CS 1110 SPRING 2016: LAB 8: PRACTICE FOR A6 (Apr 12-13)

First Name: ____________ **Last Name**: ____________ **NetID**: __________

The lab assignments are very important. Remember this: *The lab problems feed into the assignments and the assignments define what the exams are all about.*

Start *before* your lab meets. We recommend spending an hour or two on the lab *before* coming to your section, so you can use your in-person time to ask questions most efficiently.

Also, this strategy of starting beforehand increases your chances of checking in at your lab section, which will give you more staff-interaction time than consulting hours!

Getting credit. Complete all required blank boxes and lines on this handout. When you are finished, show your written answers to one of the CS 1110 lab staff in your section on Apr 12-13 or in any consulting hours up to and including April 18 (earlier days have shorter lines). The staff member will ask you a few questions to make sure you understand the material, and then swipe your Cornell ID card or directly make a notation in CMS to record your success. This physical piece of paper is yours to keep.

Getting set up. From the Lab webpage, download and unzip `Lab_8.zip` into a folder named (for example) `Lab_8`. In the command shell, navigate the file system so that this folder is THE CURRENT WORKING DIRECTORY. Review the slides from the April 4 and April 6 lectures.

1. Dictionary Basics

(a) Consider this:

```
>>> D = {"A":[1,2,3],"B":[4,5]}
>>> D["A"] = D["B"]
>>> D
```

Without using the computer, what do you think will be displayed?

(b) Consider this:

```
>>> D = {"A":2,"B":3}
>>> D[2] = "A"
>>> D
```

Course authors: D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

Without using the computer, what do you think will be displayed?

(c) Consider this:

```
>>> D = {"A":2,"B":3}
>>> D[2] = D[3]
>>> D
```

We claim the second line produces an error. Does the error stem from the lefthand side of the assignment statement, `D[2]`, or from the righthand side, `D[3]`? Explain.

Now check with the computer, and ask a staff member if you have any questions.

2. Inverting a Dictionary

Assume that `D1` is a dictionary whose keys are strings and whose values are strings. Assume also that the values are all distinct. Fill in the blanks so that upon completion, `D1` hasn't been changed, and all the keys in `D1` are values in `D2`, and all the values in `D1` are keys in `D2`. Thus, if `D1 = {"a":"x" , "b":"y", "c":"z" }` then `D2 = {"x":"a" , "y":"b", "z":"c" }`.

```
D2 = {}
for d in D1:

        ------------------------------   =    ---------------------------
```

(As you can see, our solution is one line, but you can use more than one line if you need to.)

3. Dictionaries and For-Loops

For problems (a) and (b) assume

```
Z1 = {"Lion":3, "Zebra":5, "Ape":2,"Bear":4,"Mouse":500,"Canary":12}
Z2 = {"Lion":5, "Zebra":5, "Chimp":5,"Badger":3,"Shrew":500}
```

For each of these problems, you can check out your solutions by modifying the given module `DictFunctions.py`.

(a) Consider the following function:

```
def F1(D1,D2):
    """ Specification omitted for lab purposes.
    PreC: D1 and D2 are dictionaries.
    """
    k=0
    for d1 in D1:
        if d1 not in D2:
            k+=1
    for d2 in D2:
        if d2 not in D1:
            k+=1
    return k
```

What is the value of `F1(Z1,Z2)` for the `Z1` and `Z2` given above? Describe in English what this function returns in general.

(b) Consider the following function:

```
def F2(D1,D2):
    """Specification omitted for lab purposes.
    PreC: D1 and D2 are dictionaries.
    """
    k=0
    for d1 in D1:
        # d1 is a key in D1
        for d2 in D2:
            # d2 is a key in D2
                k+=1
    return k
```

What is the value of `F2(Z1,Z2)`? Describe in English what this function returns in general.

(c) Complete the function so that it would perform as specified:

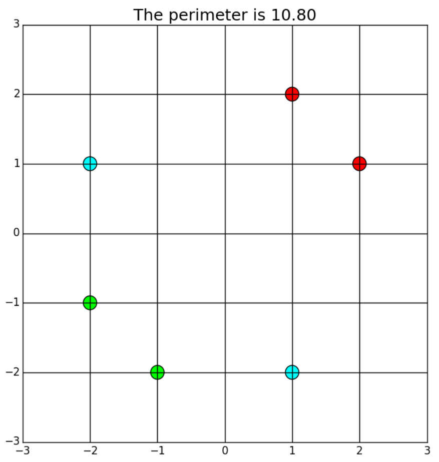
```
def KeysInBoth(D1,D2):
    """ Returns an int whose value is the number of keys that
    are in both D1 and D2.

    PreC: D1 and D2 are dictionaries.
    """
    k=0
    for d1 in D1:

        if ____________________________________________________:

            k+=1
    return k
```

