RECURSION AND HIGHER ORDER FUNCTIONS

COMPUTER SCIENCE MENTORS 61A

February 6 to February 10, 2017

1 Recursion

Every Recursive function has three things.

1. One or more base cases
2. One or more ways to break the problem down into a smaller problem
    - E.g. Given a number as input, we need to break it down into a smaller number
3. Solve the smaller problem recursively; from that, form a solution to the original problem

1. Complete the definition for `num_digits`, which takes in a number `n` and returns the number of digits it has.

```
def num_digits(n):
    """Takes in an positive integer and returns the number of
    digits.

    >>> num_digits(0)
    1
    >>> num_digits(1)
    1
    >>> num_digits(7)
    1
    >>> num_digits(1093)
    4
    """
```

2. Write a function `is_sorted` that takes in an integer `n` and returns true if the digits of that number are increasing from right to left.

```
def is_sorted(n):
    """
    >>> is_sorted(2)
    True
    >>> is_sorted(22222)
    True
    >>> is_sorted(9876543210)
    True
    >>> is_sorted(9087654321)
    False
    """
```

2 Environment Diagrams

1. Draw the environment diagram that results from running the code.

```
def bar(f):
    def g(x):
        if x == 1:
            return f(x)
        else:
            return f(x) + g(x - 1)
    return g

f = 4
bar(lambda x: x + f)(2)
```

2. What would change here?

```
x = 20

def bar():
    return lambda y: x-y

def foo(y):
    x = 5
    return bar

y = foo(7)
z = y()
print(z(2))
```