Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 8

Roll No : 352

Name : Smit Gohel

01) WAP to find their union, intersection, and difference of the given two sets.

Sample Input:

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}
```

Sample Output:

```
Union: {1, 2, 3, 4, 5, 6}
Intersection: {3, 4}
Difference (A-B): {1, 2}
```

In [3]:
```
A = set(map(int,input('Enter numbers:').split()))
B = set(map(int,input('Enter numbers:').split()))

print(f'Union A|B: {A.union(B)}')
print(f'Intersection A&B: {A.intersection(B)}')
print(f'Difference A-B: {A.difference(B)}')
```

```
Union A|B: {1, 2, 3, 4, 5, 6}
Intersection A&B: {3, 4}
Difference A-B: {1, 2}
```

02) WAP to Find elements common to all three given sets.

Sample Input:

```
A = {1, 2, 3, 4}
B = {2, 3, 5}
C = {2, 3, 6}
```

Sample Output:

```
{2, 3}
```

In [6]:
```
A = set(map(int,input('Enter numbers:').split()))
B = set(map(int,input('Enter numbers:').split()))
C = set(map(int,input('Enter numbers:').split()))

print(A.intersection(B.intersection(C))) # A & B & C
```

```
{2, 3}
```

03) WAP to remove duplicate elements from a list while preserving unique values.

Sample Input:

```
[10, 20, 10, 30, 20, 40]
```

Sample Output:

```
{10, 20, 30, 40}
```

In [6]:
```
l = list(map(int,input('Enter numbers:').split()))
s = set(l)
print(s)
```

```
{40, 10, 20, 30}
```

04) WAP to count how many elements are common between two sets.

Sample Input:

```
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}
```

Sample Output:

```
2
```

In [7]:
```
A = set(map(int,input('Enter numbers:').split()))
B = set(map(int,input('Enter numbers:').split()))

common_element = A.intersection(B)

print(len(common_element))
```

05) WAP to Merge two dictionaries. If a key exists in both dictionaries, add their values.

Sample Input:

```
Dict1 = {1: 10, 2: 20, 3: 30}
Dict2 = {2: 40, 3: 50, 4: 60}
```

Sample Output:

```
{1: 10, 2: 60, 3: 80, 4: 60}
```

In [24]:
```
l1 = list(map(int,input('Enter keys:').split()))
l2 = list(map(int,input('Enter values:').split()))

l3 = list(map(int,input('Enter keys:').split()))
l4 = list(map(int,input('Enter values:').split()))

Dict1 = dict(zip(l1,l2))
Dict2 = dict(zip(l3,l4))

Dict3 = {}

for key in Dict1.keys():
    if key in Dict2.keys():
        Dict3[key] = Dict1[key] + Dict2[key]
    else:
        Dict3[key] = Dict1[key]

for key in Dict2.keys():
    if key not in Dict1.keys():
        Dict3[key] = Dict2[key]

print(Dict3)

# for i in Dict2.keys():
#     if i in Dict1.keys():
#         Dict1.update([(i,Dict1[i]+Dict2[i])])
#     else:
#         Dict1.update([(i,Dict2[i])])

# print(Dict1)
```

```
{1: 10, 2: 60, 3: 80, 4: 60}
```

06) WAP to Sort a dictionary in ascending order of its values.

Hint: in sorted(), use udf for key parameter

Sample Input:

```
{'Math': 70, 'Physics': 85, 'Chemistry': 60}
```

Sample Output:

```
{'Chemistry': 60, 'Math': 70, 'Physics': 85}
```

In [25]:
```
key = list(map(str,input('Enter keys:').split()))
value = list(map(int,input('Enter values:').split()))

Dict = dict(zip(key,value))

def myFun(item):
    return item[1]

sdict = sorted(Dict.items(),key=myFun)   # sorted(iterable, key=function)
print(dict(sdict))

# sdict = sorted(Dict.items(), key=lambda x: x[1])
# print(dict(sdict))
```

```
{'Chemistry': 60, 'Math': 70, 'Physics': 85}
```

07) WAP to find the keys having maximum and minimum values in a dictionary.

Sample Input:

```
{'A': 50, 'B': 90, 'C': 30}
```

Sample Output:

```
Max Value Key: B
Min Value Key: C
```

In [26]:
```
l1 = list(map(str,input('Enter keys:').split()))
l2 = list(map(int,input('Enter values:').split()))

Dict = dict(zip(l1,l2))

min_value = min(Dict.values())
max_value = max(Dict.values())

for i in Dict.keys():
    if Dict[i] == min_value:
        min_key = i


for i in Dict.keys():
    if Dict[i] == max_value:
        max_key = i

print(min_key, max_key)
```

```
C B
```

08) WAP to create a dictionary with numbers from 1 to N as keys and their squares as values.

Sample Input:

```
N = 5
```

Sample Output:

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [38]:
```
N = int(input('Enter N:'))
l = list()

for i in range(1,N+1):
    l.append(i*i)

print(dict(enumerate(l, start = 1)))

# N = 5
# di=dict()

# for i in range(1,N+1):
#     di.update([(i,i*i)])
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

09) WAP to count the frequency of each character in a string using a dictionary.

