LIST

- List is an ordered sequence of items
- We can have different data types under a list. E.g We can have integer, float and string items in a same list.

List Creation

```
In [1]: list1 = []            # Empty List
```

```
In [2]: print(type(list1))
```

```
<class 'list'>
```

```
In [3]: list2 = [10,20,30,40]    # List of integers numbers
```

```
In [4]: list2
```

```
Out[4]: [10, 20, 30, 40]
```

```
In [5]: list3 = [10.77,30.66,60.89]  # List of float numbers
```

```
In [6]: list3
```

```
Out[6]: [10.77, 30.66, 60.89]
```

```
In [7]: list4 = ['one','two',"three"]   # List of strings
```

```
In [8]: list4
```

```
Out[8]: ['one', 'two', 'three']
```

```
In [9]: list5 = ['Asif',25,[50, 100],[150, 90]]     # Nested List
```

```
In [10]: list5
```

```
Out[10]: ['Asif', 25, [50, 100], [150, 90]]
```

```
In [11]: list6 = [100,'Asif',17.765]    # List of mixed data types
```

```
In [12]: list6
```

```
Out[12]: [100, 'Asif', 17.765]
```

```
In [13]: list7 = ['Asif',25,[50,100],[150, 90],{'john', 'David'}]
In [14]: list7

Out[14]: ['Asif', 25, [50, 100], [150, 90], {'David', 'john'}]

In [15]: len(list6)   # length of list

Out[15]: 3
```

List Indexing

```
In [16]: list2

Out[16]: [10, 20, 30, 40]

In [17]: list2[0]          # Retreive first element of the list

Out[17]: 10

In [18]: list4

Out[18]: ['one', 'two', 'three']

In [19]: list4[0]           # Retreive first element of the list

Out[19]: 'one'

In [20]: list4[0][0]  # Nested Indexing  - Access the first character of the first list elem

Out[20]: 'o'

In [21]: list4[-1]    # Last item of the list

Out[21]: 'three'

In [22]: list5

Out[22]: ['Asif', 25, [50, 100], [150, 90]]

In [23]: list5[-1]

Out[23]: [150, 90]
```

List Slicing

```
In [24]: mylist = ['one','two','three','four','five','six','seven','eight']

In [25]: mylist
```

```
Out[25]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [26]: mylist[0:3]  # Return all items from 0th to 3rd index location excluding the item
```

```
Out[26]: ['one', 'two', 'three']
```

```
In [27]: mylist[2:5]     # List all items from 2nd to 5th index location excluding the item
```

```
Out[27]: ['three', 'four', 'five']
```

```
In [28]: mylist[:3]     # Return first three items
```

```
Out[28]: ['one', 'two', 'three']
```

```
In [29]: mylist[:2]     # Return first two items
```

```
Out[29]: ['one', 'two']
```

```
In [30]: mylist[-3:]   # Return last three items
```

```
Out[30]: ['six', 'seven', 'eight']
```

```
In [31]: mylist[-2:]  # Return last two items
```

```
Out[31]: ['seven', 'eight']
```

```
In [32]: mylist[-1]   # Return last items of the list
```

```
Out[32]: 'eight'
```

```
In [33]: mylist[:]  # Return whole list
```

```
Out[33]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

Add, Remove & Change Items

```
In [34]: mylist
```

```
Out[34]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [35]: mylist.append('nine')  # Add an item to the end of the list
         mylist
```

```
Out[35]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

```
In [36]: mylist.insert(9,'ten') # Add item  at index location 9
         mylist
```

```
Out[36]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

In [37]:

```
mylist.insert(1,'ONE')  # Add item at index location 1
mylist
```

Out[37]:

```
['one',
 'ONE',
 'two',
 'three',
 'four',
 'five',
 'six',
 'seven',
 'eight',
 'nine',
 'ten']
```

In [38]:

```
mylist.remove('ONE')   # Remove item 'ONE'
mylist
```

Out[38]:

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

In [39]:

```
mylist.pop()   # Remove last item of the list
mylist
```

Out[39]:

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

In [40]:

