Lecture 13, 18 September 2025

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

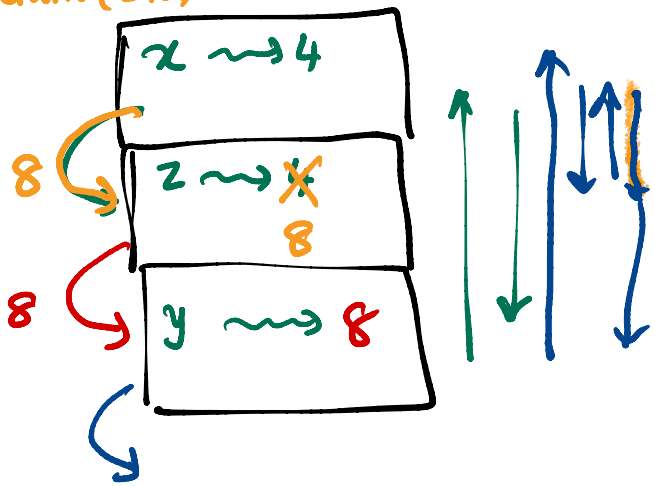
Calling functions

```
def double(x):
    return(2*x)

def set(z):
    z = double(z)
    z = triple(z)
    return(z)

y = set(4)
print(y)
```

```
def triple(a):
    return(3*a)
```



Recursive functions ⟷ Induction

Factorial

Base case $0! = 1$

$n! = (n-1)! \times n$

$(n-2)! \times (n-1)$

induction

$4! = 3! \times 4$

$3!$

$2! \times 3$

$1! \times 2$

$0! \times 1$

1

$-1!$? $\Rightarrow$ $(-2!) \times -1$ $\Rightarrow$ $-3! \times -2 \times -1$ ---

$2.5!$? $\Rightarrow$ $1.5! \times 2.5$ $\Rightarrow$ $0.5! \times 1.5 \times 2.5$

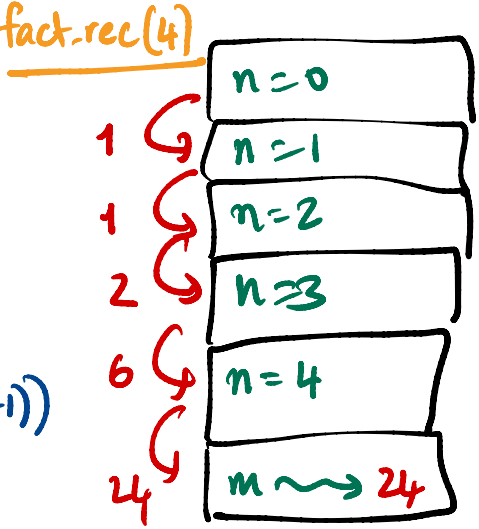
$\Rightarrow$ $-0.5! \times 0.5 \times 1.5 \times 2.5$

```
def fact(n):
    ans = 1
    while n >= 1 :
        ans = ans * n
        n = n - 1
    return (n)
```

```
def fact_rec (n):
    if n == 0:
        return (1)
    else:
        return (n * fact_rec(n-1))
```

```
def fact_rec(n):
    if n==0:
        return (1)
    else:
        return (n * fact_rec(n-1))

m = fact_rec(4)
```



```
def fib(n):
    if n==0:
        return(0)
    elif n==1:
        return(1)
    else:
        return(fib(n-1)+fib(n-2))
```

0 1 1 2 3 5 8 ...

$fib(0) = 0$

$fib(1) = 1$

$fib(n) = fib(n-1) + fib(n-2)$

Recursion / induction beyond numbers

Data Structures

Lists

Base case : `[]` length(l)

"Plus One" : `l.append(v)`

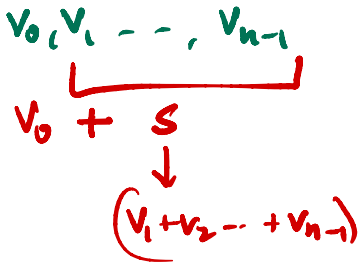
`l.insert(0,v)`

`[v]+l`

```
def length(l):
    if l==[]:
        return(0)
    else:
        # length of "rest", add 1
        1 + length(l[1:])
```

or `1 + length(l[:-1])` ✓

```
def sum(l):
    if l==[]:
        return(0)
    else:
        return (l[0]+ sum(l[1:]))
```



```
def find(l,x):
    for y in l:
        if x == y:
            return(True)
    return(False)
```

Inductively

x in l?

[] ? False

x == l[0] or x in l[1:]

```
def find(l,x):
    if l == []:
        return(False)
    else:
        return(x == l[0] or find(l[1:],x))
```

Insert v into a sorted list (ascending)

⟶ Find the place to insert

Try iteratively

Inductively insert(v, l) Base case: [] ⟶ [v]

v ≤ l[0] ⟶ [v] + l

else ⟶ l[0] + insert(v, l[1:])

```
def insert(v,l):
    if l==[]:
        return([v])
    elif v<=l[0]:
        return([v]+l)
    else:
        return([l[0]]+insert(v,l[1:]))
```