Lab 04

Tuesday, September 17th

Objectives:

- explore various techniques for iteration (looping)

Introduction

TWO CHANGES

This week I will ask you to work with a partner of your choice. You will turn in a SINGLE answer sheet for both partners.

I will talk about this at the start of the lab today, but to put it in writing - when you work with a partner you should:

- Be working together
- Only be working at ONE computer
- Both people should work and you should share duties. For example, one partner runs the computer for activity A while the other partner tells him/her what to do and writes any answers on the "answer" sheet. When you reach the end of Activity A you should change roles.

Failure to follow these guidelines may cause you to lose points for this activity.

Secondly, this week you will be **submitting some of the programs you create to eLearning**. I will give you specific instruction on when to do this in the lab as you go.

Activity A : An introduction to iteration - while

(Partner A types, Partner B writes)

There are two forms of iteration used in Python - the "while" structure and the "for" structure. Both are effective ways to loop, but each is most appropriate in different

situations. In this lab today, you will look at several examples of how each structure works. Let's begin by looking at a while structure.

A while structure is a structure that repeats a suite of code as long as a Boolean condition is True. It is called a **sentinel-controlled loop** because the Boolean condition is a "sentinal" value that helps decide whether or not the suite of code should be repeated.

Generically it looks like this:

```
while Boolean_Condition_Is_True :
    suite of code line 1
    suite of code line 2
    ...
    suite of code line n

First line of code not in the suite
```

Consider the following code that uses a while loop to calculate a student's average quiz score. From the python shell window open a new programming window by selecting "File | New Window" In this window type in the Python program :

```
"""
File : quiz.py
Author : Diesburg
Description : Uses a WHILE structure to calculate
    the average score for N scores
    where N is provided as input by the user.
"""

numberOfQuizzesStr = input("Enter the number of quizzes: ")
numberOfQuizzes = int(numberOfQuizzesStr)
total = 0
quizCount = 1

while quizCount<=numberOfQuizzes:
    scoreString = input("Enter the score on quiz " + str(quizCount) + ": "
    score = float(scoreString)
    total = total + score
    quizCount = quizCount + 1

average = total/numberOfQuizzes
print("The average quiz score is " + str(average))
```

Run this program and observe how it works.

[Q1] What is the purpose of the variable named "numberOfQuizzes"

[Q2] What is the purpose of the variable named "total"

[Q3] What is the purpose of the variable named "quizCount"

Activity B : Obnoxious kid in my car

(Partner A types at the computer)

Ann Oying contacts you about writing a simulator that demonstrates what is like to be in a car with a child on a long road trip. **Please name this program kid1.py**

For example,

```
Enter the child's age: 6
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
```

(The string "Are we there yet?" is printed once for each year of the child's age.) You should write this code based on a while loop.

If you have any questions about this code, please raise your hand and ask an instructor.

Activity C : An introduction to iteration - for

(Partner B types at the computer, Partner A writes)

The for loop is another looping structure. It is called a **count-controlled loop** because it is used in situations where an iterable data structure is involved and it's size (or "count") is known in advance. Generically speaking, it looks like the following:

```
for variableName in iterableDataStrurcture:
    suite of code line 1
    suite of code line 2
    ...
    suite of code line n

First line of code not in the suite
```

To demonstrate a for loop we have to first create an iterable data structure. The quickest and easiest to get started is the list structure produced by the range function. Remember that a list structure is a data strcture that holds one or more values. The range function produces a range of values. Combining them, we can use the range function to place a range of values in a list.

[Q4] For each of the following uses of range, first PREDICT what will be returned when you type this into the python shell. Then, test the command out and see if you were correct.

Command	Prediction	Actual result
list(range(5))		
list(range(8))		
list(range(0,5))		
list(range(3,9))		
list(range(9,3))		
list(range(2,10,1))		
list(range(2,10,2))		
list(range(2,10,3))		
list(range(10,2,2))		
list(range(10,2,-2))		

Let's consider a couple of summary questions

[Q5] Consider the list produced by the one parameter version -- range(a).

The first number in the list is:

The last number in the list is:

The distance between each number in the list is:

[Q6] Consider the list produced by the two parameter version -- range(b,c)

The first number in the list is:

The last number in the list is:

The distance between each number in the list is:

[Q7] Consider the list produced by the three parameter version -- range(x,y,z)

The first number in the list is:

The last number in the list is:

The distance between each number in the list is:

[Q8] What do you notice about the last value actually produced by the list and the "ending value" in the range function (For this question you may assume that the distance value is 1).

[Q9] What do you notice about the number of values produced in a range function? How does it relate to the "starting value" and the "ending value"? (For this question, you may assume that the distance value is 1)?