```
mylist.pop(8)
mylist
```

Out[40]:

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

In [41]:

```
del mylist[7]       # Remove item at index location 7
mylist
```

Out[41]:

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven']
```

In [42]:

```
# Change value of the string
mylist[0] = 1
mylist[1] = 2
mylist[2] = 3
mylist
```

Out[42]:

```
[1, 2, 3, 'four', 'five', 'six', 'seven']
```

In [43]:

```
mylist.clear()     # Empty list /  Delete all items in the list
mylist
```

Out[43]:

```
[]
```

In [44]:

```
del mylist   # Delete the whole list
mylist
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[44], line 2
      1 del mylist   # Delete the whole list
----> 2 mylist

NameError: name 'mylist' is not defined
```

COPY LIST

In [45]:
```
mylist = ['one','two','three','four','five','six','seven','eight','nine']
```

In [46]:
```
mylist1 = mylist    # Create  a new reference "mylist1"
```

In [47]:
```
id(mylist), id(mylist1)   # The address of  both mylist & mylist1 will be the same
```

Out[47]:
```
(2536919638080, 2536919638080)
```

In [48]:
```
mylist2 = mylist.copy()  # Create a copy  of the list
```

In [49]:
```
id(mylist2)  # The address of both mylist & mylist1 will be the same
```

Out[49]:
```
2536911025792
```

In [50]:
```
mylist
```

Out[50]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

In [51]:
```
mylist[0] = 1
```

In [52]:
```
mylist
```

Out[52]:
```
[1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

In [53]:
```
mylist1 # mylist1 will be also impacted as it is pointing to the same list
```

Out[53]:
```
[1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

In [54]:
```
mylist2  # copy of list won't be  impacted due to changes made on the original list
```

Out[54]:
```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

JOIN LISTS

In [55]:
```
list1 = ['one','two','three','four']
list2 = ['five','six','seven','eight']
```

```
In [56]: list3 = list1 + list2  # join two list by '+' operator
         list3
```

```
Out[56]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [57]: list1.extend(list2)      #Append list2 with list1
         list1
```

```
Out[57]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

LIST MEMBERSHIP

```
In [58]: list1
```

```
Out[58]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [59]: 'one' in list1  # check if 'one' exist in the list
```

```
Out[59]: True
```

```
In [60]: 'ten' in list1 # check if 'ten' exist in the list
```

```
Out[60]: False
```

```
In [61]: if 'three' in list1:   # check if 'three' exist in the list
             print('Three is present in the list')
         else:
             print('Three is not present in the list')
```

```
Three is present in the list
```

```
In [62]: if 'eleven' in list1:      # check if 'eleven' exist in the list
             print('eleven is present in the list')
         else:
             print('eleven is not present in the list')
```

```
eleven is not present in the list
```

Reverse & Sort List

```
In [63]: list1
```

```
Out[63]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [64]: list1.reverse()   # Reverse the list
         list1
```

```
Out[64]: ['eight', 'seven', 'six', 'five', 'four', 'three', 'two', 'one']
```

```
In [65]: list1 = list1[::-1]   # Reverse the list
         list1
```

```
Out[65]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [66]: mylist3 = [9,5,2,99,12,88,34]
         mylist3.sort()    # sort list in ascending order
         mylist3
```

```
Out[66]: [2, 5, 9, 12, 34, 88, 99]
```

```
In [67]: mylist3 = [9,5,2,99,12,88,34]
         mylist3.sort(reverse=True)    # sort list in ascending order
         mylist3
```

```
Out[67]: [99, 88, 34, 12, 9, 5, 2]
```

```
In [68]: mylist4 = [88,65,33,21,11,98]
         sorted(mylist4)            # Returns a new sorted list and doesn't change original
```

```
Out[68]: [11, 21, 33, 65, 88, 98]
```

```
In [69]: mylist4
```

```
Out[69]: [88, 65, 33, 21, 11, 98]
```

