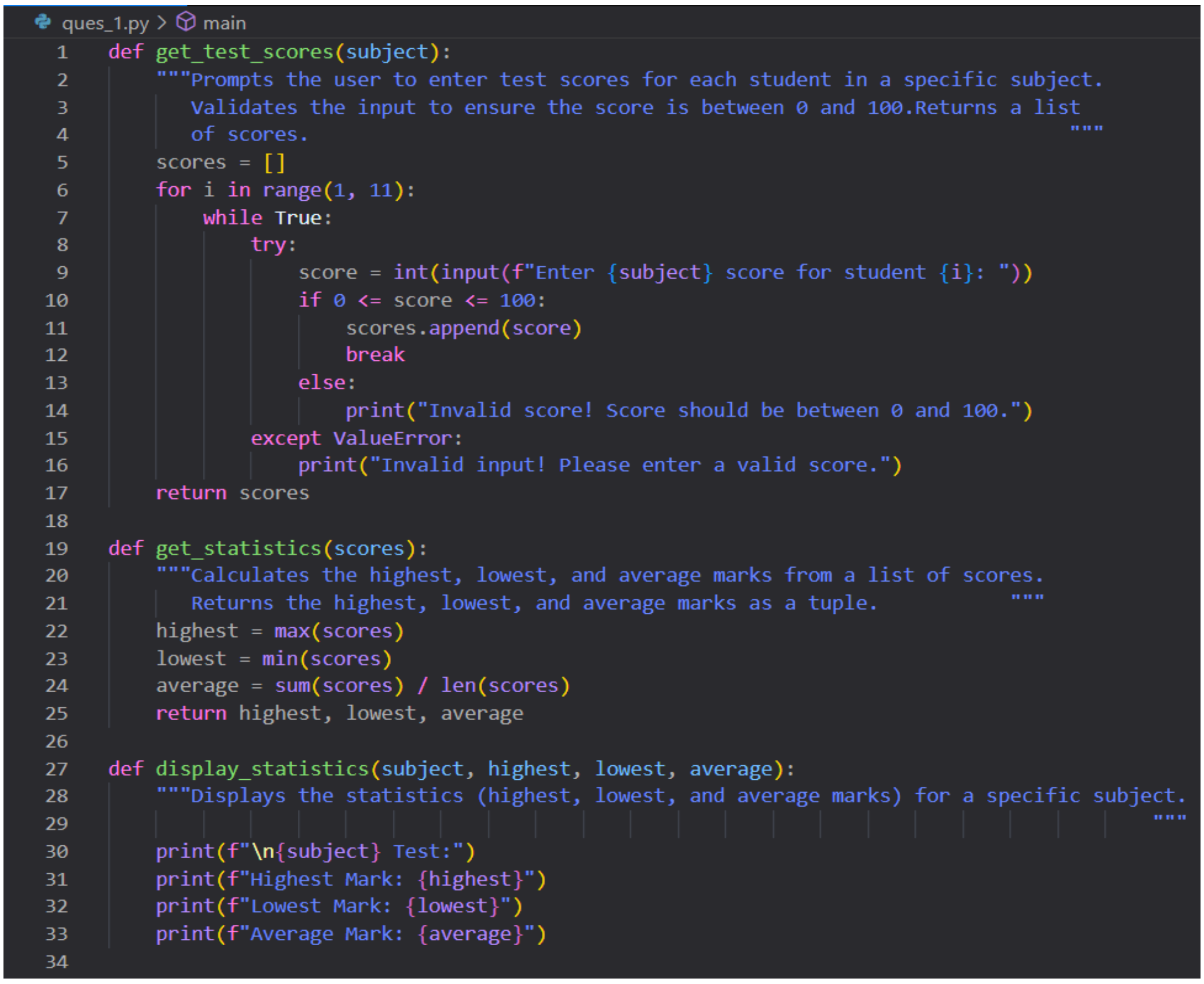
Ankushdeep Singh_102003174_COE7
ASSIGNMENT -1

ARTIFICIAL INTELLIGENCE

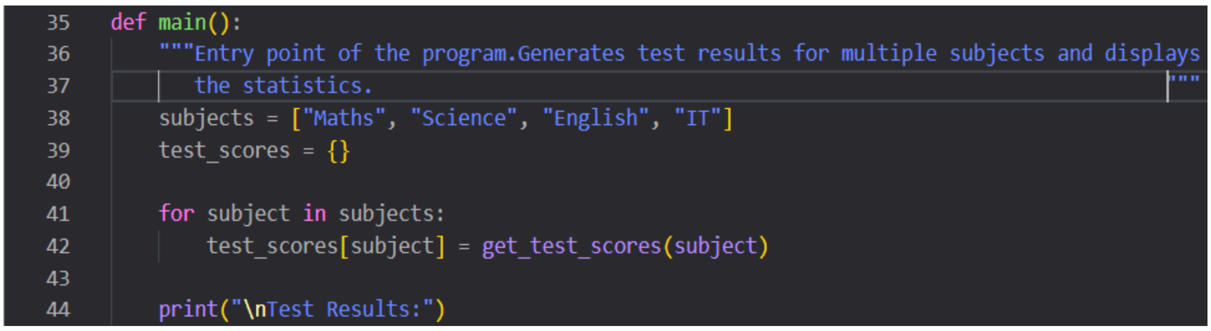
1Q.

A class with 10 students wants to produce some information from the results of the four standard tests in Maths, Science, English and IT. Each test is out of 100 marks. The information output should be the highest, lowest and average mark for each test and the highest, lowest and average mark overall. Write a program in Python to complete this task.

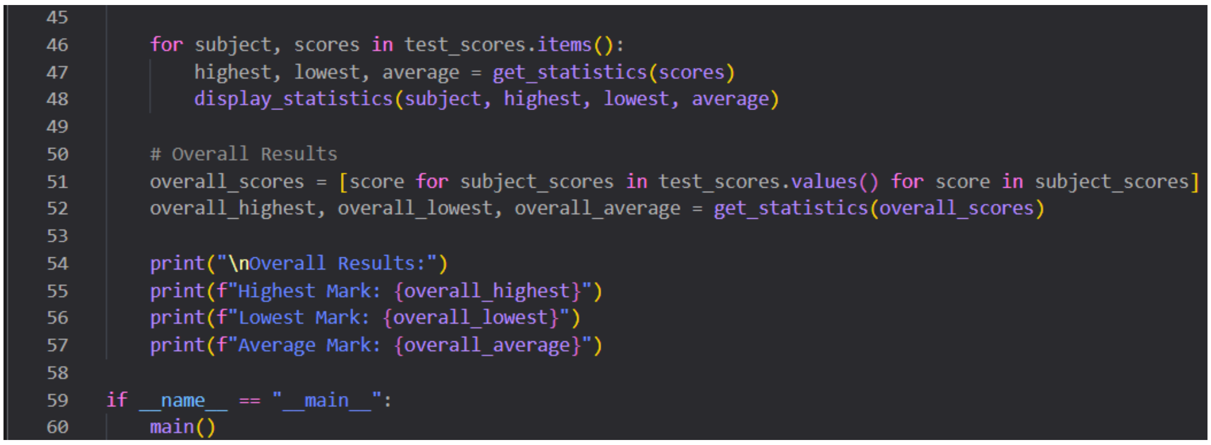
CODE:



```python
def get_test_scores(subject):
    """Prompts the user to enter test scores for each student in a specific subject.
       Validates the input to ensure the score is between 0 and 100.Returns a list
       of scores.                                                                 """
    scores = []
    for i in range(1, 11):
        while True:
            try:
                score = int(input(f"Enter {subject} score for student {i}: "))
                if 0 <= score <= 100:
                    scores.append(score)
                    break
                else:
                    print("Invalid score! Score should be between 0 and 100.")
            except ValueError:
                print("Invalid input! Please enter a valid score.")
    return scores

def get_statistics(scores):
    """Calculates the highest, lowest, and average marks from a list of scores.
       Returns the highest, lowest, and average marks as a tuple.            """
    highest = max(scores)
    lowest = min(scores)
    average = sum(scores) / len(scores)
    return highest, lowest, average

def display_statistics(subject, highest, lowest, average):
    """Displays the statistics (highest, lowest, and average marks) for a specific subject.
                                                                                         """
    print(f"\n{subject} Test:")
    print(f"Highest Mark: {highest}")
    print(f"Lowest Mark: {lowest}")
    print(f"Average Mark: {average}")

