Name: _____

Introduction to Programming in Python

Lab material prepared for Harrison High School Physics courses in collaboration with Purdue University as part of the project "Science Education in Computational Thinking (SECANT)", http://secant.cs.purdue.edu/.[1]

August 2009

Lab 4: While Loops

OBJECTIVES

In this lab you will learn:

- Writing and using while-loops
- When to use a while-loop versus a for-loop
- While-loops operating on lists and VPython objects

For-loops are natural in situations where we know ahead how many times the body of the loop should be executed. In some situations, we don't know this. We may want to execute a loop until a certain condition holds or no longer holds. This can arise when termination depends on criteria beyond our control (e.g., a random event, input of a particular value). There are also situations where determining the number of iterations is not convenient (e.g., how many times does the sphere whose velocity increases move before it leaves the track?). For such situations, Python provides **while-loops**.

To better understand when and why to use while-loops, think of it like this: Your 5 year-old cousin made mud-pies. You want him/her to take a shower to clean-up. In typical childish fashion, she/he asks, "How long do I have to shower?" You have two replies you could make:

a) 10 minutes
b) Until you are clean

When programming, the first answer would be portrayed as a for-loop because we focus on exactly how long the shower will continue:

```
for minutes in range (0,9):
    shower
```

The second answer would be portrayed as a while-loop because the length of the shower is undetermined; we instead focus on the condition of cleanliness:

```
while cousin is not clean:
    shower
```

Anything you can do with a for-loop can be rewritten using a while loop (it may result in more lines of code, however). Consider again the example of printing the integers from 0 to 99. Below is the for-loop you saw in Lab 3 and its corresponding while-loop:

```
# Lab 3 – for_loop1.py                    # Print the integers from 0 to 99
```

[1] This work is supported by the National Science Foundation under Grant No. CCF-0722210. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

```
# Lab 4 – while_loop1.py
# Print the integers from 0 to 99
for i in range(100):
    print i
print "for-loop finished"

i = 0
while i < 100:
    print i
    i = i + 1
print "while-loop finished"
```

Programs **while_loop1.py** and **for_loop1.py** do exactly the same task. The while loop does not use range; instead it does its own "book-keeping" to count up to 100. To do this, it needs a variable. The program uses variable i which is initialized to 0 and is increased every time the body of the loop is executed. The body of the loop prints the value of i and then increases it.

A **while-loop** consists of the following parts:

- It starts with the keyword **while**, followed by a Boolean condition and the symbol ":"
- The body of the loop contains statements which are indented. The statements of the body of the loop end when a statement is found that lines up with the keyword **while**

The **Boolean condition** of a loop controls when a loop ends. The Boolean condition evaluates to either True or False; evaluating to True means continue the loop. Our loop continues as long as **i < 100**, which means the loop terminates when the value of i is greater than or equal to 100.

Create file **while_loop1.py** and run it.

Change the program to print 10 instead of 100 numbers. Run the program.

Next, interchange the two statements inside the while-loop so the code looks like this:

```
i = 0
while i < 10:
    i = i + 1
    print i
print "loop finished"
```

Run the program.

A.) What was different in the outputs between the two runs? Why is there a difference?

While-loops and VPython

You will see more while-loops than for-loops during the course and it is important to fully understand how a while loop in your program is executed. We start with the VPython programs from Lab 3.

Ball on a track

Open program **for_loop4.py** from Lab 3 in which a sphere named ball moves along the track and save it as **while_loop3.py**. Change the for loop so that it shows ball moving m units in the m^{th} iteration.

```
for m in range(my_le):
    rate(10)
    ball.pos.x = ball.pos.x + m
```

The sphere stops moving after **my_le** iterations. It increases its x-coordinate by **m** units each iteration and since **m** increases, ball "flies" off the track. How do we stop the sphere when it reaches the end of the track?

This can be done with a for-loop, but it is much easier and more natural to do with a while-loop. Program **while_loop3.py** is our first solution.

```
# Lab 4: while_loop3.py
# sphere moving on a track
# Written by: Your Name

from __future__ import division
from visual import *

my_le = 100
my_he = 2
my_wi = 10

track = box(size=(my_le, my_he, my_wi), color=color.red)
# the center of the track is at (0,0,0); right end is at my_le/2

rad_ball = 2                #defines the radius of the ball

ball = sphere(pos=vector(-my_le/2, (my_he/2 + rad_ball), 0), radius=2, color=color.blue)

deltax = 1
while ball.pos.x < my_le/2:    # continue the while loop as long as the ball's x-position is on the track
   rate(2)

   ball.pos.x = ball.pos.x + deltax      # update the x-position of the sphere and print position
   print "ball: ", ball.pos.x
   print "deltax: ", deltax

   deltax = deltax + 1  # deltax corresponds to m in the for-loop program
```

Run **while_loop3.py**.
B.) Why does the sphere not stop when the end of the track is reached, but moves a little further?

