**ASSIGNEMNT III**

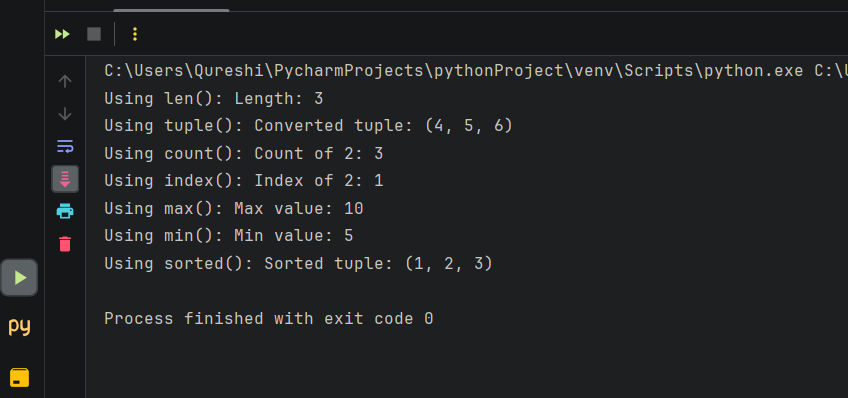
Q1. Give two differences between list and tuple (S-22) (2M)

|  |  |  |
| --- | --- | --- |
| Mutability | Mutable: Elements can be changed, added, or removed after creation. | Immutable: Elements cannot be changed after creation. |
| Memory Consumption | Generally consume more memory due to their dynamic nature and resizing capability. | Consume less memory as they are immutable and fixed in size. |
| Performance | Slower for certain operations, especially when resizing or shifting elements. | Faster for iteration and element access due to fixed size and immutability. |
| Use Cases | Suitable for collections that may change over time and require flexibility. | Appropriate for data that should remain constant and immutable. |
| Syntax | Uses square brackets **[ ]** to define elements. | Uses parentheses **( )** to define elements. |

Q2. Explain four Built-in tuple functions python with EXAMPLE. (S-22) (4M)

CODE:

my\_tuple = (1, 2, 3)  
length = len(my\_tuple)  
print("Using len(): Length:", length)  
  
my\_list = [4, 5, 6]  
my\_tuple = tuple(my\_list)  
print("Using tuple(): Converted tuple:", my\_tuple)  
  
my\_tuple = (1, 2, 2, 3, 2)  
count\_of\_2 = my\_tuple.count(2)  
print("Using count(): Count of 2:", count\_of\_2)  
  
my\_tuple = (1, 2, 3, 2)  
index\_of\_2 = my\_tuple.index(2)  
print("Using index(): Index of 2:", index\_of\_2)  
  
my\_tuple = (10, 5, 8)  
max\_value = max(my\_tuple)  
print("Using max(): Max value:", max\_value)  
  
my\_tuple = (10, 5, 8)  
min\_value = min(my\_tuple)  
print("Using min(): Min value:", min\_value)  
  
my\_tuple = (3, 1, 2)  
sorted\_tuple = tuple(sorted(my\_tuple))  
print("Using sorted(): Sorted tuple:", sorted\_tuple)  
OUTPUT:



Q3. Explain indexing and slicing in list with EXAMPLE. (S-22) (4M)

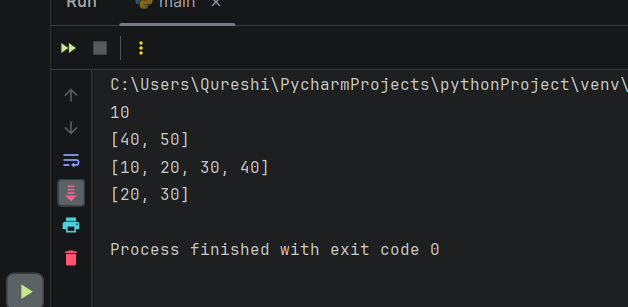
Indexing refers to accessing individual elements within a list by their position or index. In Python, indexing starts from 0, meaning the first element has an index of 0, the second element has an index of 1, and so on.

LIST INDEXING: Indexing refers to accessing individual elements within a list by their position or index. In Python, indexing starts from 0. You can use the index operator **[]** to access items in a list.

CODE:

list1 = [10, 20, 30, 40, 50]  
print(list1[0])   
print(list1[3:])   
print(list1[:4])   
print(list1[1:3])

OUTPUT:

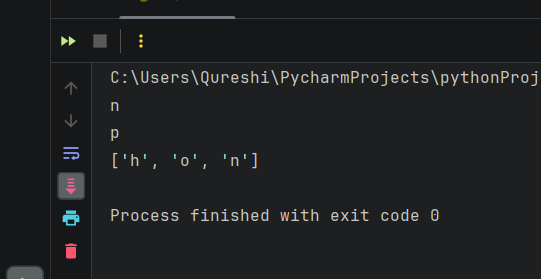


NEGATIVE INDEXING: Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 refers to the second last item, and so on.

CODE:

list2 = ['p', 'y', 't', 'h', 'o', 'n']  
print(list2[-1])  
print(list2[-6])  
print(list2[-3:])

OUTPUT:

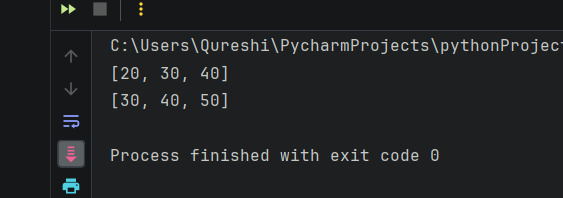


LIST SLICING: Slicing is an operation that allows you to extract elements from lists by specifying a start and end index using the colon

CODE:

list\_variable = [10, 20, 30, 40, 50]  
print(list\_variable[1:4])   
print(list\_variable[2:5])

OUTPUT:

