Python List

A list in Python is used to store the sequence of various types of data. Python lists are mutable type its mean we can modify its element after it created. However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.

A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and enclosed with the square brackets [].

Example1

```
L1 = ["John", 102, "USA"]
L2 = [1, 2, 3, 4, 5, 6]
print(type(L1))
print(type(L2))
```

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

Example2

```
emp = ["John", 102, "USA"]
Dep1 = ["CS",10]
Dep2 = ["IT",11]
HOD_CS = [10,"Mr. Holding"]
HOD_IT = [11, "Mr. Bewon"]
print("printing employee data...")
print("Name : %s, ID: %d, Country: %s"%(emp[0],emp[1],emp[2]))
print("printing departments...")
print("Department 1:\nName: %s, ID: %d\nDepartment 2:\nName: %s
print("HOD Details ....")
print("CS HOD Name: %s, Id: %d"%(HOD_CS[1],HOD_CS[0]))
print("IT HOD Name: %s, Id: %d"%(HOD_IT[1],HOD_IT[0]))
print(type(emp),type(Dep1),type(Dep2),type(HOD_CS),type(HOD_IT)
```

```
printing employee data...
Name : John, ID: 102, Country: USA
printing departments...
Department 1:
Name: CS, ID: 11
Department 2:
```

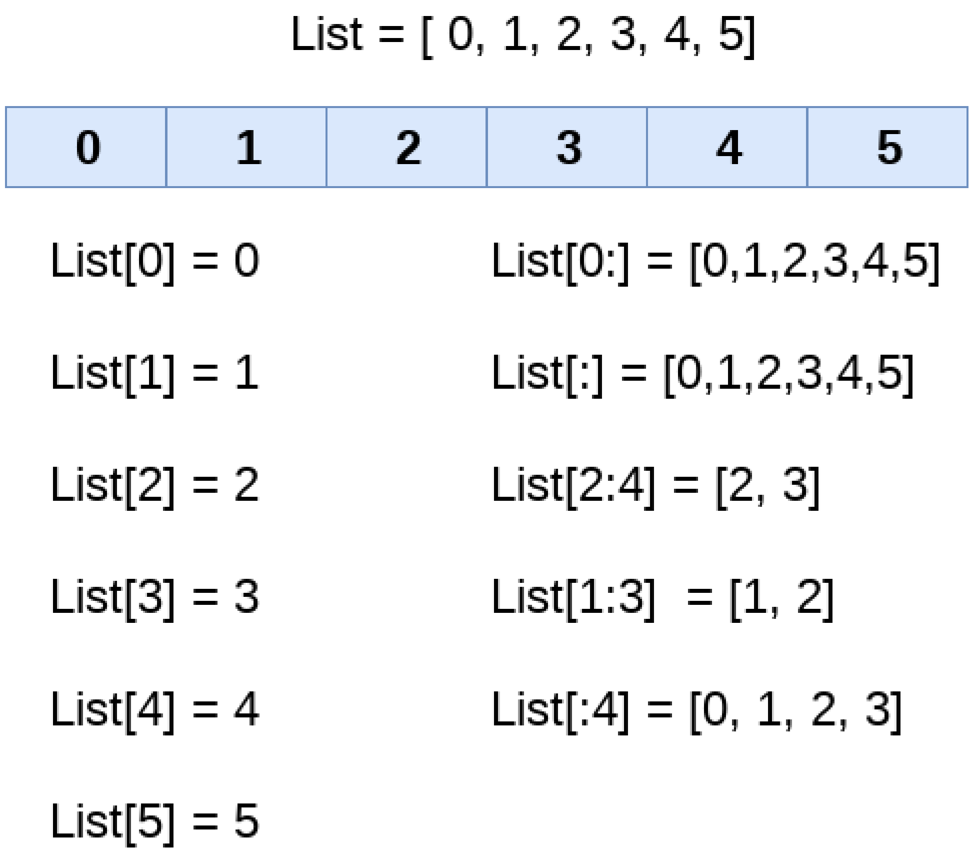
```
Name: IT, ID: 11
HOD Details ....
CS HOD Name: Mr. Holding, Id: 10
IT HOD Name: Mr. Bewon, Id: 11
<class 'list'> <class 'list'> <class 'list'> <class 'list'> <class 'list'>
```

Example3

```
emp = ["John", 102, "USA"]
Dep1 = ["CS",10]
Dep2 = ["IT",11]
HOD_CS = [10,"Mr. Holding"]
HOD_IT = [11, "Mr. Bewon"]
print("printing employee data...")
print((emp))
```

```
printing employee data...
['John', 102, 'USA']
```

List indexing and splitting



```
List[0] = 0        List[0:] = [0,1,2,3,4,5]
List[1] = 1        List[:] = [0,1,2,3,4,5]
List[2] = 2        List[2:4] = [2, 3]
List[3] = 3        List[1:3]  = [1, 2]
List[4] = 4        List[:4] = [0, 1, 2, 3]
List[5] = 5
```

The indexing is processed in the same way as it happens with the strings. The elements of the list can be accessed by using the slice operator [].

