

## 1.Understanding how to create and access elements in a list.

To create a list we place elements inside [ ] and separate them with commas

```
fruits = ["banana", "apple", "mango"]
```

list elements can be accessed by index values or range of index :

```
fruits[0]  
fruits[1]  
fruits[2]  
fruits[-1]  
fruits[0:2]
```

## 2.Indexing in lists (positive and negative indexing).

Elements of a list can be accessed by index value, index can be negative or positive

by default a list's index starts from zero

```
fruits = ["banana", "apple", "mango", "cherry"]
```

**Positive indexing** : positive index access element from starting of list

```
fruit = fruits[1] # apple  
fruit = fruits[2] # mango
```

**Negative indexing** : negative index access element from end of the list

```
fruit = fruits[-1] # cherry  
fruit = fruits[-2] # mango
```

## 3.Slicing a list: accessing a range of elements.

Slicing is a method to extract a subset of elements from a list, the new list can be modified without reflecting the changes to old list except it has a nested list portion, nested portion shares same reference to object

Multiple element of a list can be accessed by providing a range of index

a range can be specified by [start : end : steps]

```
fruits = ["banana", "apple", "mango", "cherry"]
```

```
fruits[0:2] # ["banana", "apple", "mango"]
```

## 4.Common list operations: concatenation, repetition, membership.

**Concatenation**: list can be concatenated using + operator or extend() method

```
l1 = [1,2,3]  
l2 = [4,5,6]  
l1 += l2
```

```
l1.extend(l2)
```

**repetition** : elements inside a list can be repeated using \* operator

```
l1 * 2
```

**membership** : membership operator used to check if a value is member of list or not

**in** operator checks if value is member

**not in** checks if value is not a member

if 1 in l1: print(f"{1} is a member")	If 1 not in l1: print(f"{1} is not a member")
--	--

## 5. Understanding list methods like append(), insert(), remove(), pop()

This are the methods to modify a list:

```
fruits = ["banana", "apple", "mango", "cherry"]
```

**append()** : add new element or a list at the end of a list

```
fruits.append("pinapple")
```

```
fruits.append(["papaya", "orange"])
```

**insert()**: insert a element at specific index in list

```
fruits.insert(2, "peru")
```

**remove()**: remove the first occurrence of an element in list

```
fruits.remove("peru")
```

**pop()**: returns or remove an element at specific index, by default returns last element

```
fruits.pop()
```

## 6. Iterating over a list using loops.

Iterate over a loop can be done using for loop or while loop

**for loop:**

```
for i in list:
```

```
    print(i)
```

**while loop:**

```
i = 0
```

```
while i < len(list):
```

```
    print(list[i])
```

```
    i += 1
```

## 7. Sorting and reversing a list using sort(), sorted(), and reverse().

**sort()** : sort the list in assending order by default

**sorted()**: sorted uses to sort a list but it returns a new list object

**reverse()** : reverse the order of elements in list

## 8. Basic list manipulations: addition, deletion, updating, and slicing.

```
fruits = ["banana", "apple", "mango", "cherry"]
```

**Addition :**

to add one list into another list we can use **+** operator.

**list1 += list2**

**Deletion:**

**pop()**: pop removes a element from a specific element or by default removes last element from list

```
list.pop() or list.pop(2)
```

**remove()**: remove() method removes specific elements first occurance in list

```
list.remove("banana")
```

**clear()**: clear method clear the list and make it empty

```
list.clear()
```

**del** : del can remove list or element of list from memory

```
del list or del list[3]
```

**Update:**

**insert:** insert method add a new element or list at a given index

**extend:** extend method extend a list with another list by addint it end of the lis

**9.Introduction to tuples, immutability.**

Tuples are immutable in python, after creating a tuple it have a fixed size and it can not be modified after creating,but any list nested inside a tuple can be modified.

