CS21: INTRODUCTION TO COMPUTER SCIENCE

Prof. Mathieson

Fall 2018

Swarthmore College

Outline Dec 3:

- Recursion

Notes

•Lab 10 is due TONIGHT!
•Lab 11 (last lab!) is due Sunday night (shorter)

Begin: recursive functions

What is recursion?

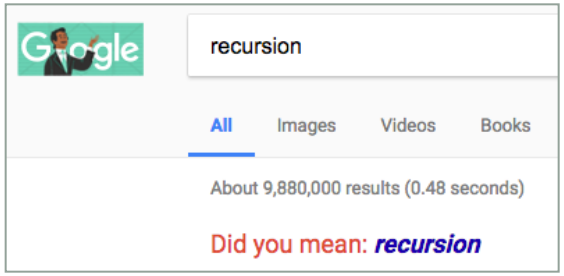
- A recursive function calls **itself**.

- Usually two key components to a recursive function:
  1) A **base case** or way to stop the recursion
  2) A **recursive call** using a modified argument

What is recursion?

- A recursive function calls **itself**.

- Usually two key components to a recursive function:
  1) A **base case** or way to stop the recursion
  2) A **recursive call** using a modified argument



"To understand recursion, you must first understand recursion"

Image: "Matryoshka Nesting Dolls: Meaning of Russian Wooden Stacking Doll" - Legomenon

Factorial function

n	recursion	n!
0	base case	1
1		
2		
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2		
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3		
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4		
5		
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5		
6		

n! = n*(n-1)*(n-2)…3*2*1

or

n! = n*(n-1)!

with

base case: 0! = 1

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5	5*24	120
6		

$n! = n*(n-1)*(n-2)\ldots3*2*1$

or

$n! = n*(n-1)!$

with

base case: $0! = 1$

Factorial function

n	recursion	n!
0	base case	1
1	1*1	1
2	2*1	2
3	3*2	6
4	4*6	24
5	5*24	120
6	6*120	720

$n! = n*(n-1)*(n-2)\ldots3*2*1$

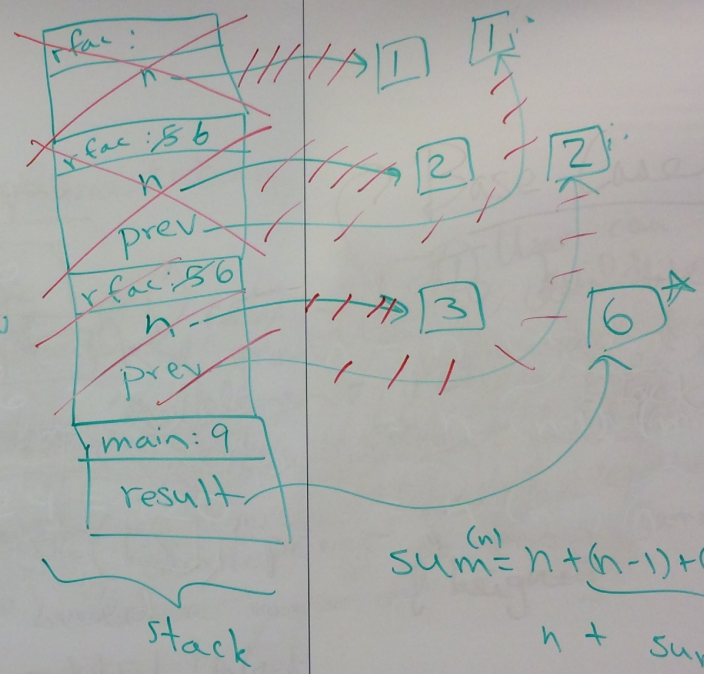
or

$n! = n*(n-1)!$

with

base case: $0! = 1$

```
1  def rfac(n):
2      if n <= 1:
3          return 1
4      else:
5          prev = rfac(n-1)
6          return n * prev
7
8  def main():
9      result = rfac(3)
10 main()
```



$$sum(n) = n + \underbrace{(n-1)+(n-2)\cdots 3+2+1}_{n + sum(n-1)}$$

Recursion stack diagram

Reverse String (recursion)

Reversing a string

```
                                reverse("summer")

                        reverse("ummer")                + "s"

                    (reverse("mmer")            + "u") + "s"

                ((reverse("mer")        + "m") + "u") + "s"

            (((reverse("er")        + "m") + "m") + "u") + "s"

        ((((reverse("r")     + "e") + "m") + "m") + "u") + "s"

    (((((reverse("") + "r") + "e") + "m") + "m") + "u") + "s"
```