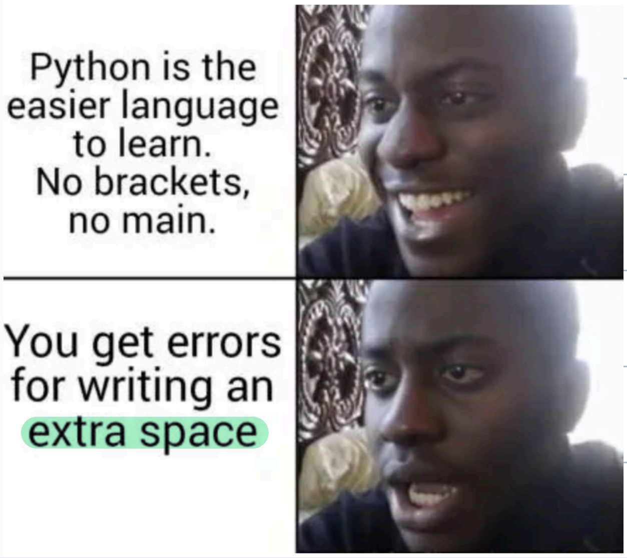
Agenda

1. List Slicing
    1. Motivation behind list slicing.
    2. List slicing with positive & negative indices
    3. List slicing with positive & negative step sizes
2. 2D Lists
    1. Introduction to 2D lists
    2. Indexing in 2D lists
    3. Iterating over 2D lists


Python is the easier language to learn. No brackets, no main.
You get errors for writing an extra space

Question-1

Given a list of all runs by Virat Kohli, create a new list of runs made in last 5 matches.

```
runs = [62, 85, 74, 10, 12, 101, 122, 99, 81, 55]
```

runs [start : end : jump]

1) +ve jump ⟶ default start = 0
default end = len

2) -ve jump → default start = -1
default end = -len - 1

Accessing List

1) Accessing an element → runs [i]

2) Accessing a range of elements →

a) runs [L : R] ⟶ From L to R-1.

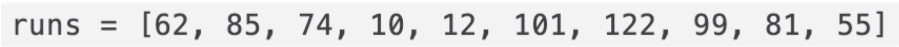
$$\underbrace{L}_{\text{included}} \quad \underbrace{R}_{\text{excluded}}$$

b) runs [L:] → From L to end.

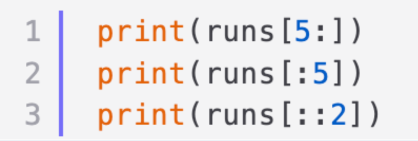
c) runs [: R] → From start to R-1.

d) runs [L : R : J] → From L to R-1 with jump J.

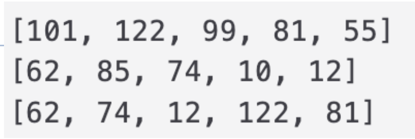
0 1 2 3 4 5 6 7 8 9



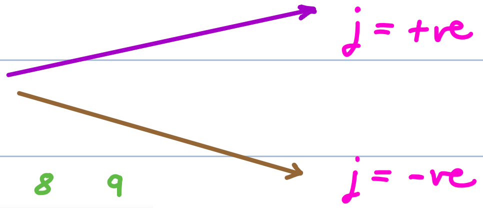
```
runs = [62, 85, 74, 10, 12, 101, 122, 99, 81, 55]
```



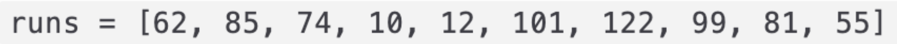
```python
print(runs[5:])
print(runs[:5])
print(runs[::2])
```



```
[101, 122, 99, 81, 55]
[62, 85, 74, 10, 12]
[62, 74, 12, 122, 81]
```



0 1 2 3 4 5 6 7 8 9



```python
runs = [62, 85, 74, 10, 12, 101, 122, 99, 81, 55]
```

