Project 1 -- Python Warm-up

Instructions

Update your report every week. Let me know you where you succeeded and where you could use more help. Turn in your progress report as a PDF file and your lab0.py file in the assignments folder for the current week. So, during week 1 of the class, go to Worldclass > MSDS688 > Assignments > Week 1 -- Project 1. Please post your questions to the Project 1 discussion thread.

Remember, you get credit by either:

1. Making progress in terms of number of tests passing, *or*
2. Describing the roadblock that has stopped you, the sequence of steps you took to solve it, and the reasons you took them. Think of applying the scientific method to programming.

Getting Help

For help with Markdown, check out the concise and helpful Mastering Markdown that lets you modify demo code and see the resulting changes live.

For help generating a PDF file from your progress report markdown file. First, add the Markdown PDF plugin to VSCode. Then follow these directions.

See the Project 1 video(s) in the discussions folder for help in completing these steps. Please post a question if you get stuck.

Please post your questions to the discussion forum for this project.

What progress did you make?

As for Lab0, I did complete all code and answer, and I passed all 68 tests.

What code are you most proud of, and why?

Primes up to function is the most code I am proud of because I had to figure out how to check all numbers from 0 to the number that passed to the function. Also, because I had experience in Java, I was able to use for loops and recursion easily.

```
def primes_up_to(x):
    """Given a number x, returns an in-order list of all primes up to and
including x"""

    primes = []

    for i in range(0, round(x)+1):
        if i == 0 or i == 1:
```

```
            continue
        else:
            for j in range(2, int(i/2)+1):
                if i % j == 0:
                    break
            else:
                primes.append(i)
    return primes
```

Did you get stuck?

I got stuck in functions that return function section fora while until I figured out that it is similar to classes and methods and how the function is built inside the parent function and how it inherits the variables from the father function. Also, once I read about decorators in python, I immediately knew what it is, and what confused me is the different naming of things between python and java when it serves the same purpose.

What steps did you take to solve the problem?

I did ask Dr. Busch about it in class, and it helped to understand. However, I was still confused, I googled a lot without getting an answer, then I emailed Dr. Busch again, and he told me to read about decorators, and then I was able to complete the code.

Did you succeed? If so, what worked and why?

Yes, at first, I thought I can write another function outside the original function and return the second function, but it didn't work because it needed to put another argument to the second function, which the test won't pass, and even if I put a random number, it won't return the function, it will return a number. then I wrote a function inside the first one in which take an argument and it will multiply the number with the M (the first function argument) and then return the second function and it worked.