Sample Input:

```
programming
```

Sample Output:

```
{'p': 1, 'r': 2, 'o': 1, 'g': 2, 'a': 1, 'm': 2, 'i': 1, 'n': 1}
```

In [10]:
```
s = input('Enter string:')
Dict = {}

for char in s:
    Dict.update([(char, s.count(char))]) #iterable obj list of tuple

print(Dict)

# Dict = {char: s.count(char) for char in s}
```

```
{'p': 1, 'r': 2, 'o': 1, 'g': 2, 'a': 1, 'm': 2, 'i': 1, 'n': 1}
```

10) WAP to swap keys and values of a dictionary.

Sample Input:

```
{1: 'A', 2: 'B', 3: 'C'}
```

Sample Output:

```
{'A': 1, 'B': 2, 'C': 3}
```

In [27]:
```
l1 = list(map(int,input('Enter keys:').split()))
l2 = list(map(str,input('Enter values:').split()))

Dict = dict(zip(l1,l2))

new_Dict = dict()
for i in Dict.keys():
    new_Dict.update([(Dict[i], i)]) #iterable obj list of tuple

print(new_Dict)
```

```
{'A': 1, 'B': 2, 'C': 3}
```

11) WAP to find the student with the highest total marks, given student names and their marks.

Sample Input:

```
{'Amit': [70, 80, 90], 'Riya': [85, 75, 80], 'Neha': [60, 70,
65]}
```

Sample Output:

```
Topper: Amit
```

In [19]:
```
n = int(input("Enter number of students: "))
Dict = {}

for _ in range(n):
    name = input("Enter student name: ")
    marks = list(map(int, input(f"Enter marks for {name} (space-separated): ").s
    Dict[name] = marks

sorted_students = sorted(Dict.items(), key=lambda x: sum(x[1]), reverse=True)

print(f"Topper: {sorted_students[0][0]}")
```

```
Topper: Amit
```

12) WAP to extract keys whose values are even numbers.

Sample Input:

```
{1: 11, 2: 20, 3: 15, 4: 40}
```

Sample Output:

```
[2, 4]
```

In [28]:
```
l1 = list(map(int,input('Enter keys:').split()))
l2 = list(map(int,input('Enter values:').split()))

Dict = dict(zip(l1,l2))

li = [key for key in Dict.keys() if Dict[key] % 2 == 0]

print(li)
```

```
[2, 4]
```

13) WAP to Find common keys between two dictionaries.

Sample Input:

```
Dict1 = {1: 'A', 2: 'B', 3: 'C'}
Dict2 = {2: 'X', 3: 'Y', 4: 'Z'}
```

Sample Output:

```
[2, 3]
```

In [23]:
```
Dict1 = {1: 'A', 2: 'B', 3: 'C'}
Dict2 = {2: 'X', 3: 'Y', 4: 'Z'}

common_keys = list(set(Dict1.keys()) & set(Dict2.keys()))
# common_keys = [key for key in Dict1 if key in Dict2]  Using dictionary compreh
# common_keys = list(Dict1.keys() & Dict2.keys())  Using intersection() method

print(common_keys)

# Taking input for dictionaries
# import ast

# Dict1 = ast.literal_eval(input("Enter first dictionary: "))
# Dict2 = ast.literal_eval(input("Enter second dictionary: "))

# Find common keys
# common_keys = list(set(Dict1.keys()) & set(Dict2.keys()))
# print(f"Common keys: {sorted(common_keys)}")
```

```
[2, 3]
```

14) WAP to handle missing keys in dictionaries.

Sample Input - 1:

```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = 'c'
```

Sample Output - 1:

```
8
```

Sample Input - 2:

```
dict1 = {'a': 5, 'c': 8, 'e': 2}
key = 'd'
```

Sample Output - 2:

```
Key Not Found
```

In [30]:
```
l1 = list(map(str,input('Enter keys:').split()))
l2 = list(map(int,input('Enter values:').split()))
key = input('Enter key:')
Dict = dict(zip(l1,l2))

value = Dict.get(key)

if value:
    print(value)
else:
    print("Key Not Found")
```

```
Key Not Found
```

15) WAP to create a dictionary using dictionary comprehension from two lists containing student names and marks. Include only those students in the dictionary whose marks are greater than or equal to 75.

Sample Input:

```
students = ['Amit', 'Riya', 'Neha', 'Karan']
marks = [78, 72, 85, 60]
```

Sample Output:

```
{'Amit': 78, 'Neha': 85}
```

In [31]:
```
students = list(map(str,input('Enter name of students:').split()))
marks = list(map(int,input('Enter marks of students:').split()))

Dict = dict(zip(students, marks))

new_Dict = {key: value for key,value in Dict.items() if value >= 75}

# new_Dict = {students[i]: marks[i] for i in range(len(students)) if marks[i] >=

# new_Dict = {name: mark for name, mark in zip(students, marks) if mark >= 75}

print(new_Dict)
```

```
{'Amit': 78, 'Neha': 85}
```