len function

In [1]:
```
#In case of dictionary the len function will return the total number of key_value pair.
x={"name":"Pratyush Srivastava","class":"M.Tech"}
len(x)
```
Out[1]:
```
2
```
In [2]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
len(x)
```
Out[2]:
```
4
```

pop Function

In [ ]:
```
pop function --> will delete the element from a dictionary based on the given key.
                and it will return the deleleted element
syntax:
    dict_name.pop(key)
```
In [3]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.pop("name"))
print(x)
```
```
Pratyush Srivastava
{'class': 'M.Tech', 'Qualification': 'M.Tech', 'Achievement': 'AIR 6'}
```
In [4]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.pop("class")) #M.Tech
print(x) #M.Tech will be not shown
```
```
M.Tech
{'name': 'Pratyush Srivastava', 'Qualification': 'M.Tech', 'Achievement': 'AIR 6'}
```
In [5]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.pop("name")) #M.Tech
print(x) #M.Tech will be not shown
```
```
Pratyush Srivastava
{'class': 'M.Tech', 'Qualification': 'M.Tech', 'Achievement': 'AIR 6'}
```
In [6]:
```
x={}
print(x.pop("name")) #M.Tech
print(x) #M.Tech will be not shown
```
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [6], in <cell line: 2>()
      1 x={}
----> 2 print(x.pop("name")) #M.Tech
      3 print(x)

KeyError: 'name'
```

popItem Function

In [26]:
```
#It will delete the last key value pair of the dictionary
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.popitem())
print(x)
```
```
('Achievement', 'AIR 6')
{'name': 'Pratyush Srivastava', 'class': 'M.Tech', 'Qualification': 'M.Tech'}
```
In [27]:
```
x={}
print(x.popitem())
print(x)
```
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [27], in <cell line: 2>()
      1 x={}
----> 2 print(x.popitem())
      3 print(x)

KeyError: 'popitem(): dictionary is empty'
```

Get Function

In [ ]:
```
get function is used to return the value associated with the given key(accessing the value)
Syntax:

    dict_name.get(key)
```
In [32]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.get("500"))  #If the key is not present in the dictionary then get will not give u an error
print(x["500"])     #You will get key error if the key is not present
```
```
None
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [32], in <cell line: 3>()
      1 x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
      2 print(x.get("500"))  #If the key is not present in the dictionary then get will not give u an error
----> 3 print(x["500"])

