Lecture 11, 19 September 2024

Madhavan Mukund

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

Programming and Data Structures with Python

Inductive definitions

Define $f(n)$ in terms of $f(n-1)$, ...

Base Case — Often $f(0)$ or $f(1)$

$$n! = n \cdot n-1 \cdot n-2 \cdots \cdot 1$$

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

$$1! = 1$$

$$fact(n) = n \cdot fact(n-1)$$

fact(3) = 3 × fact(2)

└ 2 × fact(1)

└ 1

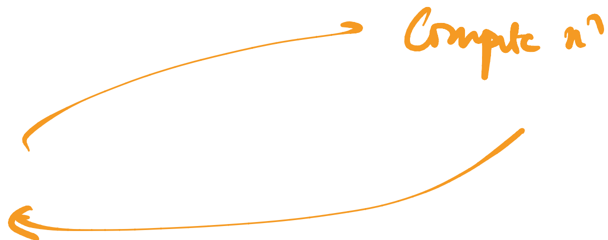
2 = 2 × 1

= 3 × 2

= 6

$$f(n) = n \times g(n)$$

$$f(n) = n \times n^2$$



$fib(0) = 0$

$fib(1) = 1$

0 1 1 2 3 5 8 ...

$\underline{n \geq 2}$ $fib(n) = fib(n-1) + fib(n-2)$

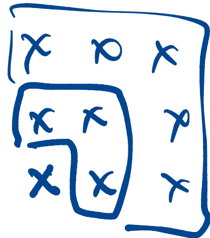
$fib(2) = 1 + 0 = 1$

$fib(3) = 1 + 1 = 2$

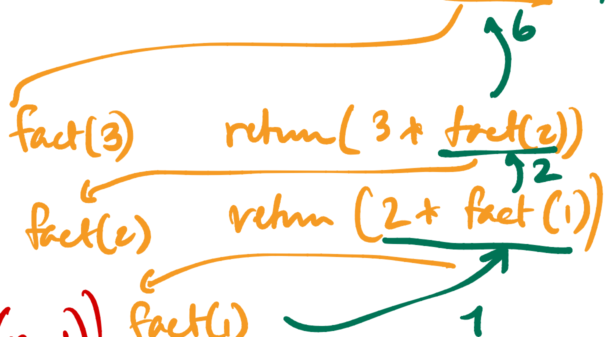
$fib(4) = 2 + 1 = 3$

$$(n+1)^2 = n^2 + 2n + 1$$

$$sqr(n+1) = (2n+1) + sqr(n)$$

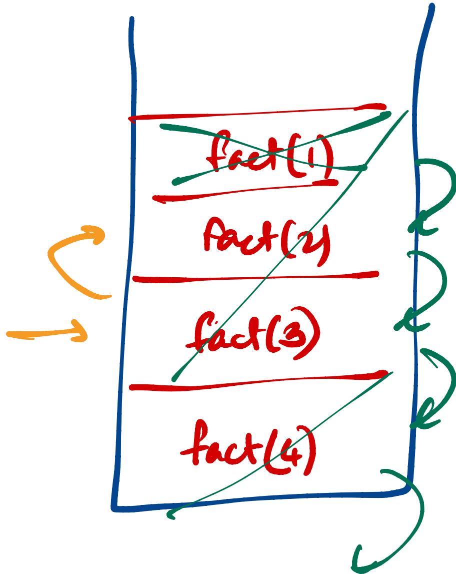


```
def fact(n):
    if n == 1:
        return(1)
    else
        return (n * fact(n-1))
```

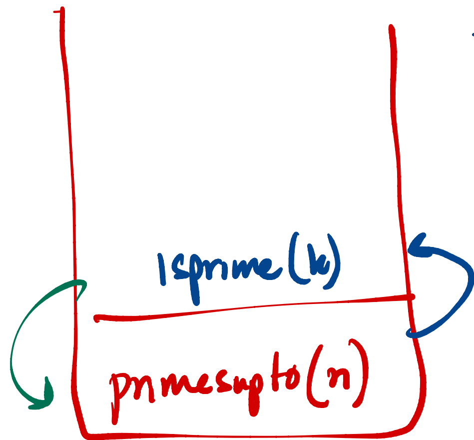


In practice

"Stack"

