List - Introduction

```
Mylist = ['john','smith','david','mark','eric','smith']
```

```
list1 = [1,2,3,4,5]
```

```
# to print david
print(Mylist[2])
```

```
david
```

```
#to print eric - positive and negative indexing
print(Mylist[-2])
```

```
eric
```

```
# use list()
list2 = list((1,2,3,4,5))
```

```
list2
```

```
[1, 2, 3, 4, 5]
```

```
Mylist = ['john', 15, 14.6, True, 'steven', 5+7j]   #list is
heterogeneous
```

```
# List is mutable
Mylist = [15, 9, 12, 18, 7, 10]
```

```
Mylist[0] = 30
```

```
Mylist[4] = 'John'
```

```
Mylist
```

```
[30, 9, 12, 18, 'John', 10]
```

```
len(Mylist) #no. of elements in the list
```

```
6
```

```
Mylist.append(50) # Modified even by adding an object
```

```
Mylist
```

```
[30, 9, 12, 18, 'John', 10, 50]
```

```
len(Mylist)
```

```
7
```

List Operations

```
list1 = [1,2,3,4,5,6,7,8,9,10]
```

Index Operator

```
print(list1[6])

7

print(list1[-4])

7

list1[6] = 15

list1

[1, 2, 3, 4, 5, 6, 15, 8, 9, 10]

x = list1[6]

x

15
```

List Slicing

```
list1 = [1,2,3,4,5,6,7,8,9,10]

list1[:]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

list1[:5]

[1, 2, 3, 4, 5]

list1[3:]

[4, 5, 6, 7, 8, 9, 10]

list1[3:8]

[4, 5, 6, 7, 8]

list1[0:10:2]

[1, 3, 5, 7, 9]

list1[-1:-11:-1]

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

List Concatenation

```
lis1 = [1,2,3]

lis2 = [8,9,10]

lis1 + list2

[1, 2, 3, 1, 2, 3, 4, 5]

lis3 = lis1 + lis2

lis3

[1, 2, 3, 8, 9, 10]

lis1 + 4

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[73], line 1
----> 1 lis1 + 4

TypeError: can only concatenate list (not "int") to list

lis3 = lis1 + [4]

lis3

[1, 2, 3, 4, 5, 6, 4]

lis1.extend([4,5,6])

lis1

[1, 2, 3, 4, 5, 6, 4, 5, 6]

lis2

[8, 9, 10]

lis2 = lis2 + [11,12,13]

lis2

[8, 9, 10, 11, 12, 13]
```

List Repitition

```
lis1 = [1,2,3]

lis1 * 2
```

```
[1, 2, 3, 1, 2, 3]
```

```
lis1
```

```
[1, 2, 3]
```

```
lis1 * 3
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
list * 2.5
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[101], line 1
----> 1 list * 2.5

TypeError: unsupported operand type(s) for *: 'type' and 'float'
```

in and not in operator

```
list1 = [1,2,3]
```

```
list1
```

```
[1, 2, 3]
```

```
1 in list1
```

```
True
```

```
if 3 in list1:
    print('Found')
else:
    print('Not Found')
```

```
Found
```

```
5 in list1
```

```
False
```

```
5 not in list1
```

```
True
```

Iterating a List

```
list1 = [5,6,7,8,9]
```

```
for i in list1:
    print(i)

5
6
7
8
9

for i in range(len(list1)):
    print(list1[i])

5
6
7
8
9

for i in range(len(list1)-1,-1,-1):
    print(list1[i])

9
8
7
6
5

for i in range(-1,-len(list1)-1,-1):
    print(list1[i])

9
8
7
6
5

i = 0
while i<len(list1):
    print(list1[i])
    i = i+1

5
6
7
8
9

i = len(list1)-1
while i>=0:
    print(list1[i])
    i = i-1
```

```
9
8
7
6
5
```

```
l1 = [5,6,7,8,9]
```

append(x)

```
l1.append(10)

len(l1)

6

l1

[5, 6, 7, 8, 9, 10]

l1.append(11,12,13)

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[10], line 1
----> 1 l1.append(11,12,13)

TypeError: list.append() takes exactly one argument (3 given)

l1

[5, 6, 7, 8, 9, 10]
```

extend(iterable)

```
l1 = [5,6,7,8,9]

l1.extend([10,11,12])

l1

[5, 6, 7, 8, 9, 10, 11, 12]

l1.extend('abc')

l1

[5, 6, 7, 8, 9, 10, 11, 12, 'a', 'b', 'c']
```

insert(i,x)

```
l1 = [5,6,7,8,9]

l1.insert(0,10)

l1
```

```
[10, 5, 6, 7, 8, 9]

l1.insert(10,0)

l1

[10, 5, 6, 7, 8, 9, 0]

l1.insert(3,20)

l1

[10, 5, 6, 20, 7, 8, 9, 0]
```

copy()

```
l1 = [5,6,7,8,9]

l2 = l1.copy()

l2

[5, 6, 7, 8, 9]

l1

[5, 6, 7, 8, 9]
```

pop()

```
l1 = [5,6,7,8,9]

l1.pop(10)

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[52], line 1
----> 1 l1.pop(10)

IndexError: pop index out of range

l1.pop()   #by default delete the last element

9

l1

[5, 6, 7, 8]

l1.pop(0) #optional argument to set index value to be poped
```

```
5

l1

[6, 7, 8]

l1.pop(1)

7

l1

[6, 8]
```

remove()

```
l1 = [5,6,7,8,9]

l1.remove(7)

l1

[5, 6, 8, 9]

l1 = [5,6,7,8,9,5,6,7,8,9]

l1.remove(6)

l1

[5, 7, 8, 9, 5, 6, 7, 8, 9]

l1 = [5,6,7,8,9]

l1.remove(10)

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[81], line 1
----> 1 l1.remove(10)

ValueError: list.remove(x): x not in list
```

clear()

```
l1 = [5,6,7,8,9]

l1

[5, 6, 7, 8, 9]
```

```
l1.clear()

l1

[]
```

index(x)

```
l1 = [5,6,7,5,8,9,6,10,6]

l1.index(8)

4

l1.index(6)

1

l1.index(6,2)

6

l1.index(6,7)

8

l1

[5, 6, 7, 5, 8, 9, 6, 10, 6]

l1.index(6,2,5)

----------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[105], line 1
----> 1 l1.index(6,2,5)

ValueError: 6 is not in list

l1.count(6)

3

l1.reverse()  #modify the original -> mutable

l1

[6, 10, 6, 9, 8, 5, 7, 6, 5]

l1.sort() #modify the original list -> mutable

l1
```

```
[5, 5, 6, 6, 6, 7, 8, 9, 10]

l1.sort(reverse=True)

l1

[10, 9, 8, 7, 6, 6, 6, 5, 5]

sorted(l1) #global function

[5, 5, 6, 6, 6, 7, 8, 9, 10]

l1

[10, 9, 8, 7, 6, 6, 6, 5, 5]

#Sort() modifies the original list but sorted() does not.
```

List Comprehension

```
# l1 = [expression for i in iterable]

li = []
for i in range(10):
    li.append(i)
print(li)

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

l1 = [ i for i in range(10)]

l1

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

l2 = [i**2 for i in range(1,6)]

l2

[1, 4, 9, 16, 25]

l3 = [i for i in (10,5,7,8,12,3) if i%2==0]

l3

[10, 8, 12]

l4 = [i.lower() for i in 'PyThoN']

l4

['p', 'y', 't', 'h', 'o', 'n']

s = 'ab12de-&fg4$hi2'

l5 = [i for i in s if i.isalpha()]

l5

['a', 'b', 'd', 'e', 'f', 'g', 'h', 'i']

# l6 = 'ajay', 'john','smith'

data = input('Enter names : ')
l6 = data.split()
print(l6)

Enter names :  ajay john smith

['ajay', 'john', 'smith']
```

```python
l6 = input('Enter names :').split()
print(l6)
```

```
Enter names : ajay john smith

['ajay', 'john', 'smith']
```

Tuple

```
#immutable -> values can not be changed, added or removed from the
tuple

# Similar to List - > Ordered - heterogeneous collection of elements

tuple1 = ('John',45, 89.3,False,5+6j,'Smith',45)

for i in range(len(tuple1)):
    print(i,tuple1[i])

0 John
1 45
2 89.3
3 False
4 (5+6j)
5 Smith
6 45

tuple1[-1]

45

tuple1[0] = 'Smith'  # immutable

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[12], line 1
----> 1 tuple1[0] = 'Smith'

TypeError: 'tuple' object does not support item assignment
```

Tuple Creation

```
t1 = (1,2,3,4,5)

t2=()

t2

()

t3 = (10)

print(type(t3))

<class 'int'>

t3 = (10,)
```

```
t3

(10,)

print(type(t3))

<class 'tuple'>

t4 = tuple((1,2,3,4,5))

t5 = tuple([1,2,3,4,5])

t5

(1, 2, 3, 4, 5)

type(t5)

tuple

t6 = tuple('Python')

t6

('P', 'y', 't', 'h', 'o', 'n')

t7 = tuple((1,2,33,4))

t7

(1, 2, 33, 4)
```

Tuple Packing & Unpacking

```
t1 = 10

t1 = 10,20,30,40,50  # Tuple Packing

t1

(10, 20, 30, 40, 50)

type(t1)

tuple

a,b,c,d,e = t1  # Tuple Unpacking

a

10

b
```

```
20

c

30

d

40

e

50

l1 = [1,2,3] #unpacking list

a,b,c = l1

a

1

b

2

c

3

a,b,c = "pyt"  # unpacking string

a

'p'

b

'y'

c

't'

# Packing is done in tuple only but unpacking can be done for list,
tuple and string too.

t1 = 10,20,30,40,50

a,b,c = t1

---------------------------------------------------------------------
-----
ValueError                               Traceback (most recent call
last)
```

```
Cell In[88], line 1
----> 1 a,b,c = t1

ValueError: too many values to unpack (expected 3)
```

```
a,b,*c = t1
```

```
a
```

```
10
```

```
b
```

```
20
```

```
c # returns a list for the remaining elements
```

```
[30, 40, 50]
```

```
a,*b,c = t1
```

```
a
```

```
10
```

```
c
```

```
50
```

```
b
```

```
[20, 30, 40]
```

```
a,*b,*c = t1
```

```
  Cell In[106], line 1
    a,*b,*c = t1
    ^
SyntaxError: multiple starred expressions in assignment
```

Tuple Comprehension

```
l1 = [i for i in range(5)]

l1

[0, 1, 2, 3, 4]

t1 = (i for i in range(5))

t1

<generator object <genexpr> at 0x000001DD5C009A80>

t1 = tuple(i for i in range(5))

t1

(0, 1, 2, 3, 4)

t2 =(*(i for i in range(5)),)

t2

(0, 1, 2, 3, 4)

t3 = (*(i for i in range(1,11,2)),)

t3

(1, 3, 5, 7, 9)

t4 = (*(i for i in 'Python'),)

t4

('P', 'y', 't', 'h', 'o', 'n')

t5 = (*(i for i in 'PyThoN' if i.islower()),)

t5

('y', 'h', 'o')

t6 = tuple(i for i in 'PyThoN' if i.islower())

t6

('y', 'h', 'o')

t7 = tuple(i**2 for i in (1,2,3,4,5,6))

t7
```

```
(1, 4, 9, 16, 25, 36)
```

Tuple Methods

```
t8 =  (1,2,3,4,5,4,4,5,6,7)

t8.count(4)

3

t8.count(5)

2

t8.index(3)

2

t8.index(4,4)

5

t8.index(10)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[49], line 1
----> 1 t8.index(10)

ValueError: tuple.index(x): x not in tuple
```

Tuple Operators

```
tuple1[1]