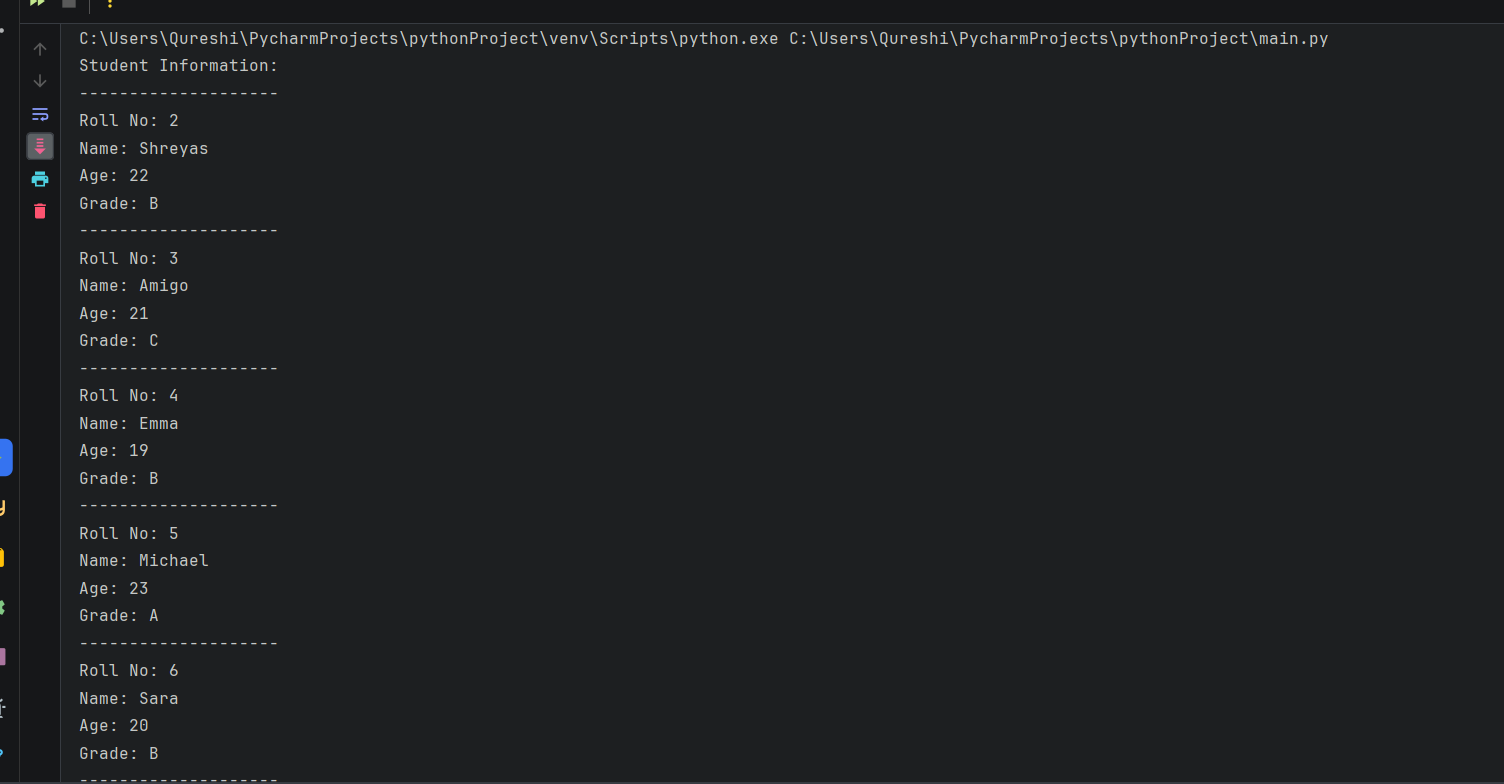


Q4. Write a program to create dictionary of students that includes their ROLL NO. and NAME. (S-22) (4M)

CODE:

students = {  
 1: {'name': 'Juan', 'age': 20, 'grade': 'A'},  
 2: {'name': 'Aande', 'age': 22, 'grade': 'B'},  
 3: {'name': 'Amigo', 'age': 21, 'grade': 'C'}  
}  
  
students[4] = {'name': 'Emma', 'age': 19, 'grade': 'B'}  
students[5] = {'name': 'Michael', 'age': 23, 'grade': 'A'}  
students[6] = {'name': 'Sara', 'age': 20, 'grade': 'B'}  
  
students[2]['name'] = 'Shreyas'  
  
del students[1]  
  
print("Student Information:")  
print("--------------------")  
for roll\_no, student in students.items():  
 print(f"Roll No: {roll\_no}")  
 print(f"Name: {student['name']}")  
 print(f"Age: {student['age']}")  
 print(f"Grade: {student['grade']}")  
 print("--------------------")

OUTPUT:

****

Q5. Write the OUTPUT of the following: (S-22) (6M)

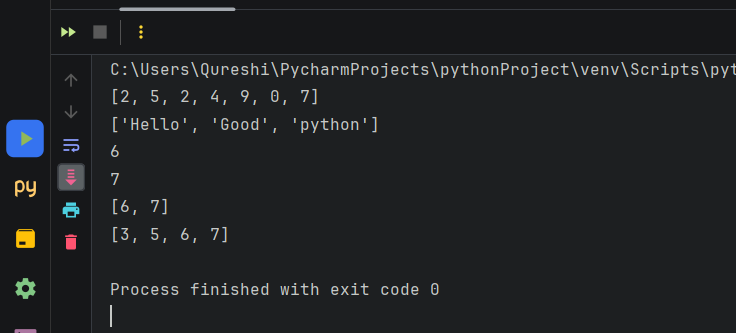
CODE:

# i) Replace elements in a list slice  
a = [2, 5, 1, 3, 6, 9, 7]  
a[2:6] = [2, 4, 9, 0]  
print(a)

# ii) Append to a list  
b = ["Hello", "Good"]  
b.append("python")  
print(b)

# iii) List slicing and indexing  
t1 = [3, 5, 6, 7]  
print(t1[2])  
print(t1[-1])  
print(t1[2:])  
print(t1[:])

OUTPUT:



Q6. Describe Tuples in Python. (W-22) (2M)

* Ordered Collection: Like lists, tuples maintain the order of elements as they are added. Each element in a tuple has a specific index, starting from 0 for the first element.
* Immutable: Tuples are immutable, meaning once they are created, their elements cannot be changed, added, or removed. This immutability makes tuples suitable for representing fixed collections of values.
* Heterogeneous Elements: Tuples can contain elements of different data types, including integers, floats, strings, other tuples, or even complex objects.
* Defined with Parentheses: Tuples are defined using parentheses (). Elements within a tuple are separated by commas.

**Q7. Write any four methods of dictionary. (W-22) (4M)**

Python Dictionary clear()

* Removes all Items

Method Dictionary copy()

* Returns the Shallow Copy of a Dictionary

Method Dictionary fromkeys()

* creates dictionary from given sequence

Method Dictionary get()

* Returns Value of The Key

Method Dictionary items()

* returns view of dictionary's (key, value) pair

Method Dictionary keys()

* Returns View Object of All Keys

Method Dictionary pop()

* removes and returns element having given key

Method Dictionary popitem()

* Returns & Removes Latest Element From Dictionary

Method Dictionary setdefault()

* Inserts Key With a Value if Key is not Present

Method Dictionary update()

* Updates the Dictionary

Method Dictionary values()

