# Recursion - 1

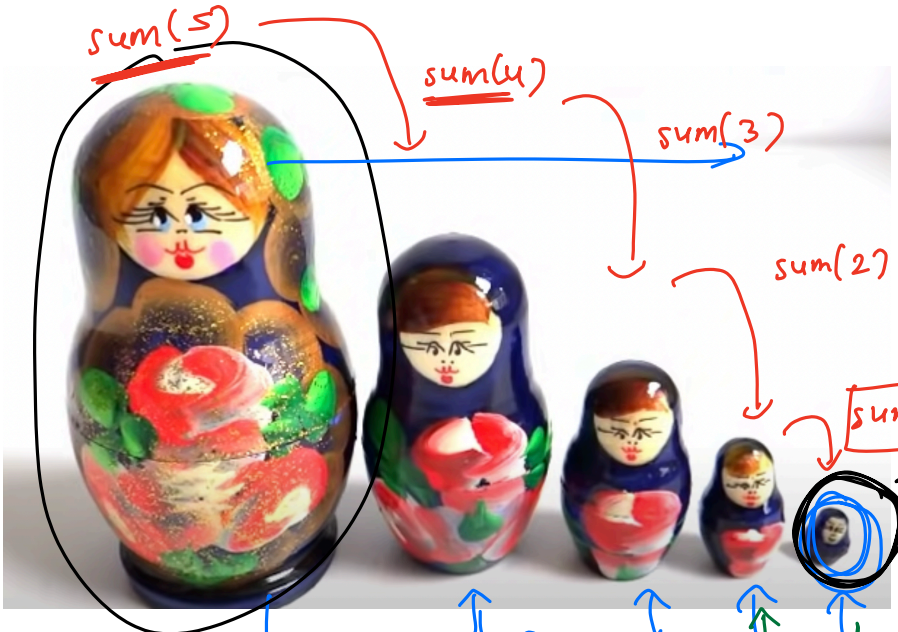
=> Python Refresher 4 -

Process on How to solve an Assignment Problem.

(1) What is recursion?

→ Merge sort } Adv. DSA
→ Trees
→ Dynamic Prg. } Adv DSA.
→ Backtracking

Video Clip => 3 observations




1) ~~smaller dolls (subproblem)~~
2) ~~Similar dolls (problem)~~
3) End doll (Base Condition)
4) ~~Larger dolls~~

Stopping Condition (3)

subproblems (2)

Recursion => Solving a problem (1) using smaller instances of the same problem.

Q1) Sum of 1st N natural numbers. => Do it using recursion.

sum(3) = 1+2+3 = 6

sum(4) = 1+2+3+4 = 10 => sum(3)+4

sum(5) = 1+2+3+4+5 = 15 => sum(4)+5

N

sum(N-1) & N

Generalize the logic for N value

sum(N) = sum(N-1) + N

## Steps of Recursion

-> Problem

1. **Assumption**: Decide what you want your fn to do
   => Define the problem.
2. **Main Logic**: Solving the assumption using subproblems.
3. **Base Condition**: When does the recursive function stop.

Steps of Recursion

① **Assumption**: Decide what you want your fn to do ← Problem
⇒ Define the problem.

⇒ Given a num N, calc. sum of 1st N numbers.

② **Main Logic**: Solving the assumption using subproblems.

③ **Base Condition**: When does the recursive function stop.

Calling the same fn within itself.

```
def sum(N):                          → Assumption.
    if N==1:                         } → Base Condition.
        return 1
    return sum(N-1) + N              ⇒ Main logic
```

sum(3) → sum(2) → sum(1) → sum(0) → sum(-1) → ...

Never stop.

1 * 10 = 10

0 + 10 = 10

(Q2) Factorial fact(N)

N = 5

fact(5) = 5*4*3*2*1 = 120

fact(6) = 6*5*4*3*2*1 = 720

= 6 * fact(5)

fact(2) = 2*1 = 2

fact(3) = 3*2*1 = 6

= 3 * fact(2)

fact(1) = 1

fact(0) = 1

① **Assumption**: Decide what you want your fn to do (Problem) => Define the problem.

② **Main Logic**: Solving the assumption using subproblems.

③ **Base Condition**: When does the recursive function stop.