45

tuple1[0]

'John'

tuple1[4]

(5+6j)

tuple1[4] = 78

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
Cell In[65], line 1
----> 1 tuple1[4] = 78

TypeError: 'tuple' object does not support item assignment
```

```
tuple1[-1]
```

```
45
```

```
tuple1[-2]
```

```
'Smith'
```

```
tuple1[::]
```

```
('John', 45, 89.3, False, (5+6j), 'Smith', 45)
```

```
tuple1[::-1]
```

```
(45, 'Smith', (5+6j), False, 89.3, 45, 'John')
```

```
tuple1[3::]
```

```
(False, (5+6j), 'Smith', 45)
```

```
tuple1[-3:-len(tuple1)-1:-1]
```

```
((5+6j), False, 89.3, 45, 'John')
```

```
tuple1[::2]
```

```
('John', 89.3, (5+6j), 45)
```

```
t1 = (1,2,3)
```

```
t2 = (4,5,6)
```

```
t1 + t2  # Concatenation
```

```
(1, 2, 3, 4, 5, 6)
```

```
t1
```

```
(1, 2, 3)
```

```
t2
```

```
(4, 5, 6)
```

```
t1 + [4,5,6]
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
```

```
Cell In[91], line 1
----> 1 t1 + [4,5,6]

TypeError: can only concatenate tuple (not "list") to tuple
```

```python
t1 + tuple([4,5,6])
```

```
(1, 2, 3, 4, 5, 6)
```

```python
t1 * 2
```

```
(1, 2, 3, 1, 2, 3)
```

```python
t1 * 3
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

```python
t1 * 2.5
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[99], line 1
----> 1 t1 * 2.5

TypeError: can't multiply sequence by non-int of type 'float'
```

```python
3 in t1
```

```
True
```

```python
5 in t1
```

```
False
```

```python
5 not in t1
```

```
True
```

```python
3 not in t1
```

```
False
```

Set

```
s1 = {'hello', 45, 89.23, 'hello', 45, True}
```

```
s1  # unordered, # unique values # No Duplicates
```

```
{45, 89.23, True, 'hello'}
```

```
for i in s1: # iterable
    print(i)
```

```
hello
89.23
45
True
```

```
s1[0]
```

```
------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[8], line 1
----> 1 s1[0]

TypeError: 'set' object is not subscriptable
```

```
s1.add(78)
```

```
s1
```

```
{45, 78, 89.23, True, 'hello'}
```

```
print(type(s1))
```

```
<class 'set'>
```

```
s2 = set(1,2,3,4,5)
```

```
------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[16], line 1
----> 1 s2 = set(1,2,3,4,5)

TypeError: set expected at most 1 argument, got 5
```

```
s2 = set((1,2,3,4,5))
```

```
s2
```

```
{1, 2, 3, 4, 5}
```

```
s3 = set('Python')

s3  #as it is unordered

{'P', 'h', 'n', 'o', 't', 'y'}

s3

{'P', 'h', 'n', 'o', 't', 'y'}

s3[1] = 'u'

-----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[28], line 1
----> 1 s3[1] = 'u'

TypeError: 'set' object does not support item assignment

s3.discard('h')
# can't edit using index, but deleting and adding the new one is
possible
# This is how it is mutable

s3

{'P', 'n', 'o', 't', 'y'}

s3.add('u')

s3

{'P', 'n', 'o', 't', 'u', 'y'}

len(s3)

6

S = {1,2,3,4,5,6,7,8,9,10}

A = {1,2,3,5,7}

B = {5,7,9,10}

C = {1,2,3,4,5,6,7,8,9,10}

D = {1,2,3,4,5}

E = {6,7,8,9,10}
```

union(iterable), intersection(iterable), difference(iterable), symmetric_difference(iterable)

```
A.union(B)

{1, 2, 3, 5, 7, 9, 10}

A.union('Python')

{1, 2, 3, 5, 7, 'P', 'h', 'n', 'o', 't', 'y'}

l1 = [11,22,33]

A.union(l1)

{1, 2, 3, 5, 7, 11, 22, 33}

A

{1, 2, 3, 5, 7}

A.intersection(B)

{5, 7}

A.difference(B)

{1, 2, 3}

B.difference(A)

{9, 10}

A.intersection_update(B)

A

{5, 7}

A = {1,2,3,5,7}

A.difference(B)

{1, 2, 3}

A

{1, 2, 3, 5, 7}

A.difference_update(B)

A

{1, 2, 3}
```

```
A = {1,2,3,5,7}

A.symmetric_difference(B)

{1, 2, 3, 9, 10}

A

{1, 2, 3, 5, 7}

A.symmetric_difference_update(B)

A

{1, 2, 3, 9, 10}

B

{5, 7, 9, 10}

A = {1,2,3,5,7}

B

{5, 7, 9, 10}

A | B #union

{1, 2, 3, 5, 7, 9, 10}

A & B #intersection

{5, 7}

A ^ B #symmetric difference

{1, 2, 3, 9, 10}

A - B #difference

{1, 2, 3}

B ^ A

{1, 2, 3, 9, 10}

A < S # A is subset o S or not

True

S < A # S is subset of A or not

False

S > A # A is superset of S or not
```

```
True

C == S #equal or not

True

C != A #both are not equal or not

True

6 in B

False

7 in B

True
```

add(), copy(), update()

```
s1 = {10,20,30,40,50}

s1.add(60)

s1

{10, 20, 30, 40, 50, 60}

s1.add(78)

s1

{10, 20, 30, 40, 50, 60, 78}

s1.add(11)

s1

{10, 11, 20, 30, 40, 50, 60, 78}

s1.add(l1)  #mutable object can't be set member

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[138], line 1
----> 1 s1.add(l1)

TypeError: unhashable type: 'list'

t1 = (1,2,3)

s1.add(t1) # immutable object can be set member
```

```
s1

{(1, 2, 3), 10, 11, 20, 30, 40, 50, 60, 78}

s1.add('python') # immutable object can be set member

s1

{(1, 2, 3), 10, 11, 20, 30, 40, 50, 60, 78, 'python'}

s1.add(A) #mutable object can't be set member

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[150], line 1
----> 1 s1.add(A)

TypeError: unhashable type: 'set'

s1.copy()

{(1, 2, 3), 10, 11, 20, 30, 40, 50, 60, 78, 'python'}

s1.update(l1)

s1

{(1, 2, 3), 10, 11, 20, 22, 30, 33, 40, 50, 60, 78, 'python'}

l1

[11, 22, 33]

s1.update(t1)

s1

{(1, 2, 3), 1, 10, 11, 2, 20, 22, 3, 30, 33, 40, 50, 60, 78, 'python'}

s1.update({60,70,80})

s1

{(1, 2, 3), 1, 10, 11, 2, 20, 22, 3, 30, 33, 40, 50, 60, 70, 78, 80,
'python'}
```

pop(), discard(), remove(), clear()

```
s1.pop()

1
```

```
s1.pop()

2

s1.discard((1,2,3))

s1

{10, 11, 20, 22, 3, 30, 33, 40, 50, 60, 70, 78, 80, 'python'}

s1.discard('python')


s1

{3, 10, 11, 20, 22, 30, 33, 40, 50, 60, 70, 78, 80}

s1.remove(33)

s1

{3, 10, 11, 20, 22, 30, 40, 50, 60, 70, 78, 80}

s1.remove('python') # generates error if the specified value is not in
the set

----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Cell In[187], line 1
----> 1 s1.remove('python')

KeyError: 'python'

s1.discard('python') #simply ignores if the specified value is not in
the set

s1.clear()

s1

set()
```

Empty set creation

```
s1 = {}

print(type(s1))

<class 'dict'>

s1 = set()
```

```
print(type(s1))
```

```
<class 'set'>
```