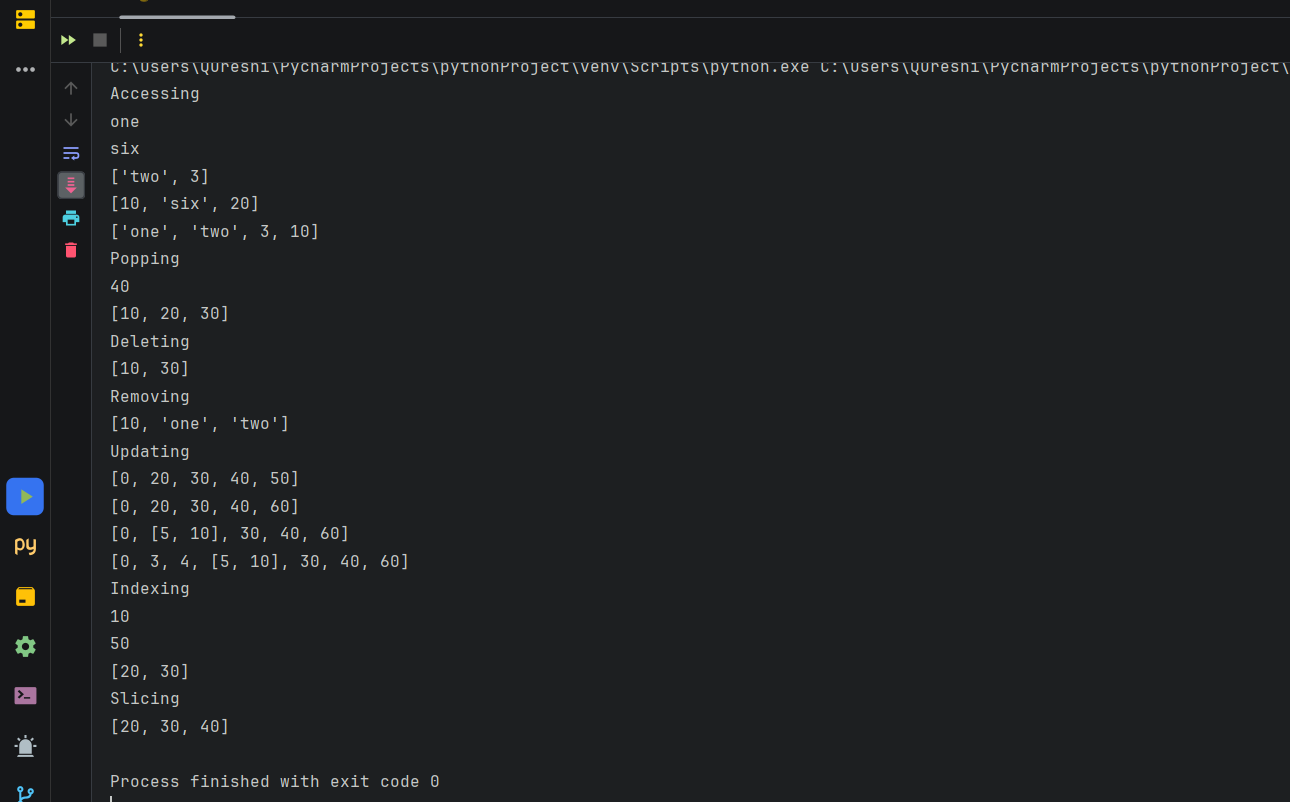
* returns view of all values in dictionary

Q8. Write basis operations of list. (W-22) (4M)

CODE:

list1 = ["one", "two", 3, 10, "six", 20]  
print("Accessing")  
print(list1[0])  
print(list1[-2])  
print(list1[1:3])  
print(list1[3:])  
print(list1[:4])  
  
list2 = [10, 20, 30, 40]  
print("Popping")  
print(list2.pop())  
print(list2)  
  
print("Deleting")  
del list2[1]  
print(list2)  
  
list3 = [10, "one", 20, "two"]  
list3.remove(20)  
print("Removing")  
print(list3)  
  
print("Updating")  
list4 = [10, 20, 30, 40, 50]  
list4[0] = 0  
print(list4)  
list4[-1] = 60  
print(list4)  
list4[1] = [5, 10]  
print(list4)  
list4[1:1] = [3, 4]  
print(list4)  
  
print("Indexing")  
list5 = [10, 20, 30, 40, 50]  
print(list5[0])  
print(list5[4])  
print(list5[1:3])  
  
print("Slicing")  
list6 = [10, 20, 30, 40, 50]  
print(list6[1:4])

OUTPUT:

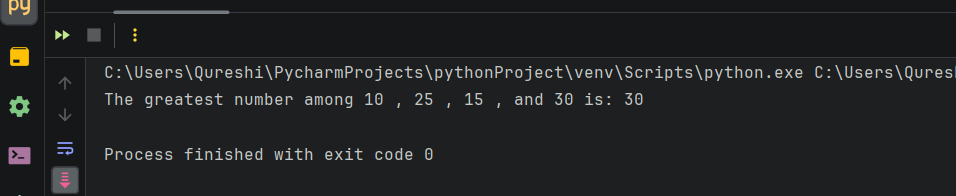
****

Q9. Write Python code for finding greatest among four numbers. (W-22) (4M)

CODE:

num1 = 10  
num2 = 25  
num3 = 15  
num4 = 30  
  
greatest = num1  
  
if num2 > greatest:  
 greatest = num2  
  
if num3 > greatest:  
 greatest = num3  
  
if num4 > greatest:  
 greatest = num4  
  
print("The greatest number among", num1, ",", num2, ",", num3, ", and", num4, "is:", greatest)

OUTPUT:

****

Q10. Compare list and dictionary.(W-22) (4M)

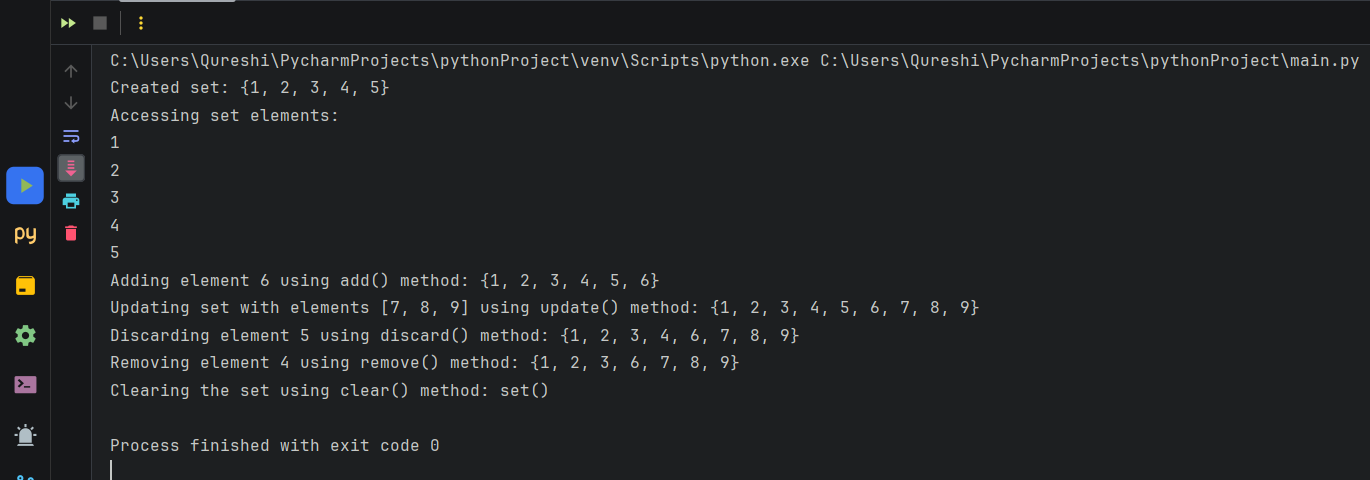
|  |  |  |
| --- | --- | --- |
| Feature | List | Dictionary |
| Definition | Ordered collection of elements | Unordered collection of key-value pairs |
| Mutability | Mutable | Mutable |
| Indexing | Accessed by index | Accessed by key |
| Memory Usage | Consumes more memory | Consumes less memory compared to list |
| Iteration | Slower due to linear search | Faster due to direct access by key |
| Elements | Homogeneous or heterogeneous | Heterogeneous (keys and values) |
| Usage | Suitable for sequences | Suitable for key-value mappings |
| Syntax | Defined by square brackets [] | Defined by curly braces {} |
| Example | my\_list = [1, 2, 3, 4, 5] | my\_dict = {'a': 1, 'b': 2, 'c': 3} |

