List

01.We can create list using square brackets [ ]
02.In List we can store different types of data
03.In List Insertion Ordered is Preserved / Ordered Elements
04.When we create a List python will create an Array like data
05.List is mutable we can modify it once we created

06.In List immutable objects are not duplicated just the memory is shared among the objects (Memory Allocation in List)

```
# Memory allocation of elements/items
l1 = [1,2,[3,1]]
print(id(l1[0])) # 1079486605616
print(id(l1[1])) # 1079486605648
print(id(l1[2][0])) # 1079486605680
print(id(l1[2][1])) # 1079486605616
```

07.List occupies more space compared to Tuple (Size of object in memory)

```
# __sizeof__() object in memory, in bytes.
import array as arr
a = arr.array('i', [])
l=[]
t=()
print(a.__sizeof__()) # 64
print(l.__sizeof__()) # 40
print(t.__sizeof__()) # 24
```

List

08.List performance is low compared to Tuple (Tuple is faster compared to List)

```
# Convenience function to create Timer object and call timeit method.
# The return value is seconds as a float.
import timeit

l1 = 'l = [1,2,3,4,5,6] '
print(timeit.timeit(l1)) # 0.11081392100000001

t1 = 't = (1,2,3,4,5,6)'
print(timeit.timeit(t1)) # 0.02389385699999999
```

09.List is not secure because of data mutability

10.In List Index Call, Slicing, Data Modifications are allowed

```
# index call
l = [1,2,3,4,5]
print(l[0]) # 1
```

```
#Slicing
#Syntax: If we need portion/part of array we can use Slice.
#string[start/begin : end(-1) : step(default value is 1)]
lst1 = [1, 2, 3, 4, 5, 6, 7, 8]
#       0  1  2  3  4  5  6  7
print(lst1[3:6]) # [4, 5, 6]
```

Arrays:

1. Arrays are homogenous whereas List is heterogeneous.
2. In Array we cannot store same data/similar data
3. In List we can store multiple types data
4. To work with array we need to import a module
   import array as arr
   from array import *

https://docs.python.org/3/library/array.html

Array Example in Python

```python
import array as arr

a = arr.array('i',[1, 2, 3, -1,-2,-3, 0, 1]) # Type Code 'i'
print(a) # array('i', [1, 2, 3, -1, -2, -3, 0, 1])
print(type(a)) # <class 'array.array'>
print(a.typecode) # i
```

Array Example in Java

```java
public class Example01 {
	public static void main(String[] args) {
		int[] a = { 10, 20, 30, 10}; //Initializing Array //we cannot store string here
		System.out.println(a);//[I@7852e922
		System.out.println(a.length);//4
		System.out.println(a[0]); //10
		System.out.println(a[1]); //20
		System.out.println(a[2]); //30
		System.out.println(a[3]); //10
	}
}
```

```
# create list using square brackets
l = [1,2,3,4,5,10.0,50j, True, False, 50.0]
print(l) # [1, 2, 3, 4, 5, 10.0, 50j, True, False, 50.0]
```

```
# find the index of the list
l = [1,2,3,4,5]
#    0,1,2,3,4 index position
print(l[0]) # 1
print(l[1]) # 2
print(l[2]) # 3
# print(l[6]) # IndexError: list index out of range
```

```
# find the length of the list
l = [1,2,3,4,5]
print(len(l)) # 5

# find the length of the nested list
l = [1,2,3,[4,5]]
print(len(l)) # 4

# find the length of string
l = "Hello Python"
print(len(l)) # 12
```

Note:
When we are finding the length of string even it count the whitespaces

How to add the single data at the end of the list?

1. We can the add the data using append() method
2. append() method takes only one argument

```
# append using empty array
l = []
l.append(10)
print(l) # [10]
```

```
# append using list of elements
l = ["NameOne", "NameTwo", "NameThree"]
l.append("NameFour")
print(l) # ['NameOne', 'NameTwo', 'NameThree', 'NameFour']
```

```
# append using nested list
l = ["NameOne", "NameTwo", ["NameThree", "NameFour"]]
l.append("NameFive")
print(l) # ['NameOne', 'NameTwo', ['NameThree', 'NameFour'], 'NameFive']
```

How to add the multiple data at the end of the list?

