In [1]:
```
#Sample program for Dictionary?
#{1:1,2:4,3:9,.....10:100}
d={}
for i in range(1,11):
    d[i]=i**2
print(d)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

In [2]:
```
#Sample program for Dictionary?
#{1:2,3:6,5:10}
d={}
for i in range(1,11,2):
    d[i]=i*2
print(d)
```

```
{1: 2, 3: 6, 5: 10, 7: 14, 9: 18}
```

In [3]:
```
#keys are duplicate or not?
x={1: 2, 3: 6, 5: 10, 1: 14, 9: 18}
x
#Note: keys are not duplicates if we are using duplicate key then the old key will be replaced
#with the new key
```

Out[3]:
```
{1: 14, 3: 6, 5: 10, 9: 18}
```

In [5]:
```
#Builtin Functions
#update()
d={"name":"Pratyush","Roll No":98,"Address":"Delhi"}
d1={"Education":"B.Tech Cse","Graduation Marks":90}
print(id(d))
d.update(d1)
print(id(d))
```

```
1446172132352
1446172132352
```

In [ ]:
```
#Functions that are used for iterations
#keys()--> returns all keys associate with dictionary
#Values()-->return all values associated with keys of the dictionary
#items--> it returns the key value pair in the form of tuple
```

In [13]:
```
#Example of Each Functions
```

In [12]:
```
#keys()-->returns all keys associate with dictionary
d={"name":"Pratyush","Roll No":98,"Address":"Delhi"}
print(d.keys())   #return keys in form of tuple inside the list
for i in d.keys():
    print(i,end=",")
```

```
dict_keys(['name', 'Roll No', 'Address'])
name-Roll No-Address-
```

In [16]:
```
#Values()->return all values associated with keys of the dictionary
d={"name":"Pratyush","Roll No":98,"Address":"Delhi"}
print(d.values())   #return values in form of tuple inside the list
for i in d.values():
    print(i,end=" ,")
```

```
dict_values(['Pratyush', 98, 'Delhi'])
Pratyush ,98 ,Delhi ,
```

In [19]:
```
#items--> it returns the key value pair in the form of tuple
d={"name":"Pratyush","Roll No":98,"Address":"Delhi"}
print(d.items())
for i, j in d.items():
    print("Keys is "+str(i)+"  Value is "+str(j))
```

```
dict_items([('name', 'Pratyush'), ('Roll No', 98), ('Address', 'Delhi')])
Keys is name  Value is Pratyush
Keys is Roll No  Value is 98
Keys is Address  Value is Delhi
```

In [ ]:
```
#Introduction to sets
--> if we want to represent a group of unique value as a single enitity the we should go for sets.
-->Duplicates are not allowed.
---> Indexing is not important because sets are unordered.
---> Indexing and slicing is not possible
--> Disimilar elements are allowes
---> Sets are mutable(changeable)
---> Sets allows mathematical operations like intersection , union , difference etc
---> {} brackets are used for sets
```

In [21]:
```
#how to create set object
x={}
print(type(x))
```

```
<class 'dict'>
```

In [22]:
```
x=set()
print(type(x))
```

```
<class 'set'>
```

In [28]:
```
x={10,20,30,40,50,50}
print(x)
print(type(x))
```

```
{50, 20, 40, 10, 30}
<class 'set'>
```

In [34]:
```
#Importannt functions related to sets
#add(x)-->add item in the set(x)
x={10,20,30,40,50,50}
print(id(x))
x.add("Python")
print(x)
print(id(x))
print(type(x))
```

```
1446172206912
{'Python', 50, 20, 40, 10, 30}
1446172206912
<class 'set'>
```

In [35]:
```
#update()--> to add multiple item in the set
x={10,20,30,40,50,50}
y={90,80,70}
x.update(y)
x
```

Out[35]:
```
{10, 20, 30, 40, 50, 70, 80, 90}
```

In [41]:
```
#update()--> to add multiple item in the set
x={10,20,30,40,50,50}
y={90,80,70}
x.update(y)
x
```

Out[41]:
```
{10, 20, 30, 40, 50, 70, 80, 90}
```

In [ ]:
```
which of the following are valid for set(s)?
#s.add(10)     #Valid
#s.add(10,20,30)    #Type error
#s.update(10)        # Type error --> int we are giving but update function is looking for sequence
#.s.update(range(1,10))  #Valid
```

In [42]:
```
#Pop()-->remove random element
x={10,20,30,40,50,50}
print(x.pop())
print(x)
```

```
50
{20, 40, 10, 30}
```

In [49]:
```
#remove function --> deleted the specific iitem of the set
x={10,20,30,40,50,50}
x.remove(20)
print(x)
```

```
{50, 40, 10, 30}
```

In [50]:
```
#Discard --> deleted the specific item of the set
x={10,20,30,40,50,50}
x.discard(200)
print(x)
```

```
{50, 20, 40, 10, 30}
```

In [ ]:
```
remove vs discard
Remove function will give you error when the element is not present
whereas discard will never give you an error
```

In [51]:
```
#clear--> remove all items from the set
x={10,20,30,40,50,50}
x.clear()
print(x)
```

```
set()
```

In [ ]:
```
extend() --> +
```

In [55]:
```
#Mathematical Operations on Sets
#!.Union -->return all the elements from both the sets but not duplicate
x={10,20,30,40,50,50}
y={90,80,70}
print(x.union(y))   #--> + Opertator
```

```
{80, 50, 20, 90, 70, 40, 10, 30}
```

In [56]:
```
#Intersection --> return common elemnt of the set
x={10,20,30,40,50,50}
y={90,40,50}
print(x.intersection(y))
```

```
{40, 50}
```

In [62]:
```
#Difference --> return the element that are present in first set but not in second
x={10,20,30,40,50,50}
y={90,40,50}
print(x.difference(y))
```

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

In [60]:
```
#Membership Opertor in not in
x={10,20,30,40,50,50}
100 in x
```

Out[60]:
```
False
```

In [64]:
```
#Vowels --> print all different vowels that are present in the given string  --> aeiou
#"Mohammedea"
str1=input()
s1=set(str1)
print(s1)
s2={"a","i","e","o","u"}
print(s2)
s3=s1.intersection(s2)
print(s3)
```

```
mohitkumar
{'r', 'm', 'i', 'u', 'h', 'o', 't', 'k', 'a'}
{'a', 'o', 'e', 'i', 'u'}
{'o', 'i', 'a', 'u'}
```

In [90]:
```
#Python program to print the sum of unqiue numbers that are present in the given list.
#[10,20,30,40,50,60,70,10,20,30,40,50]
n=int(input()) #10
list2=[]
for i in range(0,n):
    s1=int(input())
    list2.append(s1)
print(list2)
set_1=set(list2)
print(sum(set_1))
```

```
5
10
20
30
10
20
[10, 20, 30, 10, 20]
60
```

In [83]:
```
n=int(input()) #10
list2=[]
for i in range(0,n):
    s1=int(input())
    list2.append(s1)
print(list2)
set_1=set(list2)
print(set_1)
```

```
6
10
20
30
10
20
30
[10, 20, 30, 10, 20, 30]
60
```

In [88]:
```
#d={1:2,2:3,3:4,4:5}
d={}
for i in range(1,5):
    d[i]=i+1
print(d)
```

```
{1: 2, 2: 3, 3: 4, 4: 5}
```

In [ ]:
```

```