(d) Complete the function so that it would perform as specified. (You may want to ask yourself why there is no loop over D2.)

```
def ItemsInBoth(D1,D2):
    """ Returns an int whose value is the number of items that
    are in both D1 and D2.

    PreC: D1 and D2 are dictionaries.
    """

    k=0
    for d1 in D1:

        if ______________________________________________:

            k+=1
    return k
```

4. The Point Class

Study the given module ShowPointClass.py and run it. In this problem you are to complete the skeleton module ShowPointClassLab.py so that it produces a graphic like this:



This will involve (1) calling the constructor to create three points, (2) using the method ShowPoint to display them, (3) using the Dist method to compute the perimeter of the triangle that they define, (4) using the `Reflect` method to reflect the three points across the 45-degree line, and (5) using `ShowPoint` to display the reflected points. On the following page we have reproduced the skeleton `ShowPointClassLab.py` with "gaps" where you can record how you handled (1)-(5).

One thing to be careful about:

- For a method, like `Reflect`, when you call it, you prepend the name of the object it "belongs to". For example, if `p` is a point, you would say `p.Reflect()`, not `Reflect(p)`. (Don't say `p.Reflect`, either.)
- For a function that *isn't* a method, like `ShowPoint`, you don't prepend the object it "belongs to" when you call it, because it doesn't "belong" to any object — the function definition wasn't made as part of any object class. For example, if `p` is a point, you would say `ShowPoint(p)`, not `p.ShowPoint()`.

```
# ShowPointClassLab.py
# CS 1110 Profs
# April 2016

""" Illustrate the class  Point
"""

from ThePointClass import *
from SimpleGraphics import *


def ShowPoint(P,c):
    """ Displays P in the current figure window with color c
    PreC: P is a point and c is an rgb list
    """
    DrawDisk(P.x,P.y,.1,FillColor=c)


def student_function():
    """Demonstrates all the methods in the Point class
    and some functions that manipulate Point objects.
    """

    n = 3
    MakeWindow(n)
    for z in range(-n+1,n):
        DrawLineSeg(z,-n,z,n)
        DrawLineSeg(-n,z,n,z)

    ##############

    # Create 3 points at (1,2), (-2,1), and (-1,-2)
    # Display these points, coloring them as in the handout.









    # The 3 points define a triangle. Modify the next line so that
    # the perimeter is computed.
    # Make effective use of the Dist method that is defined in the Point class.
    Perimeter = 0





    Title('The perimeter is %5.2f' % Perimeter)
    # Display the reflections of the three points.
    # Make effective use of the the method Reflect in the Point class










    ShowWindow()

if __name__ == '__main__':
    student_function()
```

5. Appendix: common syntactic errors when working with classes

There's no work for you to do in this section, but you will want to take a look at the error messages provided below, in case you run into them in your future Python work.

With reference to the class definition in module `ThePointClass.py`, suppose we have done:

```
>>> import ThePointClass
>>> p = ThePointClass.Point(1,2)
>>> q = ThePointClass.Point(4,5)
```

5.1. **Trying to call a method without appending the name of its "enclosing" object.** Example:

```
>>> Dist(p,q)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Dist' is not defined
```

The problem is that `Dist` is a method, not a function declared outside of an object class. So you can't call it except in the context of a given `Point`. You should have said `p.Dist(q)` or `q.Dist(p)`.

5.2. **Forgetting to have `self` be a parameter of a method you write.** For example, *if you had deleted the* `self` *parameter from the definition of* `Dist`, then here's what would happen:

```
>>> p.Dist(q)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Dist() takes exactly 1 argument (2 given)
```

The long explanation for why this happens: When `obj` is an object of class `C` that has method `f`, Python interprets the call `obj.f(x)` as `C.f(obj, x)`, with the idea that `obj` is the argument for parameter `self` — that's how Python lets a method know which object it is being called with respect to! So, the function definition header `def f(self, input1)` says that two arguments are to be given, whereas `def f(input1)` says only one argument is to be given. And that causes an error for the implicit call `C.f(obj, x)`.

5.3. **Referring to a non-existent attribute.** Example:

```
>>> p.z
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: Point instance has no attribute 'z'
```

A common reason for getting this error message is typos, e.g., you meant to type `p.x`.

You can always check what attributes and functions are associated with a class with the `help` function, e.g., `help(ThePointClass.Point)`. This prints out all the docstring information for the class.

5.4. **Referring to the function itself rather than calling it.** A common error is to leave off the parentheses when making a function call.

Example:

```
>>> q = p.Reflect
>>> q
<bound method Point.Reflect of <ThePointClass.Point instance at 0x1028c1950>>
```

`q` is not a `Point` but a method, because `p.Reflect` isn't a function call but a reference to a method. You probably meant `q = p.Reflect()`.