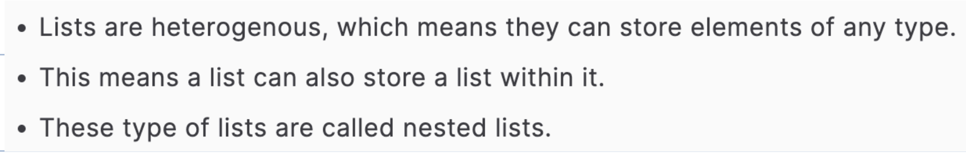
runs [8 : 3 : 2] → [ ]

runs [8 : 3 : -2] → [81 , 122 , 12]

runs [: : 2] → [62 , 74 , 12 , 122 , 81]

runs [: : -1] → Reverse

2D List

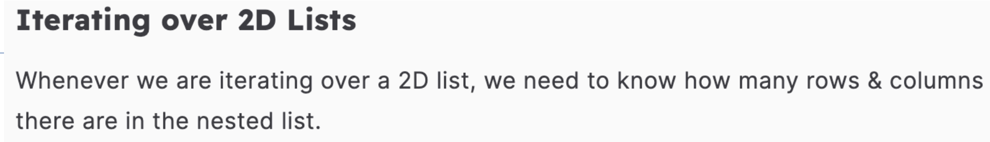


- Lists are heterogenous, which means they can store elements of any type.
- This means a list can also store a list within it.
- These type of lists are called nested lists.

$a = [\overset{0}{2} , \overset{1}{3.5} , \overset{2}{\text{"Hello"}} , \overset{3}{0.82}]$

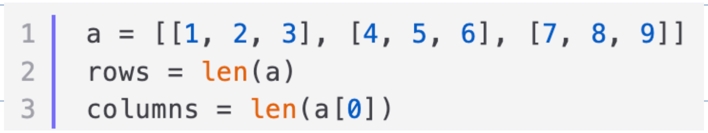
$a = [\overset{0}{1} , \overset{1}{2} , \overset{2}{[\overset{0}{10} , \overset{1}{20} , \overset{2}{30}]}]$

$a = [[1 , 2 , 3] , [4 , 5 , 6] , [7 , 8 , 9]]$

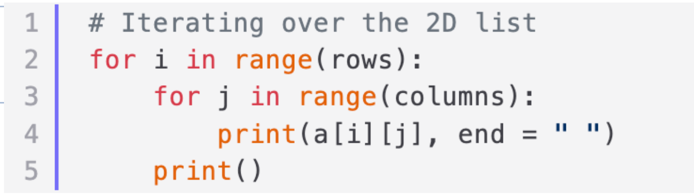


Iterating over 2D Lists

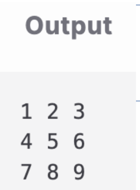
Whenever we are iterating over a 2D list, we need to know how many rows & columns there are in the nested list.



```python
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
rows = len(a)
columns = len(a[0])
```

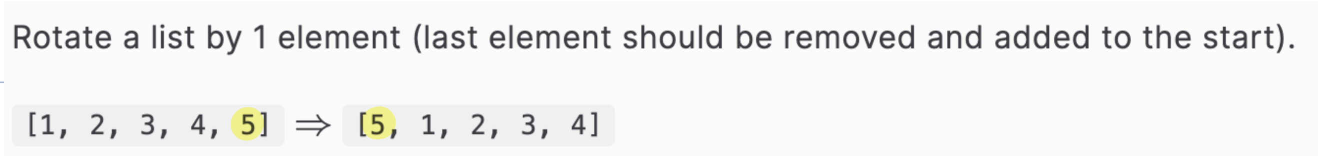


```python
# Iterating over the 2D list
for i in range(rows):
    for j in range(columns):
        print(a[i][j], end = " ")
    print()
```



Output

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



Rotate a list by 1 element (last element should be removed and added to the start).

`[1, 2, 3, 4, 5]` ⇒ `[5, 1, 2, 3, 4]`