Data Science & AI
&
NIC - Param

Python-For Data Science

Stack and Queue

Lecture No.- 01

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Linked List Part-03

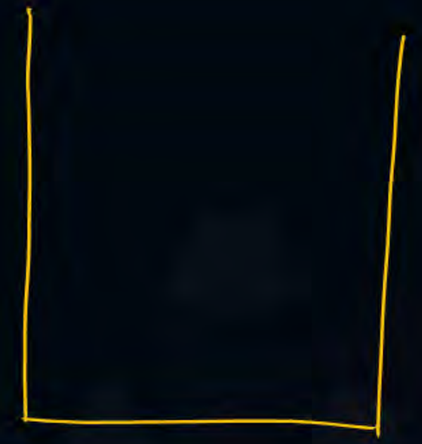
Topics to be Covered

Topic Stack and Queues Part-01

Topic : Stack and Queues

Stack

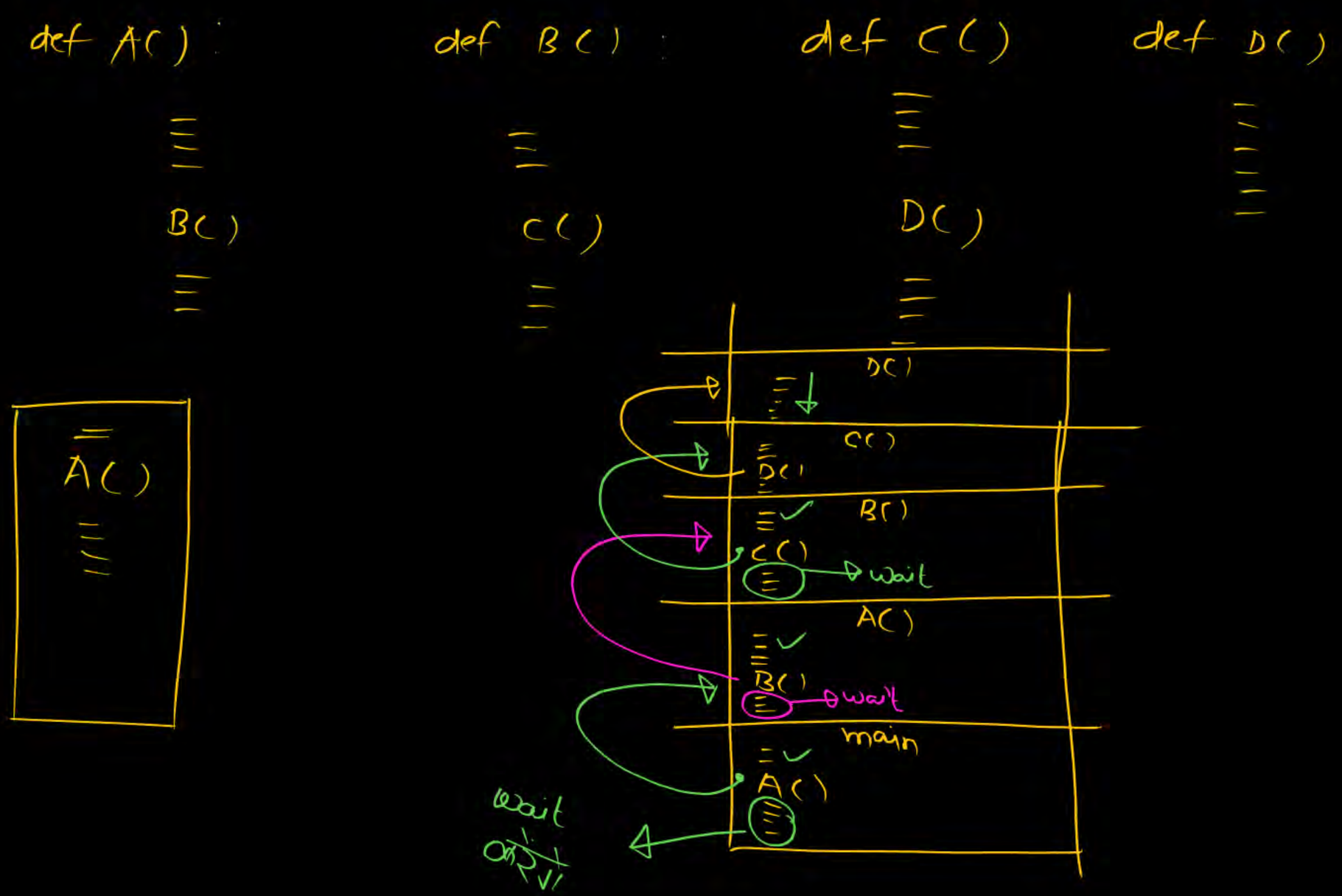
Last in First out policy

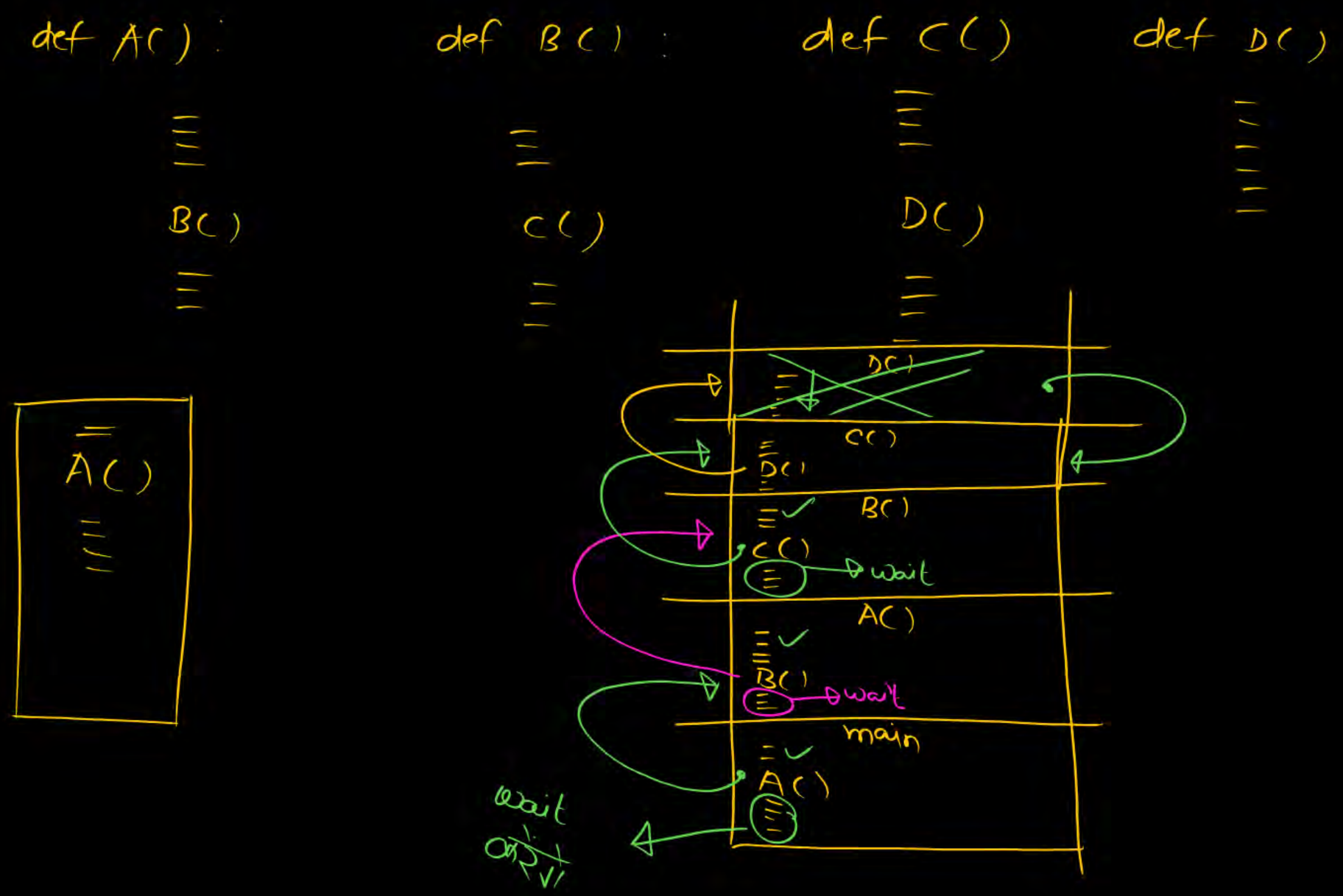


UNDO

recursion → ✓

function calling →


def A():
B()
def B():
C()
def C():
D()
def D():
A()
