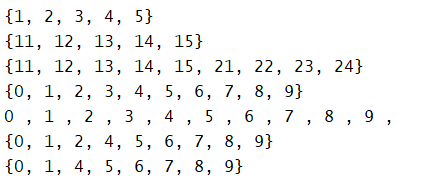
**-: Practical Set – 5 :-**

1. Create a set of integers as follows: initialize the set directly, initialize empty set and then add values (from a list, from another set, using range, updating an existing set using another set, print the elements of set iteratively, check the functionality of remove and discard.)

my\_set = {1, 2, 3, 4, 5}  
print(my\_set)  
my\_set2 = set(())  
list1 = [11, 12, 13, 14, 15]  
my\_set2.update(list1)  
print(my\_set2)  
  
set3 = set((21, 22, 23, 24))  
my\_set2.update(set3)  
print(my\_set2)  
  
for i in range(0, 10):  
 my\_set.add(i)  
print(my\_set)  
  
for element in my\_set:  
 print(element, "," , end=" ")  
print("")  
  
my\_set.remove(3)  
print(my\_set)  
  
my\_set.discard(2)  
print(my\_set)

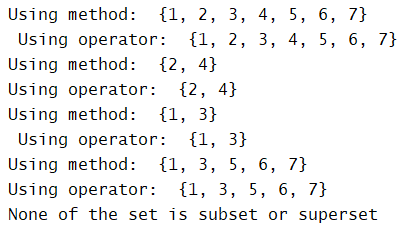
**OUTPUT:**



1. Create two sets of integers and find their difference, intersection, union and symmetric difference. Also find subset and superset from these two. Apply methods as well as operators for all operations.

A = {1, 2, 3, 4}  
B = {2, 4, 5, 6, 7}  
print("Using method: ", A.union(B), "\n Using operator: ", A|B)  
print("Using method: ", A.intersection(B), "\nUsing operator: ", A & B)  
print("Using method: ", A.difference(B), "\n Using operator: ", A-B)  
print("Using method: ", A.symmetric\_difference(B), "\nUsing operator: ", A ^ B)  
if(A.issubset(B)):  
 print("A is subset of B")  
 print("B is superset of A")  
elif(B.issubset(A)):  
 print("B is subset of A")  
 print("A is superset of B")  
else:  
 print("None of the set is subset or superset")

**OUTPUT:**



1. Write a function called find\_dups that takes a list of integers as its input argument and returns a set of those integers that occur two or more times in the list.

def find\_dups(list1):  
 duplicates = []  
 for i in list1:  
 if(list1.count(i) >= 2):  
 if i not in duplicates:  
 duplicates.append(i)  
 return duplicates  
list1 = [1, 2, 3, 4, 5, 1, 2]  
print(find\_dups(list1))

**OUTPUT:**



1. The following company details are given for analysis: customer acc no, customer name, purchased product no, product category, unit price. Marketing is interested in understanding customer purchase patterns. Find the answers of following question: How many customers have purchased bread? Butter? Bread and butter? Bread but not butter? Bread, butter and milk? Print name of the most valuable customers who have purchased all three items.

names={1:["Yash"],2:["Manan"],3:["Vivek"]}

purchases={1:[2,1,3],2:[1,0,0],3:[1,1,2]}

bread=butter=milk=cbread=cbutter=cmilk=breadnbutter=sum=0

breadnotbutter=[]

bbm=[]

valuable=[]

for i in names:

if purchases[i][0]>0:

bread+=int(purchases[i][0])

cbread+=1

if purchases[i][1]>0:

bread+=int(purchases[i][1])

cbutter+=1

if purchases[i][2]>0:

bread+=int(purchases[i][2])

cmilk+=1

if purchases[i][0]>0 and purchases[i][1]>0:

breadnbutter+=1

if purchases[i][0]>0 and purchases[i][1]==0:

breadnotbutter.append(names[i][0])

if purchases[i][0] > 0 and purchases[i][1]>0 and purchases[i][2]>0:

bbm.append(names[i][0])

print("\nHow many customers have purchased bread?")

print(cbread)

print("\nHow many customers have purchased butter?")

print(cbutter)

print("\nHow many customers have purchased bread and butter?")

print(breadnbutter)

print("\nWho has purchased bread but not butter?")

if len(breadnotbutter)==0:

print("No One")

else:

print(breadnotbutter)

print("\nWhich customers have purchased bread, butter and milk?")