Q11. Write python program to perform following operations on Set i) Create set ii) Access set Element iii) Update set iv) Delete set (W-22) (6M)

CODE:

my\_set = {1, 2, 3, 4, 5}  
print("Created set:", my\_set)  
  
print("Accessing set elements:")  
for element in my\_set:  
 print(element)  
  
my\_set.add(6)  
print("Adding element 6 using add() method:", my\_set)  
  
my\_set.update([7, 8, 9])  
print("Updating set with elements [7, 8, 9] using update() method:", my\_set)  
  
my\_set.discard(5)  
print("Discarding element 5 using discard() method:", my\_set)  
  
my\_set.remove(4)  
print("Removing element 4 using remove() method:", my\_set)  
  
my\_set.clear()  
print("Clearing the set using clear() method:", my\_set)

OUTPUT:

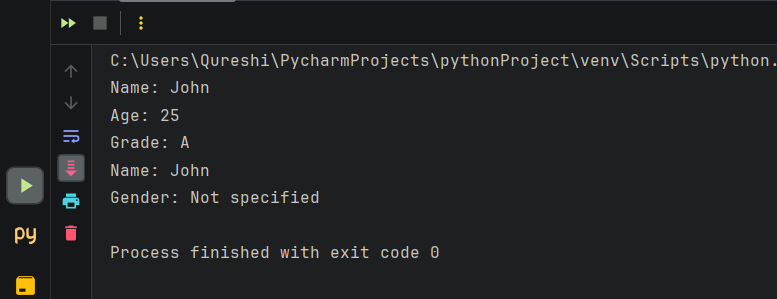
****

Q13. Explain creating dictionary and accessing dictionary elements with EXAMPLE (S-23) (4M)

CODE:

student = {  
 'name': 'John',  
 'age': 25,  
 'grade': 'A'  
}  
  
print("Name:", student['name'])  
print("Age:", student['age'])  
print("Grade:", student['grade'])  
  
print("Name:", student.get('name'))  
print("Gender:", student.get('gender', 'Not specified'))

OUTPUT:



Q14. Write a python program to input any two tuples and interchange the tuples variable (S-23) (4M)

CODE:

tuple1 = tuple(input("Enter elements of the first tuple separated by space: ").split())  
  
tuple2 = tuple(input("Enter elements of the second tuple separated by space: ").split())  
  
tuple1, tuple2 = tuple2, tuple1  
  
print("After interchanging:")  
print("First tuple:", tuple1)  
print("Second tuple:", tuple2)

OUTPUT:



Q15. Differentiate between: list and tuple (S-23) (4M)

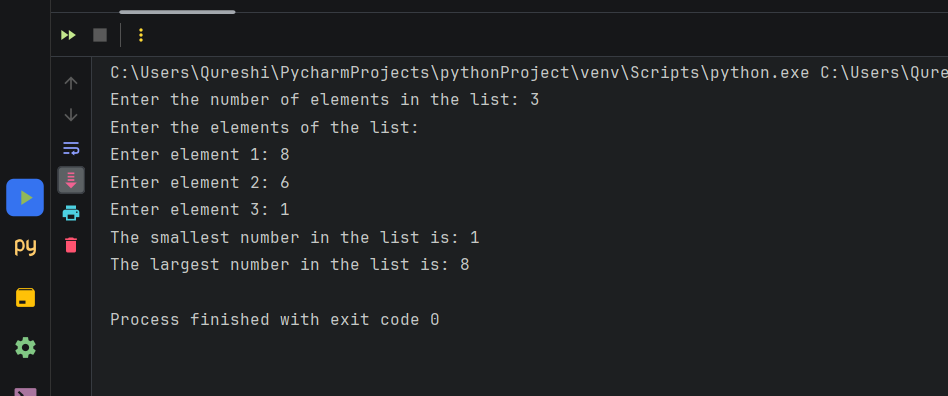
|  |  |  |
| --- | --- | --- |
| Feature | List | Tuple |
| Mutability | Mutable (Can be modified after creation) | Immutable (Cannot be modified after creation) |
| Syntax | Created using square brackets [] | Created using parentheses () |
| Performance | Slightly slower due to mutability | Slightly faster due to immutability |
| Usage | Suitable for collections of items that need to be modified or appended frequently | Suitable for fixed collections of items that do not need to be modified |
| Syntax for single element | [element] (with square brackets and a comma) | (element,) (with parentheses and a comma) |
| Example | my\_list = [1, 2, 3] | my\_tuple = (1, 2, 3) |

Q16. Write a python program to accept values from user in a list and find smallest and largest number in a list (S-23) (6M)

CODE:

num\_list = []  
num\_count = int(input("Enter the number of elements in the list: "))  
print("Enter the elements of the list:")  
for i in range(num\_count):  
 num = int(input(f"Enter element {i + 1}: "))  
 num\_list.append(num)  
  
if len(num\_list) == 0:  
 print("List is empty.")  
else:  
 smallest = min(num\_list)  
 largest = max(num\_list)  
 print(f"The smallest number in the list is: {smallest}")  
 print(f"The largest number in the list is: {largest}")

OUTPUT:

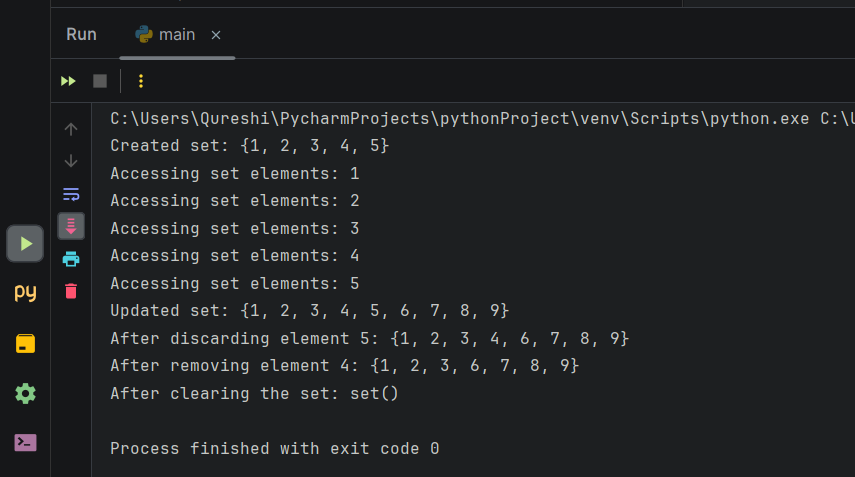
****

Q17. Explain any 6 set function with EXAMPLE (S-23) (6M)

CODE:

my\_set = {1, 2, 3, 4, 5}  
print("Created set:", my\_set)  
  
for element in my\_set:  
 print("Accessing set elements:", element)  
  
my\_set.add(6)  
my\_set.update([7, 8, 9])  
print("Updated set:", my\_set)  
  
my\_set.discard(5)  
print("After discarding element 5:", my\_set)  
  
my\_set.remove(4)  
print("After removing element 4:", my\_set)  
  
my\_set.clear()  
print("After clearing the set:", my\_set)