**10.Creating and accessing elements in a tuple.****Creating tuple:**

```
t1 = 1,2,4,5,"banana",True
```

```
t2 = (1,2,4,5,"tea",True)
```

```
t3 = tuple([1,2,3,4,5])
```

**Accessing elements:**

```
t1[0]
```

```
t1[-1]
```

```
t1[:4]
```

**11.Basic operations with tuples: concatenation, repetition, membership.**

```
t1 = 1,2,4,5,"banana",True
```

```
t2 = (1,2,4,5,"tea",True)
```

**Concatenation:**

```
t1 += t2
```

**repetition:**

```
t1 *= 3
```

**membership :**

```
result = 2 in t1
```

```
result = 2 not in t1
```

**12.Accessing tuple elements using positive and negative indexing.**

```
t1 = (1,2,4,5,"tea",True)
```

**Positive indexing :**

```
t1[1] # 2
```

**Negative indexing:**

```
t[-1] # True
```

**13.Slicing a tuple to access ranges of elements**

to slice a tuple we can use same method for slicing list and string

**tuple name followed by [start : end : step]**

```
t1 = (1,2,4,5,"tea",True)
```

```
print(t1[1:4])
```

#### 14.Introduction to dictionaries: key-value pairs.

Dictionary is one of the powerfull data structure in python, it is an unordered collection of unique key value pair and it is mutable each key in dictionary is unique and it hold its corresponding value.

#### 15.Accessing, adding, updating, and deleting dictionary elements.

To add new item we can assign a new key value pair to dictionary **dict["key"] = "value"**

To update we can assign new value to existing key in dictionary or use update method

To delete we use del key word or pop method to remove or delete a key

#### 16.Dictionary methods like keys(), values(), and items().

**Key()** method returns an object which holds keys of dictionary in a list

**value()** method returns an object which holds value of each key in dictionary

**items()** method returns an object which holds a tuples of key value pair in a dictionary

#### 17.Iterating over a dictionary using loops.

```
person = {'name': 'John', 'age': 30, 'city': 'New York'}
```

```
for key, value in person.items():  
    print(f"{key}: {value}")
```

#### 18.Merging two lists into a dictionary using loops or zip().

```
keys = ['a', 'b', 'c']
```

```
values = [1, 2, 3]
```

```
dict = {}
```

```
for i in range(len(keys)):  
    dict[keys[i]] = values[i]
```

```
print(dict)
```

#### 19.Counting occurrences of characters in a string using dictionaries.

```
input_string = "hello world"
```

```
char_count = {}
```

```
for char in input_string:  
    if char in char_count:  
        char_count[char] += 1  
    else:  
        char_count[char] = 1
```

```
print(char_count)
```

#### 20.Defining functions in Python.

To define a function in python we use **def** keyword followed by function name and parentheses and parameters in parentheses if it required any.

```
def display():  
    print("hello world")
```

## 21. Different types of functions: with/without parameters, with/without return values.

There are 4 types of function categories:

**1 with parameters and return type:** functions that accept parameters at call and return a value

**2 with parameters without return type:** function that accepts parameters but didn't return something

**3 without return type and parameters:** function that didn't accept any parameters and didn't return any value

**4. without parameters and with return type:** function that didn't accept any parameters but return some value

## 22. Anonymous functions (lambda functions).

An anonymous function is a function which doesn't have any name and is created using the lambda keyword. Also called a lambda function, it can take n numbers of arguments but only have one statement

```
lambda n : n * 2
```

## 23. Introduction to Python modules and importing modules.

Modules are python files that contain function definition, variables, classes and executable code which we can use in our program by importing that module. Modules help in keeping and organizing the code in reusable components.

We can import a module using **import keyword followed by module name** in our file

```
ex :- import math
```

## 24. Standard library modules: math, random.

Math and random are part of Python's standard library.

Math module contains important math functions and classes and executable code which can be helpful in various mathematical tasks like -->

```
sqrt(), power(), ceil(), floor(), factorial(), sin(), cos(), tan()
```

random module contains functions to work with random numbers and values,

```
random(), randint(), choice(), shuffle()
```

## 25. Creating custom modules.

To create a custom module we can make a python file and define all the logics and statements in the file and import the file in another file using **import** keyword followed by module file name and we can use all the functions and variables of that module file by using a dot method

**module.function(value)**