=> fact(N) gives factorial of N

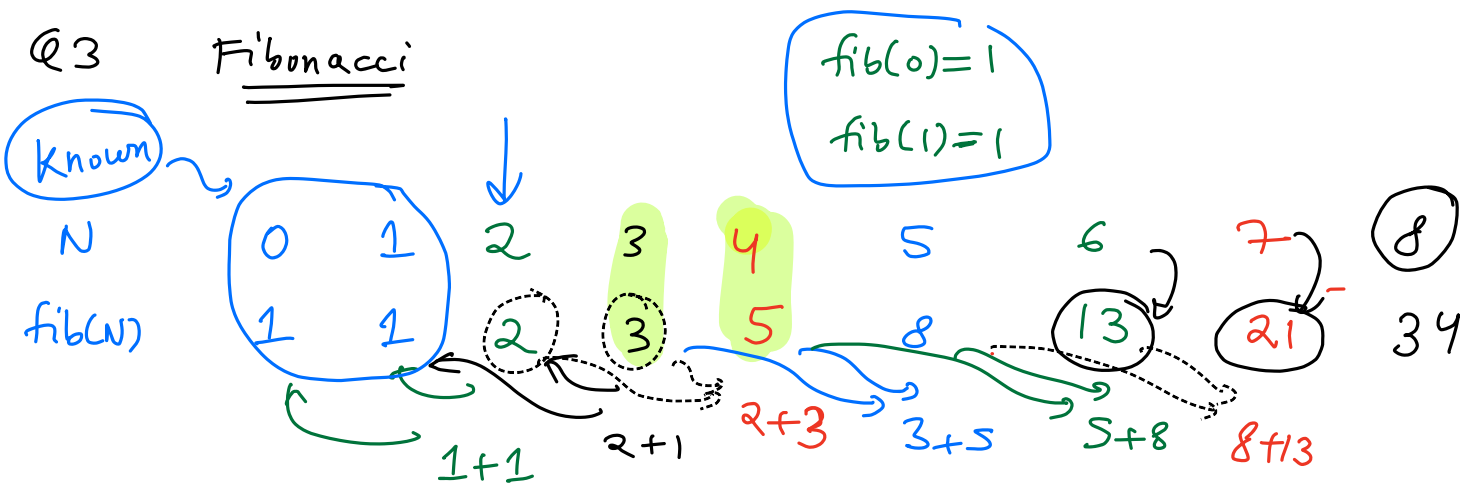
return

fact(N) = N * (N-1) * (N-2) * ... 1

= N * fact(N-1)

```
def fact(N):
    if N==0:
        return 1
    return N * fact(N-1)
```

Q3 Fibonacci




fib(5) = fib(4) + fib(3) = 5 + 3 = 8.

fib(N) = fib(N-1) + fib(N-2)

fib(8) = fib(7) + fib(6)

= 21 + 13

= 34

```
def fib(N): # return Nth fib.
    if N==0 Or N==1:
        return 1
    return fib(N-1) + fib(N-2)
```

Steps of Recursion

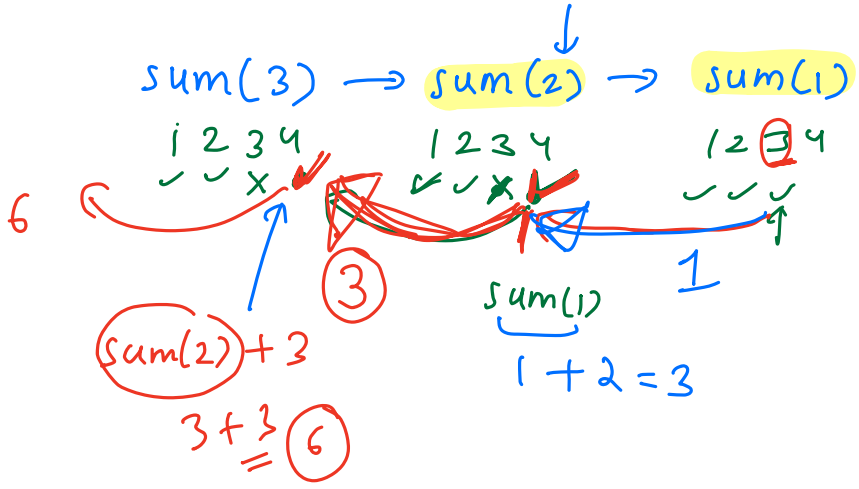
① **Assumption**: Decide what you want your fn to do → Problem
   ⇒ Define the problem.

② **Main logic**: Solving the assumption using subproblems.

③ **Base Condition**: When does the recursive function stop.

```
1 → def sum(N):                          → Assumption.
2 →     if N==1:                         → Base Condition
3 →         return 1                     → Base Condition
4 →     return sum(N-1) + N              ⇒ Main logic
```

Flow of Recursion.




Print 1-N

↳ 1, 2, 3, ---- N

using rec.

