Python Data Structures *List* Tuple *Sets* Distionary

In [2]:
```
l = []
```

In [3]:
```
len(l)
```

Out[3]: 0

In [4]:
```
l.append(10)
l
```

Out[4]: [10]

In [5]:
```
len(l)
```

Out[5]: 1

In [6]:
```
l.append(20)
l.append(30)
l.append(40)
l.append(50)
```

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

Out[7]: [10, 20, 30, 40, 50]

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

Out[8]: 5

In [9]:
```
id(l)
```

Out[9]: 2331526083840

In [10]:
```
type(l)
```

Out[10]: list

In [11]:
```
l[:]
l
```

Out[11]: [10, 20, 30, 40, 50]

In [12]:
```
l[0]
```

Out[12]: 10

In [13]:
```
l[3]
```

Out[13]: 40

In [14]:
```
l[-3]
```

Out[14]: 30

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

Out[15]: [10, 20, 30, 40, 50]

In [16]:
```
l1 = l.copy()
l1
```

Out[16]: [10, 20, 30, 40, 50]

In [17]:
```
l == l1
```

Out[17]: True

In [18]:
```
len(l1)
```

Out[18]: 5

In [19]:
```
print(len(l), len(l1))
```

```
5 5
```

In [20]:
```
l1.append(2.3)
l1.append(True)
l1.append(2+3j)
```

In [21]:
```
l1
```

Out[21]: [10, 20, 30, 40, 50, 2.3, True, (2+3j)]

In [22]:
```
l1.append(50)
l1
```

Out[22]: [10, 20, 30, 40, 50, 2.3, True, (2+3j), 50]

In [23]:
```
l1.count(50)
```

Out[23]: 2

In [24]:
```
l1.count(123)
```

Out[24]: 0

In [25]:
```
l1.count(False)
```

Out[25]: 0

In [26]:
```
l2 = l1.copy()
```

In [27]:
```
l2
```

Out[27]: [10, 20, 30, 40, 50, 2.3, True, (2+3j), 50]

In [28]:
```
l2.remove(True)
```

In [29]:
```
l2
```

Out[29]: [10, 20, 30, 40, 50, 2.3, (2+3j), 50]

In [30]:
```
l2.count(50)
```

Out[30]: 2

In [31]:
```
l2.append(123)
```

In [32]:
```
l2
```

Out[32]: [10, 20, 30, 40, 50, 2.3, (2+3j), 50, 123]

In [33]:
```
l2.clear()
```

In [34]:
```
l2
```

Out[34]: []

In [35]:
```
l2.del()
```

```
  Cell In[35], line 1
    l2.del()
       ^
SyntaxError: invalid syntax
```

In [65]:
```
del l2
```

In [67]:
```
l2
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[67], line 1
----> 1 l2

NameError: name 'l2' is not defined
```

In [74]:
```
print(l)
print(l1)
```

```
[10, 20, 30, 40, 50, [1, 2, 3, 'hi']]
[10, 20, 30, 40, 50, 2.3, True, (2+3j), 50]
```

In [76]:
```
l.append([1,2,3, hi])
l
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[76], line 1
----> 1 l.append([1,2,3, hi])
      2 l

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

In [78]:
```
l.append([1,2,3, 'hi'])
l
```

Out[78]: [10, 20, 30, 40, 50, [1, 2, 3, 'hi'], [1, 2, 3, 'hi']]

In [80]:
```
l[4]
```

Out[80]: 50

In [82]:
```
l.pop()
```

Out[82]: [1, 2, 3, 'hi']

In [84]:
```
l
```

Out[84]: [10, 20, 30, 40, 50, [1, 2, 3, 'hi']]

In [86]:
```
l.insert(35, 3)
```

In [88]:
```
l
```

Out[88]: [10, 20, 30, 40, 50, [1, 2, 3, 'hi'], 3]

In [90]:
```
l.insert(3,35)
l
```

Out[90]: [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3]

In [92]:
```
l.pop()
```

Out[92]: 3

In [94]:
```
l
```

Out[94]: [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi']]

In [96]:
```
l.index(35)
```

Out[96]: 3

In [98]:
```
l.append(-1)
```

In [100…
```
l
```

Out[100… [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], -1]

In [102…
```
l.append(3.5)
l.append(1.2)
```

In [104…
```
l
```

Out[104… [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], -1, 3.5, 1.2]

In [121…
```
l
```

Out[121… [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]

In [224…
```
for i in l: #Enumarate
    print(l)
```

