List-3

1- Inbuilt functions

2- List slicing.

3- Nested List : Introduction.


```
y = max ( [5, 1, 2, -1] )
print ( y )
```

```
l = [5, 1, 2, -1]
```

`l.append(5)`

`l.count(5)`

=> Object method => will be covered in OOP classes.

* `l.index(5)` # gives the 1st idx of given val.

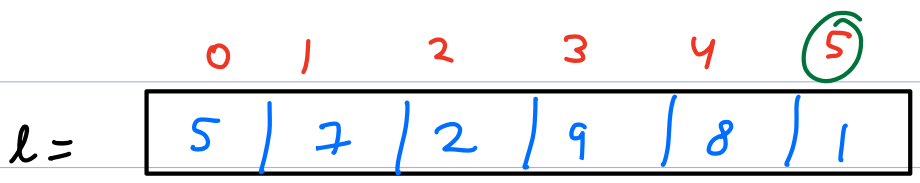
* `5 in l` ⟹ bool True/False

Q) Given a list, reverse the list.

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

print from reverse (Right to left)

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



`x = len(l)` # 6

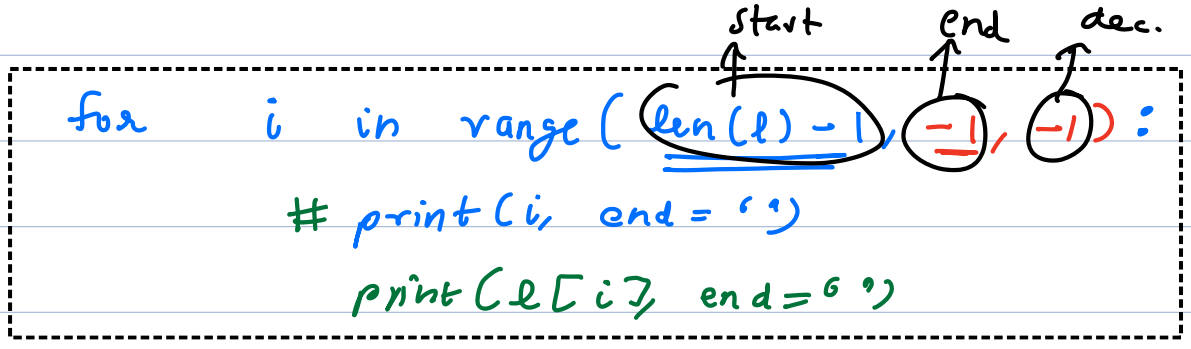
`range(5, -1, -1)`

`[5, 4, 3, 2, 1, 0]`

5 → 4 → 3 → 2 → 1 → 0 # -1

x-1

①



② l.reverse()

③ l[::-1]

inc. negative

Try this ←

default start = len(l)