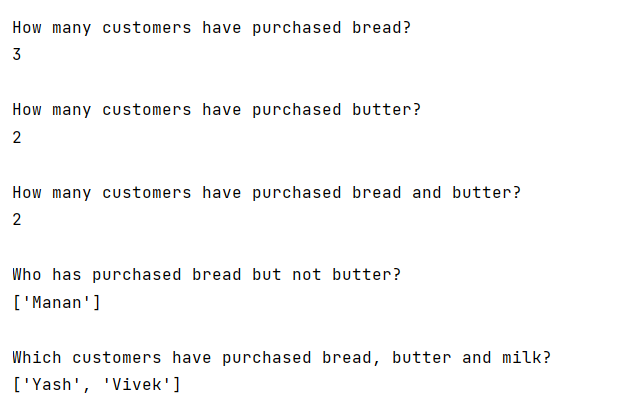
if len(bbm)==0:

print("No One")

else:

    print(bbm)

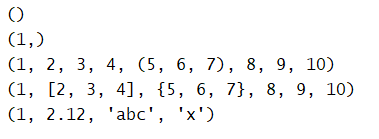
**OUTPUT:**

****

1. WAP to create an empty tuple, tuple with single value, tuple with multiple values/ collections and a tuple with different data types.

tuple1 = ()  
print(tuple1)  
tuple2 = (1,)  
print(tuple2)  
tuple3 = (1, 2, 3, 4, (5, 6, 7), 8, 9, 10)  
print(tuple3)  
tuple4 = (1, [2, 3, 4], {5, 6, 7}, 8, 9, 10)  
print(tuple4)  
tuple5 = (1, 2.12, "abc", 'x')  
print(tuple5)

**OUTPUT:**



1. Check all the methods of tuple.

tuple1 = (1, 2, 3, 4, 5, 4, 3)  
print(tuple1.count(4))  
print(tuple1.index(5))

**OUTPUT:**



1. WAP to find multiple items of a tuple.

tuple1 = (1, 2, 3, 4, 5, 6, 1, 2, 3)  
def findRepeatedItems(tuple1):  
 itemsList = []  
 for i in tuple1:  
 if tuple1.count(i) >= 2:  
 if i not in itemsList:  
 itemsList.append(i)  
 return itemsList  
print(findRepeatedItems(tuple1))

**OUTPUT:**



1. WAP script to add a key to a dictionary.

dict1 = {'name' : 'abc', 'id' : 1}  
dict1['surname'] = 'xyz'  
print(dict1)

**OUTPUT:**



1. WAP to print a dictionary where the keys are numbers between 1 and 15(both included) and the values are square of keys.

dict1 = dict()  
for i in range(1, 16):  
 dict1[i] = i\*\*2  
print(dict1)

**OUTPUT:**

****

1. Write a Python script to check if a given key already exists in a dictionary.

dict1 = {"name" : "Yash", "surname" : "Dhuliya", "city" : "Rajkot"}  
key1, key2 = "surname", "Surname"  
print(key1 in dict1)  
print(key2 in dict1)

**OUTPUT:**



1. Write a Python script to merge two Python dictionaries.

dict1 = {"name" : "Yash", "surname" : "Dhuliya", "city" : "Rajkot"}  
dict2 = {"enroll" : "33", "language" : "Python"}  
dict1.update(dict2)  
print(dict1)

**OUTPUT:**



1. Write a python program to remove a key from a dictionary.

dict1 = {"name" : "Yash", "surname" : "Dhuliya", "city" : "Rajkot"}  
dict1.pop("city")  
print(dict1)

**OUTPUT:**



1. Write a Python program to create a dictionary from two lists.

list1 = ["animal", "bird", "thing"]  
list2 = ["cat", "sparrow", "icecream"]  
dict1 = dict()  
for i in list1:  
 for j in list2:  
 dict1[i] = j  
 list2.remove(j)  
 break  
  
print(dict1)

**OUTPUT:**



1. Write a Python program to check if all dictionaries in a list are empty or not.

list1 = ["hello", {"animal" : "cat", "bird" : "sparrow"}, {}, "world"]  
for i in list1:  
 if type(i) is dict:  
 empty\_flag = False  
 if len(i) == 0:  
 empty\_flag = True  
 if empty\_flag == False:  
 break  
  
print("Not all dictionaries inside list are empty") if empty\_flag == False else print("All dictionaries inside list are empty")

**OUTPUT:**

