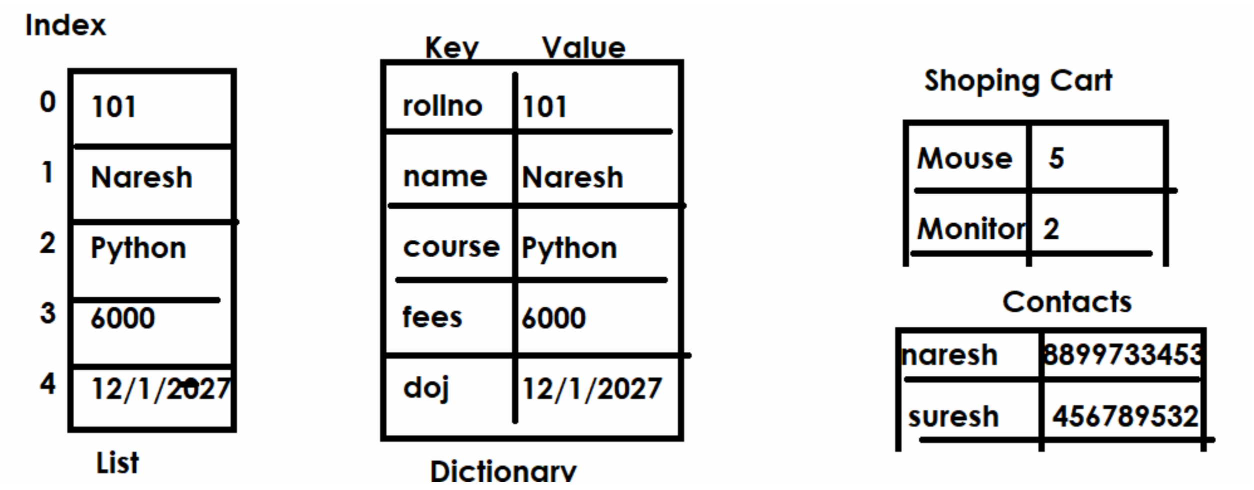
Dictionary (Mapping)

"dict" is class or data type which represents dictionary object
Dictionary is key based collection.
Dictionary is collection of items, where each item consist of 2 values

1. Key
2. Value

Each key in dictionary is mapped with one or more than one value.
Each value in dictionary is identified with key.
In application development dictionary is used to organize data as key, value pair.



How to create dictionary?
Dictionary can be created in different ways

1. Empty dictionary is created using empty curly braces {}

```
A={}
print(A,type(A))
{} <class 'dict'>
```

2. Dictionary with items are created using {}, each item within curly braces is separated with, and key and values are separated with :

Points:

1. In dictionary keys are immutable types
2. Dictionary does not allows duplicate keys
3. Dictionary allows duplicate values
4. Dictionary values can be any type (mutable/immutable)

Dictionary is a mutable collection and after creating dictionary changes can be done.

```
>>> stud1={'rollno':1,
      'name':'naresh',
      'course':'python',
      'fees':3000}
>>> print(stud1)
{'rollno': 1, 'name': 'naresh', 'course': 'python', 'fees': 3000}
sales={2010:45000,
      2011:65000,
      2012:78000,
      2013:76000,
      2014:85000}
>>> print(sales)
{2010: 45000, 2011: 65000, 2012: 78000, 2013: 76000, 2014: 85000}
d1={1:10,1:20,1:30}
print(d1)
{1: 30}
>>> d2={1:10,2:10,3:10}
>>> print(d2)
{1: 10, 2: 10, 3: 10}
>>> d3={(1,2):10,(3,4):20}
>>> print(d3)
{(1, 2): 10, (3, 4): 20}
>>> d4={[1,2]:10}
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    d4={[1,2]:10}
TypeError: unhashable type: 'list'
>>> mi={'rohit':[10,30,60,70,80],
...     'surya':[60,50,90,60,70]}
>>> print(mi)
{'rohit': [10, 30, 60, 70, 80], 'surya': [60, 50, 90, 60, 70]}
```

3. **dict()** type or function is used for creating empty dictionary

```
>>> d1=dict()
>>> print(d1)
```

```
{}
```

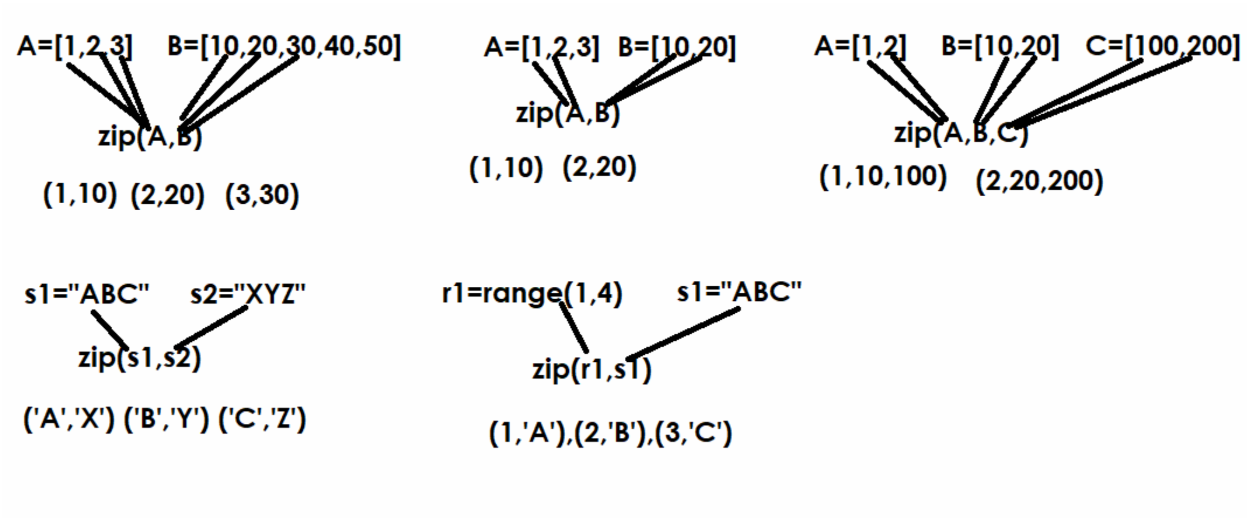
4. **dict(iteable)** type or function is used to convert other collections or iterables into dictionary type

