Python Data Structure:

Data structure is very essential part of Python programming. Using the data structure we can store any kind of data and we can transform as well if needed.

Type of Python Data Structure: ¶

- List
- Tuple
- Set
- Dict

List Data Structure:

- Rules and guideline of List Data Structure:
  - List is a **sequential or indexable** data structure (like string)
  - List is a **mutable** data structure which means we can change the element if needed
  - List elements are seperated by comma
  - List can be created using **long bracket [] and using list function**
  - We can store any kind of information within the list.

In [1]:
```
print("All Function from List: ", [i for i in dir(list) if "__" not in i])
```

```
All Function from List:  ['append', 'clear', 'copy', 'count', 'extend', 'ind
ex', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

Note:

List, Set, Tuple, Dictionary, String etc are iterables (means where we can apply the loop), Numbers are not iterables.

In [2]:
```
lst = []
```

In [3]:
```
type(lst)
```

Out[3]:
```
list
```

In [4]:
```
lst = list()
```

In [5]:
```
lst
```

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

In [6]:
```
1 lst = [1,3,5,12.23,"A","B",True,False]
```

In [7]:
```
1 print(lst)
```

```
[1, 3, 5, 12.23, 'A', 'B', True, False]
```

List is a mutable which we can change, append, remove, delete,insert etc

In [9]:
```
1 a = "KnowledgeHut"
```

In [11]:
```
1 a[-1] = "A"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 a[-1] = "A"

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

In [12]:
```
1 lst
```

Out[12]:
```
[1, 3, 5, 12.23, 'A', 'B', True, False]
```

In [14]:
```
1 lst[-3] = "Bhagat"
```

In [15]:
```
1 lst
```

Out[15]:
```
[1, 3, 5, 12.23, 'A', 'Bhagat', True, False]
```

List Append Function

Using 'append' function we can append an item (single item only) at the end of the list.

In [16]:
```
1 lst
```

Out[16]:
```
[1, 3, 5, 12.23, 'A', 'Bhagat', True, False]
```

In [17]:
```
1 lst.append("Sohan")
```

In [18]:
```
1 lst
```

Out[18]:
```
[1, 3, 5, 12.23, 'A', 'Bhagat', True, False, 'Sohan']
```

In [20]:
```
1 newlst = ["Modi","Biden","Jocinda"]
```

In [21]:
```
1 lst.append(newlst)
```

In [22]:
```
1 print(lst)
```

```
[1, 3, 5, 12.23, 'A', 'Bhagat', True, False, 'Sohan', ['Modi', 'Biden', 'Joc
inda']]
```

In [23]:
```
1 lst.append(100)
```

In [24]:
```
1 print(lst)
```

```
[1, 3, 5, 12.23, 'A', 'Bhagat', True, False, 'Sohan', ['Modi', 'Biden', 'Joc
inda'], 100]
```

In [27]:
```
1 lst.append?
```

In [26]:
```
1 print(lst.append.__doc__)
```

```
Append object to the end of the list.
```

List Extend

Using this function we can extend a existing list.

Extend function can take as input iterable only (always)

In [28]:
```
1 lst = [1,3,5,7]
```

In [29]:
```
1 lst
```

Out[29]:
```
[1, 3, 5, 7]
```

In [30]:
```
1 lst.extend(100)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 lst.extend(100)

TypeError: 'int' object is not iterable
```

In [31]:
```
1 lst.extend([100])
```

In [32]:
```
1 lst
```

Out[32]: [1, 3, 5, 7, 100]

In [34]:
```
1 newlst
```

Out[34]: ['Modi', 'Biden', 'Jocinda']

In [35]:
```
1 lst.extend(newlst)
```

In [36]:
```
1 print(lst)
```

```
[1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda']
```

In [37]:
```
1 lst.extend("ARORA")
```

In [38]:
```
1 print(lst)
```

```
[1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda', 'A', 'R', 'O', 'R', 'A']
```

Difference between append and extend in list?

List Count

In [39]:
```
1 lst
```

Out[39]: [1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda', 'A', 'R', 'O', 'R', 'A']

In [40]:
```
1 lst.count("A")
```

Out[40]: 2

In [41]:
```
1 lst.count("R")
```

Out[41]: 2

In [42]:
```
1 lst.count("Modi")
```

Out[42]: 1

In [43]:
```
1 lst.count("X")
```

Out[43]: 0

In [45]:
```
1 lst.append(5)
```

In [46]:
```
1 print(lst)
```

```
[1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda', 'A', 'R', 'O', 'R', 'A', 5, 5]
```

In [47]:
```
1 lst.count(5)
```

Out[47]: 3

List Index Function

Using this function we can find the index of an element.

