Python Data Structure

In [1]:
```
l =[] #list
l
```

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

In [2]:
```
type(l)
```

Out[2]:
```
list
```

In [3]:
```
l.append(100)
l.append(200)
l.append(300)
l.append(400)
l
```

Out[3]:
```
[100, 200, 300, 400]
```

In [4]:
```
l.append(12.5)
l.append(True)
```

In [5]:
```
l
```

Out[5]:
```
[100, 200, 300, 400, 12.5, True]
```

In [6]:
```
l.append(200) # duplication is allow
```

In [7]:
```
l
```

Out[7]:
```
[100, 200, 300, 400, 12.5, True, 200]
```

In [8]:
```
print(len(l))
```

```
7
```

In [9]:
```
l.append(12,22)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 1
----> 1 l.append(12,22)

TypeError: list.append() takes exactly one argument (2 given)
```

In [ ]:
```
l
```

In [10]:
```
l.count(300)
```

Out[10]: 1

In [11]:
```
l.count(200) # they count the number how many time they repat in list
```

Out[11]: 2

In [12]:
```
l1 =[]
```

In [13]:
```
l = l1.copy()
```

In [14]:
```
l1
```

Out[14]: []

In [15]:
```
l
```

Out[15]: []

In [16]:
```
l.append(100)
l.append(200)
l.append(300)
l.append(400)
l
```

Out[16]: [100, 200, 300, 400]

In [17]:
```
l.append(3.3)
l.append(True)
l.append(1+2j)
```

In [18]:
```
l
```

Out[18]: [100, 200, 300, 400, 3.3, True, (1+2j)]

In [19]:
```
l1.append(22)
l1.append(33)
l1
```

Out[19]: [22, 33]

In [20]:
```
l1.append(8.9)
l1.append(False)
l1.append(1000)
l1.append(5)
l1
```

Out[20]: [22, 33, 8.9, False, 1000, 5]

In [21]:
```
len(l1)
```

Out[21]: 6

In [22]:
```
l1
```

Out[22]: [22, 33, 8.9, False, 1000, 5]

In [23]:
```
l2=[]
```

In [24]:
```
l2 = l1.copy()
```

In [25]:
```
l1
```

Out[25]: [22, 33, 8.9, False, 1000, 5]

In [26]:
```
l2
```

Out[26]:
```
[22, 33, 8.9, False, 1000, 5]
```

In [27]:
```
l2.clear()
```

In [28]:
```
l2
```

Out[28]:
```
[]
```

In [29]:
```
l2 = l1.copy()
```

In [30]:
```
l2
```

Out[30]:
```
[22, 33, 8.9, False, 1000, 5]
```

In [31]:
```
print(l)
print(l1)
print(l2)
```

```
[100, 200, 300, 400, 3.3, True, (1+2j)]
[22, 33, 8.9, False, 1000, 5]
[22, 33, 8.9, False, 1000, 5]
```

In [32]:
```
l.extend(l1) # This fun will merge the set
```

In [33]:
```
l
```

Out[33]:
```
[100, 200, 300, 400, 3.3, True, (1+2j), 22, 33, 8.9, False, 1000, 5]
```

In [34]:
```
len(l)
```

Out[34]:
```
13
```

In [35]:
```
l
```

Out[35]:
```
[100, 200, 300, 400, 3.3, True, (1+2j), 22, 33, 8.9, False, 1000, 5]
```

In [36]:
```
l.index(True) # index fun tell position of the value
```

Out[36]: 5

In [37]:
```
l.index(10) # if number is not present then they ggive error
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[37], line 1
----> 1 l.index(10) # if number is not present then they ggive error

ValueError: 10 is not in list
```

In [ ]:
```
l
```

In [38]:
```
l1
```

Out[38]: [22, 33, 8.9, False, 1000, 5]

In [39]:
```
l1.append(50)
l1.append(60)
l1
```

Out[39]: [22, 33, 8.9, False, 1000, 5, 50, 60]

In [40]:
```
l1.insert(7, 55) # insert fun add the value where you want
                 # so i want to add 55 number in between 50 to 60 so that
                 # i write means i gave first index position and then i give value (7 index posistion , 55 is value)
```

In [41]:
```
l1
```

Out[41]: [22, 33, 8.9, False, 1000, 5, 50, 55, 60]

In [42]:
```
l1.insert(4, True)
```

In [43]:
```
l1
```

Out[43]: [22, 33, 8.9, False, True, 1000, 5, 50, 55, 60]

In [44]:
```
l1.pop() # pop fun remove the number by dafault last number okk
l1
```

Out[44]: [22, 33, 8.9, False, True, 1000, 5, 50, 55]

In [45]:
```
l1.pop(2) # but if you give index position in pop fun then he will remove the this value
l1        # so i give position 2 so that pop remove the 8.9 number
```

Out[45]: [22, 33, False, True, 1000, 5, 50, 55]

In [46]:
```
l1
```

Out[46]: [22, 33, False, True, 1000, 5, 50, 55]