D()
C()
D()
B()
C()
wait
A()
B()
wait
main
A()
wait


def A():
def B():
def C()
def D()
B()
C()
D()
A()
D()
C()
D()
B()
C()
wait
A()
B()
wait
main
A()
wait

def A():

B()

def B():

C()

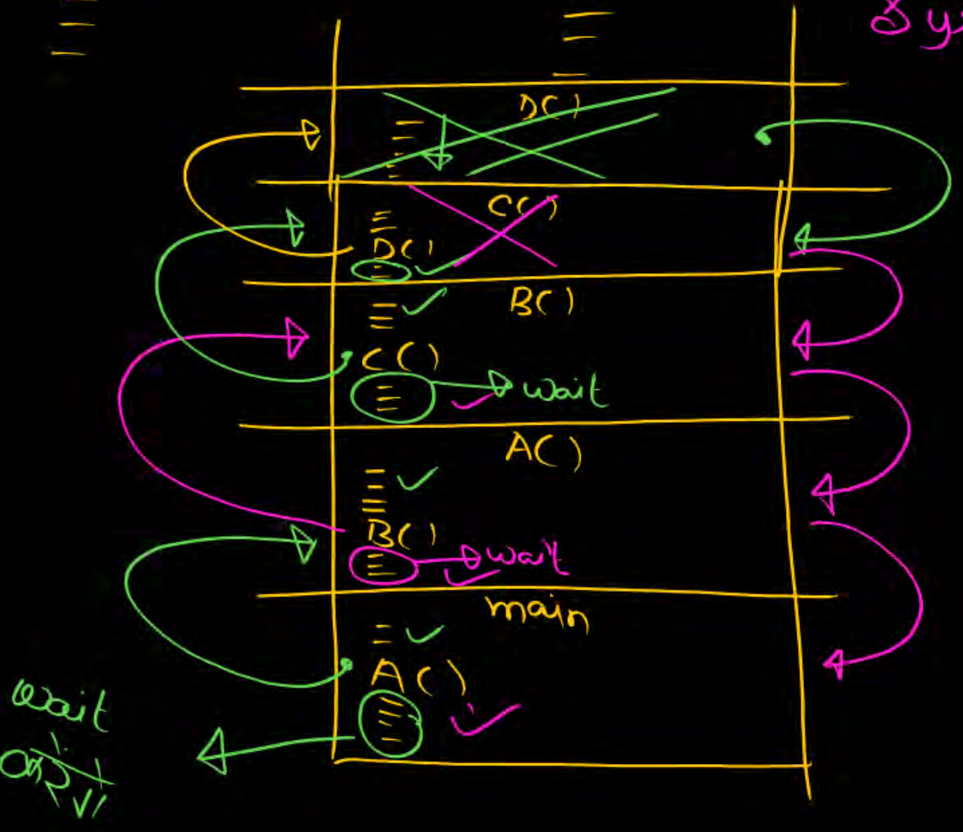
def C()

D()

def D()