Loop through a list

```
In [70]: list1
```

```
Out[70]: ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [71]: for i in list1:
             print(i)
```

```
one
two
three
four
five
six
seven
eight
```

```
In [72]: for i in enumerate(list1):
             print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

COUNT

In [73]:
```python
list10 = ['one','two','three','four','one','one','two','three']
```

In [74]:
```python
list10.count('one')   # Number of times item "one" occurred in the list
```

Out[74]:
```
3
```

In [75]:
```python
list10.count('two')    # occurence of item 'two' in the list
```

Out[75]:
```
2
```

In [76]:
```python
list10.count('four')    # occurence of item 'four' in the list
```

Out[76]:
```
1
```

All & Any

In [77]:
```python
L1 = [1,2,3,4,0]
```

In [78]:
```python
all(L1)     #  will return false as one value is false (value 0)
```

Out[78]:
```
False
```

In [79]:
```python
any(L1)     # will return True as  we have items  in the list with True value
```

Out[79]:
```
True
```

In [80]:
```python
L2 = [1,2,3,4,True,False]
```

In [81]:
```python
all(L2)    # Returns false as one value is false
```

Out[81]:
```
False
```

In [82]:
```python
any(L2)       # will return True as  we have items  in the list with True value
```

Out[82]:
```
True
```

In [83]:
```python
L3 = [1,2,3,4,True]
```

```
In [84]: all(L3)   # will retrun True as all items  in the list are True

Out[84]: True
```

```
In [85]: any(L3)

Out[85]: True
```

```
In [ ]:
```

Tuple

- 1. Tuple is similar to List except that the objects in tuple are immutable which means we cannot

change the elements of a tuple once assigned.

- 2. When we do not want to change the data over time, tuple is a preferred data type.
- 3. Iterating over the elements of a tuple is faster compared to iterating over a list.

```
In [86]: tup1 = ()             # Empty tuple
```

```
In [87]: tup1

Out[87]: ()
```

```
In [88]: tup2 = (10,30,60)     # tuple of integers numbers
```

```
In [89]: tup2

Out[89]: (10, 30, 60)
```

```
In [90]: tup3 = (10.77,30.66,60.89)     # tuple of float numbers
```

```
In [91]: tup3

Out[91]: (10.77, 30.66, 60.89)
```

```
In [92]: tup4 = ('one','two',"three")    # tuple of strings
```

```
In [93]: tup4

Out[93]: ('one', 'two', 'three')
```

```
In [94]: tup5 = ('Asif',25,(50,100),(150,90))   # Nested tuples
```

```
In [95]: tup5
```

```
Out[95]: ('Asif', 25, (50, 100), (150, 90))
```

```
In [96]: tup6 = (100,'Asif',17.765)     # tuple of mixed data types
```

```
In [97]: tup6
```

```
Out[97]: (100, 'Asif', 17.765)
```

```
In [98]: tup7 = ('Asif', 25,[50,100],[150,90],{'John','David'},(99,22,33))
```

```
In [99]: tup7
```

```
Out[99]: ('Asif', 25, [50, 100], [150, 90], {'David', 'John'}, (99, 22, 33))
```

```
In [100… len(tup7)    # Length of tuple
```

```
Out[100… 6
```

Tuple Indexing

```
In [101… tup2
```

```
Out[101… (10, 30, 60)
```

```
In [102… tup2[0]              # Retreive first element of the tuple
```

```
Out[102… 10
```

```
In [103… tup4
```

```
Out[103… ('one', 'two', 'three')
```

```
In [104… tup4[0]        # Retreive first element of the tuple
```

```
Out[104… 'one'
```

```
In [105… tup4[0][0]        # Nested Indexing  - Access the first character of the first tupl
```

```
Out[105… 'o'
```

```
In [106… tup4[-1]    # last item of the tuple
```

```
Out[106… 'three'
```

```
In [107… tup5
```

```
Out[107… ('Asif', 25, (50, 100), (150, 90))
```

```
In [108… tup5[-1]   # Last item of the tuple
```

```
Out[108… (150, 90)
```

Tuple Slicing

```
In [109… mytuple = ['one','two','three','four','five','six','seven','eight']
```

```
In [110… mytuple[0:3]     # Return all items from 0th to 3rd index location excluding the it
```

```
Out[110… ['one', 'two', 'three']
```

```
In [111… mytuple[2:5]     # list all items from 2nd to 5th index location excluding the item
```

```
Out[111… ['three', 'four', 'five']
```

```
In [112… mytuple[:3]     # Return first three items
```

```
Out[112… ['one', 'two', 'three']
```

```
In [113… mytuple[:2]    # Return first two items
```

```
Out[113… ['one', 'two']
```

```
In [114… mytuple[-3:]   # Return last three items
```

```
Out[114… ['six', 'seven', 'eight']
```

```
In [115… mytuple[-2:]   # Return last two items
```

```
Out[115… ['seven', 'eight']
```

```
In [116… mytuple[-1:]   # Return last item of the tuple
```

```
Out[116… ['eight']
```

```
In [117… mytuple[:]    # Return whole tuple
```

```
Out[117… ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

Remove & Change Items

```
In [118… mytuple
```

```
Out[118… ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [119… del mytuple[0]   # Tuples are immutable which means we can't DELETE tuple items
```

```
In [120… mytuple[0] = 1   #  Tuples are immutable which means we can't CHANGE tuple items
In [121…  del mytuple        # Deleting entire tuple object is possible
```

Loop through a tuple

```
In [122…  mytuple1 =   ('one','two','three','four','five','six','seven','eight')
```

```
In [123…  mytuple1
```

```
Out[123…  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

```
In [124…  for i in mytuple1:
              print(i)
```

```
one
two
three
four
five
six
seven
eight
```

```
In [125…  for i in enumerate(mytuple1):
              print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

COUNT

```
In [126…  mytuple2 =('one','two','three','four','one','one','two','three')
```

```
In [127…  mytuple2.count('one')  # Number of times items "one" occurred in the tuple.
```

```
Out[127…  3
```

```
In [128…  mytuple2.count('two') # occurence of item 'two' in the tuple
```

```
Out[128…  2
```

```
In [129…  mytuple2.count('four')  # occurence of item 'four' in the tuple
```

```
Out[129…  1
```

Tuple Membership

In [130…

```
mytuple1
```

Out[130…

```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [131…

```
'one' in mytuple1    # Check if 'one' exist in the tuple
```

Out[131…

```
True
```

In [132…

```
'ten' in mytuple1    # check if 'ten' exist in the tuple
```

Out[132…

```
False
```

In [133…

```
if 'three' in mytuple1:   # check if 'three' exist in the tuple
    print('Three is present in the tuple')
else:
    print('Three is not present in the tuple')
```

```
Three is present in the tuple
```

In [134…

```
if 'eleven' in mytuple1:   # check if 'eleven' exist in the tuple
    print('Eleven is present in the tuple')
else:
    print('Eleven is not present in the tuple')
```

```
Eleven is not present in the tuple
```

Index Position

In [135…

```
mytuple1
```

Out[135…

```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [136…

```
mytuple1.index('one')   # Index of first element to 'one'
```

Out[136…

```
0
```

In [137…

```
mytuple1.index('five')  # Index of first element to 'five'
```

Out[137…

```
4
```

In [138…

```
mytuple2
```

Out[138…

```
('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
```

In [139…

```
mytuple2.index('one')   # Index of first element to 'one'
```

Out[139…

```
0
```

Sorting

In [140…

```
mytuple3 = (43,67,99,12,6,90,67)
```

In [141…

```
sorted(mytuple3)  # Return a new sorted list and doesn't change original tuple
```

Out[141…

```
[6, 12, 43, 67, 67, 90, 99]
```

In [142…

```
sorted(mytuple3,reverse=True)
```

Out[142…

```
[99, 90, 67, 67, 43, 12, 6]
```

In [ ]:

```

```

SETS

- 1. Unordered & Unindexed collection of items.
- 2. Set elements are unique. Duplicate elements are not allowed.
- 3. Set elements are immutable (cannot be changed).
- 4. Set itself is mutable. We can add or remove items from it.

In [157…

```
myset = {1,2,3,4,5}  # Set of numbers
myset
```

Out[157…

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

In [158…

```
len(myset)  # Length of the set
```

Out[158…

```
5
```

In [159…

```
my_set = {1,1,2,2,3,4,5,5}
my_set                           # Duplicate elements are not allowed.
```

Out[159…

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

In [160…

```
myset1 = {1.79, 2.08,3.99,4.56,5.45}   # Set of float numbers
myset1
```

Out[160…

```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [161…

```
myset2 = {'Asif','John','Tyrion'}     # Set of strings
myset2
```

Out[161…

```
{'Asif', 'John', 'Tyrion'}
```

In [162…

```
myset3 = {10,20,"Hola",(11,22,32)}    # Mixed data types
```

```
myset3
```

Out[162…

```
{(11, 22, 32), 10, 20, 'Hola'}
```

In [163…

```
myset3 = {10,20,"Hola",[11,22,32]}  # Set doesn't allow mutable items like lists
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[163], line 1
----> 1 myset3 = {10,20,"Hola",[11,22,32]}  # Set doesn't allow mutable items like l
ists
      2 myset3

TypeError: unhashable type: 'list'
```

In [164…

```
myset4 = set()  # create an empty set
print(type(myset4))
```

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

In [165…

```
my_set1 = (('one','two','three','four'))
my_set1
```

Out[165…

```
('one', 'two', 'three', 'four')
```

Loop through a set

In [166…

```
myset = {'one','two','three','four','five','six','seven','eight'}

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

```
three
five
two
four
seven
one
eight
six
```

In [167…

```
for i in enumerate(myset):
    print(i)
```

```
(0, 'three')
(1, 'five')
(2, 'two')
(3, 'four')
(4, 'seven')
(5, 'one')
(6, 'eight')
(7, 'six')
```

Set Membership

In [168…

```
myset
```

Out[168…

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [169…

```
'one' in  myset     # Check if 'one' exist in the set
```

Out[169…

```
True
```

In [170…

```
'ten' in myset      # Check if 'ten' exist in the set
```

Out[170…

```
False
```

In [171…

```
if 'three' in myset:   # check if 'three' exist in the set
    print('Three is present in the set')
else:
    print('Three is not present in the set')
```

```
Three is present in the set
```

In [172…

```
if 'eleven' in myset:   # check if 'eleven' exist in the set
    print('Eleven is present in the set')
else:
    print('Eleven is not present in the set')
```

```
Eleven is not present in the set
```

Add & Remove Items

In [173…

```
myset
```

Out[173…

```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [174…

```
myset.add('NINE')       # Add item to a set using add() method
myset
```

Out[174…

```
{'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [175…

```
myset.update(['TEN','ELEVEN','TWELVE'])    # Add multiple item to set using update()
myset
```

Out[175…

```
{'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [176…

```
myset.remove('NINE')       # Remove item in a set using remove() method
myset
```

Out[176…

```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [177…

```
myset.discard('TEN')   # Remove item from a set using discard() method
myset
```

Out[177…

```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [178…

```
myset.clear()   # Delete all items in a set
myset
```

Out[178…

```
set()
```

In [179…

```
del myset  # Delete the set object
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[179], line 2
      1 del myset  # Delete the set object
----> 2 myset

NameError: name 'myset' is not defined
```

Copy Set

In [180…
```
myset = {'one','two','three','four','five','six','seven','eight'}
myset
```

Out[180…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [181…
```
myset1 = myset       # Create a new reference "myset1"
myset1
```

Out[181…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [182…
```
id(myset) , id(myset1)    # The address of both myset & myset1 will be the same as
```

Out[182…
```
(2536919524832, 2536919524832)
```

In [183…
```
my_set = myset.copy()   #  Create a copy of the set
my_set
```

Out[183…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [184…
```
id(my_set)    # The address of my_set will be different from myset
```

Out[184…
```
2536919526176
```

In [185…
```
myset.add('nine')
myset
```

Out[185…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [186…
```
myset1  # myset1 will be also impacted as it is pointing to the same list
```

Out[186…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [187…
```
my_set   # Copy of the set won't be impacted due to changes made on the Original se
```

Out[187…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

Set Operation

- Union

In [188…

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

In [189…

```
A|B          # Union of A and B (All elements from both sets.NO DUPLICATES)
```

Out[189…

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [190…

```
A.union(B)    # Union of A and B
```

Out[190…

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [191…

```
A.union(B,C)      # Union of A, B and C
```

Out[191…

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

In [192…

```
"""
Updates the set calling the update() method with union of A, B and C.

for below example set A will be updated with union of A, B and C.
"""
A.update(B,C)
A
```

Out[192…

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

- Intersection

In [193…

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [194…

```
A & B     # Intersection of A and B (Common items in both sides)
```

Out[194…

```
{4, 5}
```

In [195…

```
A.intersection(B)     # Intersection of A and B
```

Out[195…

```
{4, 5}
```

In [196…

```
"""
Updates the set calling the intersection_update() method with intersection of A & B

for below example set A will be updated with intersection of A & B.
"""
A.intersection_update(B)
A
```

Out[196…

```
{4, 5}
```

- Difference

In [197…

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [198…

```
A - B    # Set of element that are only in A but not in B
```

Out[198…

```
{1, 2, 3}
```

In [199…

```
A.difference(B)  # Difference of sets
```

Out[199…

```
{1, 2, 3}
```

In [200…

```
B - A    # Set of element that are only in B but not in A
```

Out[200…

```
{6, 7, 8}
```

In [201…

```
B.difference(A)   # Difference of sets
```

Out[201…

```
{6, 7, 8}
```

In [202…

```
"""
Updates the set calling the difference_update() method with differnce of sets.

for below example set B will be updated with difference of B & A.
"""
B.difference_update(A)
B
```

Out[202…

```
{6, 7, 8}
```

- Symmetric Difference

In [203…

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

In [204…

```
A ^ B  # Symmetric Difference (Set of elements of A and B but not in both)
```

Out[204…

```
{1, 2, 3, 6, 7, 8}
```

In [205…

```
A.symmetric_difference(B)  # Symmetric difference of sets
```

Out[205…

```
{1, 2, 3, 6, 7, 8}
```

In [206…

```
"""
Updates the set calling the symmetric_difference_update() method with symmetric dif

for below example set A will be updated with symmetric difference of A & B.
"""
```

```
A.symmetric_difference_update(B)
A
```

Out[206…

```
{1, 2, 3, 6, 7, 8}
```

- Subset , Superset & Disjoint

In [207…

```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

In [208…

```
B.issubset(A)     # set B is said to be the subset of  set A
```

Out[208…

```
True
```

In [209…

```
A.issuperset(B)   # set A is said to be the subset of  set B
```

Out[209…

```
True
```

In [210…

```
C.isdisjoint(A)  # Two sets are said to be disjoint sets if they have no common ele
```

Out[210…

```
True
```

In [211…

```
B.isdisjoint(A)  # Two sets are said to be disjoint sets if they have no common ele
```

Out[211…

```
False
```

Other Builtin functions

In [212…

```
A
```

Out[212…

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [213…

```
sum(A)
```

Out[213…

```
45
```

In [214…

```
min(A)
```

Out[214…

```
1
```

In [215…

```
max(A)
```

Out[215…

```
9
```

In [216…

```
len(A)
```

Out[216…

```
9
```

In [217…

```
list(enumerate(A))
```

Out[217…

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

In [218…

```
D = sorted(A,reverse=True)
D
```

Out[218…

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

In [219…

```
sorted(D)
```

Out[219…

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

DICTIONARY

- Dictionary is a mutable data type in Python.
- A python dictionary is a collection of key and value pairs separated by a colon (:) & enclosed

in curly braces {}.

- Keys must be unique in a dictionary, duplicate values are allowed.