In [48]:
```
1 lst.index(100)
```

Out[48]: 4

In [49]:
```
1 lst.index("A")
```

Out[49]: 8

In [50]:
```
1 lst.index("A")
```

Out[50]: 8

In [51]:
```
1 lst.index("A", lst.index("A") + 1)
```

Out[51]: 12

In [52]:
```
1 lst[8]
```

Out[52]: 'A'

In [53]:
```
1 lst[12]
```

Out[53]: 'A'

In [54]:
```
1 lst[4]
```

Out[54]: 100

In [55]:
```
1 lst
```

Out[55]: [1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda', 'A', 'R', 'O', 'R', 'A', 5, 5]

List Insert Function

Using this function we can insert an object or item at the specified index position.

In [56]:
```
1 lst = [1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda']
```

In [57]:
```
1 lst
```

Out[57]: [1, 3, 5, 7, 100, 'Modi', 'Biden', 'Jocinda']

In [58]:
```
1 lst.insert(4, "Manya")
```

In [59]:
```
1 lst
```

Out[59]: [1, 3, 5, 7, 'Manya', 100, 'Modi', 'Biden', 'Jocinda']

In [61]:
```
1 lst.index("Biden") + 1
```

Out[61]: 8

In [62]:
```
1 lst.insert(lst.index("Biden") + 1,"Abhishek")
```

In [63]:
```
1 lst
```

Out[63]: [1, 3, 5, 7, 'Manya', 100, 'Modi', 'Biden', 'Abhishek', 'Jocinda']

In [64]:
```
1 lst.insert(-1,"A")
```

In [65]:
```
1 lst
```

Out[65]: [1, 3, 5, 7, 'Manya', 100, 'Modi', 'Biden', 'Abhishek', 'A', 'Jocinda']

In [66]:
```
1 lst.insert(-5,"X")
```

In [67]:
```
1 lst
```

Out[67]: [1, 3, 5, 7, 'Manya', 100, 'X', 'Modi', 'Biden', 'Abhishek', 'A', 'Jocinda']

List Clear method

Using this list we can clear the element.

In [68]:
```
1 newlst
```

Out[68]: ['Modi', 'Biden', 'Jocinda']

In [69]:
```
1 newlst.clear()
```

In [70]:
```
1 newlst
```

Out[70]: []

In [71]:
```
1 newlst.extend(lst)
```

In [72]:
```
1 newlst
```

Out[72]: [1, 3, 5, 7, 'Manya', 100, 'X', 'Modi', 'Biden', 'Abhishek', 'A', 'Jocinda']

List Copy Method

make a copy of existing list.

In [73]:
```
1 lstOfNumber = [11,33,55,77,88,99,98]
```

In [74]:
```
1 numbers = lstOfNumber.copy()
```

In [75]:
```
1 numbers
```

Out[75]:
```
[11, 33, 55, 77, 88, 99, 98]
```

In [76]:
```
1 lstOfNumber
```

Out[76]:
```
[11, 33, 55, 77, 88, 99, 98]
```

List Sort Function

Using this function we can sort the list either in asc or desc (by default: asc).

Note: This function will change the sequence of your original list as well.