```

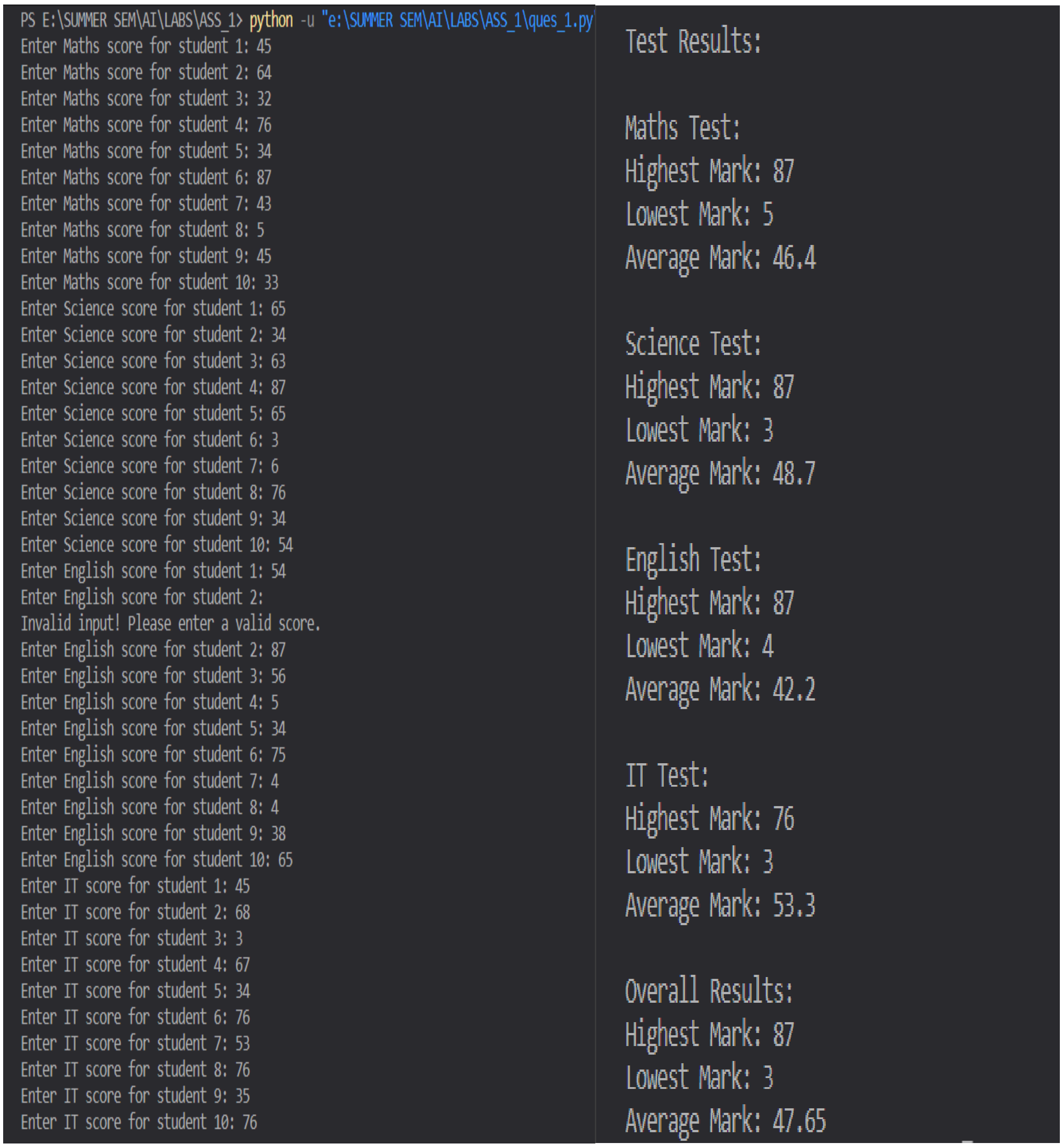


```python
def main():
    """Entry point of the program.Generates test results for multiple subjects and displays
       the statistics.                                                                  """
    subjects = ["Maths", "Science", "English", "IT"]
    test_scores = {}

    for subject in subjects:
        test_scores[subject] = get_test_scores(subject)

    print("\nTest Results:")
```



```python

    for subject, scores in test_scores.items():
        highest, lowest, average = get_statistics(scores)
        display_statistics(subject, highest, lowest, average)

    # Overall Results
    overall_scores = [score for subject_scores in test_scores.values() for score in subject_scores]
    overall_highest, overall_lowest, overall_average = get_statistics(overall_scores)

    print("\nOverall Results:")
    print(f"Highest Mark: {overall_highest}")
    print(f"Lowest Mark: {overall_lowest}")
    print(f"Average Mark: {overall_average}")

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

OUTPUT:

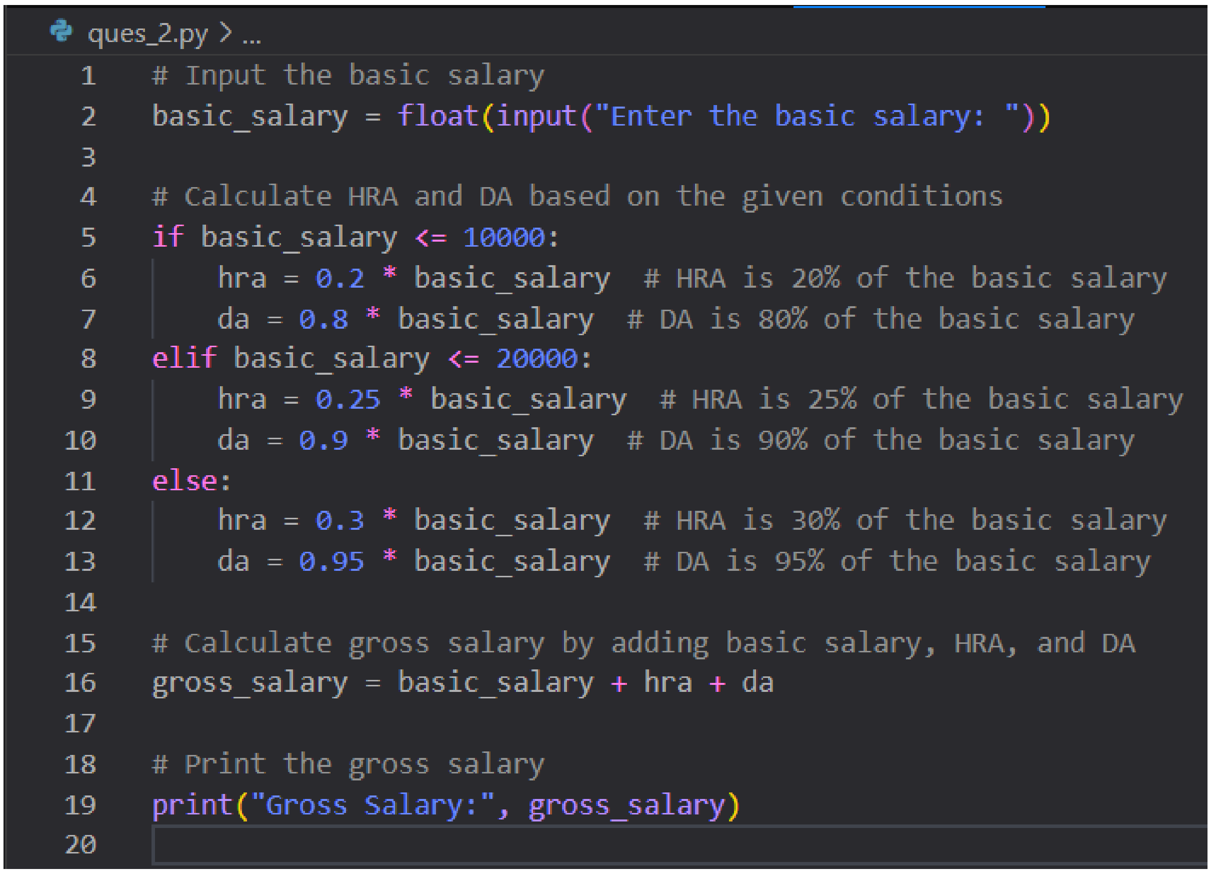


