18/1/2022

Dictionary Data type(dict()):-

Name   age   roll

sakti   22   45
vishal   23   56
shiba   24   57
gago   45   78

a = {Name:Sakti,age:22,roll:45}
b = {Name:Vishal,age:23,roll:56}

```
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = ('Name','age','roll_no')
>>> b = ('Sakti',22,45)
>>>
>>> c = dict(a,b)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    c = dict(a,b)
TypeError: dict expected at most 1 argument, got 2
>>>
>>> c = {'Name':'Sakti','age':22,'roll_no':45}
>>> type(c)
<class 'dict'>
>>> c.keys()
dict_keys(['Name', 'age', 'roll_no'])
>>> c.values()
dict_values(['Sakti', 22, 45])
>>>
>>> d = {['a','b']:1}
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    d = {['a','b']:1}
TypeError: unhashable type: 'list'
>>> d = {67:[67,90]}
>>> d[67]
[67, 90]
>>>
>>>
>>> a = []
>>> type(a)
<class 'list'>
>>> b = ()
>>> type(b)
```

```
<class 'tuple'>
>>> c = {}
>>> type(c)
<class 'dict'>
>>> d = set()
>>> type(d)
<class 'set'>
>>>
```

10/11/2021

Dictionary

If we want to represent a group of objects as key-value pairs then we should go for dictionary.

we can use list,tuple and set to represent a group of individual object as a single entity.

Imp ponits:-

1. Duplicate keys are not allowed but values can be duplicated.
2. Hetrogeneous object are allowed for both key and values.
3. Insertion order is nor preserved.
4. Dictionary are mutable.

Note:- In C++ and java Dictionary are known as 'Map' and in perl and Rubby known as 'hash'

```
>>> a = {'Name':'Krishna','Age':24,'Address':'BBSR'}
>>> type(a)
<class 'dict'>
```

```
 dir(a)
['clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem',
'setdefault', 'update', 'values']
>>>
```

1. clear():- To Remove all entries from the dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.clear()
>>> a
{}
```

Note:- del a

to delete total dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> del a
>>> print(d)
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
>>>
```

2. copy:- To create exactly duplicate dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> b = a.copy()
>>> b
{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR'}
>>> type(a)
<class 'dict'>
>>>
```

3. fromkeys():- The fromkeys() method returns a dictionary with specified keys and the specified values

syntax:- dict.fromekys(a,b)

```
a = ('Delhi','BLG','HYD','BBSR')
>>> b = ('Ragav')
>>> c = {}
>>> c.fromkeys(a,b)
{'Delhi': 'Ragav', 'BLG': 'Ragav', 'HYD': 'Ragav', 'BBSR': 'Ragav'}
>>>
```

4. get():- To get the value associated with the key.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.get('Name')
'Krishna'
>>>
```

5. itmes():- It returns list of tuples representing key-value pairs.

[(k,v),(k,v),(k,v)]

ex:-

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.items()
dict_items([('Name', 'Krishna'), ('Age', 24), ('City', 'BBSR')])
```

```
>>>
```

6. keys:- It returns all keys associated with dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.keys()
dict_keys(['Name', 'Age', 'City'])
>>>
```

7. values():- It returns all values associated with the dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.values()
dict_values(['Krishna', 24, 'BBSR'])
>>>
```

8. pop():- It removes the entry associated with the specified key and returns the corresponding values

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.pop('City')
'BBSR'
>>> a
{'Name': 'Krishna', 'Age': 24}
```

If tyhe specified key is not available then we will get KeyError

```
>>> a.pop('zip_code')
Traceback (most recent call last):
  File "<pyshell#59>", line 1, in <module>
    a.  pop('zip_code')
KeyError: 'zip_code'
>>>
```

9.popitem():- It removes an arbitaryitem(key-value) from the dictionary and returns it,

ex:-

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>> a.popitem()
('City', 'BBSR')
>>> a
{'Name': 'Krishna', 'Age': 24}
>>>
```

empty dictionary:-

```
>>> b ={}
>>> b.popitem()
Traceback (most recent call last):
  File "<pyshell#65>", line 1, in <module>
    b.  popitem()
```

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

10.setdefault():-

If the key is already available then this function returns the corresponding values.

If the key is not available then the specified key-value will be added as new item to the dictionary.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}

>>> a.setdefault('zip_code',1278)
1278
>>> a
{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip_code': 1278}
>>> a.setdefault('City','RKL')
'BBSR'
>>> a
{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip_code': 1278}
>>>
```

11.update:- All itmes present in the dictionary then we will add more key-value pair in it.

```
>>> a={'Name':'Krishna','Age':24,'City':'BBSR'}
>>>
a.update({'zip_code':1234,'post_office':'Khandagiri','PoliceStation':'CRPF
'})
>>> a
{'Name': 'Krishna', 'Age': 24, 'City': 'BBSR', 'zip_code': 1234,
'post_office': 'Khandagiri', 'PoliceStation': 'CRPF'}
>>>
```