OUTPUT:



Q18. Describe various list functions. (Any 6)

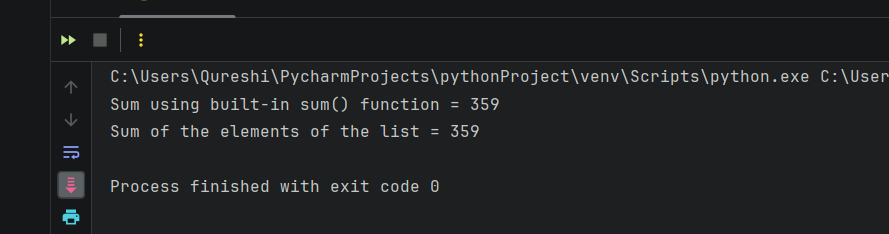
* sort(): Sorts the list in ascending order.
* type(list): It returns the class type of an object.
* append(): Adds one element to a list.
* extend(): Adds multiple elements to a list.
* index(): Returns the first appearance of a particular value.
* max(list): It returns an item from the list with a max value.
* min(list): It returns an item from the list with a min value.
* len(list): It gives the overall length of the list.
* clear(): Removes all the elements from the list.
* insert(): Adds a component at the required position.
* count(): Returns the number of elements with the required value.
* pop(): Removes the element at the required position.
* remove(): Removes the primary item with the desired value.
* reverse(): Reverses the order of the list.
* copy(): Returns a duplicate of the list.

Q19. Write a Python program to sum all the items in a list.

CODE:

aList = [73, 34, 76, 24, 55, 97]  
print("Sum using built-in sum() function =", sum(aList))  
  
Sum = 0  
for i in range(len(aList)):  
 Sum += aList[i]  
  
print("Sum of the elements of the list =", Sum)

OUTPUT:

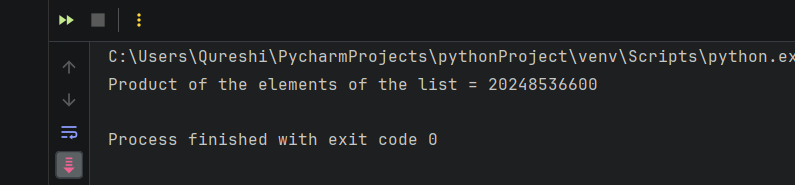
****

Q20. Write a Python program to multiplies all the items in a list.

CODE:

aList = [31, 43, 87, 24, 75, 97]  
Mul = 1  
for i in range(len(aList)):  
 Mul \*= aList[i]  
  
print("Product of the elements of the list =", Mul)

OUTPUT:

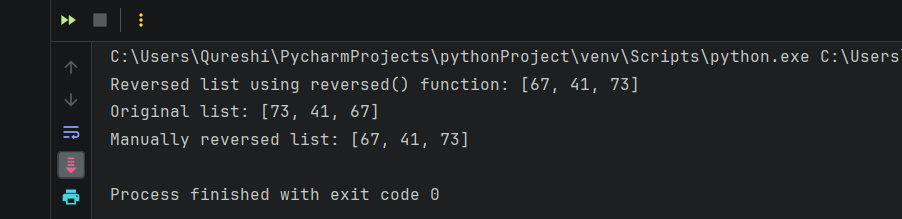


Q21. Write a Python program to reverse a list.

CODE:

aList = [73, 41, 67]  
print("Reversed list using reversed() function:", list(reversed(aList)))  
  
reversedList = list(range(len(aList)))  
iterator = 0  
for i in range(len(aList) - 1, -1, -1):  
 reversedList[iterator] = aList[i]  
 iterator += 1  
  
print("Original list:", aList)  
print("Manually reversed list:", reversedList)

OUTPUT:

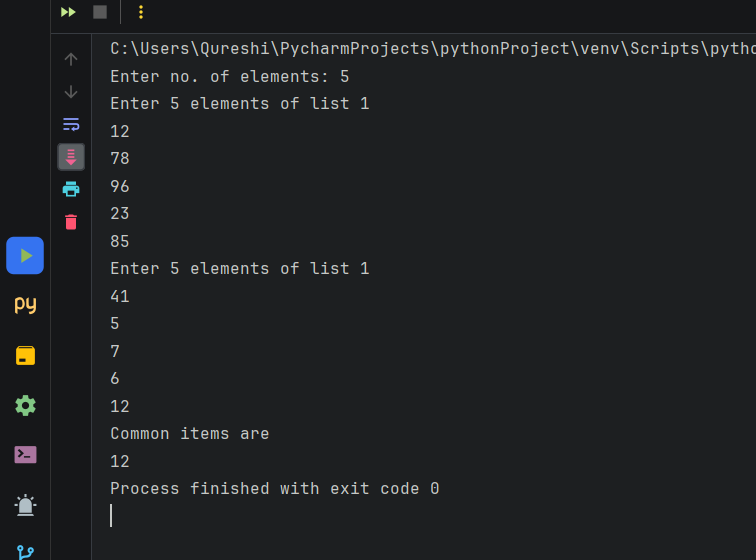


Q22. Write a Python program to find common items from two lists.

CODE:

list1 = []  
list2 = []  
n = int(input('Enter no. of elements: '))  
print('Enter', n, 'elements of list 1')  
for i in range(n):  
 list1.append(int(input()))  
  
print('Enter', n, 'elements of list 1')  
for i in range(n):  
 list2.append(int(input()))  
print('Common items are')  
for i in range(n):  
 for j in range(n):  
 if list1[i] == list2[j]:  
 print(list1[i], end="")

OUTPUT:

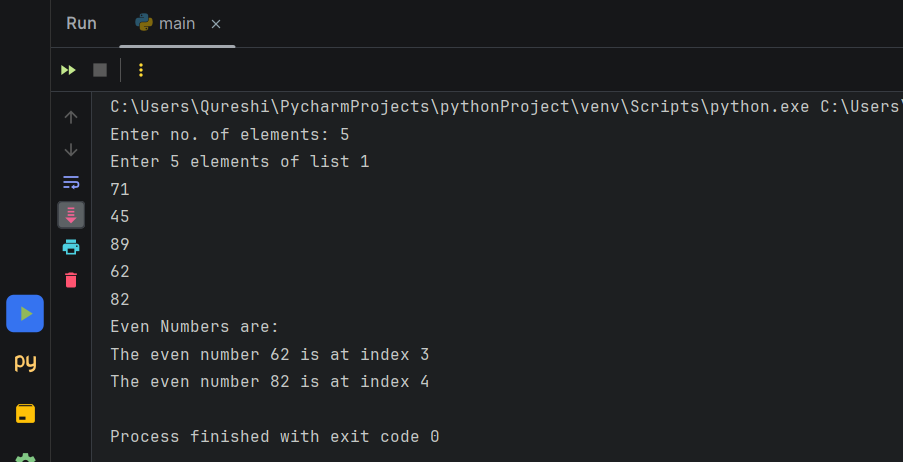


Q23. Write a Python program to select the even items of a list.

