Discussion 3

www.yellkey.com/whatever

Midterm 1

- Congrats! First exam done

- Everyone learns at their own pace
  - don't compare yourself to others

- Lower exam averages are normal in CS classes

Midterm 1

- Midterm 1 is a small part of 61a
  - 13% Midterm 1
  - 16% Midterm 2
  - 25% Final
  - 45% Assignments

Midterm 1: Resources

- Advising appointments
  - https://cs61a.org/articles/advising/

- CSM
  - Type @879 into piazza search bar
    - Title: "[CSM] Section Signups Reminder"

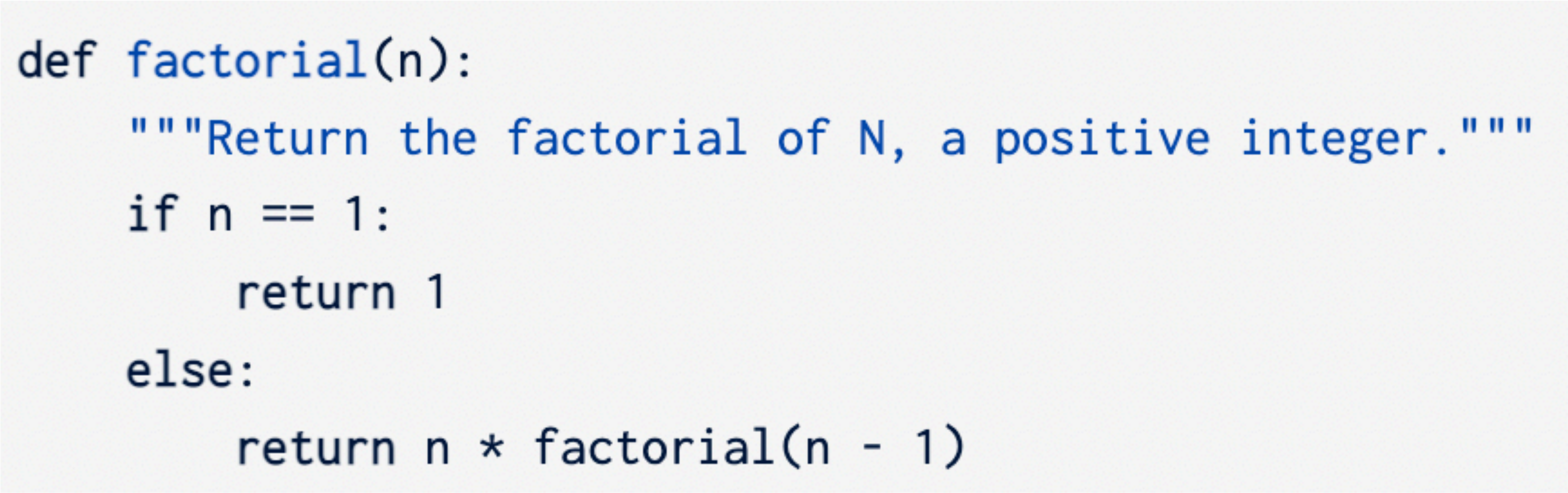
- Course Staff

Feedback!

- Go a little faster
  - I agree
- Responses to student questions are a little long

Recursion

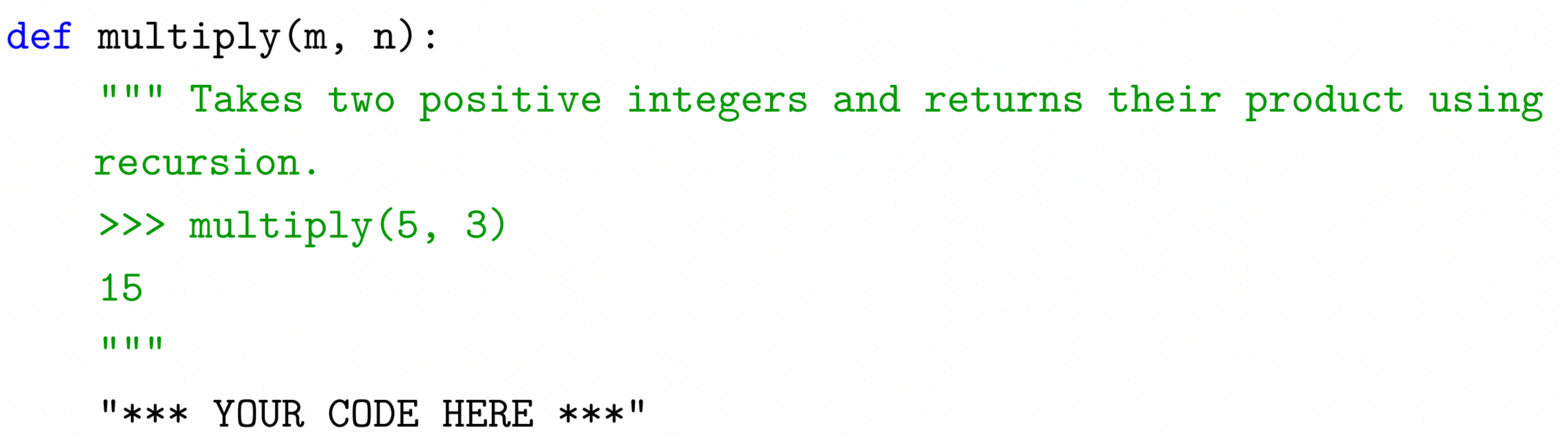
1. **Base Case:** simplest function input
2. **Recursive Call** (smaller problem)
   - **recursive leap of faith** (trust yourself!)
3. **Solve the larger problem**



```
def factorial(n):
    """Return the factorial of N, a positive integer."""
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

Q1: Recursive Multiplication

Write a function that takes two numbers m and n and returns their product. Assume m and n are positive integers. Use recursion, not mul or *.