```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_1.py
Enter Maths score for student 1: 45
Enter Maths score for student 2: 64
Enter Maths score for student 3: 32
Enter Maths score for student 4: 76
Enter Maths score for student 5: 34
Enter Maths score for student 6: 87
Enter Maths score for student 7: 43
Enter Maths score for student 8: 5
Enter Maths score for student 9: 45
Enter Maths score for student 10: 33
Enter Science score for student 1: 65
Enter Science score for student 2: 34
Enter Science score for student 3: 63
Enter Science score for student 4: 87
Enter Science score for student 5: 65
Enter Science score for student 6: 3
Enter Science score for student 7: 6
Enter Science score for student 8: 76
Enter Science score for student 9: 34
Enter Science score for student 10: 54
Enter English score for student 1: 54
Enter English score for student 2:
Invalid input! Please enter a valid score.
Enter English score for student 2: 87
Enter English score for student 3: 56
Enter English score for student 4: 5
Enter English score for student 5: 34
Enter English score for student 6: 75
Enter English score for student 7: 4
Enter English score for student 8: 4
Enter English score for student 9: 38
Enter English score for student 10: 65
Enter IT score for student 1: 45
Enter IT score for student 2: 68
Enter IT score for student 3: 3
Enter IT score for student 4: 67
Enter IT score for student 5: 34
Enter IT score for student 6: 76
Enter IT score for student 7: 53
Enter IT score for student 8: 76
Enter IT score for student 9: 35
Enter IT score for student 10: 76

Test Results:

Maths Test:
Highest Mark: 87
Lowest Mark: 5
Average Mark: 46.4

Science Test:
Highest Mark: 87
Lowest Mark: 3
Average Mark: 48.7

English Test:
Highest Mark: 87
Lowest Mark: 4
Average Mark: 42.2

IT Test:
Highest Mark: 76
Lowest Mark: 3
Average Mark: 53.3

Overall Results:
Highest Mark: 87
Lowest Mark: 3
Average Mark: 47.65
```

2Q:

Write a Python Program to input basic salary of an employee and calculate its Gross salary according to following: Basic Salary <= 10000 : HRA = 20%, DA = 80% Basic Salary <= 20000 : HRA = 25%, DA = 90% Basic Salary > 20000 : HRA = 30%, DA = 95%.

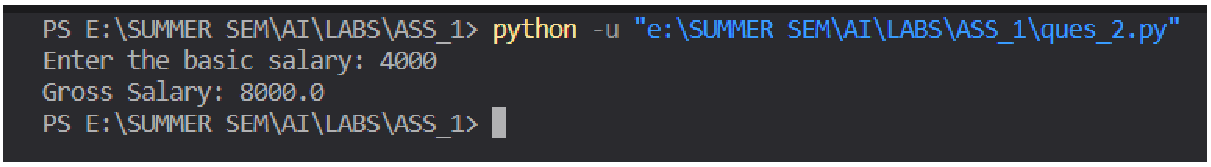
CODE:



ques_2.py > ...

```python
# Input the basic salary
basic_salary = float(input("Enter the basic salary: "))

# Calculate HRA and DA based on the given conditions
if basic_salary <= 10000:
    hra = 0.2 * basic_salary  # HRA is 20% of the basic salary
    da = 0.8 * basic_salary  # DA is 80% of the basic salary
elif basic_salary <= 20000:
    hra = 0.25 * basic_salary  # HRA is 25% of the basic salary
    da = 0.9 * basic_salary  # DA is 90% of the basic salary
else:
    hra = 0.3 * basic_salary  # HRA is 30% of the basic salary
    da = 0.95 * basic_salary  # DA is 95% of the basic salary

# Calculate gross salary by adding basic salary, HRA, and DA
gross_salary = basic_salary + hra + da

# Print the gross salary
print("Gross Salary:", gross_salary)

```

OUTPUT:



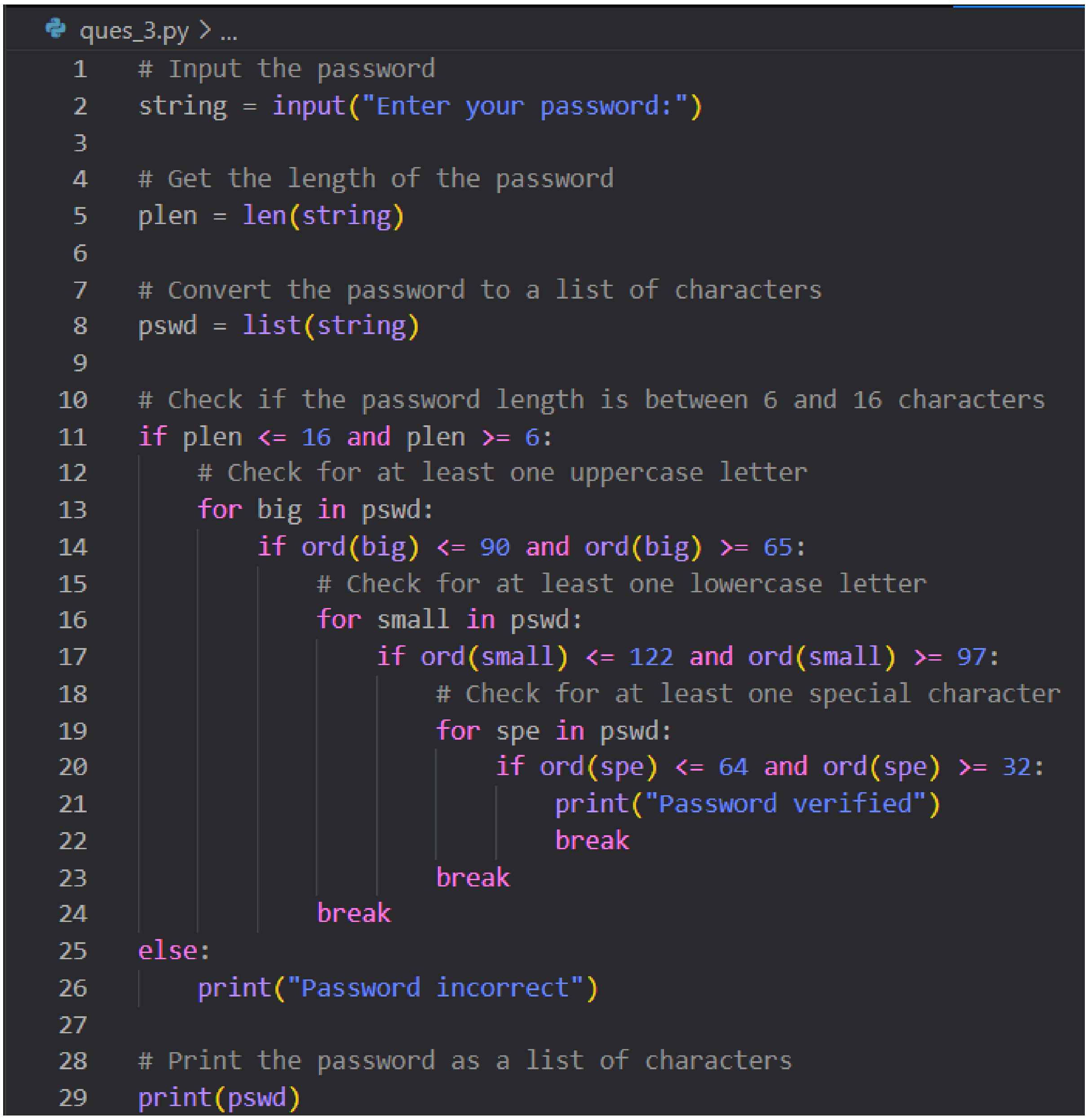
```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_2.py"
Enter the basic salary: 4000
Gross Salary: 8000.0
PS E:\SUMMER SEM\AI\LABS\ASS_1> 
```

3Q:

Write a Python program to check the validity of password input by users. **Validation:**

- At least 1 letter between [a-z] and 1 letter between [A-Z].
- At least 1 number between [0-9].
- At least 1 character from [$#@].
- Minimum length 6 characters.
- Maximum length 16 characters.

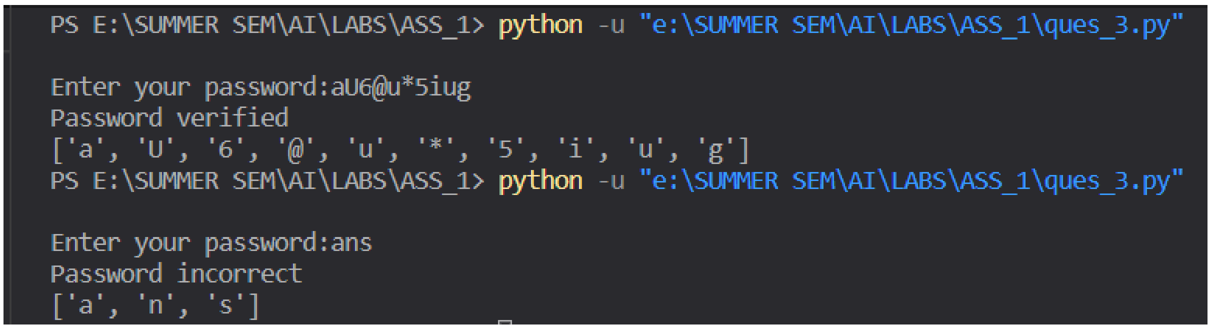
CODE:



ques_3.py > ...