Note: in order to convert existing iterable or collection into dictionary type, the iterable or collection must generate two values.

```
>>> A=[10,20,30,40,50]
>>> d1=dict(A)
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    d1=dict(A)
TypeError: cannot convert dictionary update sequence element #0 to a
sequence
>>> A=[(1,10),
   (2,20),(3,30),(4,40),(5,50)]
>>> d1=dict(A)
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> A=[10,20,30,40,50]
>>> e=enumerate(A,start=1)
>>> d2=dict(e)
>>> print(d2)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> d3=dict(range(1,6))
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    d3=dict(range(1,6))
TypeError: cannot convert dictionary update sequence element #0 to a
sequence
>>> A=[1,2,3,4,5]
>>> B=[10,20,30,40,50]
>>> C=[(A[i],B[i]) for i in range(5)]
>>> print(A)
[1, 2, 3, 4, 5]
>>> print(B)
[10, 20, 30, 40, 50]
>>> print(C)
[(1, 10), (2, 20), (3, 30), (4, 40), (5, 50)]
```

```
>>> d2=dict(C)
>>> print(d2)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
```

zip(*iterables)
Iterate over several iterables in parallel, producing tuples with an item from each one.



```
>>> dict1=dict(zip(range(1,6),range(10,60,10)))
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> dict2=dict(zip("ABC","XYZ"))
>>> print(dict2)
{'A': 'X', 'B': 'Y', 'C': 'Z'}
>>> dict3=dict(zip("ABCDE",[10,20,30,40,50]))
>>> print(dict3)
{'A': 10, 'B': 20, 'C': 30, 'D': 40, 'E': 50}
```

How to read content of dictionary?
Dictionary content can be read in different ways

1. using key
2. using for loop
3. using dictionary methods
   a. get()
   b. keys()
   c. values()
   d. items()

e. setdefault()

Using key
Dictionary is key based collection and we can read the value of dictionary using key

Syntax:
Dictionary-name[key]

If key exists, it returns value
If key not exists, it raises KeyError

```
>>> persons={'naresh':60,'suresh':45,'ramesh':50}
>>> age1=persons['suresh']
>>> print(age1)
45
>>> age2=persons['ramesh']
>>> print(age2)
50
>>> age3=persons['kishore']
Traceback (most recent call last):
  File "<pyshell#55>", line 1, in <module>
    age3=persons['kishore']
KeyError: 'kishore'
>>> 'kishore' in persons
False
>>> 'naresh' in persons
True
```

Example:
```
# Login Application

users={'naresh':'n123',
       'suresh':'s321',
       'kishore':'k567',
       'ramesh':'r567'}

print("***Login****")
uname=input("UserName :")
pwd=input("Password :")
```

```
if uname in users and users[uname]==pwd:
    print(f'{uname} welcome')
else:
    print("invalid username or password")
```

Output

```
***Login****
UserName :suresh
Password :s123
invalid username or password

***Login****
UserName :ramesh
Password :r567
ramesh welcome
```

Example

```
# Result Processing

marks={'naresh':[40,50,60],
       'suresh':[70,80,90],
       'ramesh':[30,60,70]}

name=input("Name :")
if name in marks:
    A=marks[name]
    total=sum(A)
    avg=total/3
    result="PASS"
    for m in A:
        if m<40:
            result="FAIL"
            break
    print(f'{name}\t{A}\t{total}\t{avg:.2f}\t{result}')

else:
    print("Invalid name")
```

Output

```
Name :ramesh
```

```
ramesh      [30, 60, 70] 160   53.33 FAIL

Name :kiran
Invalid name
```

Example:

```
>>> d1={1:[10,20,30],
    2:[40,50,60],
    3:[70,80,90]}
>>> d1[1]
[10, 20, 30]
>>> d1[2]
[40, 50, 60]
>>> d1[3]
[70, 80, 90]
>>> d1[1][0]
10
>>> d1[3][-1]
90
>>> d2={'a':"python",
...     'b':'java'}
>>> d2['a']
'python'
>>> d2['b']
'java'
>>> d2['a'][0]
'p'
>>> d2['a'][-1]
'n'
>>> d2['b'][:2]
'ja'
>>> d2['b'][::-1]
'avaj'
```

Using for loop

If dictionary is given to for loop, for loop iterate or read keys

Syntax:

```
for variable-name in dictionary-name:
        statement-1
        statement-2
```

Example:

```
A={1:10,2:20,3:30,4:40,5:50}

for x in A:
    print(x,A[x])
```

Output

```
1 10
2 20
3 30
4 40
5 50
```