How to create frozenset?
frozenset() → Creates empty frozenset
frozenset(iterable) → Create frozenset by converting existing iterables into set

```
>>> A=frozenset()
>>> print(A,type(A))
frozenset() <class 'frozenset'>
>>> A.add(10)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    A.add(10)
AttributeError: 'frozenset' object has no attribute 'add'

>>> B=frozenset(range(10,60,10))
>>> print(B)
frozenset({40, 10, 50, 20, 30})
>>> B.remove(10)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    B.remove(10)
```

Example:

```
A=frozenset(range(10,60,10))
for value in A:
    print(value,end=' ')

print()
B={frozenset(range(1,6)),frozenset(range(10,60,10))}
print(B)
for x in B:
    print(x)
    for y in x:
        print(y,end=' ')
    print()
```

Output

40 10 50 20 30
{frozenset({1, 2, 3, 4, 5}), frozenset({40, 10, 50, 20, 30})}
frozenset({1, 2, 3, 4, 5})
1 2 3 4 5
frozenset({40, 10, 50, 20, 30})
40 10 50 20 30

What is difference between set and frozenset?

Set	**Frozenset**
Set is mutable collection	Frozenset is an immutable collection
Set is created using curly braces	Frozenset set is created using frozenset()
Set can be represented as an element inside set	Frozenset can be represented as an element with set

What is difference between list and set?

List	**Set**
List is mutable sequence data type or list is an ordered collection (where insertion order is same)	Set is mutable unordered collection (where insertion order not same)
List allows any type of objects	Set allows only immutable objects
List support indexing and slicing	Set does not support indexing and slicing (non index based)
List allows duplicate values	Set does not allows duplicate values
In list data organized in sequential order	In set data is organized using hashing data structure
List is created using []	Set is created using {}
In application development list is used to group individual objects where insertion order is preserved allows duplicates	In application development set is used to group individual objects where duplicates are not allowed and perform mathematical set operations
List class is used for creating list	Set class is used for creating set

object	object

Dictionary or dict data type (Mapping)

Dictionary is mapping type collection
Python supports only one mapping type called dict (dict class)
In dictionary data is organized is as key and value pair (OR)
dictionary is collection items, where each item having 2 values.

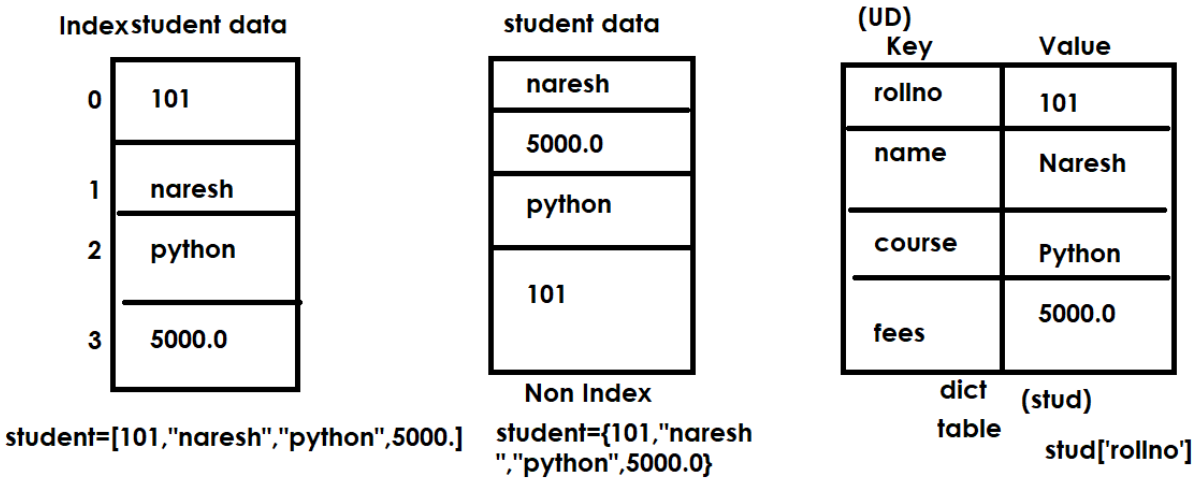
1. Key
2. Value

Every value in dictionary is mapped with key
Dictionary is key based collection, where reading and writing is done using key.

Example: PhoneBook, Shopping Cart,..

Points to remember

1. Dictionary does not allows duplicate keys but allows duplicate values
2. Dictionary keys are immutable and values can be any type
3. A dictionary key is mapped with one or more than one value
4. Dictionary is mutable collection, after creating dictionary changes can be done



How to create dictionary?