```python
# Input the password
string = input("Enter your password:")

# Get the length of the password
plen = len(string)

# Convert the password to a list of characters
pswd = list(string)

# Check if the password length is between 6 and 16 characters
if plen <= 16 and plen >= 6:
    # Check for at least one uppercase letter
    for big in pswd:
        if ord(big) <= 90 and ord(big) >= 65:
            # Check for at least one lowercase letter
            for small in pswd:
                if ord(small) <= 122 and ord(small) >= 97:
                    # Check for at least one special character
                    for spe in pswd:
                        if ord(spe) <= 64 and ord(spe) >= 32:
                            print("Password verified")
                            break
                    break
            break
else:
    print("Password incorrect")

# Print the password as a list of characters
print(pswd)
```

OUTPUT:



```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_3.py"

Enter your password:aU6@u*5iug
Password verified
['a', 'U', '6', '@', 'u', '*', '5', 'i', 'u', 'g']
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_3.py"

Enter your password:ans
Password incorrect
['a', 'n', 's']
```

4Q:

Create a List L that is defined as= [10, 20, 30, 40, 50, 60, 70, 80].

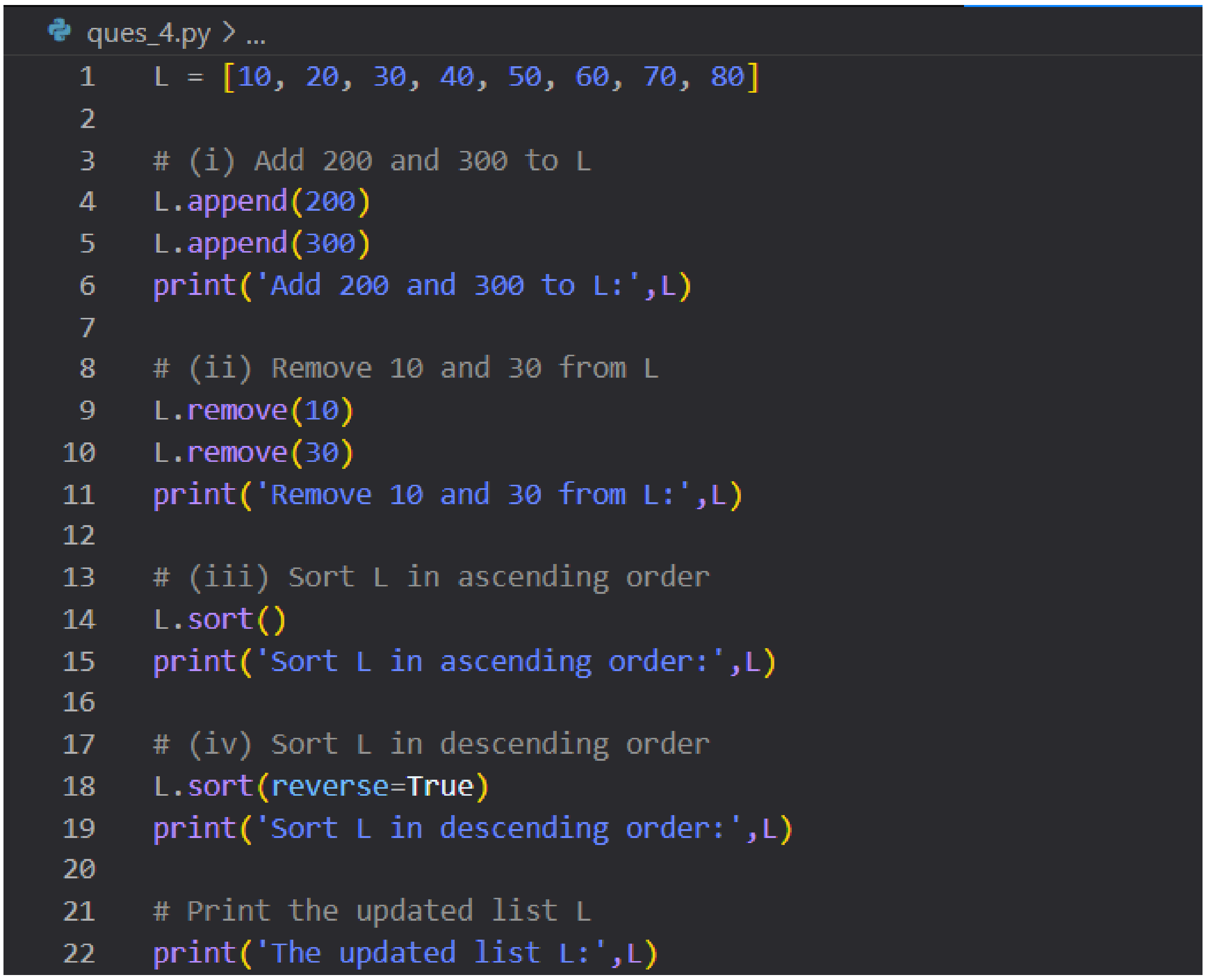
(i) WAP to add 200 and 300 to L.

(ii) WAP to remove 10 and 30 from L.

(iii) WAP to sort L in ascending order.

(iv) WAP to sort L in descending order.

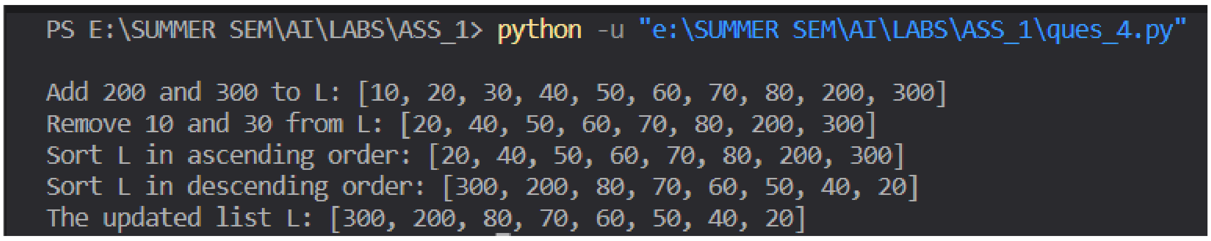
CODE:



ques_4.py > ...