```
def multiply(m, n):
    """ Takes two positive integers and returns their product using
    recursion.
    >>> multiply(5, 3)
    15
    """
    "*** YOUR CODE HERE ***"
```

Hint: 5 * 3 = 5 + (5 * 2) = 5 + 5 + (5 * 1).

Q1: Recursive Multiplication

```
def multiply(m, n):   5×3
    if n==1:
        return m

    return multiply(m,n-1) +m
                   5×2
```

m=1 ?

3*1 = 3 =3

$\overset{m}{1} * \overset{n}{3} = 1+1+1 = 3$

3+3+3+3+3

5×3 = 5+5+5 = 15

Hint: $5 * 3 = \boxed{5 + (5 * 2)} = 5 + 5 + (5 * 1)$.

Q1: Recursive Multiplication

For the recursive case, what does calling multiply(m - 1, n) do? What does calling multiply(m, n - 1) do? Do we prefer one over the other?

Q3: Find the Bug

Find the bug with this recursive function.

```
def skip_mul(n):
    """Return the product of n * (n - 2) * (n - 4) * ...

    >>> skip_mul(5) # 5 * 3 * 1
    15
    >>> skip_mul(8) # 8 * 6 * 4 * 2
    384
    """
    if n = = 1 : return 1
    if n == 2:
        return 2
    else:
        return n * skip_mul(n - 2)
```

1. Base Case
2. Recursive Call
3. Solve the larger problem

Q5: Merge Numbers

Write a procedure merge(n1, n2) which takes numbers with digits in decreasing order and returns a single number with all of the digits of the two, in decreasing order. Any number merged with 0 will be that number (treat 0 as having no digits). Use recursion