```
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
```

In [212…

```
l3 = []
l3
```

Out[212…

```
[]
```

In [214…

```
l3.append(10)
l3.append(8)
```

In [216…

```
l3
```

Out[216…

```
[10, 8]
```

In [218…

```
l3.sort()
l3
```

Out[218…

```
[8, 10]
```

In [ ]:

```

```

In [208…

```
l.sort()
l
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[208], line 1
----> 1 l.sort()
      2 l

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

In [110…

```
l.sort(reverse=True)
l
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[110], line 1
----> 1 l.sort(reverse=True)
      2 l

TypeError: '<' not supported between instances of 'list' and 'float'
```

In [112…

```
l.append(12)
l
```

Out[112…

```
[10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]
```

List Slicling

In [125…
```
l1
```
Out[125… [10, 20, 30, 40, 50, 2.3, True, (2+3j), 50]

In [127…
```
l[:]
```
Out[127… [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]

In [129…
```
l[:5]
```
Out[129… [10, 20, 30, 35, 40]

In [131…
```
l[2:10:3]
```
Out[131… [30, 50, 1.2]

In [133…
```
l[::-1]
```
Out[133… [12, -1, 1.2, 3.5, [1, 2, 3, 'hi'], 50, 40, 35, 30, 20, 10]

In [135…
```
l[::-2]
```
Out[135… [12, 1.2, [1, 2, 3, 'hi'], 40, 30, 10]

In [137…
```
l
```
Out[137… [10, 20, 30, 35, 40, 50, [1, 2, 3, 'hi'], 3.5, 1.2, -1, 12]

In [139…
```
l1.append('nit')
```
In [141…
```
l1
```
Out[141… [10, 20, 30, 40, 50, 2.3, True, (2+3j), 50, 'nit']

In [143…
```
l1[9]
```
Out[143… 'nit'

Tuple

In [146…
```
t = ()
```
In [148…
```
print(type(t))
```
```
<class 'tuple'>
```
In [150…
```
t1 = (10,20,30,40,40)
```
In [152…
```
t1
```

Out[152… (10, 20, 30, 40, 40)

In [154…
```
len(t1)
```

Out[154… 5

In [156…
```
t1.count(10)
```

Out[156… 1

In [158…
```
t1.count(40)
```

Out[158… 2

In [160…
```
t1
```

Out[160… (10, 20, 30, 40, 40)

In [162…
```
t1.index(25)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[162], line 1
----> 1 t1.index(25)

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

In [164…
```
t1.index(20)
```

Out[164… 1

In [166…
```
t1.index(40)
```

Out[166… 3

In [168…
```
t1(1) = 15   # tuples are immutable
t1
```

```
  Cell In[168], line 1
    t1(1) = 15   # tuples are immutable
    ^
SyntaxError: cannot assign to function call here. Maybe you meant '==' instead of
'='?
```

In [170…
```
t1
```

Out[170… (10, 20, 30, 40, 40)

In [172…
```
l5 = ['a', 'b', 'c', 'd']  #list are mutable
```

In [174…
```
l5
```

Out[174… ['a', 'b', 'c', 'd']

In [176…
```
l5[1] = 10
```

In [178…
```
l5
```

Out[178…
```
['a', 10, 'c', 'd']
```

In [180…
```
t2 = (100, 3.4, 'mujju', True, 1+2j, [1,2,3])
```

In [182…
```
t2
```

Out[182…
```
(100, 3.4, 'mujju', True, (1+2j), [1, 2, 3])
```

In [184…
```
print(t)
print(t1)
print(t2)
```