In [77]:
```
1 numbers = [11,66,10,5,23,10,30,-40,12,30,7]
```

In [78]:
```
1 numbers
```

Out[78]:
```
[11, 66, 10, 5, 23, 10, 30, -40, 12, 30, 7]
```

In [81]:
```
1 numbers.sort(reverse=False)
```

In [82]:
```
1 numbers
```

Out[82]:
```
[-40, 5, 7, 10, 10, 11, 12, 23, 30, 30, 66]
```

In [83]:
```
1 numbers.sort(reverse=True)
```

In [84]:
```
1 numbers
```

Out[84]:
```
[66, 30, 30, 23, 12, 11, 10, 10, 7, 5, -40]
```

In [85]:
```
1 lst = [10,3,5,102,"A"]
```

In [86]:
```
1 lst.sort()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[86], line 1
----> 1 lst.sort()

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

In [87]:
```
1 lstOfChar = ["A","B","H","A","X","O","N","D"]
```

In [88]:
```
1 lstOfChar.sort()
```

In [89]:
```
1 lstOfChar
```

Out[89]:
```
['A', 'A', 'B', 'D', 'H', 'N', 'O', 'X']
```

In [90]:
```
1 lstOfChar.sort(reverse=True)
```

In [91]:
```
1 lstOfChar
```

Out[91]:
```
['X', 'O', 'N', 'H', 'D', 'B', 'A', 'A']
```

List Reverse

In [93]:
```
1 numbers = [11,66,10,5,23,10,30,-40,12,30,7]
```

In [94]:
```
1 numbers
```

Out[94]:
```
[11, 66, 10, 5, 23, 10, 30, -40, 12, 30, 7]
```

In [95]:
```
1 numbers.reverse()
```

In [96]:
```
1 numbers
```

Out[96]:
```
[7, 30, 12, -40, 30, 10, 23, 5, 10, 66, 11]
```

In [97]:
```
1 lstOfChar = ["A","B","H","A","X","O","N","D"]
```

In [98]:
```
1 lstOfChar.reverse()
```

In [99]:
```
1 lstOfChar
```

Out[99]:
```
['D', 'N', 'O', 'X', 'A', 'H', 'B', 'A']
```

Removing, Popping, Deleting

- List Remove:

Using remove method we can remove an item based on item's name and it will not be return the removed item.

- List Pop:

Using pop method we can pop an item based on item's index, if you are not passing an index by default this function will take last index (-1) and this function always return the popped item.

In [100]:
```
1 lstOfChar
```

Out[100]:
```
['D', 'N', 'O', 'X', 'A', 'H', 'B', 'A']
```

In [101]:
```
1 lstOfChar.remove("X")
```

In [103]:
```
1 lstOfChar
```

Out[103]:
```
['D', 'N', 'O', 'A', 'H', 'B', 'A']
```

In [107]:
```
1 lstOfChar.append("X")
```

In [104]:
```
1 lstOfChar.remove("x")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[104], line 1
----> 1 lstOfChar.remove("x")

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

In [106]:
```
1 lstOfChar
```

Out[106]:
```
['D', 'N', 'O', 'A', 'H', 'B', 'A', 'X']
```

In [108]:
```
1 lstOfChar.insert(3,"X")
```

In [109]:
```
1 lstOfChar
```

Out[109]:
```
['D', 'N', 'O', 'X', 'A', 'H', 'B', 'A', 'X', 'X']
```

In [110]:
```
1 lstOfChar.remove("X")
```

In [111]:
```
1 lstOfChar
```

Out[111]:
```
['D', 'N', 'O', 'A', 'H', 'B', 'A', 'X', 'X']
```

In [112]:
```
# pop : based on item's index
```

In [113]:
```
lstOfChar
```

Out[113]:
```
['D', 'N', 'O', 'A', 'H', 'B', 'A', 'X', 'X']
```

In [114]:
```
lstOfChar.pop()
```

Out[114]:
```
'X'
```

In [115]:
```
lstOfChar
```

Out[115]:
```
['D', 'N', 'O', 'A', 'H', 'B', 'A', 'X']
```

In [116]:
```
lstOfChar.pop(3)
```

Out[116]:
```
'A'
```

In [117]:
```
lstOfChar
```

Out[117]:
```
['D', 'N', 'O', 'H', 'B', 'A', 'X']
```

Del

Del is a keyword, using del we can delete an item or whole list as well.

Item can be deleted based on item's index and does not return the delete item.

In [118]:
```
lstOfChar
```

Out[118]:
```
['D', 'N', 'O', 'H', 'B', 'A', 'X']
```

In [120]:
```
del lstOfChar[-3]
```

In [121]:
```
lstOfChar
```

Out[121]:
```
['D', 'N', 'O', 'H', 'A', 'X']
```

In [122]:
```
del lstOfChar
```

In [123]:
```
1 lstOfChar
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[123], line 1
----> 1 lstOfChar

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

Python Sorted Function

using this function we can sort the item in asc or desc, but this function always return an output in the form of list.

In [124]:
```
1 a = "AbhishekSaraswat"
```

In [126]:
```
1 print(sorted(a))
```

```
['A', 'S', 'a', 'a', 'a', 'b', 'e', 'h', 'h', 'i', 'k', 'r', 's', 's', 't',
'w']
```

In [127]:
```
1 print(sorted(a, reverse=True))
```

```
['w', 't', 's', 's', 'r', 'k', 'i', 'h', 'h', 'e', 'b', 'a', 'a', 'a', 'S',
'A']
```

In [128]:
```
1 lst = [1,5,8,0,10,3,9,3,1,5,2]
```

In [129]:
```
1 new_lst = sorted(lst)
```

In [130]:
```
1 new_lst
```

Out[130]:
```
[0, 1, 1, 2, 3, 3, 5, 5, 8, 9, 10]
```

In [131]:
```
1 lst
```

Out[131]:
```
[1, 5, 8, 0, 10, 3, 9, 3, 1, 5, 2]
```

In [132]:
```
1 lst = [1,3,5,7]
```

In [133]:
```
1 lst * 2
```

Out[133]:
```
[1, 3, 5, 7, 1, 3, 5, 7]
```

In [134]:
```
lst * 5
```

Out[134]:
```
[1, 3, 5, 7, 1, 3, 5, 7, 1, 3, 5, 7, 1, 3, 5, 7, 1, 3, 5, 7]
```

In [138]:
```
lst
```

Out[138]:
```
[1, 3, 5, 7]
```

In [139]:
```
for i in lst:
    print(i*2)
```

```
2
6
10
14
```

In [ ]:
```

```