```python
L = [10, 20, 30, 40, 50, 60, 70, 80]

# (i) Add 200 and 300 to L
L.append(200)
L.append(300)
print('Add 200 and 300 to L:',L)

# (ii) Remove 10 and 30 from L
L.remove(10)
L.remove(30)
print('Remove 10 and 30 from L:',L)

# (iii) Sort L in ascending order
L.sort()
print('Sort L in ascending order:',L)

# (iv) Sort L in descending order
L.sort(reverse=True)
print('Sort L in descending order:',L)

# Print the updated list L
print('The updated list L:',L)
```

Output:

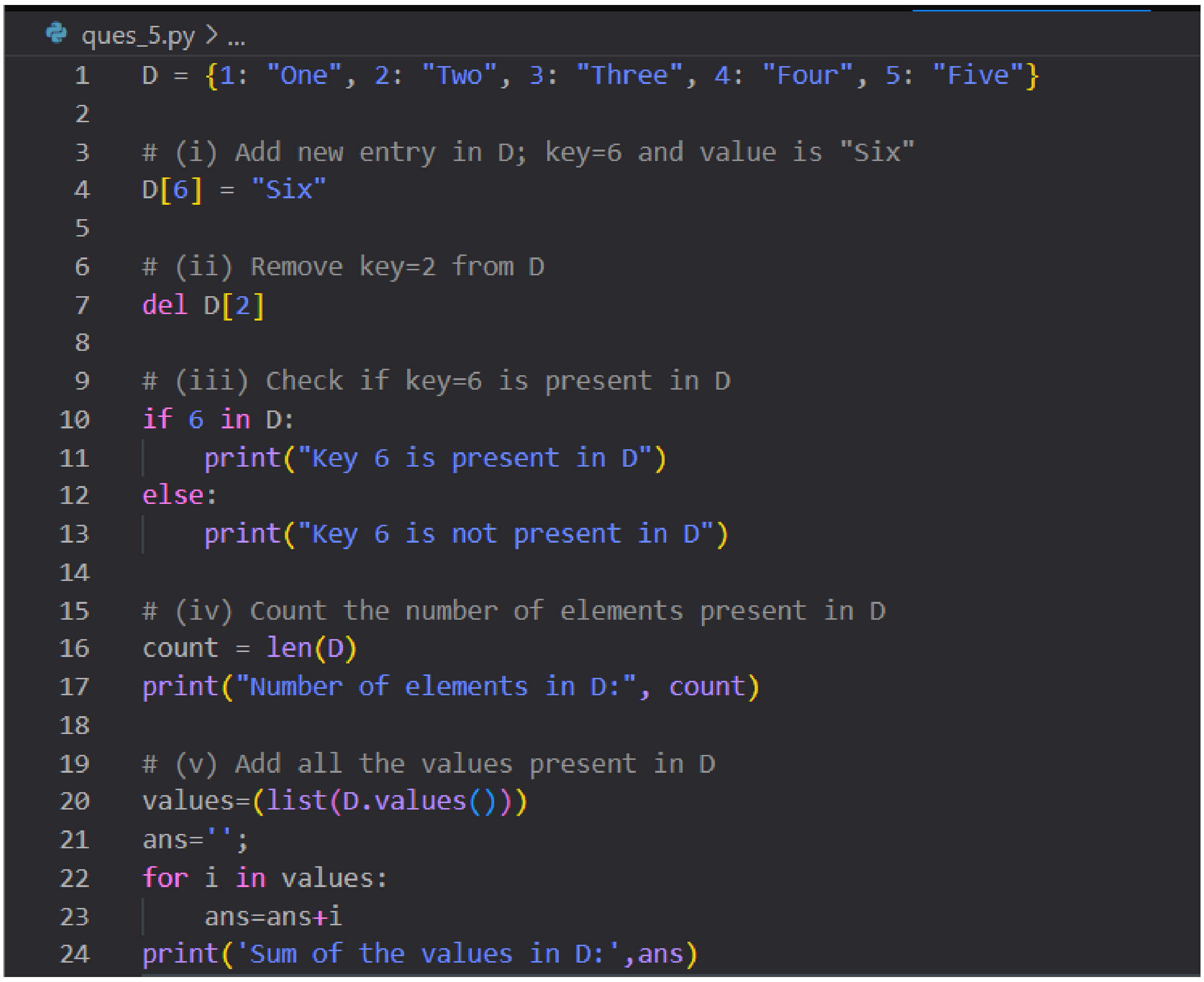


```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_4.py"

Add 200 and 300 to L: [10, 20, 30, 40, 50, 60, 70, 80, 200, 300]
Remove 10 and 30 from L: [20, 40, 50, 60, 70, 80, 200, 300]
Sort L in ascending order: [20, 40, 50, 60, 70, 80, 200, 300]
Sort L in descending order: [300, 200, 80, 70, 60, 50, 40, 20]
The updated list L: [300, 200, 80, 70, 60, 50, 40, 20]
```

5Q:

D is a dictionary defined as D= {1:"One", 2:"Two", 3:"Three", 4: "Four", 5:"Five"}. (i) WAP to add new entry in D; key=6 and value is "Six"
(ii) WAP to remove key=2.
(iii) WAP to check if 6 key is present in D.
(iv) WAP to count the number of elements present in D.
(v) WAP to add all the values present D.

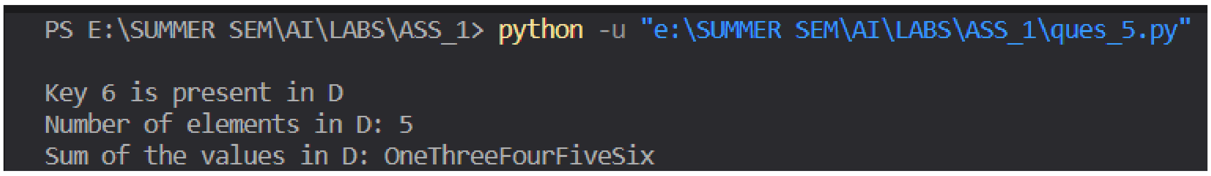
CODE:



ques_5.py > ...

```python
D = {1: "One", 2: "Two", 3: "Three", 4: "Four", 5: "Five"}

# (i) Add new entry in D; key=6 and value is "Six"
D[6] = "Six"

# (ii) Remove key=2 from D
del D[2]

# (iii) Check if key=6 is present in D
if 6 in D:
    print("Key 6 is present in D")
else:
    print("Key 6 is not present in D")

# (iv) Count the number of elements present in D
count = len(D)
print("Number of elements in D:", count)

# (v) Add all the values present in D
values=(list(D.values()))
ans='';
for i in values:
    ans=ans+i
print('Sum of the values in D:',ans)
```

OUTPUT:

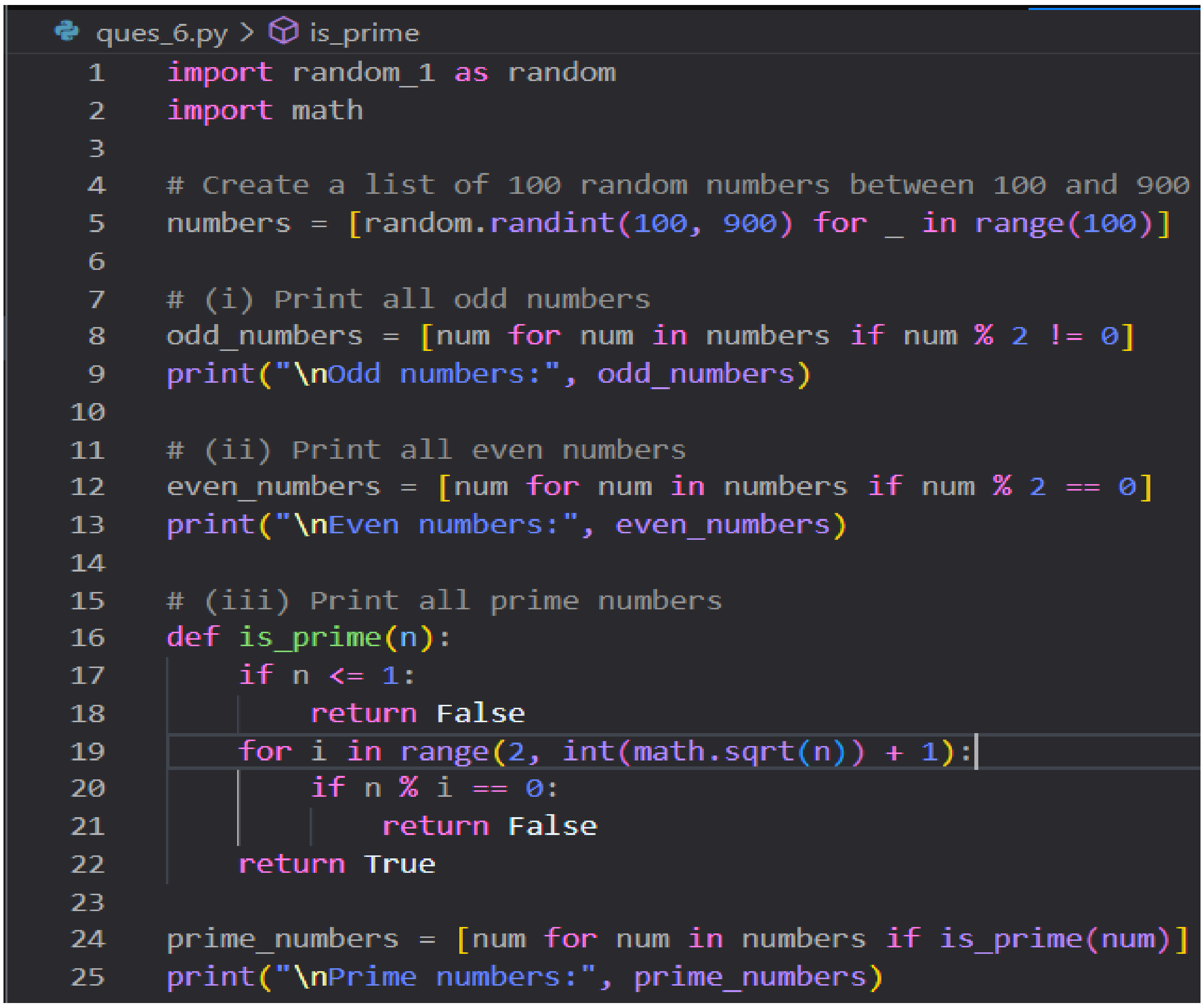