CODE:

list1=[]  
n=int(input('Enter no. of elements: '))  
print('Enter', n , 'elements of list 1')  
for i in range(n):  
 list1.append(int(input()))  
print('Even Numbers are: ')  
for i in range(n):  
 if list1[i]%2 ==0:  
 print('The even number',list1[i], 'is at index',i)

OUTPUT:

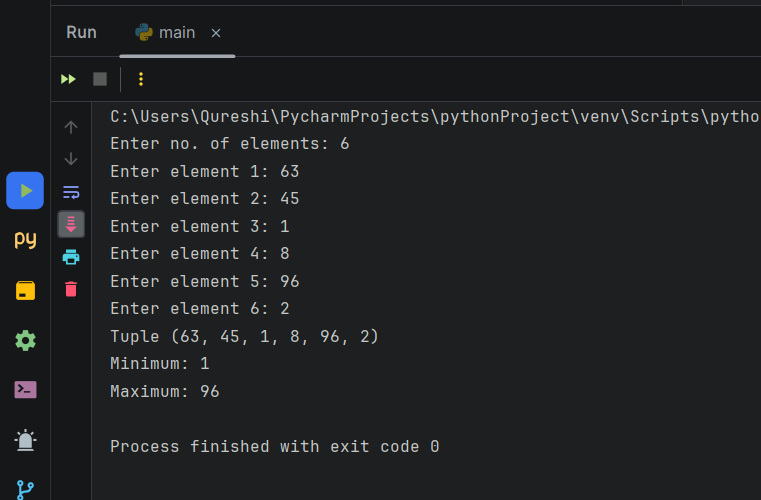


Q24. Create a tuple and find the minimum and maximum number from it.

CODE:

tup = tuple()  
n = int(input("Enter no. of elements: "))  
for i in range(n):  
 a = int(input(f"Enter element {i+1}: "))  
 tup = tup + (a,)  
 min = tup[0]  
 max = tup[0]  
  
  
for i in range(n):  
 if min > tup[i]:  
 min = tup[i]  
  
  
for i in range(n):  
 if max < tup[i]:  
 max = tup[i]  
print(f"Tuple {tup}")  
print(f"Minimum: {min}")  
print(f"Maximum: {max}")

OUTPUT:



Q25. Write a Python program to find the repeated items of a tuple.

CODE:

tup = tuple()

n = int(input("Enter no. of elements: "))

for i in range(n):

a = int(input(f"Enter element {i+1}: "))

tup = tup + (a,)

print("Repeated Element(s): ", end="")

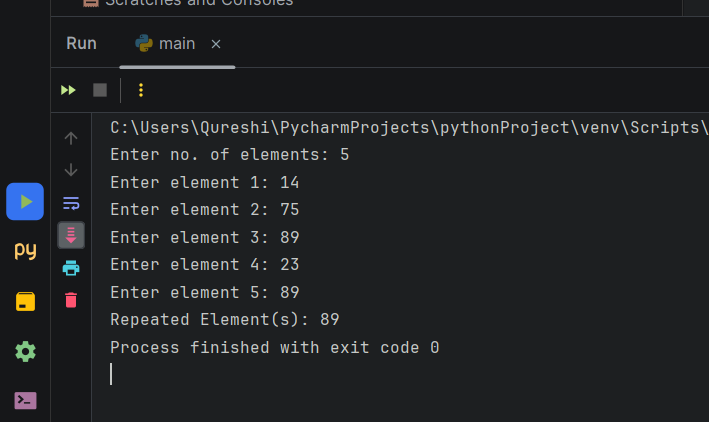
for i in range(0, len(tup)):

for j in range(i+1, len(tup)):

if tup[i] == tup[j]:

print(f"{tup[j]}", end=" ")

OUTPUT:



Q26. Describe the various methods of set

A set in Python is an unordered collection of unique elements. It is similar to a mathematical set where duplicate elements are not allowed. Sets are mutable, meaning you can add or remove elements from them.

There are various methods of set such as :-

* add()
* update()
* clear()
* remove()
* discard()
* pop()
* del

1.add(element):

Adds the specified element to the set if it's not already present.

**SYNTAX:**

set.add(element)

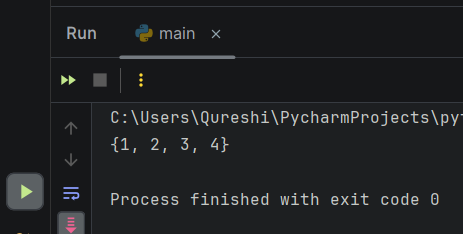
EXAMPLE:

my\_set = {1, 2, 3}

my\_set.add(4)

print(my\_set)

OUTPUT:



2.update(iterable):

Adds elements from the specified iterable to the set.

SYNTAX:

set.update(iterable)

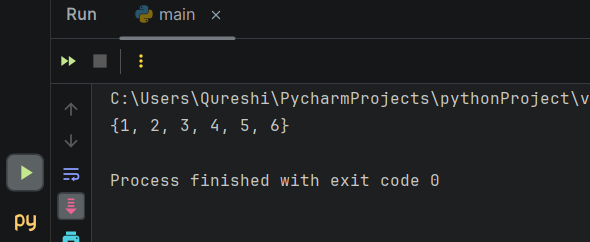
EXAMPLE:

my\_set = {1, 2, 3}

my\_set.update([4, 5, 6])

print(my\_set)

OUTPUT:



3.clear():

Removes all elements from the set.

SYNTAX:

set.clear()

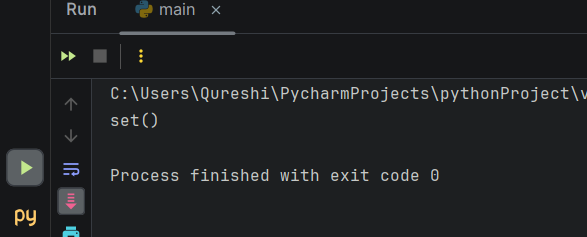
EXAMPLE:

my\_set = {1, 2, 3}

my\_set.clear()

print(my\_set)

OUTPUT:



4.remove(element):

Removes the specified element from the set. Raises a KeyError if the element is not present.

SYNTAX:

set.remove(element)

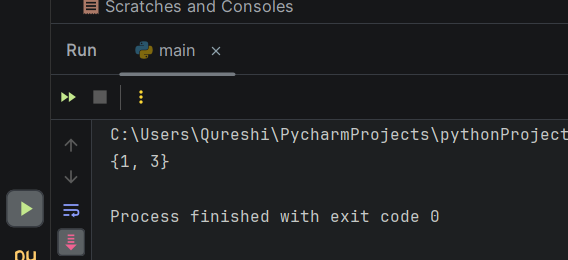
EXAMPLE:

my\_set = {1, 2, 3}

my\_set.remove(2)

print(my\_set)

OUTPUT:



5.discard(element):

Removes the specified element from the set if it's present. Does not raise an error if the element is not found.