Change the stopping condition from **ball.pos.x < my_le/2** to **ball.pos.x < my_le/2 – deltax** Run it.

C.) Explain the difference between the two stopping criteria.

How could you make the motion of the ball smoother so that it looks like it is rolling rather than jumping? (Don't just increase the rate so it flips through faster).

Check with your instructor once you have modified your program to produce a smoothly rolling ball.

Incorporating Velocity

Save existing **while_loop3.py** as **while_loop4.py** We are now going to want the ball to move at a constant velocity of (10,0,0)m/s After defining the track and ball, edit the program so it looks like this:

```
my_le = 100
my_he = 2
my_wi = 10

track = box(size=(my_le, my_he, my_wi), color=color.red)
# the center of the track is at (0,0,0); right end is at my_le/2

rad_ball = 2                #defines the radius of the ball

ball = sphere(pos=vector(-my_le/2, (my_he/2 + rad_ball), 0), radius=2, color=color.blue)

vball = vector(10,0,0)  # vector vball represents the velocity of ball
t = 0                   # variable t represents time
deltat = 1              # variable deltat represents the time step, the amount of time elapsed for
                          each iteration of the loop
while t < 10:
   rate(2)
   print "position at time", t, "is:", ball.pos

   ball.pos = ball.pos + vball*deltat   #update the position of ball using vector operations

   t = t + deltat

print "finished at time", t, "with position", ball.pos
```

D.) What equation does **ball.pos = ball.pos + vball*deltat** resemble from Matter & Interaction textbook?

E.) What does **vball*deltat** give you in terms of the ball's motion? (don't tell me the numeric answer)

F.) What is the difference between **t** and **deltat**?

G.) Why do we use **deltat** instead of **t** in **ball.pos = ball.pos + vball*deltat**? Try changing it in the program. (make sure you change it back)

H.) Put **t = t + deltat** before **ball.pos = ball.pos + vball*deltat** in the while loop. What changed? Why?

What could you do to make the ball move very smoothly? Do it.

Add to your program an arrow that visually represents the velocity vector of the ball at every point in time. Have the arrow located directly over the center of the ball and move with the ball. Call it vel_arrow. Suggestion: create the arrow **after** the line that creates the ball and the line defining velocity.

I.) What happens when you put the line of code that creates the arrow within the while loop?

Have your instructor check your work. You may continue on if they are busy with others.

Infinite while-loops

While loops have the potential to result in an infinite loop. Look at the code below.

```
L = [4, 15, 8 ,12]
i =0
while L[i] > 0:
    L[i] = L[i] + 1
    print L[i]
```

J.) What do you expect this loop to do?

In almost all situations an infinite loop is highly undesirable. However, when running simulations of physical system, you will execute infinite loops. This will allow you to watch a simulation as long as you want to (you will terminate it).

Such infinite while-loops will generally be written as **while 1 == 1.** Write the following program, save your work and run the infinite loop:

```
i=0
while 1 == 1:
        print i
        i= i+1
```

You will have to terminate the program (in SPE, close the black executable window and wait for termination).

Challenge

Using an infinite while-loop, complete one of the following problems. Note that VPython automatically imports the math module.

1. *Write a program creating a pulsating sphere.*
   First, name and define a sphere. In the infinite while-loop, change the radius of the sphere to a constant plus the sine of t, increasing t by a small constant **deltat** each iteration of the loop. You will need to use **scene.autoscale = 0** to stop VPython from zooming in and out as the sphere changes size.

2. *Write a program that moves a sphere in a circular orbit around the origin in the x-y plane.*
   First, place a sphere (or other object) at the origin and name and define a sphere orbiting the origin. In the infinite while-loop, the orbiting sphere moves on a counterclockwise circular path parameterized as $x(\theta) = r*\cos(\theta)$ and $y(\theta) = r*\sin(\theta)$, where r is the radius of the circle.

Your program should run indefinitely. After you have enjoyed watching your simulation, terminate the VPython window. Wait for your instructor to check your work.

Copy your code to the challenge problem into Word.doc and type the answers to questions A – J.