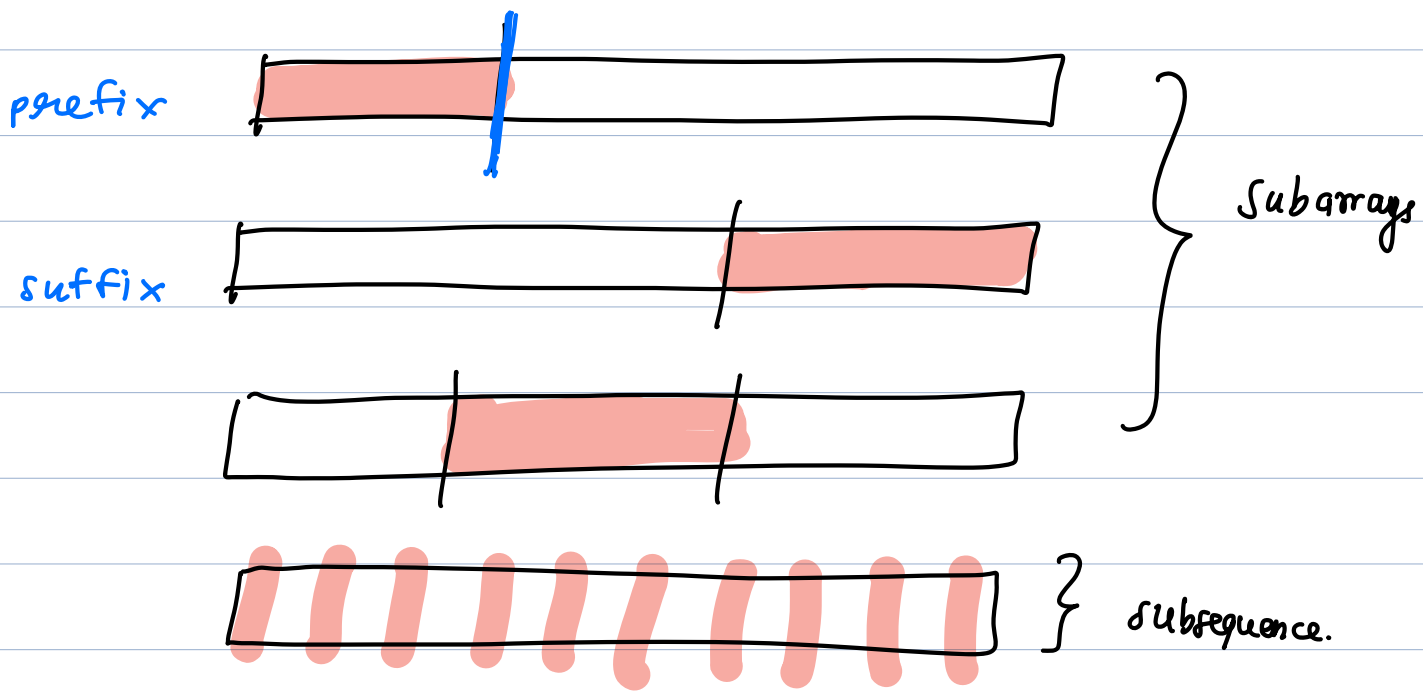
end = 0

Cover it in next class.

List Slicing

Q) Given the runs made by Sachin in a list, find the runs made in

1) odd matches (1, 3, 5, 7, ...)
2) even matches (2, 4, 6, 8, ...)
3) First 5 matches runs[:5] =>
4) Last 5 matches runs[-5:]



range (start, end, inc) :

range (10) = [0, 1, 2 . . . 9]

range (1, 10) = [1, 2 . . . 9]

range (1, 10, 2) = [1, 3, 5, 7, 9]

---

idx ⇒ 0 1 2 3 4 | 5 6

nums = [5, 1, 2, 7, 6, | 3, 4]

use a colon : inside square brackets

i) range (0, 5, 1)

[0, 1, 2, 3, 4]

nums [0 : 5 : 1]

start = 0

end = 5 (excluded)

inc = 1

ii) range (0, 5) default inc = 1

nums [0 : 5]

default inc = 1

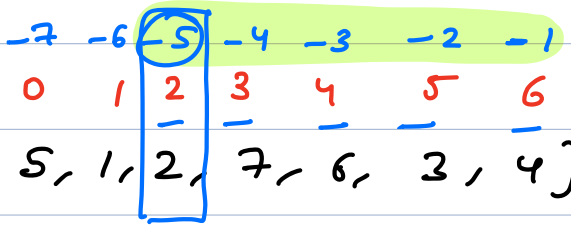
iii) range (5) → default start = 0, default inc = 1

[0, 1, 2, 3, 4]

nums [: 5]

default start = 0

default inc = 1



start

nums [-5 :] ✓

range (-5, 0, 1)

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

range (-5, 0) default inc = 1

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

range (-5) default start = 0

default inc = 1

X []

Q) nums [-5 : 0 : 1] ?

Will it work?

nums [-5 : len() : 1] [ ]

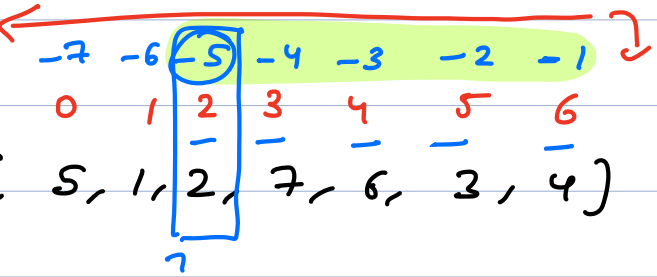
Q) nums [-5 : 0] ?

nums [-5 : len()] [ ]

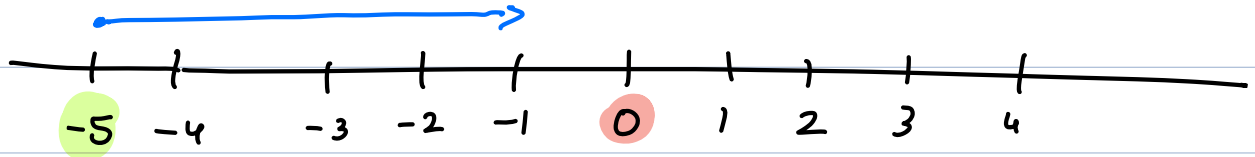
Q ? ✓ nums [-5 :]

default end = len()

range (-5, 0, 1) ⟹ [-5, -4, -3, -2, -1]



direction of -ve idx is opposite.