- Create Dictionary

In [220…

```
mydict = dict()      # empty dictionary
mydict
```

Out[220…

```
{}
```

In [221…

```
mydict = dict()      # empty dictionary
mydict
```

Out[221…

```
{}
```

In [222…

```
mydict = dict({1:'one',2:'two',3:'three'})  # Create Dictionary using dict()
mydict
```

Out[222…

```
{1: 'one', 2: 'two', 3: 'three'}
```

In [223…

```
mydict = {'A':'one', 'B':'two', 'C':'three'}  # dictionary with character keys
mydict
```

Out[223…

```
{'A': 'one', 'B': 'two', 'C': 'three'}
```

In [224…

```
mydict = {1:'one', 'A':'two', 3:'three'}    # dictionary with mixed data keys
mydict
```

Out[224…

```
{1: 'one', 'A': 'two', 3: 'three'}
```

In [225…

```
mydict.keys()     # Return Dictionary keys using keys() method
```

Out[225…

```
dict_keys([1, 'A', 3])
```

In [226…

```
mydict.values()   # Return Dictionary values using values() method
```

Out[226…

```
dict_values(['one', 'two', 'three'])
```

In [227…

```
mydict.items()    # Access each key-value pair within a dictionary
```

Out[227…

```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

In [228…

```
mydict = {1:'one' , 2:'two' , 'A':['asif' ,  'john', 'Maria'], 'B':('Bat', 'Cat', '
mydict
```

Out[228…

```
{1: 'one',
 2: 'two',
 'A': ['asif', 'john', 'Maria'],
 'B': ('Bat', 'Cat', 'Hat')}
```

In [229…

```
mydict = {1:'one' , 2:'two' , 'A':{'Name' : 'Asif' , 'Age': 20 }, 'B':('Bat', 'Cat'
mydict
```

Out[229…

```
{1: 'one',
 2: 'two',
 'A': {'Name': 'Asif', 'Age': 20},
 'B': ('Bat', 'Cat', 'Hat')}
```

In [230…

```
keys = {'a', 'b', 'c', 'd'}
mydict3 = dict.fromkeys(keys)      # Create a dictionary from a sequence of keys
mydict3
```

Out[230…

```
{'c': None, 'd': None, 'a': None, 'b': None}
```

In [231…

```
keys = {'a', 'b', 'c', 'd'}
value = 10
mydict3 = dict.fromkeys(keys,value)      # Create a dictionary from a sequence of k
mydict3
```

Out[231…

```
{'c': 10, 'd': 10, 'a': 10, 'b': 10}
```

In [232…

```
keys = {'a', 'b', 'c', 'd'}
value = [10, 20, 30]
mydict3 = dict.fromkeys(keys,value)      # Create a dictionary from a sequence of k
mydict3
```

Out[232…

```
{'c': [10, 20, 30], 'd': [10, 20, 30], 'a': [10, 20, 30], 'b': [10, 20, 30]}
```

In [233…

```
value.append(40)
mydict3
```

Out[233…

```
{'c': [10, 20, 30, 40],
 'd': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'b': [10, 20, 30, 40]}
```

- Accessing Items

In [234…

```
mydict = {1:'one', 2:'two', 3:'three', 4:'four'}
mydict
```

Out[234…

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [235…

```
mydict[1]    # Access item using key
```

Out[235…

```
'one'
```

In [236…

```
mydict.get(1)   # Access item using key
```

Out[236…

```
'one'
```

In [237…

```
mydict1 = {'Name':'Asif' , 'ID': 74123 , 'DOB': 1991, 'job':'Analyst'}
mydict1
```

Out[237…

```
{'Name': 'Asif', 'ID': 74123, 'DOB': 1991, 'job': 'Analyst'}
```

In [238…

```
mydict1['Name']  # Access item using key
```

Out[238…

```
'Asif'
```

In [239…

```
mydict1.get('job')   # Access item using get() method
```

Out[239…

```
'Analyst'
```

Add , Remove & Change Items

In [240…

```
mydict1 = {'Name':'Asif', 'ID': 12345, 'DOB': 1991, 'Address' : 'hilsinki'}
mydict1
```

Out[240…

```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'hilsinki'}
```

In [241…