1. We can the add the multiple data using extend() method
2. extend() method takes iterables of data
3. extend() takes exactly one argument

```
# extend() empty list using iterables
l = []
l.extend("NameOne")
print(l) # ['N', 'a', 'm', 'e', 'O', 'n', 'e']
```

```
# extend() empty list using iterables
l = []
l.extend(["NameOne", "NameTwo"])
print(l) # ['NameOne', 'NameTwo']
```

```
# extend() list using iterables
l = ["NameOne","NameTwo"]
l.extend(["NameThree","NameFour"])
print(l) # ['NameOne', 'NameTwo', 'NameThree', 'NameFour' ]
```

```
# Add the data in the middle of the list using insert() method
l = ["NameOne", "NameTwo", "NameThree", "NameFour"]
l.insert(1,"NameFive") # insert(index, element)
print(l) # ['NameOne', 'NameFive', 'NameTwo', 'NameThree', 'NameFour']
```

```
# Remove the end of the element using pop() method
l = [1,2,3,4,5]
l.pop()
print(l) # [1, 2, 3, 4]

# Remove the element from the middle of list
l = [1,2,3,4,5]
l.pop(1) # pop(index)
print(l) # [1, 3, 4, 5]
```

```
# find the index of the given list
l = [1,2,3,4,5]
print(l.index(2)) # 1 # index(element)
print(l.index(5)) # 4
print(l.index(6)) # ValueError: 6 is not in list
```

```
# remove the elements from the middle of list
l = [1,2,3,4,5]
l.remove(5) # remove(element)
print(l) # [1, 2, 3, 4]
l.remove(6)
print(l) # ValueError: list.remove(x): x not in list
```

```
# return the number of items with same occurrence use count()
lst = [1,2,3,4,5,6,1,2,3,2]
print(lst.count(2)) # 3 # count(element)
print(lst.count(1)) # 2
print(lst.count(6)) # 1
print(lst.count(8)) # 0
```

```
# remove all items from the list use clear()
l = [1,2,3,4,5]
l.clear()
print(l) # []
```

```
# copy() method returns shallow copy
l = ["a","e","i","o","u"]
print(l.copy()) # ['a', 'e', 'i', 'o', 'u']
```

```
If a key function is given, apply it once to each list item and sort them, ascending
or descending, according to their function values.
The reverse flag can be set to sort in descending order.
Syntax: sort(self, /, *, key=None, reverse=False)

# Sort the list in ascending order and return None.
l1 = [1,3,5,2,4]
l1.sort(reverse=False) # reverse=False is optional, by sorting order
print('Ascending Order: ', l1) # Ascending Order:  [1, 2, 3, 4, 5]
```

```
# Sort the list in descending order and return None.
l1 = [1,3,5,2,4]
l1.sort(reverse=True)
print('Descending Order: ', l1) # Ascending Order:  [1, 2, 3, 4, 5]
```

ASCII stands for American Standard Code for Information Interchange.