Python Tutor code visualizer: V

pythontutor.com/render.html#mode=display

# Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

Python 3.6
(known limitations)

```
1
2 def fact(n): # assumption
3     if n == 0: # base condition
4         return 1
5     return n * fact(n - 1) # main logic
6
7 print(fact(3))
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 15 of 18

Customize visualization

Print output (drag lower right corner to resize)

Frames Objects

Global frame
fact → function fact(n)

fact
n 3

fact
n 2

fact
n 1

fact
n 0
Return value 1

Wait on 5 line

Wait on line 5

Wait on line 5

Base

f(3) → f(2) → f(1) → f(0)

1

Python 3.6
(known limitations)

```
1
2 def fact(n): # assumption
3     if n == 0: # base condition
4         return 1
5     return n * fact(n - 1) # main logic
6
7 print(fact(3))
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 15 of 18

Customize visualization

Print output (drag lower right corner to resize)

Frames Objects

Global frame
fact → function fact(n)

fact
n 3

fact
n 2

fact
n 1

fact
n 0
Return value 1

Wait on 5 line

Wait on line 5

Wait on line 5

Base

1*1

2*1

3*2

f(3) → f(2) → f(1) → f(0)

6 ← 3*2

2 ← 2*1

1 ← 1*1

1

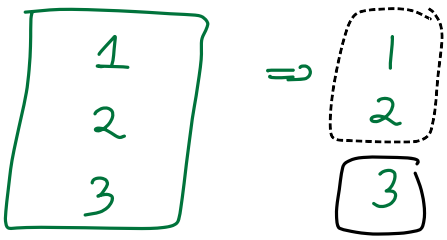
Target: Confident in writing recursive codes.

Q4) Print Increasing

Given N, print nums from 1 to N using recursion..

N = 3

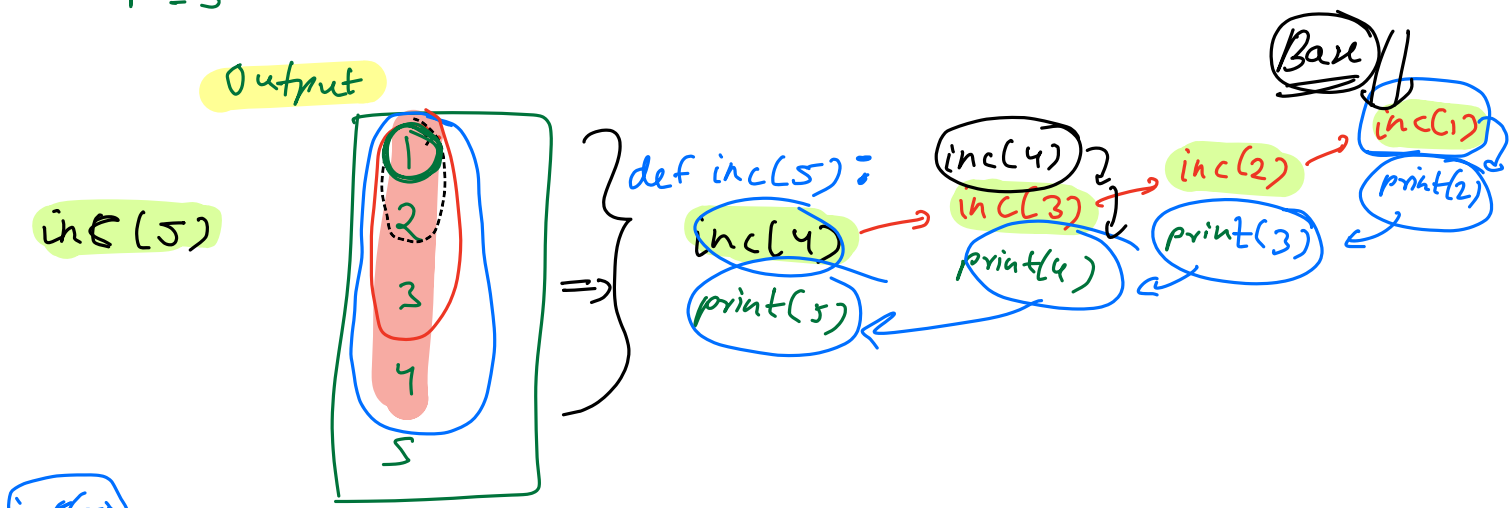
Output




inc(2)

print(3)