Code

```
# MIT 6.034 Lab 0: Getting Started
# Written by jb16, jmn, dxh, and past 6.034 staff

from point_api import Point

#### Multiple Choice ##########################################################

# These are multiple choice questions. You answer by replacing
# the symbol 'None' with a letter (or True or False), corresponding
# to your answer.

# True or False: Our code supports both Python 2 and Python 3
# Fill in your answer in the next line of code (True or False):
ANSWER_1 = False

# What version(s) of Python do we *recommend* for this course?
#   A. Python v2.3
```

```
#   B. Python V2.5 through v2.7
#   C. Python v3.2 or v3.3
#   D. Python v3.4 or higher
# Fill in your answer in the next line of code ("A", "B", "C", or "D"):
ANSWER_2 = "D"


################################################################################
# Note: Each function we require you to fill in has brief documentation        #
# describing what the function should input and output. For more detailed      #
# instructions, check out the lab 0 wiki page!                                 #
################################################################################


#### Warmup ####################################################################

def is_even(x):
    """If x is even, returns True; otherwise returns False"""
    if (x % 2) == 0:
        return True
    else:
        return False


def decrement(x):
    """Given a number x, returns x - 1 unless that would be less than
    zero, in which case returns 0."""
    x = x - 1
    if x > 0:
        return x
    else:
        return 0

def cube(x):
    """Given a number x, returns its cube (x^3)"""
    return x**3

#### Iteration #################################################################

def is_prime(x):
    """Given a number x, returns True if it is prime; otherwise returns False"""
    x = int(x)
    if x > 1:
        for j in range(2, int(x/2)+1):
            if x % j == 0:
                return False
                break
        else:
            return True
    else:
        return False
```

```
def primes_up_to(x):
    """Given a number x, returns an in-order list of all primes up to and
including x"""

    primes = []

    for i in range(0, round(x)+1):
        if i == 0 or i == 1:
            continue
        else:
            for j in range(2, int(i/2)+1):
                if i % j == 0:
                    break
            else:
                primes.append(i)
    return primes


#### Recursion ###############################################################

def fibonacci(n):
    """Given a positive int n, uses recursion to return the nth Fibonacci
number."""
    if n in {0,1}:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)


def expression_depth(expr):
    """Given an expression expressed as Python lists, uses recursion to return
    the depth of the expression, where depth is defined by the maximum number of
    nested operations."""

    if not isinstance(expr, list):
        return 0
    return max(map(expression_depth,expr))+ 1



#### Built-in data types #####################################################

def remove_from_string(string, letters):
    """Given an original string and a string of letters, returns a new string
    which is the same as the old one except all occurrences of those letters
    have been removed from it."""

    for i in letters:
        string = string.replace(i,"")
    return string


def compute_string_properties(string):
```

```
    """Given a string of lowercase letters, returns a tuple containing the
    following three elements:
        0. The length of the string
        1. A list of all the characters in the string (including duplicates, if
           any), sorted in REVERSE alphabetical order
        2. The number of distinct characters in the string (hint: use a set)
    """
    st1 = []
    st1[:] = string
    st1.sort(reverse=True)
    t = (len(string),st1, len(set(string)))
    return t


def tally_letters(string):
    """Given a string of lowercase letters, returns a dictionary mapping each
    letter to the number of times it occurs in the string."""
    tally = {}
    for i in string:
        if i in tally:
            tally[i] += 1
        else:
            tally[i] = 1
    return tally



#### Functions that return functions ###############################################

def create_multiplier_function(m):
    """Given a multiplier m, returns a function that multiplies its input by m."""
    def multiply(x):
        return x*m
    return multiply



def create_length_comparer_function(check_equal):
    """Returns a function that takes as input two lists. If check_equal == True,
    this function will check if the lists are of equal lengths. If
    check_equal == False, this function will check if the lists are of different
    lengths."""
    def compare(z , y):
        if check_equal == True:
            return len(z) == len(y)
        else:
            return len(z) != len(y)
    return compare


#### Objects and APIs: Copying and modifying objects ##############################

def sum_of_coordinates(point):
    """Given a 2D point (represented as a Point object), returns the sum
```

```
    of its X- and Y-coordinates."""
    # x-coor + y-coor
    return point.getX() + point.getY()

def get_neighbors(point):
    """Given a 2D point (represented as a Point object), returns a list of the
    four points that neighbor it in the four coordinate directions. Uses the
    "copy" method to avoid modifying the original point."""
    # first neighbor
    p1 = point.copy()
    p1.setX(point.getX() - 1)
    # second neighbor
    p2 = point.copy()
    p2.setX(point.getX() +1)
    # third neighbor
    p3 = point.copy()
    p3.setY(point.getY()-1)
    # fourth neighbor
    p4 = point.copy()
    p4.setY(point.getY()+1)

    neighbors = [p1,p2,p3,p4]

    return neighbors

#### Using the "key" argument ########################################################

def sort_points_by_Y(list_of_points):
    ## I don't know how to sort it Without reverse argument. the default is False.
    newList = sorted(list_of_points,key=lambda x:x.getY(), reverse=True)
    return newList

def furthest_right_point(list_of_points):
    maxPoint = max(list_of_points, key=lambda x: x.getX())
    return maxPoint


#### SURVEY ##########################################################################

# How much programming experience do you have, in any language?
#     A. No experience (never programmed before this lab)
#     B. Beginner (just started learning to program, e.g. took one programming
class)
#     C. Intermediate (have written programs for a couple classes/projects)
#     D. Proficient (have been programming for multiple years, or wrote programs
for many classes/projects)
#     E. Expert (could teach a class on programming, either in a specific language
or in general)

PROGRAMMING_EXPERIENCE = "D"


# How much experience do you have with Python?
#     A. No experience (never used Python before this lab)
```

```
#     B. Beginner (just started learning, e.g. took 6.0001)
#     C. Intermediate (have used Python in a couple classes/projects)
#     D. Proficient (have used Python for multiple years or in many
classes/projects)
#     E. Expert (could teach a class on Python)

PYTHON_EXPERIENCE = "C"


# Finally, the following questions will appear at the end of every lab.
# The first three are required in order to receive full credit for your lab.

NAME = 'Ahmad Alqurashi'
COLLABORATORS = "No one, except I helped Julio with this lab"
HOW_MANY_HOURS_THIS_LAB_TOOK = "took 4 days, one or two hours a day"
SUGGESTIONS = "Directons are a little bit messy but I figured it out eventually"
#optional
```