+1 +1 +1 +1

nums [-5 : 0 : 1]

-5 → -4, -3, -2 → -1

0 1 2 3 4 5 6 → 7

nums = [5, 1, 2, 7, 6, 3, 4]

nums [-5 : -1 : 1]

excluded

[-5, -4, -3, -2]

diff.

nums [-5 : 7 : 1]

range(-5, 7, 1) =

[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6]

len(l)

-7 -6 -5 -4 -3 -2 -1 → 0

0 1 2 3 4 5 6 → 7

Taken by last elt

nums = [5, 1, 2, 7, 6, 3, 4]

nums [-5 : 7]

default inc = 1

len(l)

nums [-5 :]

⇒ default end = len(l)

⇒ default inc = 1

nums [: 5]

end +ve

✓ default start = 0

✓ default inc : 1

nums [-5 :]

start -ve

✓ default end = len(l)

✓ 1

nums = [5, 1, 2, 7, 6, 3, 4] (0 1 2 3 4 5 6) N = 7

Odd idx Even idx

nums[1::2]

nums[0::2]

~~nums[:7:2]~~ → nums[0:7:2]

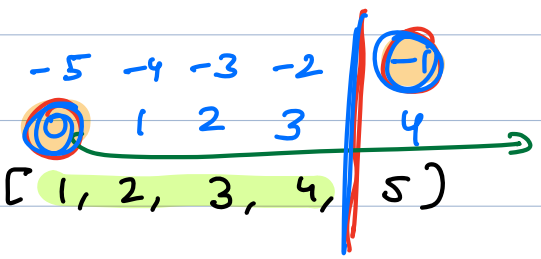
default start = 0

nums[1::2]

nums[0::2]

default start = 0

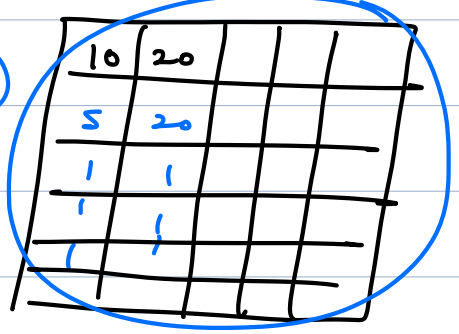
default end = N



nums = [1, 2, 3, 4, 5]

pixels 0-255

Images.



Grid OpenCv

All elements except last.

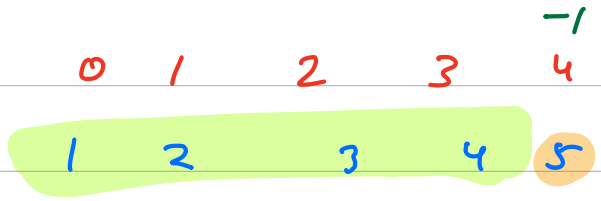
nums[0:-1]

nums[:-1]

Q) Rotate a given array. (Only one)

l = [1, 2, 3, 4, 5] R to L

res = [5, 1, 2, 3, 4]



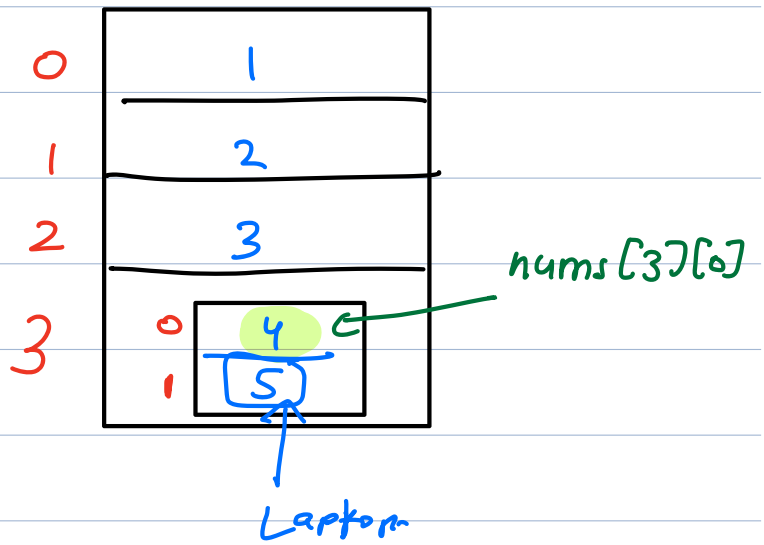
nums [-1:] # [5]

Doll inside doll

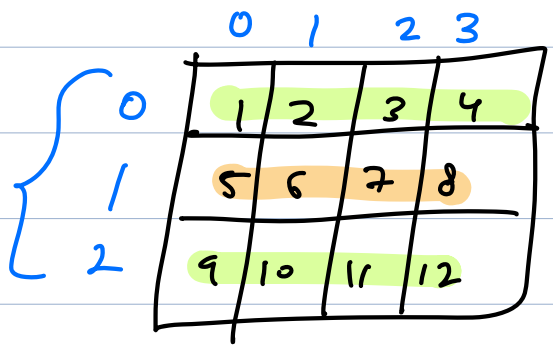
Bag inside bag.

nums = [1, 2, 3]

nums. append ([4, 5])



Image



Matrix

[[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12]]

nums = [1, 2, 3]

nums.extend([4, 5]) # adds one by one

[1, 2, 3, 4, 5]

Doubts:

ls[-1:len(l):1]