In [3]:

```
#@author: Bhanu Prakash

a={}

a={1:'raja',2:'rani',3:'mantri'} #dictionary contains keys and values (key should be immutable and unique type)

print(type(a)) #dict

print(a) # uniform keys (int)

b={4:'sainik','praja':5,6:'pashu'}

print(b) # different keys (all)
```

```
<class 'dict'>
{1: 'raja', 2: 'rani', 3: 'mantri'}
{4: 'sainik', 'praja': 5, 6: 'pashu'}
```

In [6]:

```
a={}

a={1:'raja',2:'rani',3:'mantri'}

a.keys() #keys
```

Out[6]:

```
dict_keys([1, 2, 3])
```

In [1]:

```
a={}

a={1:'raja',2:'rani',3:'mantri'}

a.values() #values
```

Out[1]:

```
dict_values(['raja', 'rani', 'mantri'])
```

In [4]:

```
a={}

a={1:'raja',2:'rani',3:'mantri'}

a[3] #we can fetch value by using key
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Out[4]: 'mantri'
```

```
In [6]: a={}

        a={1:'raja',2:'rani',3:'mantri'}

        a[4] #we we don't have that particular key
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-6-ac74c3fb24d5> in <module>
      3 a={1:'raja',2:'rani',3:'mantri'}
      4
----> 5 a[4] #we we don't have that particular key

KeyError: 4
```

```
In [8]: a={}

        a={'r':'raja','rr':'rani','m':'mantri'}

        a['s']='saynadi'

        print(a) #adding new key
```

```
{'r': 'raja', 'rr': 'rani', 'm': 'mantri', 's': 'saynadi'}
```

```
In [9]: a={}

        a={'r':'raja','rr':'rani','m':'mantri'}

        print(a.get("r")) #get (assigned one) this is one type of fetching data

        print(a.get("a")) #get (not assigned one) it doesn't show error it shows none
```

```
raja
None
```

```
In [10]: a={}

         a={'r':'raja','rr':'rani','m':'mantri'}

         print(a.get("rr","not found")) #here we do have that particular key
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print(a.get("s","not found")) #if we don't have that particular key
```

```
rani
not found
```

In [12]:

```
a={}

a={'r':'raja','rr':'rani','m':'mantri'}

a.pop("m")

print(a) #pop
```

```
{'r': 'raja', 'rr': 'rani'}
```

In [15]:

```
a={}

a={'r':'raja','rr':'rani','m':'mantri'}

a.popitem() #it removes last one
```

Out[15]:

```
('m', 'mantri')
```

In [2]:

```
a={}

a={'r':'raja','rr':'rani','m':'mantri'}

a.items() #it gives all items present in dictionary
```

Out[2]:

```
dict_items([('r', 'raja'), ('rr', 'rani'), ('m', 'mantri')])
```

In [3]:

```
a={}

a={'r':'raja','rr':'rani','m':'mantri'}

a.copy() #it copies the dictionary
```

Out[3]:

```
{'r': 'raja', 'rr': 'rani', 'm': 'mantri'}
```

In [6]:

```
a={}
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```python
a={'r':'raja','rr':'rani','m':'mantri'}

a.clear()

print(a) #clear
```

```
{}
```

In [23]:

```python
a={}

print(dir(a))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__le
n__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__setattr__', '__set
item__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'po
pitem', 'setdefault', 'update', 'values']
```

In [ ]:

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD