied: 10% off on total above ₹1000
Total bill: ₹270000.00

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 4
Exiting. Thank you for shopping!
```

Explanation:

1. A **class** `ShoppingCart` is created to manage cart operations.
2. The `__init__` method initializes an empty list `items` to store cart items.
3. The `add_item` method adds a product with name, price, and quantity.
4. It checks if the item already exists (case-insensitive) and updates the quantity if found.
5. If the item does not exist, it appends a new dictionary with name, price, and quantity.
6. The `remove_item` method removes an item by filtering out the matching name (case-insensitive).
7. The `calculate_total` method calculates the bill and applies discounts.
8. It iterates over all items and prints each item's details in the cart.
9. If quantity > 5, a bulk discount of ₹1 per item is applied.
10. The price after discount is multiplied by quantity to get `item_total`.
11. All `item_total` values are summed into `total`.
12. If `total > 1000`, a general discount of **10%** is applied.
13. The total bill is displayed with proper formatting.
14. The `main` function starts an infinite loop to interact with the user.
15. It displays a menu with 4 options: Add, Remove, View Total, Exit.
16. For **choice 1**, the user inputs item details, and `add_item` is called.
17. For **choice 2**, the user enters an item name, and `remove_item` is called.
18. For **choice 3**, the `calculate_total` method is called to display the bill.
19. For **choice 4**, the program exits with a thank-you message.
20. The program runs only if `__name__ == "__main__"` to ensure it starts from `main()`.