SYNTAX:

set.discard(element)

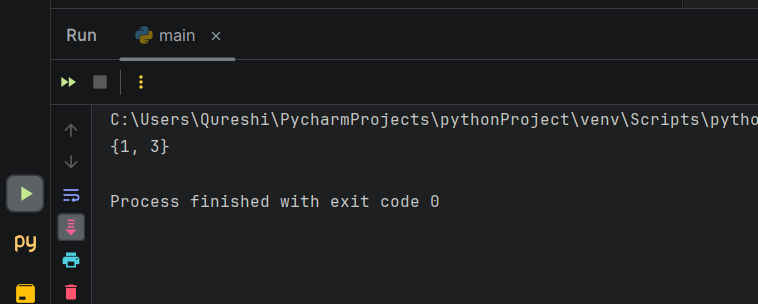
EXAMPLE:

my\_set = {1, 2, 3}

my\_set.discard(2)

print(my\_set)

OUTPUT:



6.pop():

Removes and returns an arbitrary element from the set.

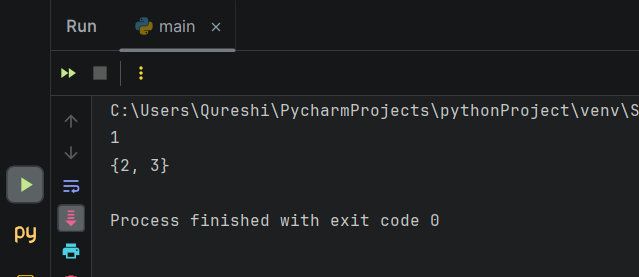
SYNTAX:

set.pop()

EXAMPLE:

my\_set = {1, 2, 3}  
popped\_element = my\_set.pop()  
print(popped\_element)  
print(my\_set)

OUTPUT:



7.del set\_name:

Deletes the entire set.

SYNTAX:

del set\_name

EXAMPLE:

my\_set = {1, 2, 3}  
del my\_set  
print(my\_set)

OUTPUT:



Q27. Write a Python program to create a set, add member(s) in a set and remove one item from set.

**CODE:**

a={1,2,3}

b=set([1,2,3])

c=set()

print ("set A=",a)

print ("set B=",b)

c.add(3)

c.update([9,10,11])

print ("set C after add and update=",c)

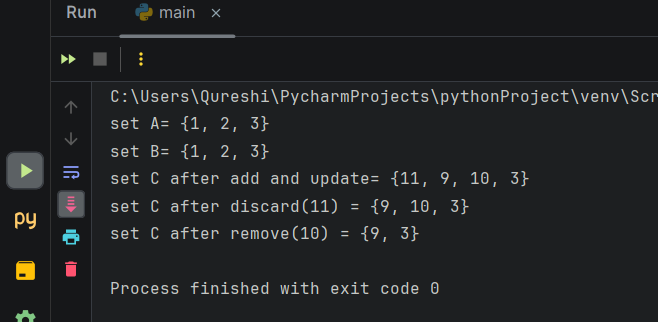
c.discard(11)

print ("set C after discard(11) =",c)

c.remove(10)

print ("set C after remove(10) =",c)

**OUTPUT:**

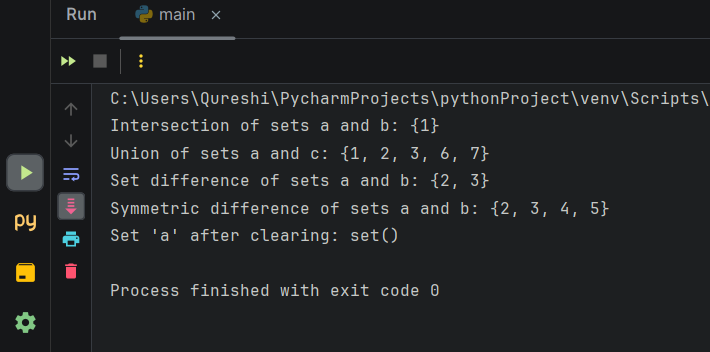


Q28. Write a Python program to perform following operations on set: intersection of sets, union of sets, set difference, symmetric difference, clear a set.

**CODE:**

a = {1, 2, 3}  
b = {4, 5, 1}  
c = {6, 7, 2}  
  
intersection\_result = a.intersection(b)  
print("Intersection of sets a and b:", intersection\_result)  
  
union\_result = a.union(c)  
print("Union of sets a and c:", union\_result)  
  
difference\_result = a.difference(b)  
print("Set difference of sets a and b:", difference\_result)  
  
symmetric\_difference\_result = a.symmetric\_difference(b)  
print("Symmetric difference of sets a and b:", symmetric\_difference\_result)  
  
a.clear()  
print("Set 'a' after clearing:", a)

**OUTPUT:**



Q29. Write a Python program to find maximum and the minimum value in a set.

**CODE:**

a={1,2,3,22,21,40,12}

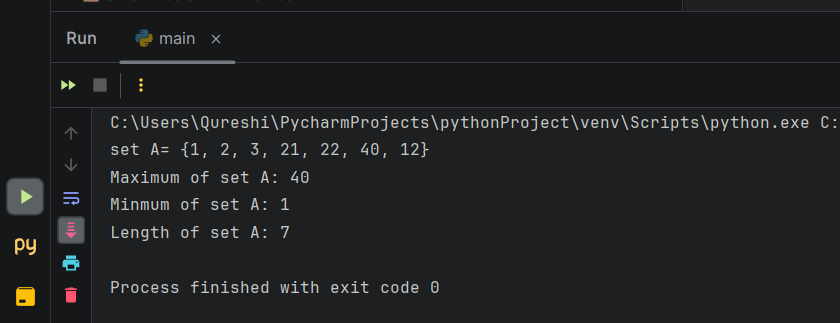
print ("set A=",a)

print("Maximum of set A:", max(a))

print("Minmum of set A:", min(a))

print("Length of set A:", len(a))

**OUTPUT:**

****

Q30. Write a Python program to find the length of a set.

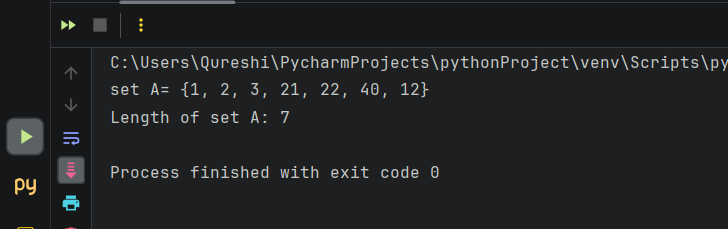
**CODE:**

a={1,2,3,22,21,40,12}

print ("set A=",a)

print("Length of set A:", len(a))

**OUTPUT:**



Q31. Write a Python script to sort (ascending and descending) a dictionary by value.

