29/09/2025

Slicing:

Slicing is used to extract a portion (sub-list) from a list using

list[start:end:step]

the default start position is 0

the default stop position is end

the default step position is 1

l1 = ['b', 101, 'info', True, 125.9, 32, 'bengaluru']

Printing elements from index 0 to 4

l1[0:5] or l1[:5]

o/p:

['b', 101, 'info', True, 125.9]

Printing elements at odd positions

l1[1::2]

o/p:

[101, True, 32]

Printing last three elements

l1[-3::1] or l1[3:]

o/p:

[125.9, 32, 'bengaluru']

- Negative index starts counting from the end of the list.

Printing elements whose positions are multiples of 3

l1[3::3]

o/p:

[True, 'bengaluru']

- Starts at index 3 and takes every 3rd element afterward.

Nested Lists {List inside list}

l2=['b',101,'info',True,[41,45,59],32,'bengaluru',[[[1,2,3]]]]

Accessing nested list elements

l2[5]

o/p:

32

Accessing sub-list elements

l2[4][1]

o/p:

45

- **Explanation:**

  l2[4] → [41, 45, 59]

  l2[4][1] → 45

Accessing deeply nested elements

l2[7][0][0][2]

o/p:

3

- **Explanation:**

  l2[7] → [[[1,2,3]]]

  Access inner lists step by step → final element 3.

Nested loop inside list (string iteration)

Using enumerate() inside nested loops

```
list1=['python','MySQL','PowerBI','ML&DL']
for i in list1:
    print(i)
    for j in enumerate(i):
        print(f"pos={j[0]},val={j[1]}")
    print()
```

o/p:

```
python
pos=0,val=p
pos=1,val=y
pos=2,val=t
pos=3,val=h
pos=4,val=o
pos=5,val=n

MySQL
pos=0,val=M
pos=1,val=y
pos=2,val=S
pos=3,val=Q
pos=4,val=L
```

PowerBI

pos=0,val=P

pos=1,val=o

pos=2,val=w

pos=3,val=e

pos=4,val=r

pos=5,val=B

pos=6,val=I

ML&DL

pos=0,val=M

pos=1,val=L

pos=2,val=&

pos=3,val=D

pos=4,val=L

- **Explanation:**

  Outer loop → iterates over each string in the list.

  enumerate(i) → gives both index (pos) and character (val) for each letter in the string.

Finding max value manually

Finding maximum number without using max()

```
list2=[3,8,1,4,2,9]
max_num=list2[0]
for i in list2:
    if i>max_num:
```

```
    max_num=i
print(max_num)
```

o/p:

9

- **Explanation:**

  Initially assume the first element is the largest.

  Compare each element using a loop.

  If a bigger number is found, update max_num.

Sorting of list

```
list2=[3,8,1,4,2,9]
list2.sort()
list2
```

o/p:

[1,2,3,4,8,9]

Sorting in reverse order

```
list2.sort(reverse=True)
list2
```

or

```
list2=[3,8,1,4,2,9]
list2[::-1]
```

o/p:

[9, 8, 4, 3, 2, 1]

Perform union and intersection on 2 given list

```
l1=[1,2,3,4,5]

l2=[4,5,6,7,8]

inter=[]

for i in l1:

    if i in l2:

        inter.append(i)

print("intersection:",inter)

union=[]

for i in l1:

    if i not in union:

        union.append(i)

for i in l2:

    if i not in union:

        union.append(i)

print("union:",union)
```

o/p:

intersection: [4, 5]

union: [1, 2, 3, 4, 5, 6, 7, 8]