The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on.

Example1

```
list = [1,2,3,4,5,6,7]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
# Slicing the elements
print(list[0:6])
# By default the index value is 0 so its starts from the 0th el
print(list[:])
print(list[2:5])
print(list[1:6:2])
```

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

Python provides the flexibility to use the negative indexing also

Example2

```
list = [1,2,3,4,5]
print(list[-1])
print(list[-3:])
print(list[-3:-1])
```

```
5
[3, 4, 5]
[3, 4]
```

Updating List values

Example1

```
ist = [1, 2, 3, 4, 5, 6]
print(list)
```

```
# It will assign value to the value to the second index
list[2] = 10
print(list)
# Adding multiple-element
list[1:3] = [89, 78]
print(list)
# It will add value at the end of the list
list[-1] = 25
print(list)
```

```
[1, 2, 3, 4, 5]
[1, 2, 10, 4, 5]
[1, 89, 78, 4, 5]
[1, 89, 78, 4, 25]
```

Example2

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

```
['apple', 'banana', 'watermelon', 'cherry']
```

Example3

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
```

Example4

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]

thislist.extend(tropical)

print(thislist)
```

```
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Remove Specified Item

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

```
['apple', 'cherry']
```

Remove Specified Index

The pop() method removes the specified index.

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

```
['apple', 'cherry']
```

Remove the last item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

```
['apple', 'banana']
```

The del keyword can also delete the list completely.

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Clear the List

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

```
[]
```

List methods

len() method used to find the length of the list

```
thislist=["apple","banana","cherry"]
print(len(thislist))
```

```
3
```

count method is used to the count number of elements with the specified value

```
thislist=["apple","banana","cherry"]
thislist.count("apple")
```

```
1
```

index method is used to return index value of spcified item

```
thislist=["apple","banana","cherry"]
thislist.index("apple")
```

```
0
```

reverse () is used to reverese the list

```
thislist=["apple","banana","cherry"]
thislist.reverse()
print(thislist)
```

```
['cherry', 'banana', 'apple']
```

copy method is used to make copy of the list

```
new_list = thislist.copy()
```

```
print(new_list)
```

```
['apple', 'banana', 'cherry']
```

extend method is used to add the element at the end of the current list

```
list1 = ["apple","banana","cherry"]
list2 = [1,2,3]
list1.extend(list2)
print(list1)
```

difference between extend and insert method

```
list4=[10.2,45.6]
list1.insert(3,list4)
print(list1)
```

```
['apple', 'banana', 'cherry', [10.2, 45.6], [10.2, 45.6], 1, 2, 3]
```

Tuble

Tuples are used to store hetrogeneous collection of comma-seperated values (items) between round brackets in a single variable.

A tuple is a collection which is ordered and unchangeable.

It is ordered and immutable(Not Changeable)

It allows dublicate values

```
tuble_1=("welcome",1,12.3,True,1+6j)
print(tuble_1)
```

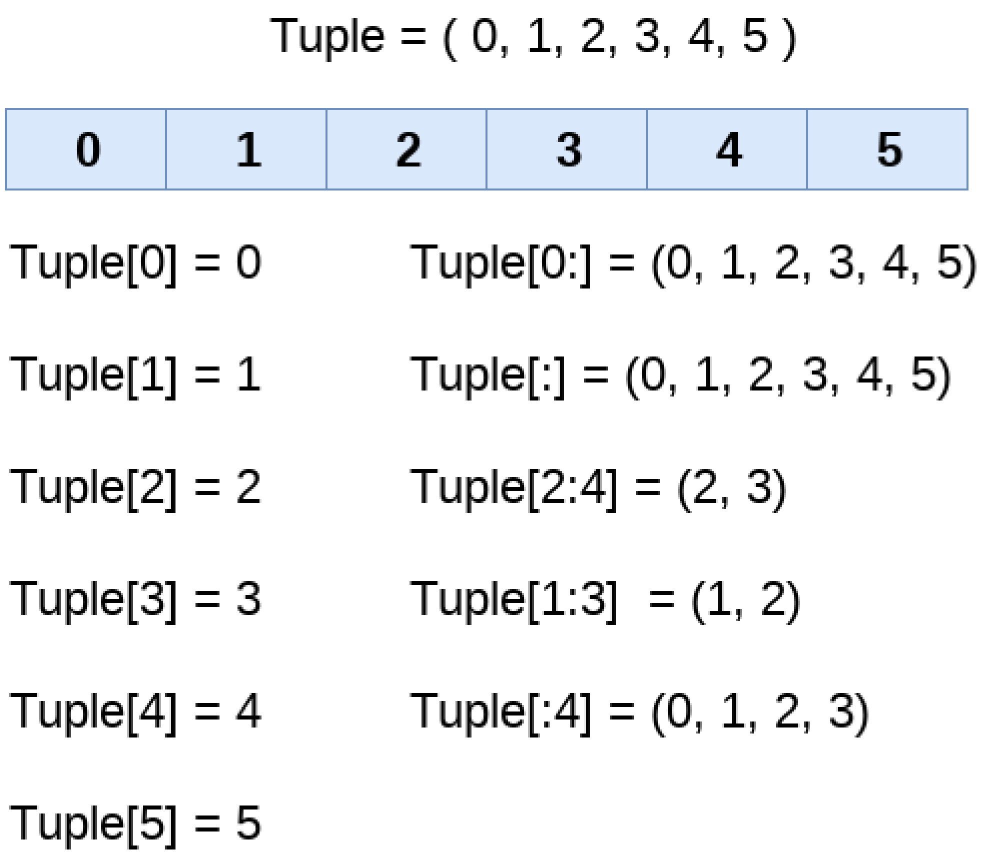
```
('welcome', 1, 12.3, True, (1+6j))
```

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