1. ord() function to convert a character to an integer (ASCII value)
2. This function returns the Unicode code point of that character.
3. Unicode has more than 100,000 characters
4. ASCII only encodes 128 characters

```
# find the ASCII value of the given character
result = 'a'
print(ord(result)) # 97
result = 'e'
print(ord(result)) # 101
result = 'i'
print(ord(result)) # 105
result = 'o'
print(ord(result)) # 111
result = 'u'
print(ord(result)) # 117
```

```
chr() function get characters from their corresponding ASCII values
result = chr(97)
print(result) # a
result = chr(101)
print(result) # e
result = chr(105)
print(result) # i
result = chr(111)
print(result) # o
result = chr(117)
print(result) # u
```

```
# Sort the list in ascending order and return None.
l1 = ["a","i","o","u","e"]
l1.sort(reverse=False) # Sorting Order
print('Ascending Order: ', l1) # Ascending Order:  ['a', 'e', 'i', 'o', 'u']
```

```
# Sort the list in Descending order and return None.
l1 = ["a","i","o","u","e"]
l1.sort(reverse=True)
print('Descending Order: ', l1) # Descending Order:  ['u', 'o', 'i', 'e', 'a']
```

```
# adding lists
l1 = [1,2,3,4,5]
l2 = [1,2,3,4,5]
print(l1+l2) # [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

# adding strings
l1 = ["Hello"]
l2 = ["Python"]
print(l1+l2) # ['Hello', 'Python']
```

```
# Swap list values
a,b,c,d = [1,2,3,4]
print(a,d,c,b) # 1 4 3 2
```

```
#Slicing
#Syntax: If we need portion/part of array we can use Slice.
#string[start/begin : end(-1) : step(default value is 1)]
lst1 = [1, 2, 3, 4, 5, 6, 7, 8]
#       0  1  2  3  4  5  6  7
print(lst1[3:6]) # [4, 5, 6]

# end Optional
lst3 = [1, 2, 3, 4, 5, 6, 7, 8]
#       0  1  2  3  4  5  6  7
print(lst3[3:]) # [4, 5, 6, 7, 8]

# No begin No end --> We get total String
lst4 = [1, 2, 3, 4, 5, 6, 7, 8]
#       0  1  2  3  4  5  6  7
print(lst4[ : ]) # [1, 2, 3, 4, 5, 6, 7, 8]
```

```
#Slicing Negative Values
#string[start/begin : end(-1) : step(default value is 1)]
lst1 = [1, 2, 3, 4, 5, 6, 7, 8]
#      -8 -7 -6 -5 -4 -3 -2  -1
print(lst1[-3:-1]) # [6, 7] #start : end

print(lst1[-3:-7]) # [] #start : end

print(lst1[-3:-4]) # [] #start : end

# Note: -7 is in left side, but it should move right side
```

```
#Slicing Step Over
#string[start/begin : end(-1) : step(default value is 1)]
lst1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
#         0  1  2  3  4  5  6  7, 8, 9

print(lst1[0:9:1]) # [1, 2, 3, 4, 5, 6, 7, 8, 9]

print(lst1[0:9:2]) # [1, 3, 5, 7, 9]

print(lst1[0:9:4]) # [1, 5, 9]
```

```
#Slicing Negative Values
#string[start/begin : end(-1) : step(default value 1)

lst = [1, 2, 3, 4, 5, 6, 7 ,8]
#      0, 1, 2, 3, 4, 5, 6, 7
#     -8,-7,-6,-5,-4,-3,-2,-1

# Start is opt, end is opt, step over is -1
print(lst[ : : -1])  # [8, 7, 6, 5, 4, 3, 2, 1]

print(lst[5 : 2 : -1])  # [6, 5, 4]

# Note when step over is -1 index starts backward
```

```
# extend list, tuple, range
lst = [1, 2]
lst.extend([3, 4, 5]) # list
lst.extend((6, 7, 8)) # tuple
lst.extend(range(9, 15)) # range()
print(lst) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
# deep copy --> In the deep copy the original list will not be reflected
import copy
list1 = [[1,2], [3,4]]
list2 = copy.deepcopy(list1)
print(list2) # [[1, 2], [3, 4]]
list2[0][0] = 10
print(list2) # [[10, 2], [3, 4]]
print(list1) # [[1, 2], [3, 4]]
```

```
# Shallow copy --> In shallow copy the original list also reflected if changes done
import copy
list1 = [[1,2], [3,4]]
list2 = copy.copy(list1)
print(list2) # [[1, 2], [3, 4]]
print(list1) # [[1, 2], [3, 4]]
list2[0][0] = 10
print(list2) # [[10, 2], [3, 4]]
print(list1) # [[10, 2], [3, 4]]
```

```
# Modify the list values in between
l1 = [10,20,30,40,50,60,70,80,90]
#      0  1   2    3  4    5    6   7   8

l1[0:2] = [100,200]
print(l1) # [100, 200, 30, 40, 50, 60, 70, 80, 90]

l1 = ["NameOne", "NameTwo", "NameThree", "NameFour"]
print(l1[2:3]) # ['NameThree']
l1[2:3] = ["NameFive"]
print(l1) # ['NameOne', 'NameTwo', 'NameFive', 'NameFour']
```

```
# Get random elements
l = [10,20,30,40,50,60,70,80]
#   -8, -7  -6  -5  -4  -3  -2  -1
print(l[-1], l[-2], l[-5]) # 80 70 40

l = [10,20,30,40,50,60,70,80]
#    0   1   2   3   4   5   6   7
print(l[2], l[5]) # 30 60
```

```
# index method using start and end
l1 = [10,20,30,40,50,30]
#       0  1   2   3   4   5
print(l1.index(30,3,6)) # index(element, start, end)

5
```

```
# Flattering
l1 = [[1,2,3], [4,5,6]]
l2 = []
for i in l1:
    for j in i:
        l2.append(j)
print(l2) # [1, 2, 3, 4, 5, 6]
```

1. **What is a list in Python?**
   - **Answer:** A list is a built-in data type in Python that represents a collection of ordered and mutable elements. Lists are defined by square brackets `[ ]` and can contain items of different data types.
2. **How do you create an empty list in Python?**
   - **Answer:** You can create an empty list by using square brackets: `my_list = []` or by using the `list()` constructor: `my_list = list()`.
3. **Explain the difference between append() and extend() methods for lists.**
   - **Answer:** The `append()` method adds a single element to the end of a list, while the `extend()` method adds elements from an iterable (e.g., another list) to the end of the list.
4. **How do you access elements in a list?**
   - **Answer:** You can access elements in a list using indexing. Indexing starts from 0. For example, `my_list[0]` would access the first element.
5. **What is the difference between remove() and pop() methods?**
   - **Answer:** The `remove()` method removes the first occurrence of a specified value from the list, while the `pop()` method removes and returns an element at a specified index (default is the last element).
6. **How can you reverse a list in-place?**
   - **Answer:** You can reverse a list in-place using the `reverse()` method. For example: `my_list.reverse()`.
7. **Explain list comprehension in Python.**
   - **Answer:** List comprehension is a concise way to create lists. It consists of an expression followed by a `for` loop inside square brackets. For example: `[x**2 for x in range(5)]`.
8. **What is the difference between sorted() and sort() for a list?**
   - **Answer:** `sorted()` is a built-in function that returns a new sorted list from the elements of any iterable, while the `sort()` method is a list method that sorts the elements of a list in-place.
9. **How do you find the length of a list?**
   - **Answer:** The `len()` function is used to find the length (number of elements) of a list. For example: `length = len(my_list)`.
10. **Explain the difference between shallow copy and deep copy of a list.**
    - **Answer:** A shallow copy creates a new list but does not create new objects for the elements; it copies references to the objects. A deep copy, on the other hand, creates a new list and recursively copies the objects within the original list, resulting in a completely independent copy.