Dict.

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection
  - which is ordered
  - changeable and
  - do not allow duplicates.

```
In [ ]:
```

Rules in Dict

- Dict. have no indexing
- Dict. is a mutable data type
- Dict :
  - keys - immutable
  - values - can be mutable
- Dict. keys should be unique

```
In [ ]:
```

Mutable and Immutable data types

- Mutable
  - List
  - Sets
  - Dict.
- Immutable
  - String
  - Tuples
  - Int
  - Float
  - Boolean
  - Complex

```
In [ ]:
```

Creating an empty dic

```
In [3]: d1 = {}
        d1
Out[3]: {}
```

```
In [4]: type(d1)
Out[4]: dict
```

```
In [ ]:
```

In [5]:
```
d2 = {'name':'harsh',
     'age':19,
     'college':'GIT'}
d2
```

Out[5]:
```
{'name': 'harsh', 'age': 19, 'college': 'GIT'}
```

In [ ]:

Proving dic. keys should be immutable

In [6]:
```
d2 = {[1,2,3]:'list'}
d2
```

```
-------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
Input In [6], in <module>
----> 1 d2 = {[1,2,3]:'list'}
      2 d2

TypeError: unhashable type: 'list'
```

In [8]:
```
'''
doing same with tuple :
its success beacuse tuple is immutable
and key in dic are immutable
'''

d2 = {(1,2,3):'list'}
d2
```

Out[8]:
```
{(1, 2, 3): 'list'}
```

In [ ]:

Proving duplicate keys is not allowed in dic.

- Whenever we use the repeating key
- the value of the key is updated with the latest value associated witht he same key

In [11]:
```
d3 = {1:'harsh',
         2:'rishi',
         1:'rin'}
```

In [12]:
```
d3
```

Out[12]:
```
{1: 'rin', 2: 'rishi'}
```

In [ ]:

In [13]:
```
d4 = {'name':'harsh',
     'name':'rin',
     'age':21}
```

```
In [14]: d4
```

```
Out[14]: {'name': 'rin', 'age': 21}
```

```
In [ ]:
```

We can create 2D 3D 4D dic

```
In [15]: d1 = {'name':'Harsh',
              'branch':'CE',
              'sem':8,
              'marks':{'sub1':90,'sub2':80,'sub3':90}}
```

```
In [16]: d1
```

```
Out[16]: {'name': 'Harsh',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 90, 'sub2': 80, 'sub3': 90}}
```

```
In [ ]:
```

We can access the items from the dic

- as we know there is no concept of indexing and slicing in dic
- therefore we need to pass the name of the key to access the vlaue from the dic

```
In [17]: d1 = {'name':'Harsh',
              'branch':'CE',
              'sem':8,
              'marks':{'sub1':90,'sub2':80,'sub3':90}}
```

```
In [18]: d1['name']
```

```
Out[18]: 'Harsh'
```

```
In [19]: d1['marks']
```

```
Out[19]: {'sub1': 90, 'sub2': 80, 'sub3': 90}
```

```
In [ ]:
```

```
In [20]: #Accessing data from the 2D dic
         d1['marks']['sub1']
```

```
Out[20]: 90
```

```
In [21]: d1['marks']['sub2']
```

```
Out[21]: 80
```

```
In [ ]:
```

Edit

- In dic we can edit the dic in the same way we are able to access it

```
In [22]: d1 = {'name':'Harsh',
              'branch':'CE',
              'sem':8,
              'marks':{'sub1':90,'sub2':80,'sub3':90}}
```

```
In [23]: d1['name']='rin'
```

```
In [24]: d1
```

```
Out[24]: {'name': 'rin',
           'branch': 'CE',
           'sem': 8,
           'marks': {'sub1': 90, 'sub2': 80, 'sub3': 90}}
```

```
In [ ]:
```

```
In [25]: # editing data inside the 2D dic
         d1['marks']['sub1'] = 100
```

```
In [26]: d1
```

```
Out[26]: {'name': 'rin',
           'branch': 'CE',
           'sem': 8,
           'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90}}
```

```
In [ ]:
```

We can access the data from the dic in 2 ways

- using get() : but is only applicable to 1D array

```
In [28]: d1 = {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90}}
```

```
In [29]: d1
```

```
Out[29]: {'name': 'rin',
           'branch': 'CE',
           'sem': 8,
           'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90}}
```

```
In [30]: d1['name']
```

```
Out[30]: 'rin'
```

```
In [31]: d1.get('name')
```

```
Out[31]: 'rin'
```

```
In [ ]:
```

