# UNIT-3

# LISTS, TUPLES & DICTIONARIES

## LISTS

- Similar to arrays.
- Can store different types of elements.

### How to create a list?

- Square Bracket [ ]

**Example:**

```
list=[] #empty list
list=[1,2,3] #List of integers
list=[1,"Hello",3.4] #list of mixed datatypes
list=["Hello",[7,8,9],['s']] #nested List
```

## ACCESSING LISTS

1. **List Index / Indexing**

- Index operator [ ]
- Index Error
- Type Error
- Nested Indexing
- Negative Indexing

2. **Slicing**

- Slicing Operator :
- [start : stop: step_size]
- By default, ‘start’ will be 0, ‘stop’ will be the last element and ‘step_size’ will be 1.

# OPERATION & WORKING WITH LISTS

## 1. Creating lists using range() Function

**Syntax: range(start, stop, stepsize)**

**Example:**

```
lst=list(range(0,10,1))
print(lst)

for i in range(0,10,1):
   print(i)

lst=range(0,10,1)
print(lst)
```

## 2. Updating the Elements of a List

- Mutable
- Assignment operator =
- One item: **append()**
- Add several items in the list : **extend ()**
- One item at a desired location : **insert()**

**Example:**

```
odd=[1,2,3]
print(odd)
odd.append(7)
print(odd)
odd.extend([9,11,13])
print(odd)

#insert method
odd=[1,9]
odd.insert(1,3)
print(odd)

odd[2:2]=[5,7]
print(odd)

lst=list(range(1,5))
```

```
print(lst)

lst.append(7)
print(lst)

#indexing
lst[1]=6 #Update 1st element of the list
print(lst)

#slicing
lst[1:3]=12, 14 #update 1st and 2nd element of the list
print(lst)

#Deleting
del lst[2] #index
print(lst)

lst.remove(12)
print(lst)
```

## 3. Delete/ Remove list Elements

- Del keyword
- Remove () method
- Pop()
- Clear() method

**Example:**

```
list=['m','o','r','n','i','n','g']
print(list)

#delete one item
del list[2]
print(list)

#delete multiple items
del list[1:5]
print(list)

del list #delete the entire list
print(list)
```

```
my_list=['P','y','t','h','o','n']
print(my_list)
my_list.remove('P')
print(my_list)
print(my_list.pop(1))
print(my_list)
print(my_list.pop())
print(my_list)
my_list.clear()
print(my_list)
```

## 4. Concatenation of Two Lists

**Example:**

```
a=[1,2,3,4,5]
b=[10,20,30]
print(a+b)

odd=[1,3,5]
print(odd+[9,7,5])
```

## 5. Repetition of Lists

**Example:**

```
print(["hello"]*3)

x=[1,2,3,4,5]
print(x*2)
```

## 6. Membership in Lists

**Example:**

```
a=[10,20,30,40,50]
x=20
print(x in a)
print(x not in a)

list=['H','e','l','l','o']
print('e' in list)
print('i' in list)
print('y' not in list)
```

## 7. Iterating through a list

**Example:**

```
for fruit in ['apple','orange','kiwi']:
   print("I like", fruit)
```

## 8. Aliasing and Cloning List

- Giving a new name to an existing list is called '**aliasing'.**
- The new name is called **alias name**.

**Example:**

```
x=[10,20,30,40,50]
y=x #alias
print(x)
print(y)

x[1]=48
print(x)
print(y)
```

- If the programmer wants two independent lists, he should not go for aliasing.
- He should use **cloning or copying**.
- Obtaining exact copy of an existing object (or list) is called **cloning.**

**Example:**

```
y=x[:]          #x is cloned as y
print(x)
print(y)

x[1]=60
print(x)
print(y)

y[1]=66
print(x)
print(y)
```

- In cloning, modifications to a list are confined only to that list.
- The same can be achieved by **copying the elements of one list to another using copy () method.**

  **Example:**

  ```
  y=x.copy() #x is copied as y
  ```

## METHODS

- len ()
- index ()
- append ()
- insert ()
- copy ()
- extend ()
- count ()
- max ()
- min ()
- reverse ()
- remove ()
- pop ()
- sort ()
- clear ()
- intersection ()