```
thistuple = ("apple", "banana", "cherry","apple")
thistuple = ("apple", "banana", "cherry", "apple")
print(thistuple)
```

```
('apple', 'banana', 'cherry', 'apple')
```



Accessing the tuple elements

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[0])
```

```
apple
```

Negative indexing

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

```
cherry
```

```
thistuple = ("apple", "banana", "cherry","orange","kiwi","mango
print(thistuple[2:5])
```

```
('cherry', 'orange', 'kiwi')
```

```python
thistuple = ("apple", "banana", "cherry","orange","kiwi","mango
print(thistuple[:-2])
```

```
('apple', 'banana', 'cherry', 'orange', 'kiwi')
```

```python
thistuple[2]="papaya"
print(thistuple)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-b2705f6c25cb> in <module>()
----> 1 thistuple[2]="papaya"
      2 print(thistuple)

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

SEARCH STACK OVERFLOW

Tuple Methods

```python
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

count the number of elements in the tuple

```python
thistuple = ("apple", "banana", "cherry")
print(len(thistuple ))
```

```
3
```

To print the index of the edlement

```python
thistuple = ("apple", "banana", "cherry")
thistuple.index("apple")
```

```
0
```

count the occurance of the item

```
thistuple = ("apple", "banana", "cherry","banana")
thistuple.count("banana")
```

```
2
```

```
thistuple.append("papaya")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-23-c229d3ad993b> in <module>()
----> 1 thistuple.append("papaya")

AttributeError: 'tuple' object has no attribute 'append'
```

SEARCH STACK OVERFLOW

```
thistuple.remove("apple")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-25-90aad26e1865> in <module>()
----> 1 thistuple.remove("apple")

AttributeError: 'tuple' object has no attribute 'remove'
```

SEARCH STACK OVERFLOW

```
del(thistuple)
```

```
print(thistuple)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-f6a69648c0ad> in <module>()
----> 1 print(thistuple)

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

SEARCH STACK OVERFLOW

Sets

Sets are used to store multiple items in a single variable.

A set is a collection which is both unordered and unindexed.

Sets are written with curly brackets.

Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable Sets are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can add new items.

Duplicates Not Allowed Sets cannot have two items with the same value.

Example1

```
thisset = {"apple", "banana", "cherry"}

print(thisset)
```

```
{'banana', 'cherry', 'apple'}
```

Example2

Duplicate values will be ignored:

```
thisset = {"apple", "banana", "cherry", "apple"}

print(thisset)
```

```
{'banana', 'cherry', 'apple'}
```

Access Items

You cannot access items in a set by referring to an index or a key.

But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

```
banana
cherry
apple
```

Check if "banana" is present in the set:

```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

```
True
```

Join Two Sets

There are several ways to join two or more sets in Python.

You can use the union() method that returns a new set containing all items from both sets, or the update() method that inserts all the items from one set into another:

Example1

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3= set1.union(set2)
print(set1)
```

```
{'c', 'b', 'a'}
```

Example2

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

```
{1, 2, 'b', 3, 'a', 'c'}
```

Set Methods

add() method

Once a set is created, you cannot change its items, but you can add new items.

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

```
{'banana', 'cherry', 'orange', 'apple'}
```

Remove () method

To remove an item in a set, use the remove(), or the discard() method.

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

```
{'cherry', 'apple'}
```

len() method is used to find the length of the set

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

```
3
```

The del () method completely delete the set completely

```
del(thisset)
```

Dictionary

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered*, changeable and does not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values:

```
Example
```

Example

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

```
thisdict =  {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
print(x)
```

```
Mustang
```

Change Values

You can change the value of a specific item by referring to its key name:

```
thisdict = {
  "brand": "Ford"
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

Copy a Dictionary

There are ways to make a copy, one way is to use the built-in Dictionary method copy().

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Add the items

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict["color"]="red"
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Update the value of existing element

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict["color"]="blue"
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'blue'}
```

Dictionary method

To print the key value

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict.keys()
```

```
dict_keys(['brand', 'model', 'year'])
```

To print the item values

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict.values()
```

```
dict_values(['Ford', 'Mustang', 1964])
```

pop() is used to remove the item with key value

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict.pop("year")
```

```
1964
```

```
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang'}
```

To Clear the items or values

```
thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}
thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}
thisdict.clear()
print(thisdict)
```

✓ 0s completed at 4:05 PM