```
>>> merge(31, 42)
4321
>>> merge(21, 0)
21
>>> merge (21, 31)
3211
```

```
>>> merge(54, 32)
5432
```

% 10

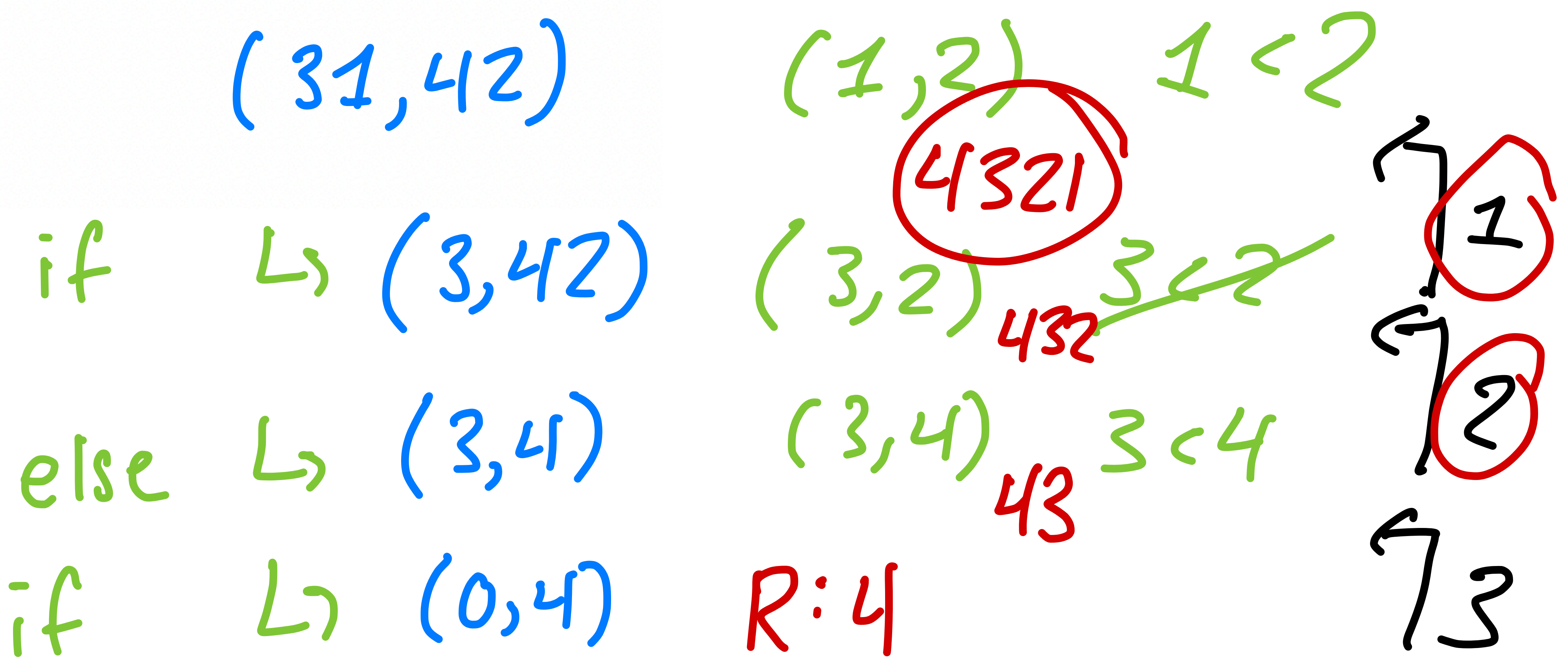
// 10

3-4m

Q5: Merge Numbers

```
def merge(n1, n2): # n1 // 10
    """ Merges two numbers by digit in decreasing order
    >>> merge(31, 42)
    4321
    if n1 ==0:
        return n2
    if n2 == 0:
        return n1

    if n1 % 10 < n2 % 10:
        merge (n1//10, n2) * 10 + (n1 % 10)      # n1%10 at end
    else:
        merge(n1, n2// 10) * 10 + (n2 % 10)      # n2%10 at end
```



(31,42) (1,2) 1<2

4321

if ↳ (3,42) (3,2) 432 ~~3<2~~ 1

else ↳ (3,4) (3,4) 43 3<4 2

if ↳ (0,4) R:4 3

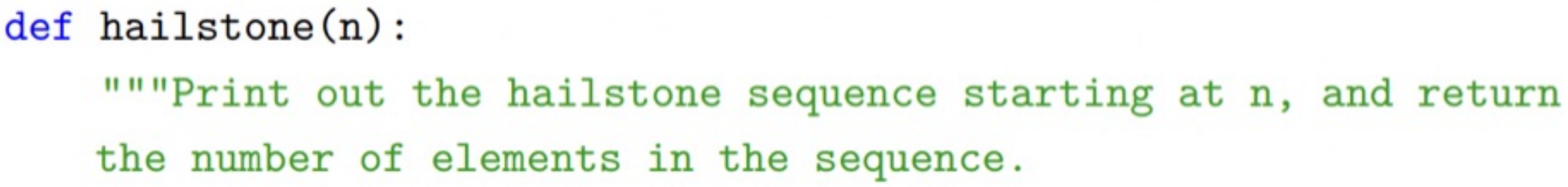
Q4: Recursive Hailstone

Recall the hailstone function from Homework 2. First, pick a positive integer n as the start. If n is even, divide it by 2. If n is odd, multiply it by 3 and add 1. Repeat this process until n is 1. Write a recursive version of hailstone that prints out the values of the sequence and returns the number of steps.