```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_5.py"

Key 6 is present in D
Number of elements in D: 5
Sum of the values in D: OneThreeFourFiveSix
```

6Q:

WAP to create a list of 100 random numbers between 100 and 900. Count and print the: (i) All odd numbers
(ii) All even numbers
(iii) All prime numbers

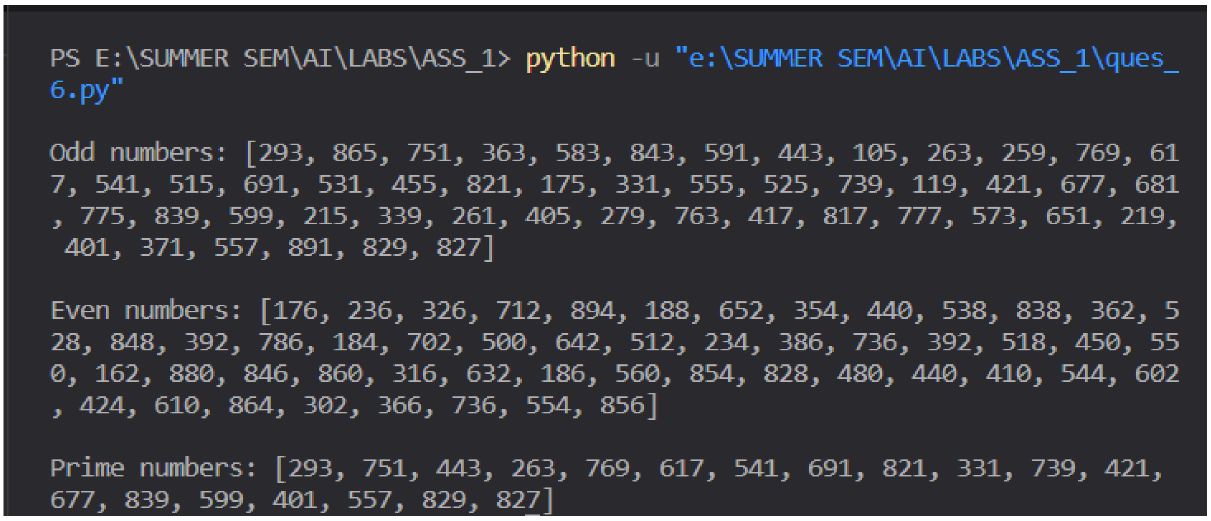
CODE:



ques_6.py > is_prime

```python
import random_1 as random
import math

# Create a list of 100 random numbers between 100 and 900
numbers = [random.randint(100, 900) for _ in range(100)]

# (i) Print all odd numbers
odd_numbers = [num for num in numbers if num % 2 != 0]
print("\nOdd numbers:", odd_numbers)

# (ii) Print all even numbers
even_numbers = [num for num in numbers if num % 2 == 0]
print("\nEven numbers:", even_numbers)

# (iii) Print all prime numbers
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

prime_numbers = [num for num in numbers if is_prime(num)]
print("\nPrime numbers:", prime_numbers)
```

OUTPUT:

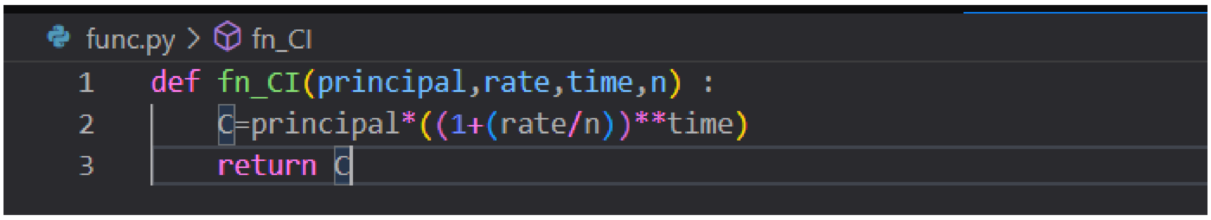


```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_
6.py"

Odd numbers: [293, 865, 751, 363, 583, 843, 591, 443, 105, 263, 259, 769, 61
7, 541, 515, 691, 531, 455, 821, 175, 331, 555, 525, 739, 119, 421, 677, 681
, 775, 839, 599, 215, 339, 261, 405, 279, 763, 417, 817, 777, 573, 651, 219,
 401, 371, 557, 891, 829, 827]

Even numbers: [176, 236, 326, 712, 894, 188, 652, 354, 440, 538, 838, 362, 5
28, 848, 392, 786, 184, 702, 500, 642, 512, 234, 386, 736, 392, 518, 450, 55
0, 162, 880, 846, 860, 316, 632, 186, 560, 854, 828, 480, 440, 410, 544, 602
, 424, 610, 864, 302, 366, 736, 554, 856]

Prime numbers: [293, 751, 443, 263, 769, 617, 541, 691, 821, 331, 739, 421,
677, 839, 599, 401, 557, 829, 827]
```

7Q:

(i) Write a function which takes principal amount, interest rate and time. This function returns compound interest. Call this function to print the output.
(ii) Save this function (as a module) in a python file and call it in another python file.

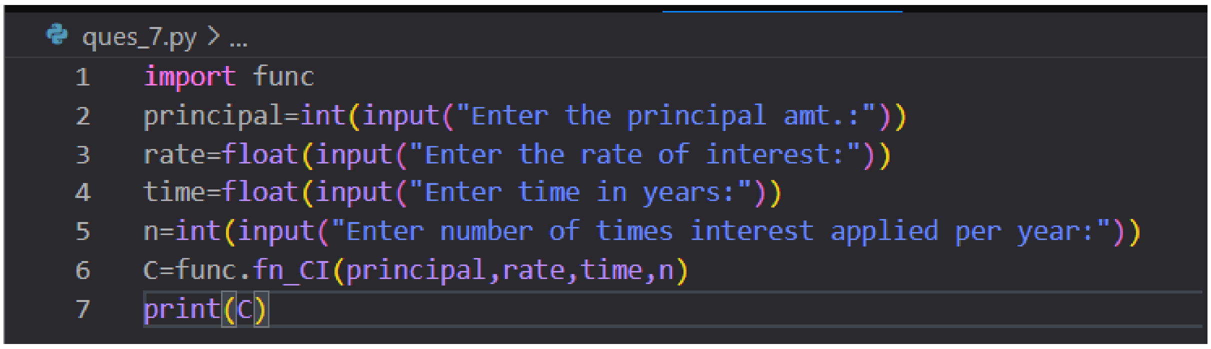
func.py



func.py > fn_CI

```python
def fn_CI(principal,rate,time,n) :
    C=principal*((1+(rate/n))**time)
    return C
```

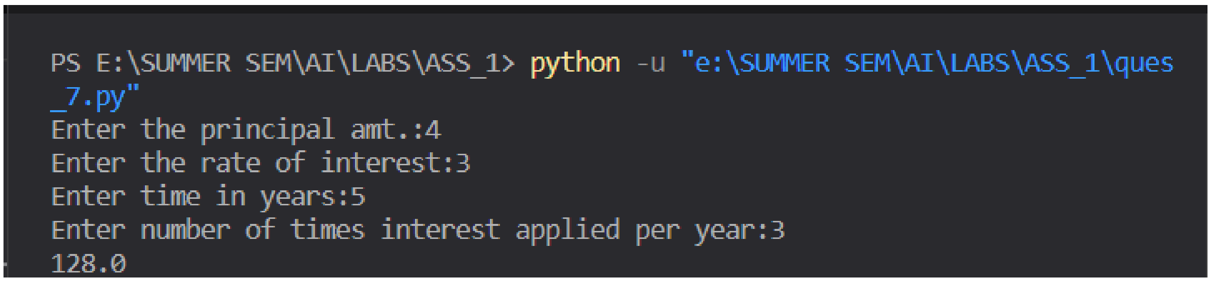
CODE:



ques_7.py > ...