Adding new key-value pairs into the dic

```
In [32]: d1 = {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90}}

In [33]: d1['college']='GIT'

In [34]: d1

Out[34]: {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90},
          'college': 'GIT'}

In [ ]:

In [35]: # adding new data into the 2D dic

         d1['marks']['newsub']=999

In [36]: d1

Out[36]: {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90, 'newsub': 999},
          'college': 'GIT'}

In [ ]:
```

Deleting key value pair from the dic

```
In [37]: d1

Out[37]: {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90, 'newsub': 999},
          'college': 'GIT'}

In [38]: # deleting from 2D dic
         del d1['marks']['newsub']

In [39]: d1

Out[39]: {'name': 'rin',
          'branch': 'CE',
          'sem': 8,
          'marks': {'sub1': 100, 'sub2': 80, 'sub3': 90},
          'college': 'GIT'}
```

```
In [ ]:

In [40]: del d1['marks']

In [41]: d1

Out[41]: {'name': 'rin', 'branch': 'CE', 'sem': 8, 'college': 'GIT'}

In [ ]:

In [42]: del d1

In [43]: d1

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [43], in <module>
----> 1 d1

NameError: name 'd1' is not defined

In [ ]:
```

Using clear(), to empty the dic.

```
In [44]: d1 = {'name': 'rin',
          'branch': 'CE',
          'sem': 8,}

In [45]: d1

Out[45]: {'name': 'rin', 'branch': 'CE', 'sem': 8}

In [46]: d1.clear()

In [47]: d1

Out[47]: {}

In [ ]:
```

Operations

- Concatenation (+) and mul (*) does not work with dic
- But we can iterate or perform looping opern on it
- also membership is supported here : but checking always get performed on keys not values

```
In [48]: d1 = {'name': 'rin', 'branch': 'CE', 'sem': 8}
In [49]: d1
Out[49]: {'name': 'rin', 'branch': 'CE', 'sem': 8}
```

```
In [50]: for i in d1:
             print(i)

name
branch
sem
```

```
In [ ]:
```

```
In [51]: for i in d1.keys():
             print(i)

name
branch
sem
```

```
In [52]: for i in d1.values():
             print(i)

rin
CE
8
```

```
In [ ]:
```

```
In [56]: for i in d1:
             print(i,d1[i])

name rin
branch CE
sem 8
```

```
In [ ]:
```

```
In [57]: for i in d1.items():
             print(i)

('name', 'rin')
('branch', 'CE')
('sem', 8)
```

```
In [ ]:
```

```
In [59]: for i,j in d1.items():
             print(i,j)

name rin
branch CE
sem 8
```

```
In [ ]:
```

Using memebrship operator on dict

```
In [60]: d1 = {'name': 'rin', 'branch': 'CE', 'sem': 8}
```

```
In [61]: 'rin' in d1
Out[61]: False
```

```
In [62]: 'name' in d1
Out[62]: True
```

```
In [ ]:
```

Functions : supported in dic

- min/max
- len
- sorted
- If our keys are int : sum() can also be used

```
In [63]: d1 = {'name': 'rin', 'branch': 'CE', 'sem': 8}
```

```
In [66]: # on the basis of ascii value :
         # lexiographically

         min(d1)
Out[66]: 'branch'
```

```
In [67]: max(d1)
Out[67]: 'sem'
```

```
In [68]: sorted(d1)
Out[68]: ['branch', 'name', 'sem']
```

```
In [69]: sorted(d1,reverse=True)
Out[69]: ['sem', 'name', 'branch']
```

```
In [ ]:
```

```
In [76]: d2 = {4:'a',2:'b',3:'c'}
```

```
In [77]: min(d2)
Out[77]: 2
```

```
In [78]: max(d2)
Out[78]: 4
```

```
In [79]: sorted(d2)
Out[79]: [2, 3, 4]
```

```
In [82]: sorted(d2,reverse=True)

Out[82]: [4, 3, 2]
```

```
In [80]: sum(d2)

Out[80]: 9
```

```
In [ ]:
```

Fucntions specific to dic

- keys
- values

```
In [83]: d2

Out[83]: {4: 'a', 2: 'b', 3: 'c'}
```

```
In [84]: d2.keys()

Out[84]: dict_keys([4, 2, 3])
```

```
In [85]: d2.values()

Out[85]: dict_values(['a', 'b', 'c'])
```

```
In [ ]:
```