KeyError: '500'
```

update Function

In [ ]:
```
update function  --> It will simply add two dictionary into single one
Syntax:
    first_dict.update(Second_dict)
```
In [35]:
```
d1={1:2,"Module":"Python","Name":"Pratyush"}
d2={7:8,9:10,10:11}
d1.update(d2)
print(d1)
```
```
{1: 2, 'Module': 'Python', 'Name': 'Pratyush', 7: 8, 9: 10, 10: 11}
```

Iterations Functions Like keys,values and items

In [ ]:
```
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
For iterate over the dictionary we are having three functions:
    keys --> return the keys
    values --> return the values
    items --> both key and value
```

Keys Function

In [38]:
```
#keys --> It is used to return all the keys of the dictionary
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.keys())
for i in x.keys():
    print(i)
```
```
dict_keys(['name', 'class', 'Qualification', 'Achievement'])
name
class
Qualification
Achievement
```

Values Function

In [40]:
```
#values --> It is used to return all the values of the dictionary
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.values())
for i in x.values():
    print(i)
```
```
dict_values(['Pratyush Srivastava', 'M.Tech', 'M.Tech', 'AIR 6'])
Pratyush Srivastava
M.Tech
M.Tech
AIR 6
```

Items Function

In [7]:
```
#items--> It is used to return all the keys and values of the dictionary
x={"name":"Pratyush Srivastava","class":"M.Tech","Qualification":"M.Tech","Achievement":"AIR 6"}
print(x.items()) #Return the answer in form of tuple inside the list
for i ,j in x.items():
    print("Key is ",i)
    print("Value is ",j)
```
```
dict_items([('name', 'Pratyush Srivastava'), ('class', 'M.Tech'), ('Qualification', 'M.Tech'), ('Achievement', 'AIR 6')])
Key is  name
Value is  Pratyush Srivastava
Key is  class
Value is  M.Tech
Key is  Qualification
Value is  M.Tech
Key is  Achievement
Value is  AIR 6
```

Dictionary and Mutability

In [49]:
```
x={1:2,3:4,5:6,7:8}
print(id(x))
y={13:4,50:60,70:90}
x.update(y)
print(id(x))
x
Note --> Dictionaries are mutable that means if you are going to perform any change
in the dictionary then that change will be done into the original dictionary a new dictionary object
will not be created because of that change.
```
```
2534593172992
2534593172992
```
Out[49]:
```
{1: 2, 3: 4, 5: 6, 7: 8, 13: 4, 50: 60, 70: 90}
```
In [51]:
```
x=10
print(id(x))
x=x+10
print(id(x))
```
```
2534506130000
2534506130320
```

Introduction to Sets

In [ ]:
```
1.Set is also a collection of eleemnt inwhich duplicates are not allowes
2.Indexing is not possible (Unordered collection of eleement)
3.slicing and indexing is not possible
4.Sets are mutable that means we can perform any change in it.
5.Disimilar elements are also allowed in set
6.Curly braces are used for represneitn a set
7.Matematical operations are also possible in case of set(Intersection,union,diffrence)
```

Creation of Set Object

In [ ]:
```
#creation of set object:
empty set:
    s = set()
if you know th number of element:
    s={10,20,30,40,50}
```
In [53]:
```
s=set()
type(s)
```
Out[53]:
```
set
```
In [54]:
```
s={10,20,30,40,50}
type(s)
```
Out[54]:
```
set
```

Insertion Functions of Set --> add and update

In [ ]:
```
for adding element into a set we are having two functions:
    1.add()
    2.update()
```

add Function

In [60]:
```
#add()--> at a single element into the set
s={10,20,30,40,"hello"}
print(id(s))
s.add("Python")
print(id(s))
print(s)
#You can only add one element at a time
```
```
2534593578592
2534593578592
{'hello', 20, 'Python', 40, 10, 30}
```

update Function

In [8]:
```
#If you want to add multiple element into your set then we should use update functio
s={10,20,30,40,"hello"}
s1={"Python","Good","Boy"}
s.update([10,20])
print(s)
#Update function will always take a sequence if you are giving any lother thing then it will give
#u an error
```
```
{20, 30, 40, 10, 'hello'}
```
In [ ]:
```
Which of the following is valid with respect to set?
1 s.add(hello)   #Invalid
2 s.add(10,20,30)  #Invalid
3 s.add("10 20")    #valid
4 s.update(10)     #invalid
5.  s.update([10])   #valid
6.s.update("100")  #valid
7.s.update([10,20,30])  #valid
```

Deletion Function of Set --> pop,remove and discard

In [ ]:
```
For deletion in   set we are having  3 functions:
    pop
    remove
    discard
```

pop Function

In [69]:
```
#return the deleted element and it will delete any element randomly
x={10,20,30,40,50,60}
print(x.pop())
print(print(x))
```
```
50
{20, 40, 10, 60, 30}
None
```
In [72]:
```
x={10,20,30,40,50,60}
print(x.pop())
print(print(x))
```
```
50
{20, 40, 10, 60, 30}
None
```

remove Function

In [ ]:
```
remove -->delete the specified element from a set.
syntax:
    set_name.remove(deleted_item)
```
In [73]:
```
x={10,20,30,40,50,60}
x.remove(50)
x
```
Out[73]:
```
{10, 20, 30, 40, 60}
```
In [75]:
```
x={10,20,30,40,50,60}
x.remove(900)
x
```
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [75], in <cell line: 2>()
      1 x={10,20,30,40,50,60}
----> 2 x.remove(900)
      3 x

KeyError: 900
```

discard Function

In [ ]:
```
it will also deleted the specified item of the set
```
In [76]:
```
x={10,20,30,40,50,60}
x.discard(50)
x
```
Out[76]:
```
{10, 20, 30, 40, 60}
```
In [77]:
```
x={10,20,30,40,50,60}
x.discard(500)
x
```
Out[77]:
```
{10, 20, 30, 40, 50, 60}
```
In [ ]:
```
remove vs discard
if the element is not present in set then remove function will give you an error where as
discard will neverr give you an error
```