Activity D : Using the for loop with Average Quiz Score

(Partner A types at the computer)

From the python shell window open a new programming window by selecting "File | New Window"

In this window type in the Python program below which calculates the average score from several scores, save it, and name it **quiz.py**.

```
"""
File : quiz.py
Author : Diesburg
Description : Calculates the average score for N scores
        where N is provided as input by the user.

        This version is broken.
"""

numberOfQuizzes = int( input("Enter the number of quizzes: "))

total = 0

for quizNumber in range(1,numberOfQuizzes):
    score = int(input("Enter the score on quiz " + str(quizNumber) + ":
    total = total + score

average = total/numberOfQuizzes
print("The average quiz score is " + str(average))
```

Test this program by running it and using the 3 quiz scores of 10, 20, and 30 (which should have an average of 20).

Notice that this program does not work the way it should. We have made a logical error (as opposed to a syntax error). Fix this problem.

[SIG1] Demonstrate both programs **kid1.py** and your fixed version of **quiz.py** to an instructor for a signature. **In addition to this**, please submit both kid1.py and quiz.py to eLearning. Only ***one partner*** needs to submit the programs.

Directions for submitting **kid1.py** and **quiz.py** to eLearning:

- Log into eLearning
- Go to this class
- Click on the "Lab Submission" link on the left menu
- Click on the "Lab04" folder
- Upload the files to their proper locations.

Activity E : Obnoxious kid in my car, for loop

(Partner B types at the computer)

Rewrite your solution to Activity B so that it uses a for loop instead of a while loop. Save this program as **kid2.py** .

If you have any questions about this code, please raise your hand and ask an instructor.

Activity F: More with the while structure

(Partner A types, Partner B writes)

So far we have looked at both the while structure and the for structure and it doesn't look like they are that different. Both were used to solve the same basic problem. If they are that similar, why have both structures?

We have two different structures because not all situations where we are looping have known finish times or "counts" when we begin. Consider a situation where we are entering quiz scores. Not all situations involve knowing the number of quizzes when we get started For example, suppose you are reading scores from a file. You don't know how many there are until you get to the end of the file.

In this situation, we want to use a while loop and define a "sentinal value" that signals that we are done taking in data.

For example, type in the following program:

```
"""
File : average.py
Author : Diesburg
Description : Calculates the average score for a set of scores
        The user enters a -1 to signal that they are done
        entering scores.
"""

print("Enter the scores one at at a time.")
print("Enter -1 to signal that you are done.")

total = 0
count = 0

oneScore = int(input("Enter a score: "))

while oneScore != -1:
    total = total + oneScore
    count = count + 1
    oneScore = int(input("Enter a score: "))


average=total/count
print("The average of those "+str(count)+" scores is: ",average)
```

At that point the computer should calculate and print the average. Something like:

```
Enter the scores one at at a time.
Enter -1 to signal that you are done.
Enter a score: 10
Enter a score: 20
Enter a score: 30
Enter a score: 40
Enter a score: -1
The average of those 4 scores is:  25.0
```

[Q10] What is the purpose of the variable named "total"

[Q11] What is the purpose of the variable named "count"

[Q12] What is the purpose of the variable named "oneScore"

[Q13] Why do we ask the "Enter a score" question twice?

Activity G : Obnoxious Kid

(Partner B types at the computer)

You may recall from our earlier conversations that the computer uses 0 for False and 1 (among other values) for True. Let's think about the obnoxious kid problem as a sentinel value problem. Revise your code to keep asking "Are we there yet" But this time, actually wait for an answer. If the human types in 0 then they are saying "False. We aren't there." When you finally get there the human will type in "1" or True. Please name this code **kid3.py** .

Your running code should look like this:

```
Are we there yet? 0
Are we there yet? 0
Are we there yet? 0
Are we there yet? 0
Are we there yet? 0
Are we there yet? 0
Are we there yet? 1
YAY!!!!
```

[SIG2] We are now nearing the end of the lab. I may not have time to come over and watch you demonstrate your **kid2.py** and **kid3.py** programs for a signature. No matter what happens, please submit both your **kid2.py** and **kid3.py** programs to eLearning so that I can grade them later. Only***one partner*** needs to submit the programs.

Directions for submitting **kid2.py** and **kid3.py** to eLearning:

- Log into eLearning

- Go to this class
- Click on the "Lab Submission" link on the left menu
- Click on the "Lab04" folder
- Upload the files to their proper locations.

If there is still time, raise your hand to demonstrate both programs **kid2.py** and **kid3.py** to an instructor for a signature.

Don't forget to turn in your answer sheet to an instructor before you leave to get points for the lab!!