Discussion 3:
Recursion & Tree Recursion

Caroline Lemieux (clemieux@berkeley.edu)

February 14, 2018

Administrativa

Homeworks

HW 3 due next Thursday 2/21

Projects

Optional Hog strategy contest ends next Friday 2/22

CSM

Sign up for this by the weekend: see piazza post @684

Recursion

Recap Quiz

Join: socrative.com
Room: CARO61A

WWPD?

```
def count_down(n):
    if n == 0:
        print(“TAKEOFF!”)
    else:
        print(“Seconds left:”, n)
        count_down(n-1)

>>> count_down(2)
```

(D): I don’t know

(A)

```
Seconds left: 2
Seconds left: 1
TAKEOFF!
```

(B)

```
Seconds left: 2
Seconds left: 1
Seconds left: 0
TAKEOFF!
```

(C)

```
Seconds left: 2
Seconds left: 2
Seconds left: 2
```

… (forever)

WWPD?

```
def count_down(n):
    if n == 0:
        print(“TAKEOFF!”)
    else:
        print(“Seconds left:”, n)
        count_down(n)

>>> count_down(2)
```

(D): I don’t know

(A)

```
Seconds left: 2
Seconds left: 2
Seconds left: 1
TAKEOFF!
```

(B)

```
Seconds left: 2
Seconds left: 1
Seconds left: 0
TAKEOFF!
```

(C)

```
Seconds left: 2
Seconds left: 2
Seconds left: 2
```

… (forever)

WWPD?

```
def count_down(n):
    print(“Seconds left:”, n)
    count_down(n-1)

>>> count_down(2)
```

(D): I don’t know

(A)

```
Seconds left: 2
Seconds left: 2
Seconds left: 1
TAKEOFF!
```

(B)

```
Seconds left: 2
Seconds left: 1
Seconds left: 0
Seconds left: -1
```

… (forever)

(C)

```
Seconds left: 2
Seconds left: 2
Seconds left: 2
```

… (forever)

WWPD?

```
def count_down(n):
    if n == 0:
        print(“TAKEOFF!”)
    else:
        print(“Seconds left:”, n)
        count_down(n-1)

>>> count_down(2)
```

(D): I don’t know

(A)

```
Seconds left: 2
Seconds left: 1
TAKEOFF!
```

(B)

```
Seconds left: 2
Seconds left: 1
Seconds left: 0
TAKEOFF!
```

(C)

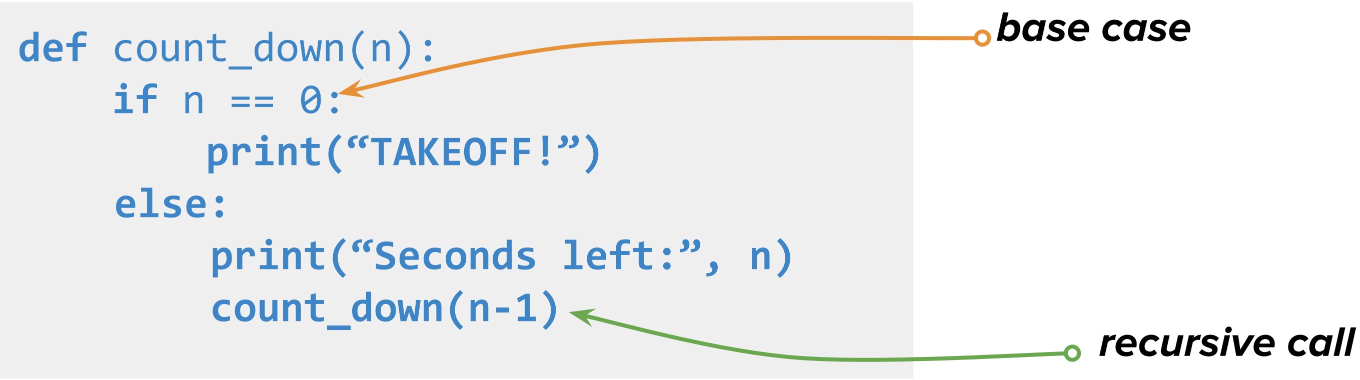
```
Seconds left: 2
Seconds left: 2
Seconds left: 2
```

… (forever)

Recursive functions

A **recursive function** is one that calls itself in its own body.

