Data Structures

1. Introduction to Lists
2. Creating Lists
3. Accessing List Elements
4. Modifying List Elements
5. List Methods
6. Slicing Lists
7. Iterating Over Lists
8. List Comprehensions
9. Nested Lists
10. Practical Examples and Common Errors

Introduction To Lists

- Lists are ordered, mutable collections of items.
- They can contain items of different data types.

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

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

```
names = ["rishi","jack","job",1,2,3,4,5]
print(names)
```

```
['rishi', 'jack', 'job', 1, 2, 3, 4, 5]
```

```
mixed_list = [1,"Hello",3.14,True]
print(mixed_list)
```

```
[1, 'Hello', 3.14, True]
```

```
### Accessing List Elements

fruits=["apple","banana","cherry","kiwi","gauva"]

print(fruits[0])
print(fruits[2])
print(fruits[4])
print(fruits[-1])
```

```
apple
cherry
gauva
gauva
```

```
print(fruits[1:])
print(fruits[1:3])
```

```
['banana', 'cherry', 'kiwi', 'gauva']
['banana', 'cherry']

## Modifying The List elements
fruits

['apple', 'banana', 'cherry', 'kiwi', 'gauva']

fruits[1] = "watermelon"
print(fruits)

['apple', 'watermelon', 'cherry', 'kiwi', 'gauva']

fruits[0:] = "watermelon"
fruits

['w', 'a', 't', 'e', 'r', 'm', 'e', 'l', 'o', 'n']

fruits=["apple","banana","cherry","kiwi","gauva"]

## List Methods

fruits.append("orange") ## Add an item to the end
print(fruits)

['apple', 'banana', 'cherry', 'kiwi', 'gauva', 'orange']

fruits.insert(1,"watermelon")
print(fruits)

['apple', 'watermelon', 'banana', 'cherry', 'kiwi', 'gauva', 'orange']

fruits.insert("watermelon")
print(fruits)

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[44], line 1
----> 1 fruits.insert("watermelon")
      2 print(fruits)

TypeError: insert expected 2 arguments, got 1

fruits.remove("banana") ## Removing the first occurance of an item
print(fruits)

['watermelon', 'apple', 'watermelon', 'cherry', 'kiwi', 'gauva',
'orange']

## Remove and return the last element
popped_fruits = fruits.pop()
```

```
print(popped_fruits)
print(fruits)
```

```
orange
['apple', 'watermelon', 'cherry', 'kiwi', 'gauva']
```

```
index = fruits.index("cherry")
print(index)
```

```
2
```

```
fruits.insert(2,"banana")
print(fruits)
print(fruits.count("banana"))
```

```
['watermelon', 'apple', 'banana', 'banana', 'banana', 'watermelon',
'cherry', 'kiwi', 'gauva', 'orange']
3
```

```
len(fruits)
```

```
10
```

```
print(fruits)
fruits.sort() ## Sorts the list in ascending order
print(fruits)
```

```
['watermelon', 'apple', 'banana', 'banana', 'banana', 'watermelon',
'cherry', 'kiwi', 'gauva', 'orange']
['apple', 'banana', 'banana', 'banana', 'cherry', 'gauva', 'kiwi',
'orange', 'watermelon', 'watermelon']
```

```
fruits.reverse() ## REverse the list
print(fruits)
```

```
['watermelon', 'watermelon', 'orange', 'kiwi', 'gauva', 'cherry',
'banana', 'banana', 'banana', 'apple']
```

```
fruits.clear() ## Remove all items from the list
print(fruits)
```

```
[]
```

```
## Slicing List
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(numbers[2:5])
print(numbers[:5])
print(numbers[5:])
print(numbers[::2])
print(numbers[::-1])
```

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

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

```
numbers[::3]
```

```
[1, 4, 7, 10]
```

```
numbers[::-2]
```

```
[10, 8, 6, 4, 2]
```

```
st = 'samahit'
for i in st:
    print(i)
```

```
s
a
m
a
h
i
t
```

```
### Iterating Over List
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for number in numbers:
    print(number)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
## Iterating with index
for index,number in enumerate(numbers):
    print(index,number)
```

```
0 1
1 2
2 3
3 4
4 5
5 6
6 7
```

```
7 8
8 9
9 10
```

```
numbers = ['apple', 'banana', 'cherry', 'gauva', 'kiwi', 'orange',
'watermelon', 'watermelon']
for index,number in enumerate(numbers):
    print(index,number)
    if index == 4:
        break
```

```
0 apple
1 banana
2 cherry
3 gauva
4 kiwi
```

```
## List comprehension
lst=[]
for x in range(10):
    lst.append(x**2)

print(lst)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
[x**2 for x in range(10)]
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

List Comprehension

Basics Syantax [expression for item in iterable]

with conditional logic [expression for item in iterable if condition]

Nested List Comprehension [expression for item1 in iterable1 for item2 in iterable2]

```
### Basic List Comphrension

sqaure=[num**2 for num in range(10)]
print(sqaure)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
### List Comprehension with Condition
lst=[]
for i in range(10):
    if i%2==0:
        lst.append(i)

print(lst)
```

```
[0, 2, 4, 6, 8]
```

```
[i**3 for i in range(10) if i%2==0]
```

```
[0, 8, 64, 216, 512]
```

```
### List Comprehension with Condition
lst=[]
for i in range(10):
    if i%2==0:
        lst.append(i)

print(lst)
```

```
[0, 2, 4, 6, 8]
```

```
lst1=[1,2,3,4]
lst2=['a','b','c','d']

# for l1 in lst1:
#     for l2 in lst2:
#         print(l1, l2)

print([[l1,l2] for l1 in lst1 for l2 in lst2])
```

```
[[1, 'a'], [1, 'b'], [1, 'c'], [1, 'd'], [2, 'a'], [2, 'b'], [2, 'c'],
[2, 'd'], [3, 'a'], [3, 'b'], [3, 'c'], [3, 'd'], [4, 'a'], [4, 'b'],
[4, 'c'], [4, 'd']]
```

```
## Nested List Comphrension

lst1=[1,2,3,4]
lst2=['a','b','c','d']

pair=[[i,j] for i in lst1 for j in lst2]

print(pair)
```

```
[[1, 'a'], [1, 'b'], [1, 'c'], [1, 'd'], [2, 'a'], [2, 'b'], [2, 'c'],
[2, 'd'], [3, 'a'], [3, 'b'], [3, 'c'], [3, 'd'], [4, 'a'], [4, 'b'],
[4, 'c'], [4, 'd']]
```

```
## List Comprehension with function calls
words = ["hello", "world", "python", "list", "comprehension"]
lengths = [len(word) for word in words]
print(lengths)  # Output: [5, 5, 6, 4, 13]
```

```
[5, 5, 6, 4, 13]
```