N = 5



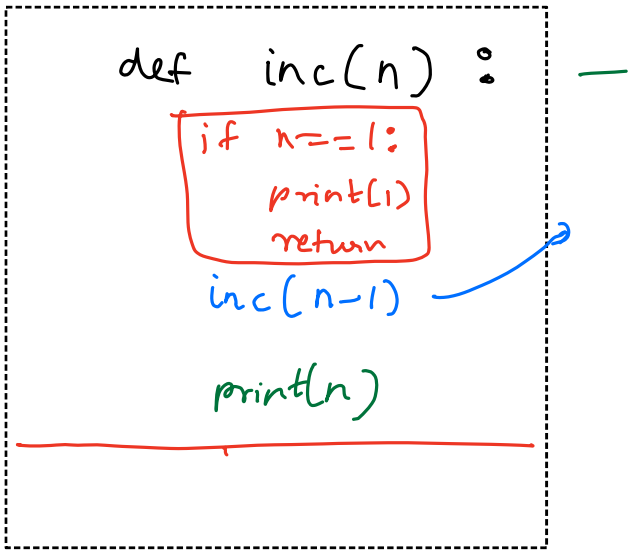

inc(5)

```
def inc(n):
    if n==1:
        print(1)
    inc(n-1)
    print(n)
```

# print from 1 to n in inc. order

my subordinate print from 1 to (n-1)

inc(5) → inc(4) → inc(3) → inc(2) → inc(1) → inc(0) → inc(-1) → inc(-2) ...




Python 3.6
(known limitations)

```
1  def inc(n): # assumption
2      if n == 1: # basee condition
3          print(1)
4          return
5      # main logic below
6      inc(n - 1)
7      print(n)
8
9  inc(5)
```

does return None

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Step 19 of 27

Customize visualization

Print output (drag lower right corner to resize)

```
1
```

Frames Objects

Global frame
inc → function inc(n)

inc
n 5

inc
n 4

inc
n 3

inc
n 2

inc
n 1
Return value None

HW: Print from N to 1 using recursion

→ : Do your assignments on recursion.

# Recap

⇒ Problem Solving Session on Sat, Sep 10, 9 pm

Time Complexity + Python Refresher + Sorting

- 4-5 problems from Assignment HW.
  - least solved ones
  - You have mentioned on Slack.

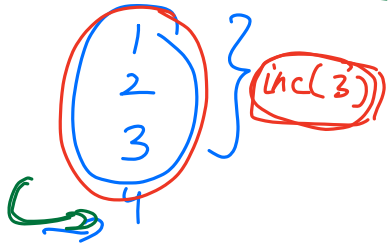
1 hr/2hr/3hr

# Doubts

→ (1) Inc. Print ( Return)

→ (2) Flow of sum function

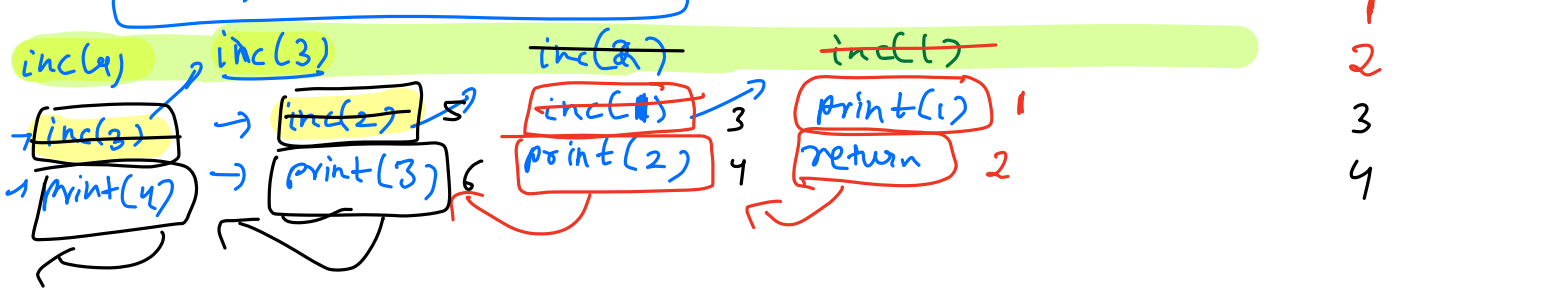
inc(4)




```
1  def inc(n):
2      if n==1:
3          print(1)
4          return
5      inc(n-1)
6      print(n)
```

# print from 1 to n

# print from 1 to (n-1)




```
def inc(n):
  if n == 1:
    print(1)
    return 5
  x = inc(n - 1)
  print("-----", x)
  print(n)
  return
```

(variable) x: Literal[5] | None

inc(5)

inc(4)

inc(3) x=None

inc(2) x=5

inc(1)

5

None

by default last statement is return.

Output

```
1
----- 5
2
----- None
3
----- None
4
----- None
5
```

```
[8] inc(5)
```

Python 3.6
(known limitations)

```
1 def sum(N):
2     if N == 1:
3         return 1
4
5     return sum(N - 1) + N
6
7 print(sum(4))
```

Edit this code

line that just executed
next line to execute

<< First | < Prev | Next > | Last >>

Done running (18 steps)

Customize visualization

Print output (drag lower right corner to resize)

10

Frames Objects

Global frame
sum

function
sum(N)