Recursion

Every Recursive function has three things.

One or more base cases

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

Solve the smaller problem recursively; from that, form a solution to the original problem

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

def num$_digits(n) : """Takesinanpositiveintegerandreturnsthenumberofdigits.$

¿¿¿ num$_digits(0)1 >>> num_digits(1)1 >>> num_digits(7)1 >>> num_digits(1093)4"""$

[1in] if n ¡ 10: return 1 else: return 1 + num$_digits(n//10)$

Write a function `is_sorted` that takes in an integer `n` and returns true if the digits of that number a

def is$_sorted(n) : """Returntrueifthedigitisinincreasingorderfromrightmostdigittoleftmostdigit.(C$

¿¿¿ is$_sorted(2)True >>> is_sorted(22222)True >>> is_sorted(9876543210)True >>> is_sorted(90876$

[1in] right$_digit = nrest = n//10ifrest == 0 : returnTrueelifright_digit > restreturnFalseelse : retu$

Environment Diagrams 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))

[2in] Output: 3

[scale=0.5]img/foobar.png

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))

[0.3in] Output: 18

[scale=0.5]img/foobar2.png

Higher Order Functions

Write a higher order function that passes the following doctests. *Challenge:* Write the function body

""" ¿¿¿ from operator import add, mul ¿¿¿ a = mystery(add, 3) ¿¿¿ a(4) equivalent to add(3,4) 7 ¿¿¿

[2in] def mystery(f, x): def helper(y): return f(x,y) return helper

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

What do these print out? ¿¿¿foo = mystery(lambda a,b: a(b), lambda c: 5 + square(c)) ¿¿¿foo(-2)

9