```
>>> a = hailstone(10)
10
5
16
8
4
2
1
>>> a
7
```

Q4: Recursive Hailstone

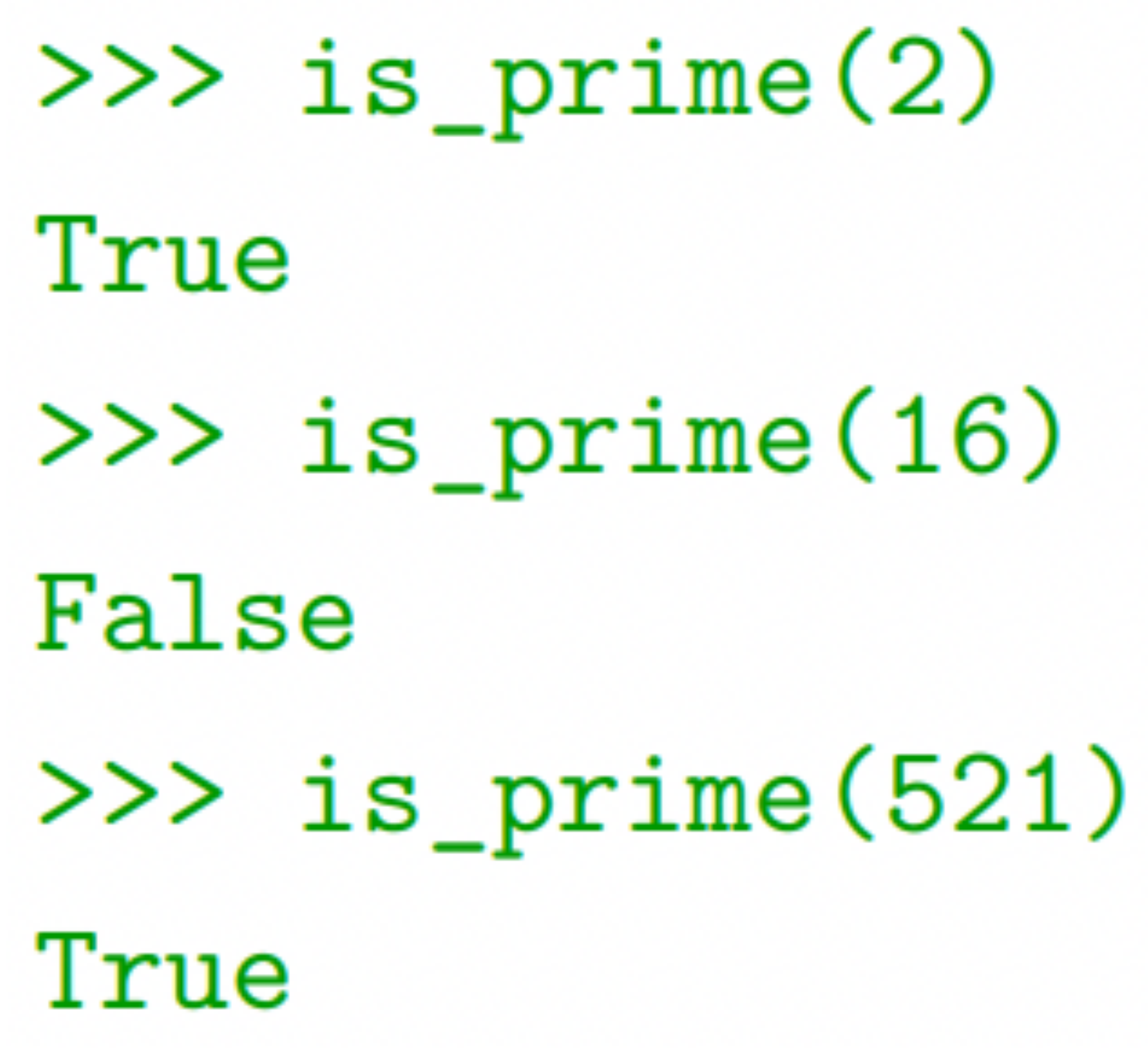
pick a positive integer n as the start. If n is even, divide it by 2. If n is odd, multiply it by 3 and add 1. Repeat this process until n is 1. Write a recursive version of hailstone that prints out the values of the sequence and returns the number of steps.



```python
def hailstone(n):
    """Print out the hailstone sequence starting at n, and return
    the number of elements in the sequence.
```

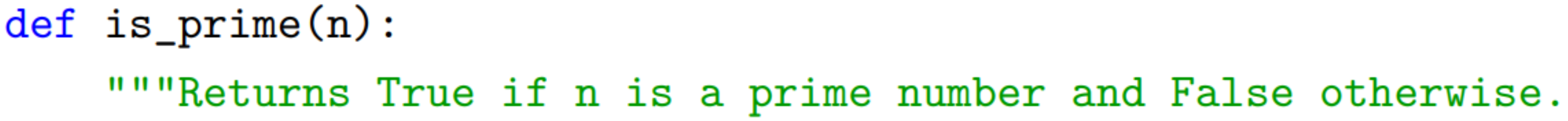
Q6: Merge Numbers

Write a function is_prime that takes a single argument n and returns True if n is a prime number and False otherwise. Assume n > 1.



```
>>> is_prime(2)
True
>>> is_prime(16)
False
>>> is_prime(521)
True
```

Q6: Merge Numbers



```python
def is_prime(n):
    """Returns True if n is a prime number and False otherwise.
```

Feedback + Attendance

www.yellkey.com/another

Pwd: granola