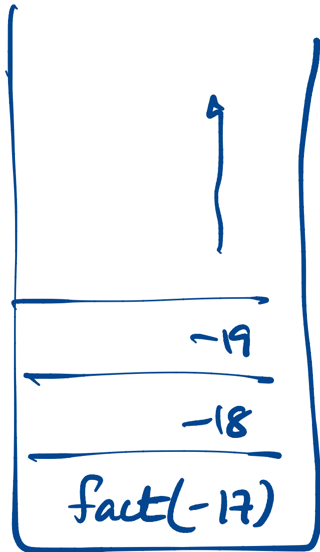


True/False




```
for i in range(2,n+1):
    if isprime(i):
        ...
```

```
def fact(n):
    if n==1 :        n<=1 :
        return(1)
    else:
        return (n * fact(n-1))
```



"Halting Problem"

of Alan Turing

```
def fact(n):
    ans = 1
    while n > 0:
        ans = ans * n
        n = n-1
    return(ans)
```

Binary search [0 .. n-1] (l = 0, r = n-1)

mid = (l+r)//2

BinarySearch(l,r)

if x > A[mid]
  l = mid+1
if x < A[mid]
  r = mid-1

If $f(0)$ is defined

&

I can define $f(n+1)$ from $f(0), \ldots, f(n)$

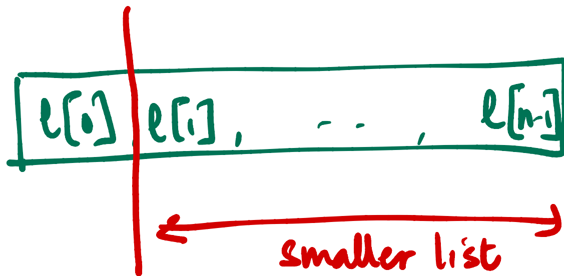
then

$f(n)$ is defined for **all** $n \geq 0$

Structural Induction

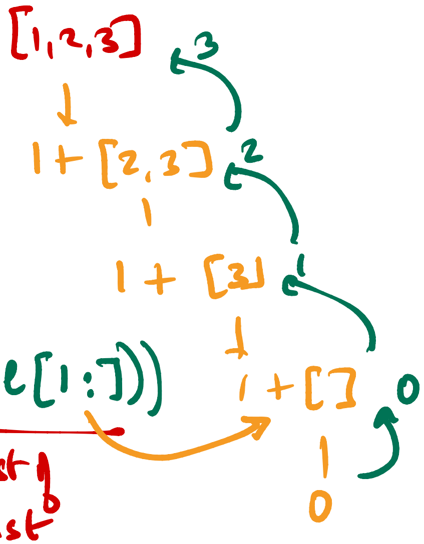
len(l)

```
count = 0
for x in l:
    count = count + 1
return(count)
```



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

rest of list



Sum(l)

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

Sorted list :
(not array)

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

12

Insert 12

(maintain ascending order)

Discard duplicates

```
def sortedinsert(l,v):
    for i in range(len(l)):
        if v < l[i]:
            return(l[:i] + [v] + l[i:])
```

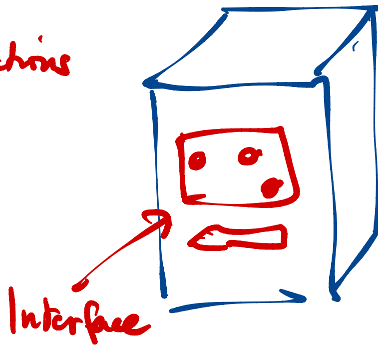
Inductively

```
if v < l[0]:
    [v]+l
else:
    l[:1] + sortedInsert(l[1:],v)
    [l[0]]
```

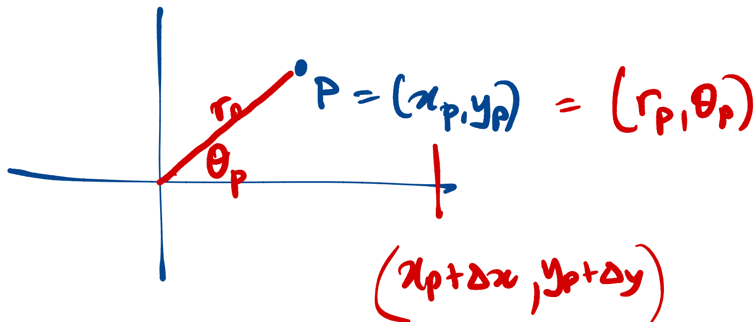
Abstract Data Types (ADT)

List — allowed functions

Dictionary



Point in space (2D)



- translate $(\Delta x, \Delta y)$
- distance from origin()

Separate interface - functionality

from

implementation - efficiency

PL needs a way to define ADTs

l = []   let l be a new List

d = {}

l = createlist()