RECURSION AND HIGHER ORDER FUNCTIONS

COMPUTER SCIENCE MENTORS 61A

February 8 to February 12, 2016

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. Write `numDigits`, which takes in a number `n` and returns the number of digits it has:

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

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

Solution:

```
if n == 0:
  return 0
else:
  return 1 + numDigits(n//10)
```

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

```
def isSorted(n):
    """Return true if the digit is in increasing order from
       rightmost digit to leftmost digit. (Consecutive same
       digits are allowed). Also return true if it has only
       onedigit. Return false otherwise.
    >>> isSorted(2)
    True
    >>> isSorted(22222)
    True
    >>> isSorted(9876543210)
    True
    >>> isSorted(9087654321)
    False
    """
```

Solution:

```
right_digit = number % 10
rest =  number // 10
if rest == 0:
    return True
elif right_digit > rest % 10:
    return False
else:
    return isSorted(rest)
```

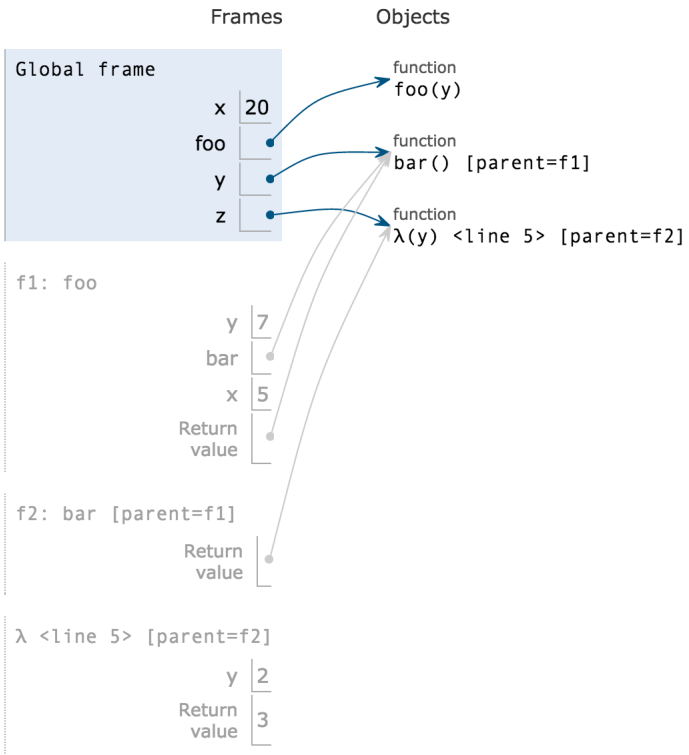
2 Environment Diagrams

3. Draw an environment diagram for the following code:

```
x = 20
def foo(y):
    x = 5
    def bar():
        return lambda y: x-y
    return bar

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

Solution:
Output: 3



4. What would change here?

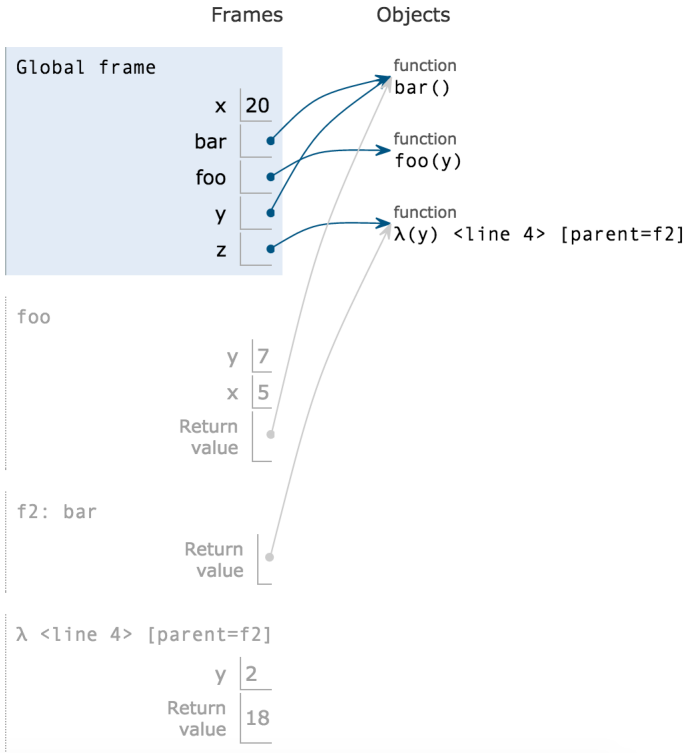
```
x = 20

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

def foo(y):
    x = 5
    return bar

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

Solution:
Output: 18



3 Higher Order Functions

1. Write a higher order function that passes the following doctests. *Challenge:* Write the function body in one line.

```
"""
>>> from operator import add, mul
>>> a = mystery(add, 3)
>>> a(4) #equivalent to add(3,4)
7
>>> a(12)
15
>>> b = mystery(mul, 5)
>>> b(7) #equivalent to mul(5,7)
35
>>> b(1)
5
>>> c = mystery(lambda x, y: x*x + y, 4)
>>> c(5)
21
>>> c(7)
23
"""
```

Solution:

```
def mystery(f, x):
    def helper(y):
        return f(x,y)
    return helper

One-line solution: return lambda y : f(x,y)
```

2. What do these print out?

```
>>>foo = mystery(lambda a,b: a(b), lambda c: 5 + square(c))
>>>foo(-2)
```

Solution: 9