```
()
(10, 20, 30, 40, 40)
(100, 3.4, 'mujju', True, (1+2j), [1, 2, 3])
```

In [186…
```
t2
```

Out[186…
```
(100, 3.4, 'mujju', True, (1+2j), [1, 2, 3])
```

In [188…
```
t2[:2]
```

Out[188…
```
(100, 3.4)
```

In [190…
```
t4 = t2*3
t4
```

Out[190…
```
(100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3])
```

In [192…
```
t4[::]
```

Out[192…
```
(100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3])
```

In [194…
```
t4
```

Out[194…
```
(100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3],
 100,
 3.4,
 'mujju',
 True,
 (1+2j),
 [1, 2, 3])
```

In [198…
```
t1
```

Out[198…
```
(10, 20, 30, 40, 40)
```

In [202…
```
t5 = t1*3
t5
```

Out[202…
```
(10, 20, 30, 40, 40, 10, 20, 30, 40, 40, 10, 20, 30, 40, 40)
```

In [204…
```
t5[:5]
```

Out[204…
```
(10, 20, 30, 40, 40)
```

In [206…
```
t5[1:8:2]
```

Out[206…
```
(20, 40, 10, 30)
```

Sets

In [3]:
```
s1 = {}      #Set written by cutly braces
s1
```

Out[3]:
```
{}
```

In [5]:
```
type (s1)
```

Out[5]:
```
dict
```

In [7]:
```
s2 = {1,4,19,12,200,49,35}  #Always sets is execute as sorted
s2
```

Out[7]:
```
{1, 4, 12, 19, 35, 49, 200}
```

In [11]:
```
s3 = {'e','k','s','p','m','a','b'}
s3
```

Out[11]:
```
{'a', 'b', 'e', 'k', 'm', 'p', 's'}
```

In [13]:
```
s4 = {12,'mujju', [1,2,3],(6,4,8), 2,5}
s4
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 s4 = {12,'mujju', [1,2,3],(6,4,8), 2,5}
      2 s4

TypeError: unhashable type: 'list'
```

In [15]:
```
s5 = {15, 4.6, 2+6j, True}
s5
```

Out[15]:
```
{(2+6j), 15, 4.6, True}
```

In [17]:
```
print(s1)
print(s2)
print(s3)
print(s5)
```

```
{}
{1, 49, 35, 4, 19, 200, 12}
{'e', 'p', 'a', 'm', 's', 'b', 'k'}
{True, (2+6j), 4.6, 15}
```

In [19]:
```
s2.add(50)
s2
```

Out[19]:
```
{1, 4, 12, 19, 35, 49, 50, 200}
```

In [21]:
```
s2.add(3)
s2
```

Out[21]: {1, 3, 4, 12, 19, 35, 49, 50, 200}

In [25]:
```
s2[:] #index slicing is not allowed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 s2[:]

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

In [27]:
```
s2[1:5]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 s2[1:5]

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

In [31]:
```
s4 = s5.copy()
s4
```

Out[31]: {(2+6j), 15, 4.6, True}

In [33]:
```
s4
```

Out[33]: {(2+6j), 15, 4.6, True}

In [35]:
```
s4.add(15) #Dublicate value not allowed
```

In [37]:
```
s4
```

Out[37]: {(2+6j), 15, 4.6, True}

In [39]:
```
s5.clear()
s5
```

Out[39]: set()

In [41]:
```
s4
```

Out[41]: {(2+6j), 15, 4.6, True}

In [43]:
```
s4.remove((2+6j))
```

In [45]:
```
s4
```

Out[45]: {True, 4.6, 15}

In [47]:
```
s3
```

Out[47]: {'a', 'b', 'e', 'k', 'm', 'p', 's'}

In [51]:
```
s3.discard('n') # if the value is not available they dont give you bug
```

In [53]:
```
s3
```

Out[53]:
```
{'a', 'b', 'e', 'k', 'm', 'p', 's'}
```

In [55]:
```
s3.remove('n') # if the value is not avalable they give you bug.
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[55], line 1
----> 1 s3.remove('n')

KeyError: 'n'
```

In [57]:
```
s3
```

Out[57]:
```
{'a', 'b', 'e', 'k', 'm', 'p', 's'}
```

In [59]:
```
s3.pop() #Its takes random value and delete it
```

Out[59]:
```
'e'
```

In [61]:
```
s3
```

Out[61]:
```
{'a', 'b', 'k', 'm', 'p', 's'}
```

In [63]:
```
s3.pop()
```

Out[63]:
```
'p'
```

In [65]:
```
s3
```

Out[65]:
```
{'a', 'b', 'k', 'm', 's'}
```

In [67]:
```
s2
```

Out[67]:
```
{1, 3, 4, 12, 19, 35, 49, 50, 200}
```

In [69]:
```
s2.pop(4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[69], line 1
----> 1 s2.pop(4)