Anatomy of a recursive function



```
def count_down(n):
    if n == 0:
        print(“TAKEOFF!”)
    else:
        print(“Seconds left:”, n)
        count_down(n-1)
```

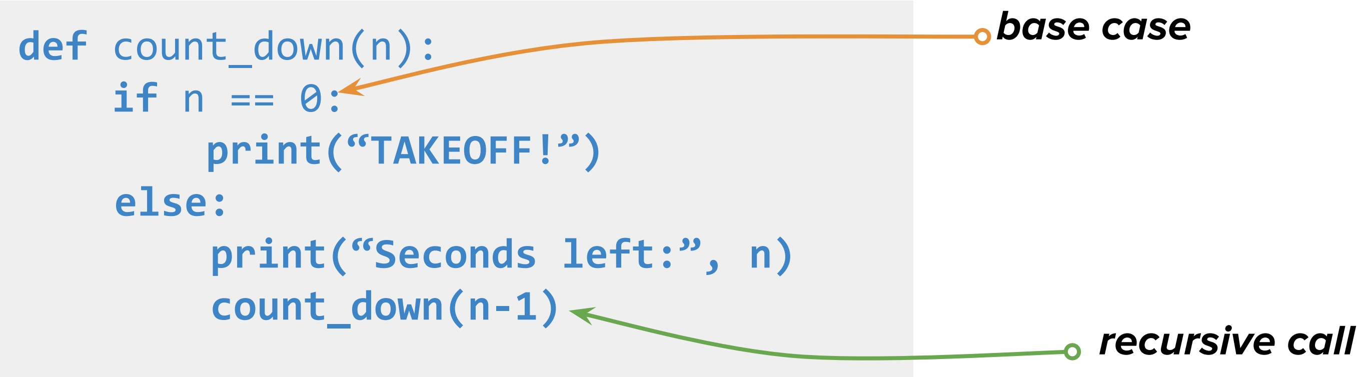
base case

recursive call

Recursive functions

A **recursive function** is one that calls itself in its own body.

Anatomy of a recursive function



```
def count_down(n):
    if n == 0:
        print(“TAKEOFF!”)
    else:
        print(“Seconds left:”, n)
        count_down(n-1)
```

base case

recursive call

What questions do you have about recursion?

Steps for writing recursive functions

1. *Figure out your **base case**:* What is the simplest argument we could possibly get?

    For example, factorial(0) is 1 by definition.

2. *Make a recursive call with a simpler argument:* Simplify your problem, and assume that a recursive call for this new problem will simply work. This is called the "**leap of faith**".

    For factorial, we reduce the problem by calling factorial(n-1).

3. *Use your recursive call to solve the full problem:* Remember that we are assuming the recursive call works. With the result of the recursive call, how can you solve the original problem you were asked?

    For factorial, we just multiply (n – 1)! by n.

Attendance

links.cs61a.org/caro-disc

Tree Recursion

Tree Recursion

Tree recursion: when a function calls itself more than once in one frame.

Often, each recursive call represents a “choice” or “possibility”

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

Fibonacci sequence: the nth term is the sum of the previous two terms

```
def count_partitions(n, m):
    if n == 0:
        return 1
    elif n < 0:
        return 0
    elif m == 0:
        return 0
    else:
        with_m = count_partitions(n-m, m)
        without_m = count_partitions(n, m-1)
        return with_m + without_m
```

Count partitions: the number of ways to partition n elements into groups of size at most m