In [47]:
```
l1.remove() # this fun must be one arg
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[47], line 1
----> 1 l1.remove() # this fun must be one arg

TypeError: list.remove() takes exactly one argument (0 given)
```

In [ ]:
```
l1.remove(22) # This fun also remove the value but it will remove value wise means
              # whatever you want to remove value give direct value dont need to index position here.
```

In [48]:
```
l1
```

Out[48]: [22, 33, False, True, 1000, 5, 50, 55]

In [49]:
```
l1.remove(False)
l1
```

Out[49]: [22, 33, True, 1000, 5, 50, 55]

In [50]:
```
l1
```

Out[50]: [22, 33, True, 1000, 5, 50, 55]

In [51]:
```
l1.reverse() # this fun do reverse the whole set
l1
```

Out[51]: [55, 50, 5, 1000, True, 33, 22]

In [52]:
```
l
```

Out[52]: [100, 200, 300, 400, 3.3, True, (1+2j), 22, 33, 8.9, False, 1000, 5]

In [53]:
```
l.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 1
----> 1 l.sort()

TypeError: '<' not supported between instances of 'complex' and 'int'
```

In [ ]:
```
l1
```

In [54]:
```
l1.sort() # this fun make set in assending order
l1
```

Out[54]: [True, 5, 22, 33, 50, 55, 1000]

In [55]:
```
l1.sort(reverse=True) # desending order
l1
```

Out[55]: [1000, 55, 50, 33, 22, 5, True]

In [56]:
```
w = 'welcome'
w
```

Out[56]: 'welcome'

In [57]:
```
w[:]
```

Out[57]: 'welcome'

In [58]:
```
w[1:5]
```

Out[58]: 'elco'

In [59]:
```
len(w)
```

Out[59]: 7

In [60]:
```
w[3:]
```

Out[60]: 'come'

In [61]:
```
w[:3]
```

Out[61]: 'wel'

In [62]:
```
w[1:10]
```

Out[62]: 'elcome'

In [63]:
```
w[0:10]
```

Out[63]: 'welcome'

In [64]:
```
w[0:7:2]
```

Out[64]: 'wloe'

In [65]:
```
w[::2]
```

Out[65]: 'wloe'

In [66]:
```
w[::3]
```

Out[66]: 'wce'

In [67]:
```
w[::-1]
```

Out[67]: 'emoclew'

Tuple

In [68]:
```
t = ()
t
```

Out[68]: ()

In [69]:
```
type(t)
```

Out[69]: tuple

In [70]:
```
t = (10 , 20 , 30 , 40)
t
```

Out[70]: (10, 20, 30, 40)

In [71]:
```
t.count(10)
```

Out[71]: 1

In [72]:
```
t.index(10)
```

Out[72]: 0

In [73]:
```
t.index(30)
```

Out[73]: 2

Set

In [74]:
```
s = {} # if you does not define value inside the set then it consideered dict bydafualt
s
```

Out[74]: {}

In [75]:
```
type(s)
```

Out[75]: dict

In [76]:
```
s = {2,3,4,5,6,8}
s
```

Out[76]:
```
{2, 3, 4, 5, 6, 8}
```

In [77]:
```
s1 = {4,6,2,1,50,20,10,3} #set by default print teh value in assending way if you pass the same data type value
s1
```

Out[77]:
```
{1, 2, 3, 4, 6, 10, 20, 50}
```

In [78]:
```
s2 = {True , False , 1+2j, 3.3 , 2 , 1 } # but if give multiple datatype then order is impossible
s2
```

Out[78]:
```
{(1+2j), 2, 3.3, False, True}
```

In [79]:
```
s1
s2
s
```

Out[79]:
```
{2, 3, 4, 5, 6, 8}
```

In [80]:
```
s.add(1)
```

In [81]:
```
s
```

Out[81]:
```
{1, 2, 3, 4, 5, 6, 8}
```

In [82]:
```
s.add(2.3)
s
```

Out[82]:
```
{1, 2, 2.3, 3, 4, 5, 6, 8}
```

In [83]:
```
s
```

Out[83]:
```
{1, 2, 2.3, 3, 4, 5, 6, 8}
```

In [85]:
```
print(s)
print(s1)
print(s2)
```

```
{1, 2, 3, 4, 5, 6, 2.3, 8}
{1, 2, 3, 4, 6, 10, 50, 20}
{False, True, 2, 3.3, (1+2j)}
```

In [86]:
```
s = s2.copy()
```

In [87]:
```
s
```

Out[87]:
```
{(1+2j), 2, 3.3, False, True}
```

In [88]:
```
s.pop()
```

Out[88]:
```
False
```

In [89]:
```
s
```

Out[89]:
```
{(1+2j), 2, 3.3, True}
```

In [90]:
```
s.pop(2) # in pop index is not allow
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[90], line 1
----> 1 s.pop(2)

TypeError: set.pop() takes no arguments (1 given)
```

In [91]:
```
s.pop()
```

Out[91]:
```
True
```

In [92]:
```
s
```

Out[92]:
```
{(1+2j), 2, 3.3}
```

In [93]:
```
s.remove(3.3) # remove the element
```

In [94]:
```
s
```

Out[94]: {(1+2j), 2}

In [95]:
```
s2
```

Out[95]: {(1+2j), 2, 3.3, False, True}

In [96]:
```
s2.discard(1) #clam guy , even if value is not present in set still they dont give error
```