Set Comprehension

```
s = set()
for i in range(5):
    s.add(i)
print(s)
```

```
{0, 1, 2, 3, 4}
```

```
s1 = {i for i in range(5)}
```

```
s1
```

```
{0, 1, 2, 3, 4}
```

```
s2 = { i**2 for i in [-2,-1,0,1,2]}
```

```
s2
```

```
{0, 1, 4}
```

```
s3 = { i for i in (3,10,8,6,9,12,11) if i%2==0}
```

```
s3
```

```
{6, 8, 10, 12}
```

```
s4 = { i.upper() for i in 'phillipines'}
```

```
s4
```

```
{'E', 'H', 'I', 'L', 'N', 'P', 'S'}
```

unhashable (mutable) objects can not be set member

```
s5 = {1,2.3,6+5j,[1,2,3]}  # mutable object i.e. list can not be part
of set
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[224], line 1
----> 1 s5 = {1,2.3,6+5j,[1,2,3]}

TypeError: unhashable type: 'list'
```

```
s5 = {1,2.3,6+5j,(1,2,3)}
```

```
s5
```

```
{(1, 2, 3), (6+5j), 1, 2.3}
```

```
s5 = {1,2.3,6+5j,'python'}
```

```
s5
```

```
{(6+5j), 1, 2.3, 'python'}
```

```
s5 = {1,2.3,6+5j,{1,2,3}}  # mutable object i.e.set can not be part of set
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[234], line 1
----> 1 s5 = {1,2.3,6+5j,{1,2,3}}

TypeError: unhashable type: 'set'
```

Dictionary

```
d1 = {} # Empty Dictionary
```

```
type(d1)
```

```
dict
```

```
d1 = dict()
```

```
type(d1)
```

```
dict
```

```
d1
```

```
{}
```

```
d1['name'] = 'Ajay'  # Add an element
```

```
d1
```

```
{'name': 'Ajay'}
```

```
d1['rollno'] = 101
```

```
d1
```

```
{'name': 'Ajay', 'rollno': 101}
```

```
d1['spi'] = 8.9
```

```
d1
```

```
{'name': 'Ajay', 'rollno': 101, 'spi': 8.9}
```

```
d1[0]  # can't be accessed using index
```

```
-----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Cell In[24], line 1
----> 1 d1[0]

KeyError: 0
```

```
d1['name'] # can be accessed using key
```

```
'Ajay'
```

```
d1['rollno'] = 102  # update the value using key
```

```
d1
```

```
{'name': 'Ajay', 'rollno': 102, 'spi': 8.9}
```

```
d1['fname'] = 'Ajay'
```

```
d1
```

```
{'name': 'Ajay', 'rollno': 102, 'spi': 8.9, 'fname': 'Ajay'}
```

```
d2 = {101:'Ajay', 102:'Riya', 103:'Keyur', 104:'Shreya'}
```

```
d2
```

```
{101: 'Ajay', 102: 'Riya', 103: 'Keyur', 104: 'Shreya'}
```

```
for i in d2:
    print(i, d2[i])
```

```
101 Ajay
102 Riya
103 Keyur
104 Shreya
```

```
for i in d2:
    print(i, d2.get(i))
```

```
101 Ajay
102 Riya
103 Keyur
104 Shreya
```

keys()

```
for i in d2.keys():
    print(i, d2[i])
```

```
101 Ajay
102 Riya
103 Keyur
104 Shreya
```

values()

```
for i in d2.values():
    print(i)
```

```
Ajay
Riya
Keyur
Shreya
```

items()

```
for i in d2.items(): # returns tuple of all elements, in the form of
(key, value)
    print(i)
```

```
(101, 'Ajay')
(102, 'Riya')
(103, 'Keyur')
(104, 'Shreya')
```

```
for  a,b  in d2.items():  # can be unpacked
    #a, b = i
    print(a, b)
```

```
101 Ajay
102 Riya
103 Keyur
104 Shreya
```

```
list(d2.keys())
```

```
[101, 102, 103, 104]
```