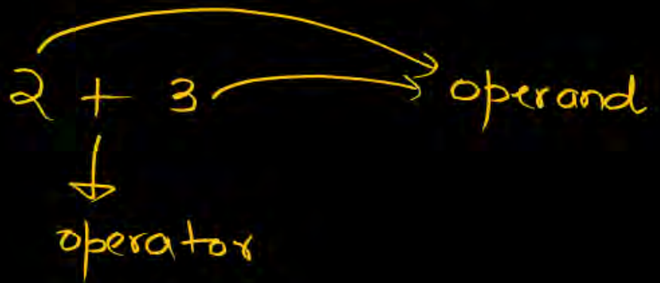
System stack

A()



Infix to Postfix

12th → infix



infix : operator is in between operands

CS : postfix
prefix

postfix: operands then operator

$2+3 \Rightarrow 2\ 3+$

prefix: operator then operands

$2+3 \Rightarrow +\ 2\ 3$

priority Associativity

infix: $2+3\times 4/6 \uparrow\uparrow 2 + \sim 6$


infix
Postfix
Evaluation

Prince

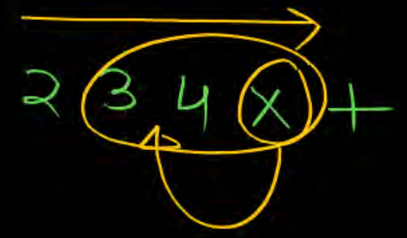
infix : 2 + 3 × 4

(1)

2 + [3 4 ×]

