Day 05 – Lists in Python

Understand lists as a built-in Python data structure and practice operations like creation, indexing, slicing, adding, removing, copying, joining, reversing, and sorting.

What is a Data Structure?

A data structure is a way to store and organize data efficiently. Python provides both built-in and user-defined data structures.

Built-in examples: list, tuple, set, dictionary

User-defined examples: stack, queue, linked list, tree, graph

List Characteristics

- Lists are defined using square brackets `[]`
- They can store mixed data types and allow duplicates
- Lists are mutable (modifiable) and growable
- They support indexing, slicing, and many built-in methods

1. List Creation and Types

In [1]:

```
list1 = []
print(type(list1))
```

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

In [3]:

```
list2 = [10,20,30,90,80,70,50] # List of Integer Numbers
list3 = [10.5,8.6,99.99,4.6] # List of Float Numbers
list4 = ['One','Two','Three','four','Five'] # List of Strings
list5 = ['Arman',25,[50,90],[5.6,9.5],['AA','hhh']] # Nested list & mixed data
list6 = ['Asif', 25 ,[50, 100],[150, 90] , {'John' , 'David'}]
```

In [4]:

```
print('Integer list :', list2)
print('Float list :', list3)
print('String list :', list4)
print('Nested and mixed list :', list5)
print('Another mixed list :', list6)
```

```
Integer list : [10, 20, 30, 90, 80, 70, 50]
Float list : [10.5, 8.6, 99.99, 4.6]
String list : ['One', 'Two', 'Three', 'four', 'Five']
Nested and mixed list : ['Arman', 25, [50, 90], [5.6, 9.5], ['AA', 'hhh']]
Another mixed list : ['Asif', 25, [50, 100], [150, 90], {'John', 'David'}]
```

In [5]:

```
# Length of each list
print('Length of Integer list:', len(list2))
print('Length of Float list:', len(list3))
print('Length of String list:', len(list4))
print('Length of Nested and mixed list:', len(list5))
print('Length of Another mixed list:', len(list6))
```

```
Length of Integer list: 7
Length of Float list: 4
Length of String list: 5
Length of Nested and mixed list: 5
Length of Another mixed list: 5
```

2. List Indexing

In [6]:

```
# Indexing examples
print(list4[0]) # 'one' - first element
print(list4[0][0]) # 'o' - first character of first element
print(list4[-1]) # 'Five' - last element
print(list5[-1]) # ['AA', 'hhh'] - last element (nested list)
```

```
One
O
Five
['AA', 'hhh']
```

3. List Slicing

In [7]:

```
# List slicing
mylist = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
print(mylist[0:3]) # ['one', 'two', 'three']
print(mylist[2:5]) # ['three', 'four', 'five']
print(mylist[:3]) # ['one', 'two', 'three']
print(mylist[:2]) # ['one', 'two']
print(mylist[-3:]) # ['six', 'seven', 'eight']
print(mylist[-2:]) # ['seven', 'eight']
print(mylist[-1]) # 'eight'
print(mylist[:]) # whole list
```

```
['one', 'two', 'three']
['three', 'four', 'five']
['one', 'two', 'three']
['one', 'two']
['six', 'seven', 'eight']
['seven', 'eight']
eight
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

4. Adding Items

In [8]:

```
mylist.append('nine') # add at end
mylist.insert(9, 'ten') # add at index 9
print(mylist)
```

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

5. Changing List Items

In [9]:

```
mylist[0] = 1
mylist[1] = 2
mylist[2] = 3
print(mylist)
```

```
[1, 2, 3, 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
```

6. Insert and Remove Items

In [10]:

```
# Inserting and removing items
mylist.insert(1, 'ONE')
print(mylist)
mylist.remove('ONE')
print(mylist)
mylist.pop() # remove last item
print(mylist)
```

```
[1, 'ONE', 2, 3, 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
[1, 2, 3, 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten']
[1, 2, 3, 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

7 & 8. Remove or Delete Specific Index

In [11]:

```python
if len(mylist) > 8:
    mylist.pop(8)
if len(mylist) > 7:
    del mylist[7]
print(mylist)
```

```
[1, 2, 3, 'four', 'five', 'six', 'seven']
```

9. Clear Entire List

In [12]:

```python
mylist.clear()
print(mylist)
```

```
[]
```

In [13]:

```python
# Recreate list for next steps
mylist = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

10. Copying Lists

In [14]:

```python
mylist1 = mylist # Reference copy (both point to same list)
mylist2 = mylist.copy() # Actual copy with different address
```

In [15]:

```python
print(id(mylist), id(mylist1)) # same id
print(id(mylist2))
```

```
1683598163904 1683598163904
1683598056064
```

In [16]:

```python
# Change original to show effect on copies
mylist[0] = 1
print(mylist) # [1, 'two', 'three', ...]
print(mylist1) # [1, 'two', 'three', ...] same because reference copy
print(mylist2) # ['one', 'two', 'three', ...] unaffected copy
```

```
[1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
[1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

11. Joining Lists

In [17]:

```python
# Joining lists
list1 = ['one', 'two', 'three', 'four']
list2 = ['five', 'six', 'seven', 'eight']
list3 = list1 + list2
print(list3)
```

```
['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

12 & 13. Reversing Lists

In [18]:

```python
# Reverse list
list1.reverse()
print(list1)

# Or using slicing
list1 = list1[::-1]
print(list1)
```

```
['four', 'three', 'two', 'one']
['one', 'two', 'three', 'four']
```

14. Sorting Lists

In [19]:

```
# Sorting lists
mylist3 = [9, 5, 2, 99, 12, 88, 34]
mylist3.sort() # Ascending sort
print(mylist3)
mylist3.sort(reverse=True) # Descending sort
print(mylist3)
mylist4 = [88, 65, 33, 21, 11, 98]
print(sorted(mylist4)) # returns sorted list without modifying original
print(mylist4)
```

```
[2, 5, 9, 12, 34, 88, 99]
[99, 88, 34, 12, 9, 5, 2]
[11, 21, 33, 65, 88, 98]
[88, 65, 33, 21, 11, 98]
```

15. Loop Through List

In [20]:

```
for i in list1:
    print(i)
```

```
one
two
three
four
```

16. all() and any()

In [22]:

```
# all() and any() examples
lst_bool1 = [True, True, False]
lst_bool2 = [True, True, True]
print(all(lst_bool1)) # False, because one False
print(all(lst_bool2)) # True, all True
print(any(lst_bool1)) # True, because at least one True
print(any([False, False])) # False, none True
```

```
False
True
True
False
```

17. List Membership

In [23]:

```
list1
```

Out[23]:

```
['one', 'two', 'three', 'four']
```

In [24]:

```
'one' in list1 # Check if 'one' exist in the list
```

Out[24]:

```
True
```

In [25]:

```
'ten' in list1 # Check if 'ten' exist in the list
```

Out[25]:

```
False
```

In [26]:

```
if 'three' in list1: # Check if 'three' exist in the list
    print('Three is present in the list')
else:
    print('Three is not present in the list')
```

```
Three is present in the list
```

In [27]:

```python
if 'eleven' in list1: # Check if 'eleven' exist in the list
    print('eleven is present in the list')
else:
    print('eleven is not present in the list')
```

```
eleven is not present in the list
```