In [97]:
```
s2
```

Out[97]: {(1+2j), 2, 3.3, False}

In [98]:
```
s2.discard(1+2j)
```

In [99]:
```
s2
```

Out[99]: {False, 2, 3.3}

set operation

In [104…
```
s5 = {1,2,3,4,5}
s6 = {4,5,6,7,8}
s7 = {8,9,10}
```

In [105…
```
s5
```

Out[105… {1, 2, 3, 4, 5}

In [106…
```
s6
```

Out[106… {4, 5, 6, 7, 8}

In [107…
```
s7
```

Out[107… {8, 9, 10}

In [108…
```
s5.union(s6) # union means it will merge the value or elements and symbol is |
```
Out[108…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```
In [109…
```
s5
```
Out[109…
```
{1, 2, 3, 4, 5}
```
In [110…
```
s6
```
Out[110…
```
{4, 5, 6, 7, 8}
```
In [111…
```
s5|s6|s7
```
Out[111…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```
In [112…
```
#intersection
s5.intersection(s6) # common value , symbo is &
```
Out[112…
```
{4, 5}
```
In [113…
```
s5 & s6
```
Out[113…
```
{4, 5}
```
In [114…
```
s5 & s7
```
Out[114…
```
set()
```
In [115…
```
s6 & s7
```
Out[115…
```
{8}
```
In [116…
```
#difference
s5.difference(s6)  # s5 me jo hai par s6 me nahi and symbol is -
```
Out[116…
```
{1, 2, 3}
```

In [117…
```
s5 - s6
```

Out[117…
```
{1, 2, 3}
```

In [118…
```
s6 - s7
```

Out[118…
```
{4, 5, 6, 7}
```

In [120…
```
print(s5)
print(s6)
print(s7)
```

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

In [121…
```
#symmetric difference
s5.symmetric_difference(s6) # they remove common value and symbol is ^
```

Out[121…
```
{1, 2, 3, 6, 7, 8}
```

In [122…
```
s5 ^ s6
```

Out[122…
```
{1, 2, 3, 6, 7, 8}
```

In [123…
```
s6 ^ s7
```

Out[123…
```
{4, 5, 6, 7, 9, 10}
```

In [124…
```
s7
```

Out[124…
```
{8, 9, 10}
```

In [125…
```
s6
```

Out[125…
```
{4, 5, 6, 7, 8}
```

In [126…
```
s6.symmetric_difference_update(s7)
```

In [127…
```
print(s6)
```

```
{4, 5, 6, 7, 9, 10}
```

SUPERSET , SUBSET , DISJOINT

In [130…
```
a = {1,2,3,4,5,6,7,8,9}
b = {3,4,5,6,7,8}
c = {10,20,30,40}
```

In [131…
```
a.issuperset(b)
```

Out[131… True

In [132…
```
b.issuperset(a)
```

Out[132… False

In [133…
```
a.issuperset(c)
```

Out[133… False

In [134…
```
b.issubset(a)
```

Out[134… True

In [135…
```
a.issubset(b)
```

Out[135… False

In [137…
```
c.isdisjoint(a)
```

Out[137… True

In [138…
```
c.isdisjoint(b)
```

Out[138… True

Dictionary

In [139…
```
mydict = {1:'one', 2:'two', 3:'three', 4:'four'}
```

In [141…
```
mydict
```

Out[141…
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

In [144…
```
mydict.values()
```

Out[144…
```
dict_values(['one', 'two', 'three', 'four'])
```

In [146…
```
mydict.items()
```

Out[146…
```
dict_items([(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four')])
```

In [147…
```
mydict[4] = 'four'
mydict
```

Out[147…
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

range

In [148…
```
range() # range () expected at least one arg
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[148], line 1
----> 1 range()

TypeError: range expected at least 1 argument, got 0
```

In [149…
```
range(5)
```

Out[149…
```
range(0, 5)
```

In [150…
```
list(range(0,5))
```

Out[150…
```
[0, 1, 2, 3, 4]
```

In [151…
```
list(range(0,20,2)) # range expected at most 3 arg
```

Out[151…
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Number System

In [152…
```
#Binary number
bin(25)
```

Out[152…
```
'0b11001'
```

In [153…
```
bin(12)
```

Out[153…
```
'0b1100'
```

In [154…
```
bin(11)
```

Out[154…
```
'0b1011'
```

In [155…
```
oct(12)
```

Out[155…
```
'0o14'
```

In [156…
```
hex(12)
```

Out[156…
```
'0xc'
```

In [ ]:
```
# swap the number
```

In [164…
```
a=10
b=20
```

In [165…
```
a,b = b,a
print(a)
print(b)
```

```
20
10
```

In [166…
```
a = a+b
b = a-b
a = a-b
```

In [167…
```
print(a)
print(b)
```

```
10
20
```

In [ ]:
```
```