```
mydict1['DOB']  = 1992      # Changing Dictionary Items
mydict1['Address']  = 'Delhi'
mydict1
```

Out[241…

```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1992, 'Address': 'Delhi'}
```

In [242…

```
dict1 = {'DOB':1995}
mydict1.update(dict1)
mydict1
```

```
Out[242…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}

In [243…  mydict1['job']  = 'Analyst'      # Adding items in the dictionary
          mydict1

Out[243…  {'Name': 'Asif',
           'ID': 12345,
           'DOB': 1995,
           'Address': 'Delhi',
           'job': 'Analyst'}

In [244…  mydict1.pop('job')    # Removing items in the dictionary using pop method
          mydict1

Out[244…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}

In [245…  mydict1.popitem()    # A random item is removed

Out[245…  ('Address', 'Delhi')

In [246…  mydict1

Out[246…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1995}

In [247…  del[mydict1['ID']]  # Removing item using del method
          mydict1

Out[247…  {'Name': 'Asif', 'DOB': 1995}

In [248…  mydict1.clear()   # Deleting all items of the dictionary using clear method
          mydict1

Out[248…  {}

In [249…  del mydict1   # Delete the dictionary object
          mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[249], line 2
      1 del mydict1   # Delete the dictionary object
----> 2 mydict1

NameError: name 'mydict1' is not defined
```

Copy Dictionary

```
In [250…  mydict = {'Name':'Asif' , 'ID': 12345, 'DOB': 1991 , 'Address': 'Hilsinki'}
          mydict

Out[250…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

```
In [251…  mydict1 = mydict  # Create a new reference "mydict1"
```

```
In [252…  id(mydict), id(mydict1)  # The address of both mydict & mydict1 will be the same
```

```
Out[252…  (2536920402304, 2536920402304)
```

```
In [253…  mydict2 = mydict.copy()    # Create a copy of the dictionary
```

```
In [254…  id(mydict2)
```

```
Out[254…  2536920374784
```

```
In [255…  mydict['Address'] = 'Mumbai'
```

```
In [256…  mydict
```

```
Out[256…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```

```
In [257…  mydict1    # mydict1 will be also impacted as it is pointing to the same dictionary
```

```
Out[257…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```

```
In [258…  mydict2    # copy of dict won't be impacted due to the changes made in the origina
```

```
Out[258…  {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

Loop through a dictionary

```
In [259…  mydict1 = {'Name':'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hlinski', 'job':'An
          mydict1
```

```
Out[259…  {'Name': 'Asif',
           'ID': 12345,
           'DOB': 1991,
           'Address': 'Hlinski',
           'job': 'Analyst'}
```

```
In [260…  for i in mydict1:
              print(i , ':' , mydict1[i])  # key & value pair
```

```
Name : Asif
ID : 12345
DOB : 1991
Address : Hlinski
job : Analyst
```

```
In [261…  for i in mydict1:
              print(mydict1[i])     # Dictionary Items
```

```
Asif
12345
1991
Hlinski
Analyst
```

Dictionary Membership

In [262…
```
mydict1 = {'Name':'Asif', 'ID': 12345, 'DOB': 1991, 'job':'Analyst'}
mydict1
```

Out[262…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'job': 'Analyst'}
```

In [263…
```
'Name' in mydict1   # Test if a key is in a dictionary or not
```

Out[263…
```
True
```

In [264…
```
'Asif' in mydict1   # Membership test can be only done for keys
```

Out[264…
```
False
```

In [265…
```
'ID' in mydict1
```

Out[265…
```
True
```

In [266…
```
'Address' in mydict1
```

Out[266…
```
False
```

In [267…
```
mydict1 = {'Name':'Asif', 'ID': 12345, 'DOB': 1991, 'job':'Analyst'}
mydict1
```

Out[267…
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'job': 'Analyst'}
```

In [268…
```
all(mydict1)   # will return false as one value is false (value 0)
```

Out[268…
```
True
```

In [269…
```
any(mydict1)
```

Out[269…
```
True
```

Datastructures are completed