Mathematical Operations of Set Like Intersection,Union and Difference

In [ ]:
```
1.Intersection
2.Union
3.Difference
```

Intersection

In [79]:
```
#Intersection --> Return the common element from both the set.
s1={10,20,30,40}
s2={60,70}
print(s1.intersection(s2))
```
```
set()
```

Union

In [80]:
```
#Union --> Return all the elements of both the set.
s1={10,20,30,40}
s2={60,70,10,20}
print(s1.union(s2)) #+ operator
```
```
{70, 40, 10, 20, 60, 30}
```

Difference

In [ ]:
```
difference --> return the elements that are present in first set and but not present in second set
--> common element got removed and the first rest of the elements of first yyou will get
```
In [83]:
```
x={10,20,30,40,50,60}
y={90,80,70,60,50}
print(y.difference(x))
```
```
{80, 90, 70}
```

Membership Operator

In [84]:
```
x={10,20,30,40,50,60}
20 in x
```
Out[84]:
```
True
```
In [85]:
```
y={10,20,30}
500 in y
```
Out[85]:
```
False
```

Practice Problems

Python Program to find the Sum of number digits in List

x=[34,56,34,23,12] output: [7,11,7,5,3]

In [9]:
```
x=[34,56,34,23,12]
ans=[]
for i in x:
    sum=0
    for j in str(i):
        sum+=int(j)
    ans.append(sum)
print(ans)
```
```
[7, 11, 7, 5, 3]
```

Write a Python program which will return all the different unique vowels that are present in the given string.

Test Case: 1.String may be in upper case or lowr case 2.Answer must give unqiue vowels list 3.If any string is having n repeating characters then that character will be considered only once.

In [89]:
```
input_string = input("Enter the String :").lower()
set1=set(input_string)
print(set1)
vowels = {"a","e","i","o","u"}
set2=set1.intersection(vowels)
print(list(set2))
```
```
Enter the String :dipali
{'p', 'd', 'i', 'a', 'l'}
['a', 'i']
```

Ways to create Your Own Dictionaries

In [10]:
```
#Create Your own Dictionary Like This :     d={1:2,2:3,3:4,4:5,5:6}
d = {}
for i in range(1,6):
    d[i]=i+1
print(d)
```
```
{1: 2, 2: 3, 3: 4, 4: 5, 5: 6}
```
In [11]:
```
#Create Your own Dictionary Like This : d={2:4,3:6,4:8,5:10,6:12}
d={}
for i in range(2,7):
    d[i]=i+i
print(d)
```
```
{2: 4, 3: 6, 4: 8, 5: 10, 6: 12}
```
In [12]:
```
#Create Your own Dictionary Like This : d={2:4,4:8,6:12,8:16,10:20}
d={}
for i in range(2,11,2):
    d[i]=i*2
print(d)
```
```
{2: 4, 4: 8, 6: 12, 8: 16, 10: 20}
```
In [13]:
```
#Create Your own Dictionary Like This : d={1:1,2:8,3:27,4:64,5:125}
d={}
for i in range(1,6):
    d[i]=i**3
print(d)
```
```
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```
In [14]:
```
#Create Your own Dictionary Like This : d={5:15,7:21 ,9:27, 11:33 , 13:39}
d={}
for i in range(5,14,2):
    d[i]=i*3
print(d)
```
```
{5: 15, 7: 21, 9: 27, 11: 33, 13: 39}
```
In [15]:
```
#Create Your own Dictionary Like This : d={1:3,2:6 ,3:11, 4:18 , 5:27}
d={}
for i in range(1,6):
    d[i]=i**2+2
print(d)
```
```
{1: 3, 2: 6, 3: 11, 4: 18, 5: 27}
```
In [16]:
```
d={}
for i in range(ord("A"),ord("C")):
    d[i]=i+2
print(d)

#{65:67 , 66:68}
```
```
{65: 67, 66: 68}
```
In [98]:
```
#can we create for this 1:1 2:4 3:27 4:256 5:3125?
d={}
for i in range(1,6):
    d[i]=i**i
print(d)
```
```
{1: 1, 2: 4, 3: 27, 4: 256, 5: 3125}
```