```python
import func
principal=int(input("Enter the principal amt.:"))
rate=float(input("Enter the rate of interest:"))
time=float(input("Enter time in years:"))
n=int(input("Enter number of times interest applied per year:"))
C=func.fn_CI(principal,rate,time,n)
print(C)
```

OUTPUT:

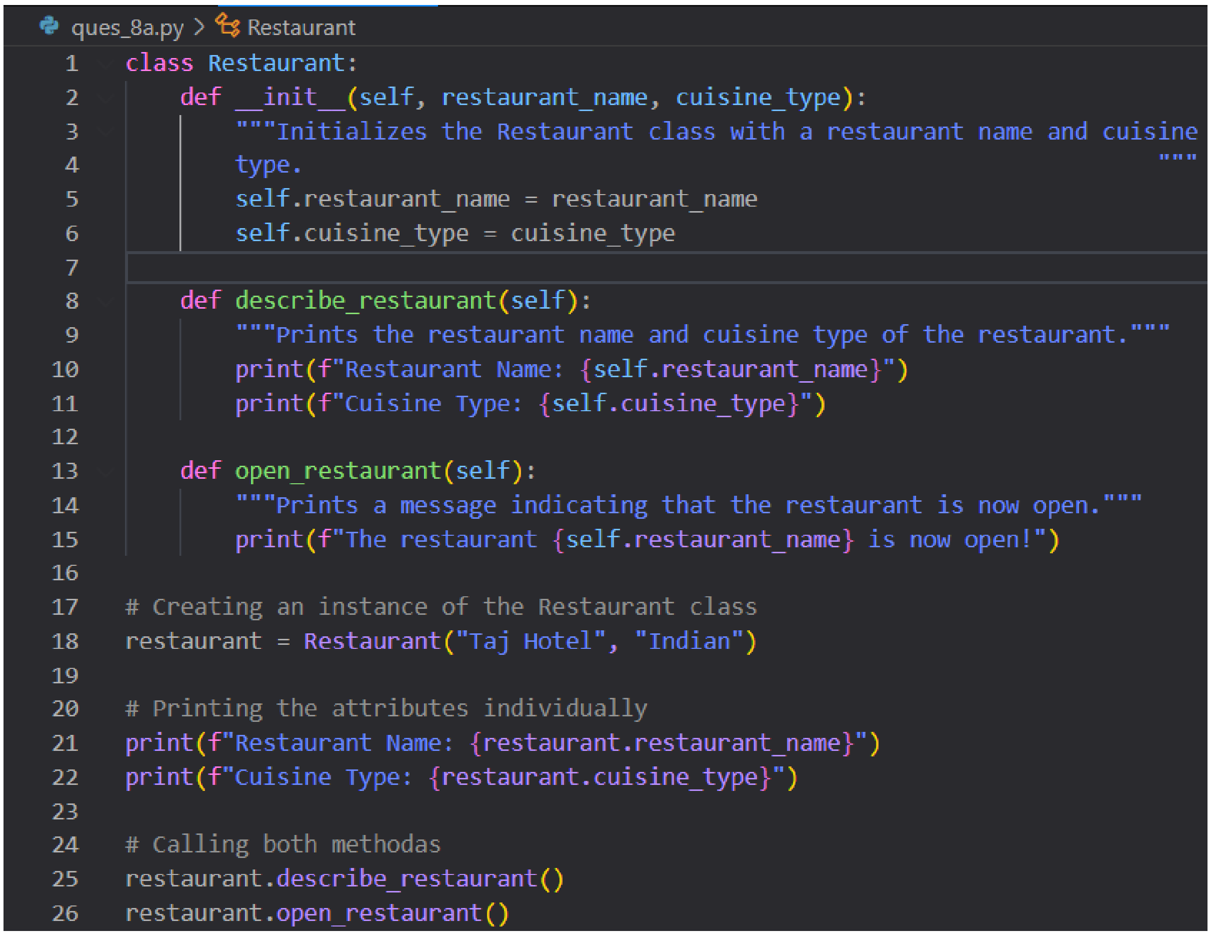


```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques
_7.py"
Enter the principal amt.:4
Enter the rate of interest:3
Enter time in years:5
Enter number of times interest applied per year:3
128.0
```

8Q:

A) Make a class called Restaurant. The __init__() method for Restaurant should store two attributes: a restaurant_name and a cuisine_type. Make a method called describe_restaurant() that prints these two pieces of information, and a method called open_restaurant() that prints a message indicating that the restaurant is open. Make an instance called restaurant from your class. Print the two attributes individually, and then call both methods.

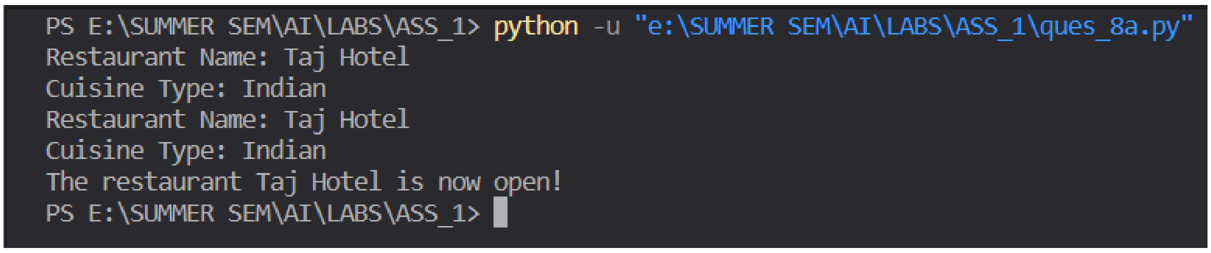
CODE:



ques_8a.py > Restaurant

```python
class Restaurant:
    def __init__(self, restaurant_name, cuisine_type):
        """Initializes the Restaurant class with a restaurant name and cuisine type."""
        self.restaurant_name = restaurant_name
        self.cuisine_type = cuisine_type

    def describe_restaurant(self):
        """Prints the restaurant name and cuisine type of the restaurant."""
        print(f"Restaurant Name: {self.restaurant_name}")
        print(f"Cuisine Type: {self.cuisine_type}")

    def open_restaurant(self):
        """Prints a message indicating that the restaurant is now open."""
        print(f"The restaurant {self.restaurant_name} is now open!")

# Creating an instance of the Restaurant class
restaurant = Restaurant("Taj Hotel", "Indian")

# Printing the attributes individually
print(f"Restaurant Name: {restaurant.restaurant_name}")
print(f"Cuisine Type: {restaurant.cuisine_type}")