op1 op2

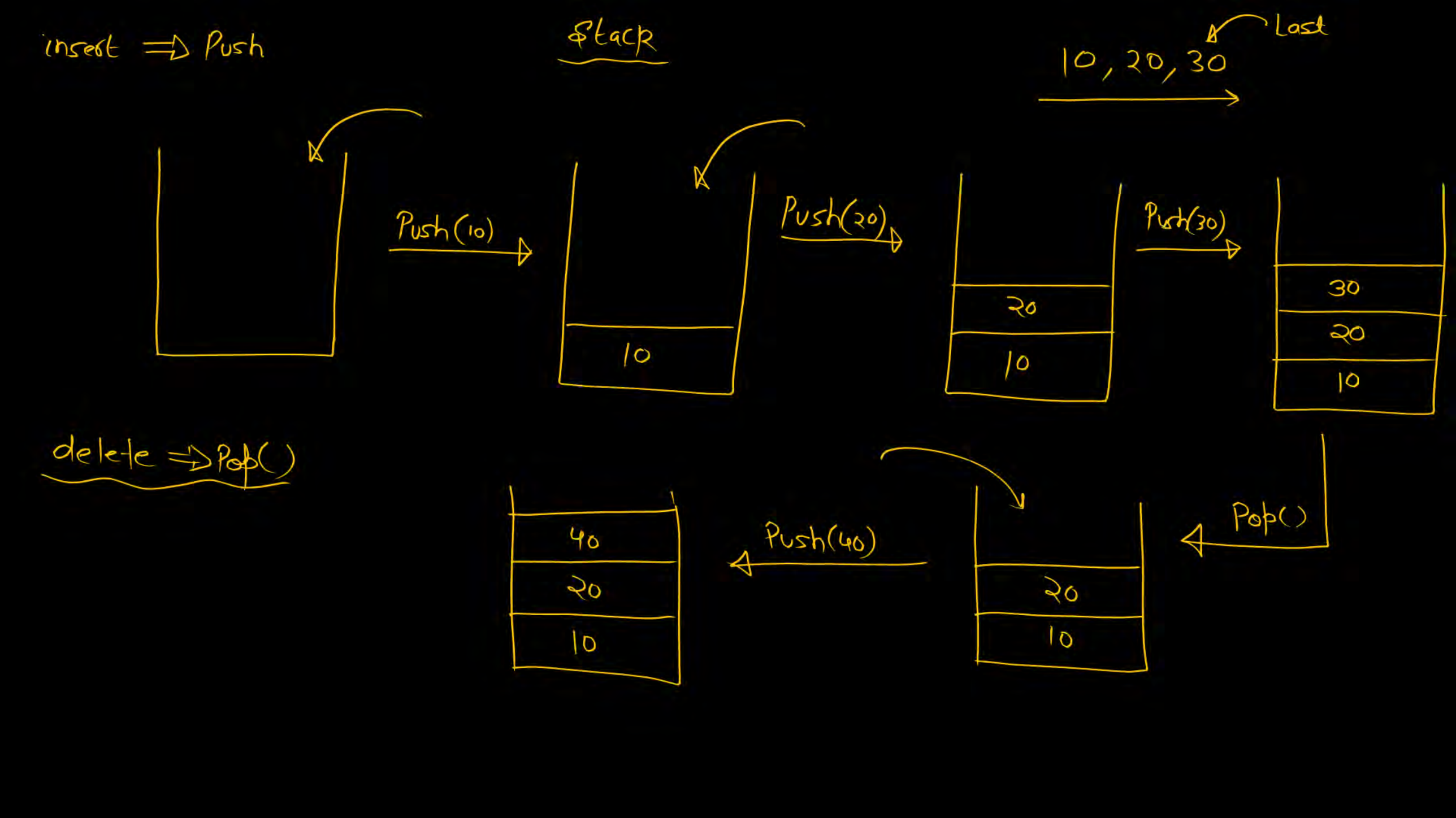
postfix : 2 3 4 × +



(14)

insert ⇒ Push

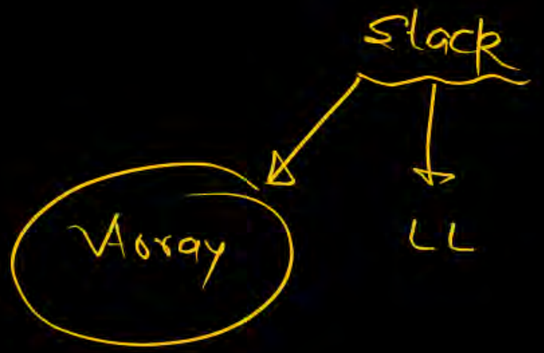
Stack



delete ⇒ Pop()

List/Array

Linked List



1. Push(x)
2. Pop()
3. IsEmply() → True / False
4. top() : returns the top ele. on stack
5. size() : No. of elem. in stacks

ArrayList → append

Push(10)
Push(20)

l.append(10)
l.append(20)

Pop() → l.pop()

top() → access top element (last element)

l[len(l) - 1]

l[-1]

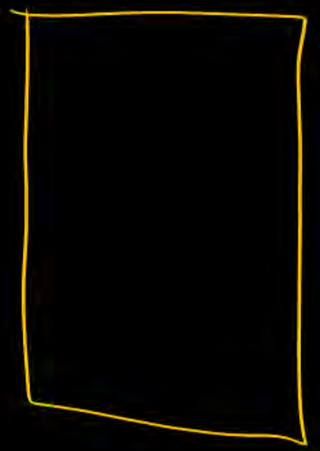
size
↳ len(l)

IsEmpty()

using list

① In list, we can access any element

ATM M/C



Private

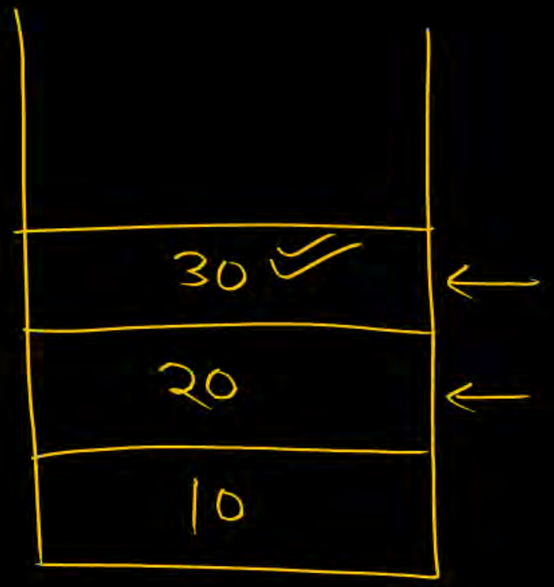
list ✓

Push

pop

IsEmpty

Operations allowed

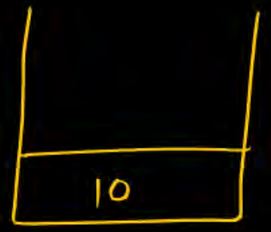


Push → O(1)