**CODE:**

d = {100: 1, 90: 4, 99: 3, 92: 1, 101: 1}

od = collections.OrderedDict(sorted(d.items()))

print("Using OrderedDict sorted by key:")

print(od)

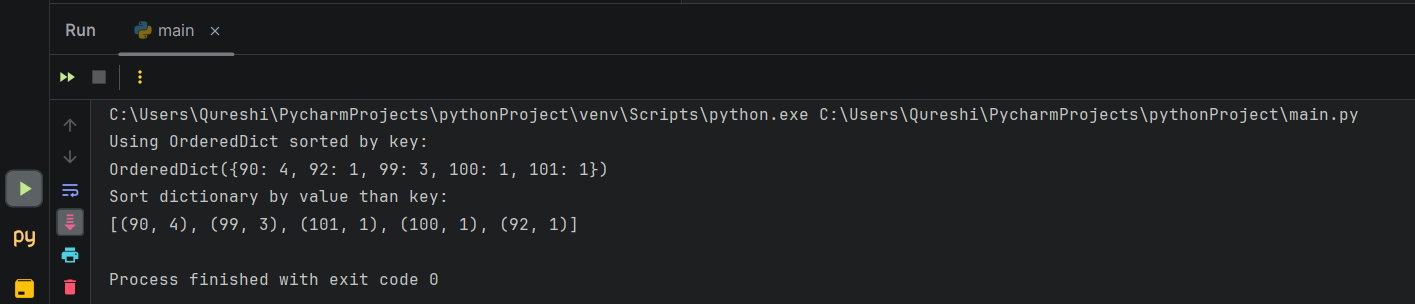
y = {100: 1, 90: 4, 99: 3, 92: 1, 101: 1}

n = sorted(y.items(), key=lambda x: (x[1], x[0]), reverse=True)

print("Sort dictionary by value than key:")

print(n)

**OUTPUT:**



Q32. Write a Python script to concatenate following dictionaries to create a new one.

a. Sample Dictionary:

b. dic1 = {1:10, 2:20}

c. dic2 = {3:30, 4:40}

d. dic3 = {5:50,6:60}

**CODE:**

DictionaryOne = {1: 10, 2: 20}

DictionaryTwo = {3: 30, 4: 40}

DictionaryThree = {5: 50, 6: 60}

DictionaryFour = DictionaryOne.copy()

DictionaryFour.update(DictionaryTwo)

DictionaryFour.update(DictionaryThree)

print("Method 1...")

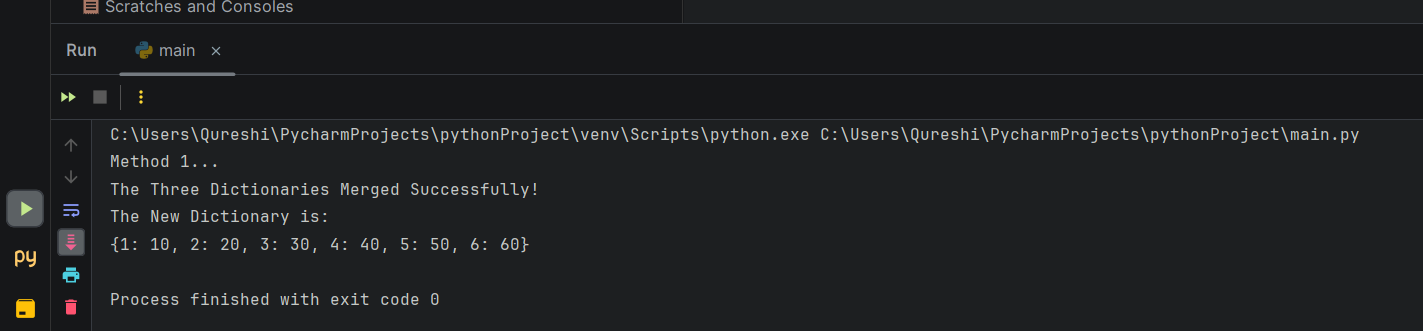
print("The Three Dictionaries Merged Successfully!")

print("The New Dictionary is:")

print(DictionaryFour)

DictionaryFour = {\*\*DictionaryOne, \*\*DictionaryTwo, \*\*DictionaryThree}

**OUTPUT:**

****

Q33. Write a Python program to combine two dictionary adding values for common keys.

a. d1 = {'a': 100, 'b': 200, 'c':300}

b. d2 = {'a': 300, 'b': 200, 'd':400}

**CODE:**

d1 = {'a': 100, 'b': 200, 'c': 300}

d2 = {'a': 300, 'b': 200, 'd': 400}

combined\_dict = {}

for key in d1.keys() | d2.keys():

combined\_dict[key] = d1.get(key, 0) + d2.get(key, 0)

print("Combined Dictionary with Added Values:")

print(combined\_dict)

**OUTPUT:**



Q34. Write a Python program to find the highest 3 values in a dictionary.

**CODE:**

my\_dict = {'a': 50, 'b': 30, 'c': 80, 'd': 20, 'e': 90, 'f': 60}

# Sorting the dictionary by values in descending order

sorted\_dict = sorted(my\_dict.items(), key=lambda x: x[1], reverse=True)

# Extracting the highest 3 values

highest\_3\_values = sorted\_dict[:3]

print("Original Dictionary:")

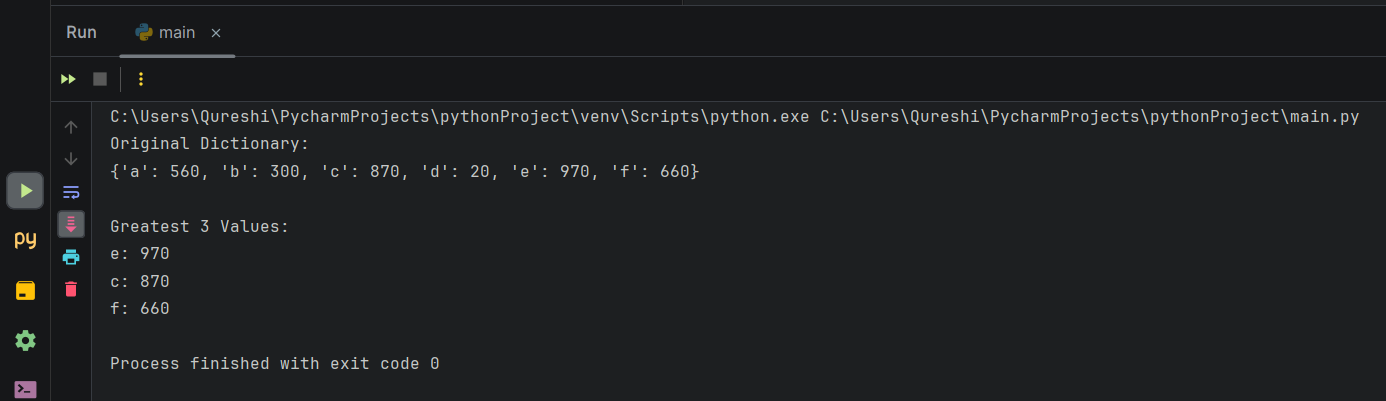
print(my\_dict)

print("\nTop 3 Values:")

for key, value in highest\_3\_values:

print(f"{key}: {value}")

**OUTPUT:**



Q35. Write a Python program to print all unique values in a dictionary.

Sample Data: [{"V":"S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII":"S005"}, {"V":"S009"}, {"VIII":"S007"}]

**CODE:**

L = [

{"V": "S001"},

{"V": "S002"},

{"VI": "S001"},

{"VI": "S005"},

{"VII": "S005"},

{"V": "S009"},

{"VIII": "S007"}

]

print("Original List:", L)

unique\_values = {val for dic in L for val in dic.values()}

print("Unique Values:", unique\_values)

**OUTPUT:**