copy()

```
d = d1.copy()
```

```
d
```

```
{'name': 'Ajay', 'rollno': 102, 'spi': 8.9, 'fname': 'Ajay'}
```

update()

```
d2['106'] = 'Raaj'
```

```
d2
```

```
{101: 'Ajay', 102: 'Riya', 103: 'Keyur', 104: 'Shreya', '106': 'Raaj'}
```

```
d2.update(d1)
```

```
d2
```

```
{101: 'Ajay',
 102: 'Riya',
 103: 'Keyur',
 104: 'Shreya',
 '106': 'Raaj',
 'name': 'Ajay',
 'rollno': 102,
 'spi': 8.9,
 'fname': 'Ajay'}
```

get(key)

```
d2.get('106')  # returns valus of the key passed
```

```
'Raaj'
```

```
d2.get(110,'NA') # returns value specified in the second parameter if
it is not available
```

```
'NA'
```

```
d2
```

```
{101: 'Ajay',
 102: 'Riya',
 103: 'Keyur',
 104: 'Shreya',
 '106': 'Raaj',
 'name': 'Ajay',
 'rollno': 102,
 'spi': 8.9,
 'fname': 'Ajay'}
```

setdefault(key, value)

```
d2.setdefault(110, 'NA')   # also adds the same element in the
dictionary
```

```
'NA'
```

```
d2
```

```
{101: 'Ajay',
 102: 'Riya',
 103: 'Keyur',
 104: 'Shreya',
 '106': 'Raaj',
 'name': 'Ajay',
 'rollno': 102,
 'spi': 8.9,
 'fname': 'Ajay',
 110: 'NA'}
```

```
d2.setdefault(111)
```

```
d2
```

```
{101: 'Ajay',
 102: 'Riya',
 103: 'Keyur',
 104: 'Shreya',
 '106': 'Raaj',
 'name': 'Ajay',
 'rollno': 102,
 'spi': 8.9,
 'fname': 'Ajay',
 110: 'NA',
 111: None}
```

```
d2.setdefault(101,'NA')
```

```
'Ajay'
```

```
d2
```

```
{101: 'Ajay',
 102: 'Riya',
 103: 'Keyur',
 104: 'Shreya',
 '106': 'Raaj',
 'name': 'Ajay',
 'rollno': 102,
 'spi': 8.9,
 'fname': 'Ajay',
 110: 'NA',
 111: None}
```

fromkeys(iterable,value)

```
l1 = [1,2,3,4,5]
```

```
d3 = dict.fromkeys(l1,'NA')
```

```
d3
```

```
{1: 'NA', 2: 'NA', 3: 'NA', 4: 'NA', 5: 'NA'}
```

pop() & popitem() & clear()

```
d3
```

```
{1: 'NA', 2: 'NA', 3: 'NA', 4: 'NA', 5: 'NA'}
```

```
d3.pop(6)
```

```
----------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
Cell In[101], line 1
----> 1 d3.pop(6)

KeyError: 6
```

```
d3.pop(3)

'NA'

d3

{1: 'NA', 2: 'NA', 4: 'NA', 5: 'NA'}

d3.popitem()

(5, 'NA')

d3.clear()

d3

{}
```

Dictionary Comprehension

```
d1 = { i: i**2 for i in  range(1,6)}

d1

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

d2 = { i:i.upper() for i in 'abcdefabc'}

d2

{'a': 'A', 'b': 'B', 'c': 'C', 'd': 'D', 'e': 'E', 'f': 'F'}

d3 = dict({ (i,i**2) for i in  range(1,6)})

d3

{2: 4, 4: 16, 1: 1, 3: 9, 5: 25}

l1 = [1,2,3,4,5]

l2 = [11,22,33,44,55]

d3 = { i:j   for i,j in zip(l1,l2)}

d3

{1: 11, 2: 22, 3: 33, 4: 44, 5: 55}

d4 = { i:j for i,j in enumerate(l2)}

d4

{0: 11, 1: 22, 2: 33, 3: 44, 4: 55}
```