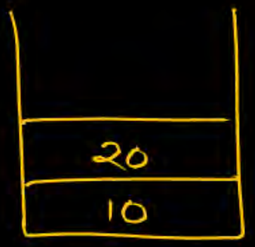
Pop → O(1)

Stack using linked list

Push(10)



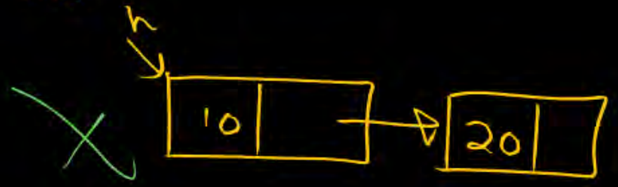
Push(20)



Push(30)

(i) 10

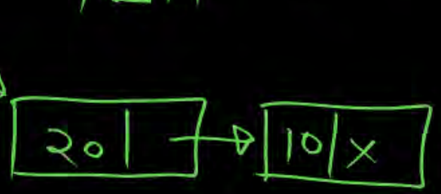
(ii) 20



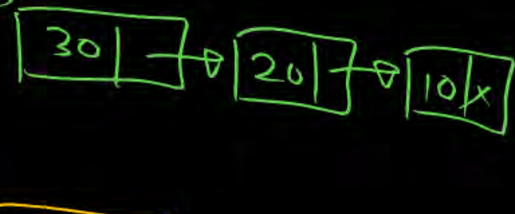
(iii) 30

(i)

(ii) Insert at begin → O(1)
Push



(iii)

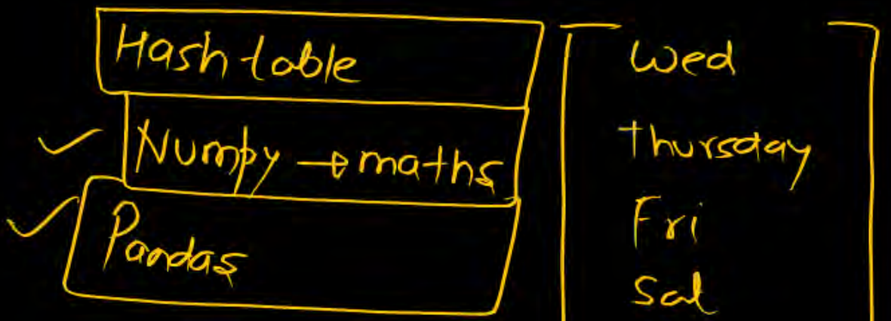


Stack → Problems

Queue

Mon, Tue (4 - lectures)

Tree



Recursion → next class

t.me/PWpankajsirP

In [1]:
```
class stack :
    def __init__(self):
        self.__array=[]
    def push(self,ele):
        self.__array.append(ele)
    def pop(self):
        #if the stack is empty ===>we can not pop
        if self.Isempty():
            print("empty stack")
            return
        return self.__array.pop()

    def top(self):
        #if the stack is empty ===>no top element
        if self.Isempty():
            print("no element")
            return
        return self.__array[-1]
    def size(self):
        return len(self.__array)
    def Isempty(self):
        return self.size()==0
```

In [2]:
```
s1=stack() #s1 object hai stack class ka
```

In [3]:
```
s1.push(10) #push 10
```

In [4]:
```
s1.push(20)#push 20
```

In [5]:
```
s1.push(30) #push(30)
```

In [6]:
```
print(s1.pop()) #print 30
```

```
30
```

In [7]:
```
print(s1.pop()) #20
```

```
20
```

In [8]:
```
print(s1.pop()) #10
```

```
10
```

In [9]:
```
print(s1.pop())
```

```
empty stack
None
```

In [ ]:
```

```

THANK - YOU