# Calling both methodas
restaurant.describe_restaurant()
restaurant.open_restaurant()
```

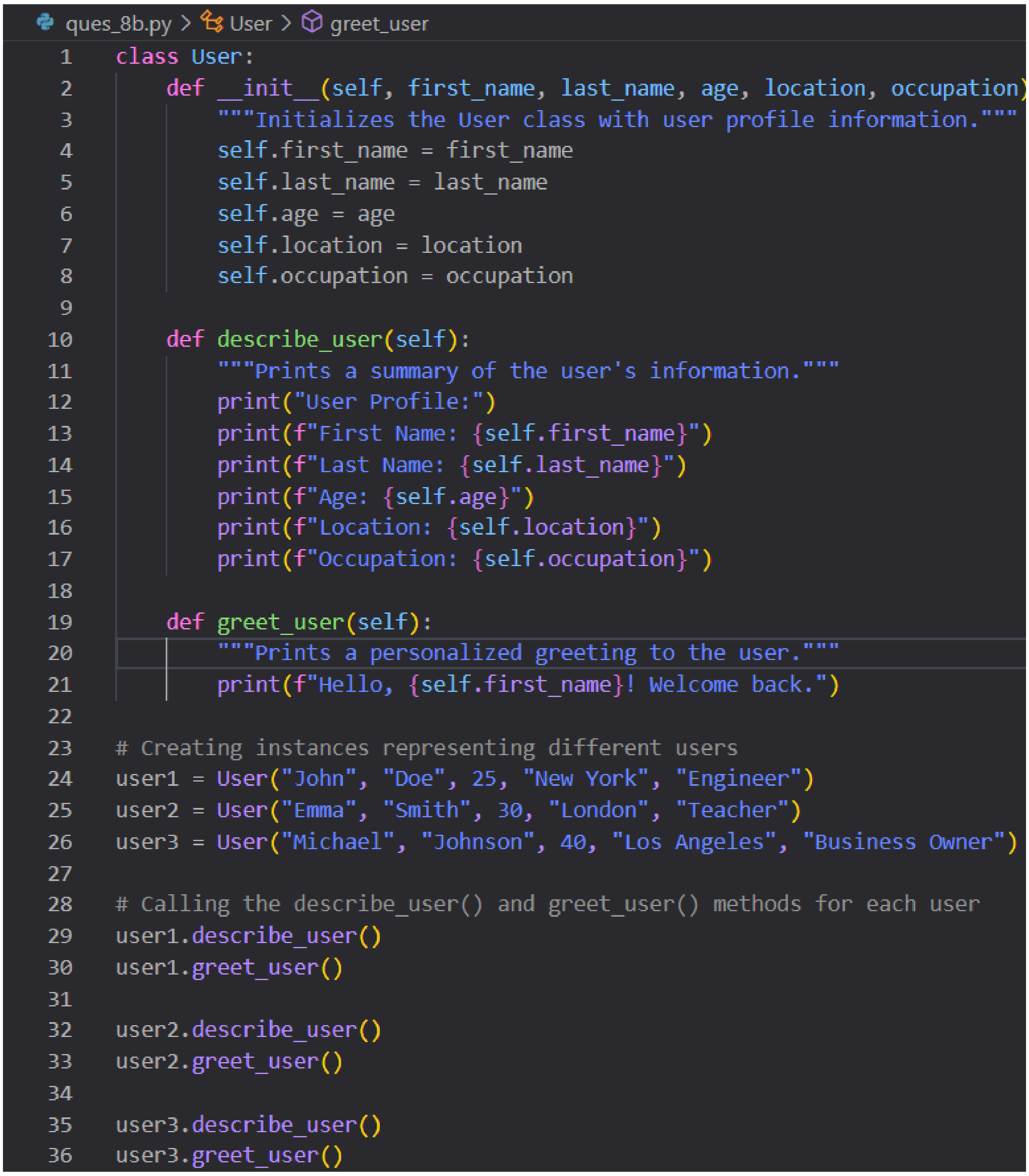
OUTPUT:



```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_8a.py"
Restaurant Name: Taj Hotel
Cuisine Type: Indian
Restaurant Name: Taj Hotel
Cuisine Type: Indian
The restaurant Taj Hotel is now open!
PS E:\SUMMER SEM\AI\LABS\ASS_1>
```

B) Make a class called User. Create two attributes called first_name and last_name, and then create several other attributes that are typically stored in a user profile. Make a method called describe_user() that prints a summary of the user's information. Make another method called greet_user() that prints a personalized greeting to the user. Create several instances representing different users, and call both method for each user.

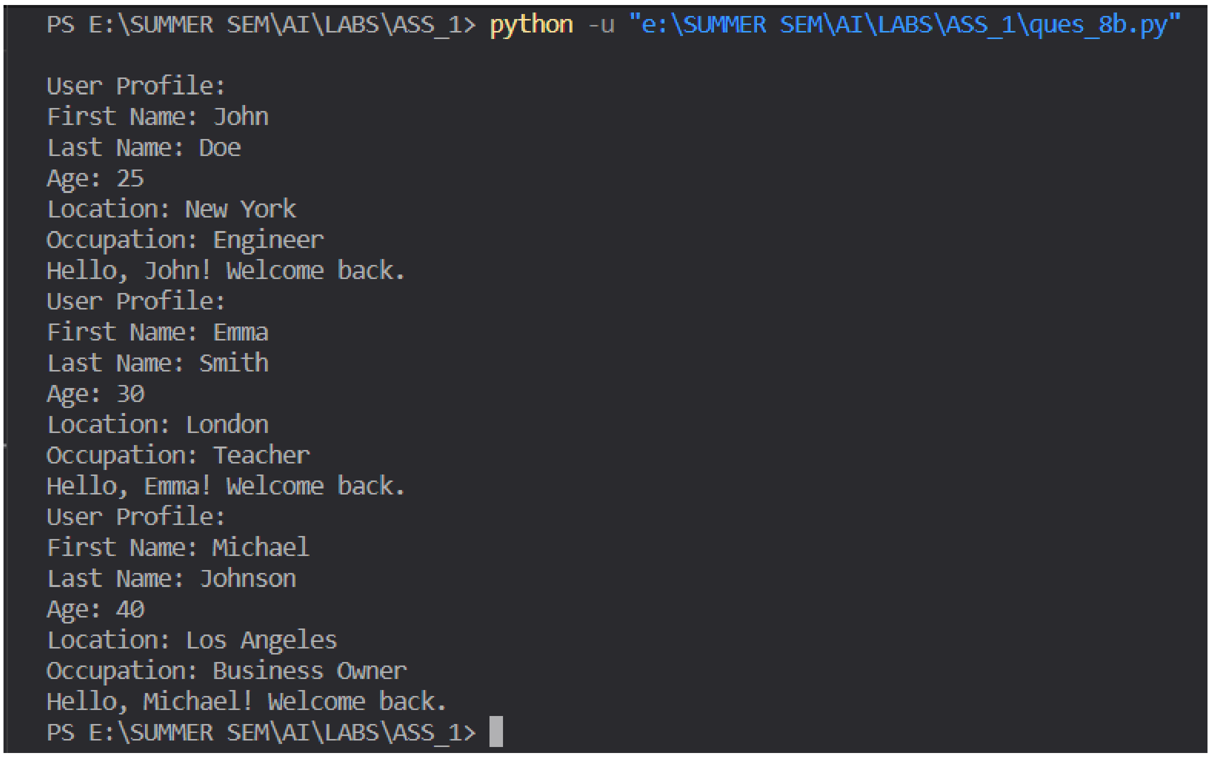
CODE:



ques_8b.py > User > greet_user

```python
class User:
    def __init__(self, first_name, last_name, age, location, occupation)
        """Initializes the User class with user profile information."""
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.location = location
        self.occupation = occupation

    def describe_user(self):
        """Prints a summary of the user's information."""
        print("User Profile:")
        print(f"First Name: {self.first_name}")
        print(f"Last Name: {self.last_name}")
        print(f"Age: {self.age}")
        print(f"Location: {self.location}")
        print(f"Occupation: {self.occupation}")

    def greet_user(self):
        """Prints a personalized greeting to the user."""
        print(f"Hello, {self.first_name}! Welcome back.")

# Creating instances representing different users
user1 = User("John", "Doe", 25, "New York", "Engineer")
user2 = User("Emma", "Smith", 30, "London", "Teacher")
user3 = User("Michael", "Johnson", 40, "Los Angeles", "Business Owner")

# Calling the describe_user() and greet_user() methods for each user
user1.describe_user()
user1.greet_user()

user2.describe_user()
user2.greet_user()

user3.describe_user()
user3.greet_user()
```

OUTPUT:



```
PS E:\SUMMER SEM\AI\LABS\ASS_1> python -u "e:\SUMMER SEM\AI\LABS\ASS_1\ques_8b.py"

User Profile:
First Name: John
Last Name: Doe
Age: 25
Location: New York
Occupation: Engineer
Hello, John! Welcome back.
User Profile:
First Name: Emma
Last Name: Smith
Age: 30
Location: London
Occupation: Teacher
Hello, Emma! Welcome back.
User Profile:
First Name: Michael
Last Name: Johnson
Age: 40
Location: Los Angeles
Occupation: Business Owner
Hello, Michael! Welcome back.
PS E:\SUMMER SEM\AI\LABS\ASS_1>
```