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

In [71]:
```
s2.pop()
```

Out[71]:
```
1
```

In [73]:
```
s2
```

Out[73]:
```
{3, 4, 12, 19, 35, 49, 50, 200}
```

In [79]:
```
for i in s2:
    print(i)
```

```
3
4
200
12
19
35
49
50
```

In [81]:
```
for i in enumerate(s2):
    print(i)
```

```
(0, 3)
(1, 4)
(2, 200)
(3, 12)
(4, 19)
(5, 35)
(6, 49)
(7, 50)
```

In [83]:
```
s2
```

Out[83]:
```
{3, 4, 12, 19, 35, 49, 50, 200}
```

In [85]:
```
s3
```

Out[85]:
```
{'a', 'b', 'k', 'm', 's'}
```

In [87]:
```
s3.update(s2)
```

In [89]:
```
s3
```

Out[89]:
```
{12, 19, 200, 3, 35, 4, 49, 50, 'a', 'b', 'k', 'm', 's'}
```

Set opetations

In [1]:
```
s6 = {1,2,3,4,5,6}
s7 = {5,6,7,8}
s8 = {8,9,10,11}
```

In [3]:
```
s6.union(s7)
```

Out[3]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [5]:
```
s6.union(s7,s8)
```

Out[5]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

In [7]:
```
s6 | s7 | s8 #'|' This symbol is called pipe and its function is same as union
```

Out[7]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}
```

In [9]:
```
print(s6)
print(s7)
print(s8)
```

```
{1, 2, 3, 4, 5, 6}
{8, 5, 6, 7}
{8, 9, 10, 11}
```

In [11]:
```
s6.intersection(s7)
```

Out[11]: {5, 6}

In [13]:
```
s6.intersection(s8)
```

Out[13]: set()

In [15]:
```
s6.intersection(s7,s8)
```

Out[15]: set()

In [17]:
```
s7.intersection(s8)
```

Out[17]: {8}

In [21]:
```
s6 & s7  #'&' is a symbol of intersection
```

Out[21]: {5, 6}

In [25]:
```
print(s6)
print(s7)
print(s8)
```

```
{1, 2, 3, 4, 5, 6}
{8, 5, 6, 7}
{8, 9, 10, 11}
```

In [23]:
```
s6.difference(s7)
```

Out[23]: {1, 2, 3, 4}

In [27]:
```
s6 - s7
```

Out[27]: {1, 2, 3, 4}

In [29]:
```
s7 - s8
```

Out[29]: {5, 6, 7}

In [31]:
```
s8 -s7
```

Out[31]: {9, 10, 11}

In [33]:
```
s8 - s6
```

Out[33]: {8, 9, 10, 11}

In [35]:
```
print(s6)
print(s7)
print(s8)
```

```
{1, 2, 3, 4, 5, 6}
{8, 5, 6, 7}
{8, 9, 10, 11}
```

In [39]:
```
s6.symmetric_difference(s7) # Symmetric difference is a elements that remove the
```

Out[39]:
```
{1, 2, 3, 4, 7, 8}
```

In [41]:
```
s7.symmetric_difference(s8)
```

Out[41]:
```
{5, 6, 7, 9, 10, 11}
```

In [45]:
```
s6.symmetric_difference(s8)
```

Out[45]:
```
{1, 2, 3, 4, 5, 6, 8, 9, 10, 11}
```

In [49]:
```
s6 ^ s7 # '^' Symmetric difference
```

Out[49]:
```
{1, 2, 3, 4, 7, 8}
```

-->Superset (parent) -->Subset (Child) -->Disjointset (Neighbour)

In [54]:
```
s9 ={1,2,3,4,5,6,7,8,9}
s10 = {3,4,5,6,7,8,9}
s11 = {10,20,30,40,50}
```

In [56]:
```
s9
```

Out[56]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [58]:
```
s9.issubset(s10)
```

Out[58]:
```
False
```

In [60]:
```
s10.issubset(s9)
```

Out[60]:
```
True
```

In [62]:
```
s10.issubset(s11)
```

Out[62]:
```
False
```

In [64]:
```
s11.issubset(s10)
```

Out[64]:
```
False
```

In [66]:
```
s9.isdisjoint(s10)
```

Out[66]:
```
False
```

In [ ]:
```

```