**CS/IT 200**

**Lab 0: Python Review**

**Recommended Due Date**: Thursday 9/10/2020, 11:59pm

**NOTE:** This lab is not graded. You will still receive feedback about your submission, but this lab does not count toward your final grade.

**Submission**

* Submit all code in a single zip file to the Lab 0 assignment folder on Kodiak.

**Goal:** Review Python syntax and programming concepts.

**Expected Python Knowledge**

For the purposes of this lab, and this course in general, I assume that you are familiar with the majority of Python syntax and concepts covered in the following sections:

Sections 1.1-1.6, 1.7.2, 1.10-1.11, 2.1-2.2, 2.3.1-2.3.3, 2.4.1-2.4.2.

There may be some details in those sections that were not covered in CS/IT 102 or CS 171. You are free to use any concept covered in Chapters 1-2, including sections not listed above, for this assignment and all future assignments.

If you have used Python before and feel weak on any of the concepts in those sections (or discover that weakness while completing this lab), go back and review those sections. If you are still stuck, **ASK FOR HELP**.

If you are new to Python, take the time to review these sections. Chapter 1 covers Python syntax for basic programming concepts that exist in other common programming languages, such as Java and C#. Chapter 2 describes classes and inheritance in Python. Unlike Java and C#, Python does not require that all code is part of a class.

**Part I**

Write each of the following four functions in a file called lab0.py. In addition, include a main() function that tests your functions with different input.

Make sure to comment your code (this applies to the whole assignment!). At bare minimum, I expect a file header (see syllabus) and a brief comment describing each function and class.

1. Write a function called difference that takes a list of numbers as a parameter and returns the difference between the largest number in the list and the smallest number. You can assume the list given is not empty. Your function should not use any of the following built in functions: min(), max(), sort(), sorted()

* difference([1,2,3,4,5]) 🡪 4
* difference([10,7,42,107,53,99]) 🡪 100
* difference([0.5,0.25,0.4,0.9,-0.1]) 🡪 1.0
* difference([5,5,5,5,5]) 🡪 0

1. Write a function called sum\_even\_cubes that takes a positive integer as a parameter and returns the sum of the cube of every even number that is less than the given integer.

* sum\_even\_cubes(5) 🡪 🡪 72
* sum\_even\_cubes(6) 🡪 🡪 72
* sum\_even\_cubes(9) 🡪 🡪 800
* sum\_even\_cubes(2) 🡪 0

1. Write a function called has\_duplicate that takes a list as a parameter. The function should return True if it finds an element that appears two or more times in the list. If there are no duplicates, return False.
   * has\_duplicate([7,0,3,0]) 🡪 True
   * has\_duplicate([7,6,5,4,3,2,1,0]) 🡪 False
   * has\_duplicate([]) 🡪 False
2. Write a function list\_product that takes two lists of integers as arguments. The function should return a list of integers created by multiplying the values at each index. You can assume the two lists have equal lengths.
   * list\_product([1,2,3], [4,5,6]) 🡪 [4,10,18]
   * list\_product([2,2,2,2], [7,8,9,10]) 🡪 [14,16,18,20]

**Part II**

Create a class called Dog in dog.py. Dogs should have attributes for name, age, and breed. Your constructor should accept parameters for each of those attributes, in that order.

Include getter and setter methods for each attribute. Name your getter methods get\_name, get\_age, get\_breed. Your setter methods should follow the same naming convention.

In addition, include and call a main() function, where you create at least one instance of a Dog and you demonstrate the methods of the class.

**Part III**

Review the class Vector that was distributed with this lab (also described on p. 77-78) and vectordemo.py, which demonstrates some of the features of the Vector class. This class takes advantage of **Operator Overloading**, which allows us to use so-called “special methods” to define how our own classes will act when presented with symbolic operators. For example, the **\_\_add\_\_** method defines what will happen when two Vectors are added using the + operator.

If you have two vectors, v1 and v2, and add them by saying v3 = v1 + v2, then Python calls the \_\_add\_\_ method for v1. The other parameter is set to v2. The function returns a new Vector object, independent from v1 and v2, which we assigned to v3.

Additionally, the \_\_add\_\_ method makes use of a new keyword: **raise**. The raise keyword allows us to “raise” an exception. Rather than handling exceptions raised when an error occurs or bad data is passed to a function, raising an exception allows us to declare that an error has occurred. For example, the raise command on line 23 of the Vector class raises the ValueError exception when the length of the two vectors are different. It would then be the responsibility of whoever misused the + operator to handle the exception (likely using a try-except block).

The vectordemo.py module demonstrates successfully adding two Vectors, as well as handling the error raised by attempting to add two Vectors with different lengths. Note that we never actually call the \_\_add\_\_ method. Instead, Python calls it for us when it encounters the + symbol. If you don’t understand something in the vector or vectordemo modules, ask the professor!

You may wish to review section 1.7.1 for further information about raising exceptions and/or section 2.3.2 for further information about Operator Overloading.

* Complete exercise R-2.12 (p. 104) by modifying the Vector class. Note that the 3 in the problem description is an example and that your solution should work for any value that we might wish to multiply by.
* Running vectordemo.py includes multiple tests of whether your solution works. Make sure that you pass all of the tests to be sure that the \_\_mul\_\_ method you wrote is correct. Do not modify vectordemo.py.

**What to Submit:**

* A zip file containing lab0.py, dog.py, vector.py, and vectordemo.py

**Rubric:**

There is no rubric for this lab since this lab is not graded.