Assignment 3 Solution

Submitted by : Yameen Ali

Go to Github repository

Question 1

Lists

1. Create an empty list my_list

In [23]:
```
my_list = []
```

2. Append three elements to my_list using the append method

In [24]:
```
my_list.append("apple")
my_list.append("orange")
my_list.append("mango")
print(my_list)
```

```
['apple', 'orange', 'mango']
```

3. Insert an element at a specific position in my_list using the insert method

In [37]:
```
# Inserting "banana" at index 1
my_list.insert(1, "banana")
print(my_list)
```

```
['apple', 'banana', 'orange', 'mango']
```

4. Remove an element from my_list using the remove method.

In [38]:
```
# Removing "banana" from the list
my_list.remove("banana")
```

5. Print the final state of my_list.

In [40]:
```
print(my_list)
```

```
['apple', 'orange', 'mango']
```

Question 2

Tuples

1. Create a tuple my_tuple with at least five elements.

In [42]:
```
my_tuple = ('yameen','rafay','saad','ahmad','haseeb')
```

2. Use tuple unpacking to assign values to multiple variables.

In [47]:
```
# Unpack the tuple and assign values to variables
a, b, c, d, e = my_tuple

print("a :", a)
print("b :", b)
print("c :", c)
print("d :", d)
print("e :", e)
```

```
a : yameen
b : rafay
c : saad
d : ahmad
e : haseeb
```

3. Concatenate two tuples and print the result.

In [48]:
```
# Define two tuples
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)

# Concatenate the tuples
result_tuple = tuple1 + tuple2
print(result_tuple)
```

```
(1, 2, 3, 4, 5, 6)
```

4. Access and print the elements of my_tuple using both positive and negative indexing.

In [53]:
```
my_tuple = ('saad','rafay','yameen','ahmad','haseeb')

# Access and print "yameen" using positive indexing
print("Index 2:",my_tuple[2])

# Access and print "yameen" using negative indexing
print("Index -3:",my_tuple[-3])
```

```
Index 2: yameen
Index -3: yameen
```

5. Perform any operation that demonstrates that tuples are immutable.

In [60]:
```
my_tuple = (1, 2, 3, 4, 5)

# Try to modify an element of the tuple
try:
    my_tuple[0] = 10  # Trying to change the first element
except TypeError as e:
    print("Error:", e)
```

```
Error: 'tuple' object does not support item assignment
```

Question 3

Input and output

1. Prompt the user to enter their name using input.

In [64]:
```
name = input("Enter your name: ")
```

```
Enter your name:  yameen
```

2. Print a welcome message using the entered name.

In [67]:
```
name = input("Enter your name: ")
print(f"Hello Welcome {name}!")
```

```
Enter your name:  yameen
Hello Welcome yameen!
```

3. Ask the user to input two numbers, convert them to integers, and calculate their sum and product.
4. Display the results using formatted output.

In [71]:
```
num_1 = int(input("Enter first number: "))
num_2 = int(input("Enter Second number: "))

# sum
s = num_1 + num_2
print(f"The sum of {num_1} + {num_2} is = {s}")

# product
p = num_1 * num_2
```

```
print(f"The product of {num_1} + {num_2} is = {p}")
```

```
Enter first number:  10
Enter Second number:  8
The sum of 10 + 8 is = 18
The product of 10 + 8 is = 80
```

Question 4

Operators

1. Create a variable number and initialize it with an integer value.

In [75]:
```
num = 10
```

2. Use the modulus operator to check if the num is even or odd.

In [76]:
```
# Check if the number is even or odd using the modulus operator
if num % 2 == 0:
    print(num, "is even")
else:
    print(num, "is odd")
```

```
10 is even
```

3. Use a comparison operator to check if the num is greater than or equal to 10.

In [77]:
```
# Check if the number is greater than or equal to 10 using a comparison operator
if num >= 10:
    print(num, "is greater than or equal to 10")
else:
    print(num, "is less than 10")
```

```
10 is greater than or equal to 10
```

4. Combine logical operators to create a complex condition and print the result.

In [78]:
```
# Combine logical operators to create a complex condition
if num % 2 == 0 and num >= 10:
    print(num, "satisfies the complex condition")
else:
    print(num, "does not satisfy the complex condition")
```

```
10 satisfies the complex condition
```

End