List in Python (Part-2)

Operations on Lists:

There are three types of Opeartion in List:

1. Arithmatic
2. Membership
3. Loop

In [ ]:
```
# Arithmatic opeartion (+, *)

L1 = [1,2,3,4,5]
L2 = [9,8,7,6,5]

# concatenation/Merge
print(L1 + L2)


print(L1*3)
print(L2*4)
```

```
[1, 2, 3, 4, 5, 9, 8, 7, 6, 5]
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
[9, 8, 7, 6, 5, 9, 8, 7, 6, 5, 9, 8, 7, 6, 5, 9, 8, 7, 6, 5]
```

In [ ]:
```
# membership
L1 = [1,2,3,4,5]
L2 = [1,2,3,4,[5,6]]

print(5 not in L1)
print([5,6] in L2)
```

```
False
True
```

In [ ]:
```
# Loops
L1 = [1,2,3,4,5]
L2 = [1,2,3,4,[5,6]]
L3 = [[[1,2],[3,4]],[[5,6],[7,8]]]

for i in L2:
    print(i)
```

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

List Functions

In [ ]:
```
# len/min/max/sorted
L = [2,1,5,7,0]

print(len(L))
print(max(L))
print(min(L))
print(sorted(L))
```

```
5
7
0
[0, 1, 2, 5, 7]
```

In [ ]:
```
# count
l = [1,2,3,456,67867]

l.count(456)
```

Out[ ]:
```
1
```

In [ ]:
```
# index
l = [3,5,7,9,3,23]

l.index(5)
```

Out[ ]:
```
1
```

In [ ]:
```
# reverse
l = [1,2,3,4,6,78]

l.reverse()
print(l)
```

```
[78, 6, 4, 3, 2, 1]
```

In [ ]:
```
# sort vs sorted
L = [2,1,5,7,0]
print(L)

print(sorted(L))

print(L)

L.sort()

print(L)
```

```
[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]
[2, 1, 5, 7, 0]
[0, 1, 2, 5, 7]
```

If you want to sort a list in-place, you should use the sort method. If you want to create a new sorted list without modifying the original list, you should use the sorted function

List Comprehension

List Comprehension provides a concise way of creating lists.

newlist = [expression **for** item **in** iterable **if** condition == **True**]

Advantages of List Comprehension

- More time-efficient and space-efficient than loops.

- Require fewer lines of code.
- Transforms iterative statement into a formula.

In [ ]:
```
# Add 1 to 10 numbers to the list

# if we use for loop
L=[]
for i in range(1,11):
    L.append(i)

print(L)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [ ]:
```
# By List Comprehension
L = [i for i in range(1,11)]
print(L)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [ ]:
```
# Scaler multiplication on vecrtor
c = [2,3,4]
v = -3

L = [v*i for i in c]
print(L)
```

```
[-6, -9, -12]
```

In [ ]:
```
# Add sqaures
L = [2,3,4,5,6]

[i**2 for i in L]
```

Out[ ]:
```
[4, 9, 16, 25, 36]
```

In [ ]:
```
# Print all numbers divisible by 5 in the range of 1 to 50

[i for i in range(1,51) if i%5 == 0]
```

Out[ ]:
```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

Disadvantages of Python Lists

- Slow
- Risky usage
- eats up more memory

In [ ]:
```
a = [1,2,3]
b = a.copy()

print(a)
print(b)

a.append(4)
print(a)
print(b)

# lists are mutable
```

```
[1, 2, 3]
[1, 2, 3]
[1, 2, 3, 4]
[1, 2, 3]
```

```
In [ ]:
```