1. Empty dictionary is created using empty curly brace
   Syntax: dictionary-name={}

```
>>> d1={}
>>> print(d1,type(d1))
{} <class 'dict'>
```

2. Dictionary with items are created by representing items within curly brace. Each item having key and value, which are separated using **:** and each item is separated using **,**

Syntax: {key:value,key:value,key:value,…}

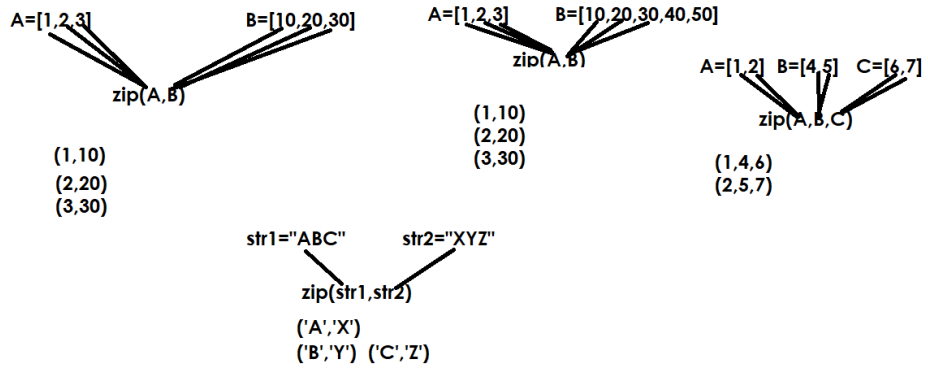
```
>>> scores={'virat':40,
        'rohit':100,
        'surya':120}
>>> sales={2000:45000,
       2001:54000,
       2002:65000,
       2003:76000,
...        2004:86000}
>>> print(scores)
{'virat': 40, 'rohit': 100, 'surya': 120}
>>> print(sales)
{2000: 45000, 2001: 54000, 2002: 65000, 2003: 76000, 2004: 86000}
>>> A={1:10,1:20,1:30,1:40}
>>> print(A)
{1: 40}
>>> B={1:10,2:10,3:10,4:10}
>>> print(B)
{1: 10, 2: 10, 3: 10, 4: 10}
>>> scores={'virat':[10,60,20,50,70],
...         'rohit':(10,20,80,90)}
>>> print(scores)
{'virat': [10, 60, 20, 50, 70], 'rohit': (10, 20, 80, 90)}
```

3. Creating dictionary using dict() type or function

dict() → This creates empty dictionary
dict(iterable) → This converts existing iteable/collection into dictionary. The iterable or collection must generate 2 values.

```
>>> dict2=dict([10,20,30,40,50])
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    dict2=dict([10,20,30,40,50])
TypeError: cannot convert dictionary update sequence element #0 to a sequence
>>> dict2=dict([(1,10),(2,20),(3,30),(4,40),(5,50)])
>>> print(dict2)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> dict3=dict(range(1,6))
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    dict3=dict(range(1,6))
TypeError: cannot convert dictionary update sequence element #0 to a sequence
```

zip()
zip() is a predefined function in python.
zip(**iterables*)
Iterate over several iterables in parallel, producing tuples with an item from each one.
More formally: zip() returns an iterator of tuples, where the *i*-th tuple contains the *i*-th element from each of the argument iterables.




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

How read content of dictionary?

1. Using key
2. Using for loop
3. Using dictionary methods
    a. get()
    b. keys()
    c. values()
    d. items()
    e. setdefault()
    f. reversed()

using key

dictionary value can be read using key

Syntax: dictionary-name[key]

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

Example:

```
employee={'naresh':56000,
          'suresh':78000,
          'ramesh':54000}

sal1=employee['naresh']
sal2=employee['ramesh']
print(sal1,sal2)
if 'kishore' in employee:
    sal3=employee['kishore']
    print(sal3)
else:
    print("key not found or invalid employeename")
```

Output

```
56000 54000
key not found or invalid employeename
```

Example:

```
# Login Application
users={'naresh':'n123',
       'ramesh':'r456',
       'kishore':'k478'}

while True:
    uname=input("UserName :")
    pwd=input("Password :")
    if uname in users and pwd==users[uname]:
        print(f'{uname} welcome,')
        break
    else:
        print("invalid uname or password")
```

Output

UserName :kishore
Password :k345
invalid uname or password
UserName :kishore
Password :k478
kishore welcome,

Using for loop
Dictionary can be given to for loop
for loop iterate/read keys from dictionary

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

Example:
```
sales={2010:45000,
       2011:65000,
       2012:75000,
       2013:85000}
tot=0
for year in sales:
   print(year,sales[year])
   tot=tot+sales[year]

print(f'Total sales {tot}')
```

Output
```
2010 45000
2011 65000
2012 75000
2013 85000
Total sales 270000
```

get() method
This method returns value of given key
If key exists it returns value
If key not exists it return default value

Syntax:
Dictionary-name.get(key,[value])

If default value is not given, it defaults to None

```
d1={1:10,2:20,3:30,4:40,5:50}
x=d1.get(1)
print(x)
10
>>> y=d1.get(2)
>>> print(y)
20
>>> z=d1.get(5)
>>> print(z)
50
>>> p=d1.get(9)
>>> print(p)
None
>>> q=d1.get(9,90)
>>> print(q)
90
>>> q=d1[9]
Traceback (most recent call last):
  File "<pyshell#50>", line 1, in <module>
    q=d1[9]
KeyError: 9
```

Dictionary is having 3 view objects

1. key view object
2. values view object
3. items view object