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 5: Conditionals

OBJECTIVES

In this lab you will learn:

- Writing and using if-statements
- Revisit earlier programs to include conditionals
- Write a program solving quadratic equations
- Make a ball bounce forever between two walls

In many programs, some statements should only be executed when a certain condition is true. If the condition is not true, the statements should not be executed. Every programming language allows such conditional statements, called **if-statements**.

In Python an **if-statement** consists of the keyword **if**, followed by a Boolean condition, the symbol **:,** and the "body" which contains the statements executed when the condition is true (these statements are indented as done for the body of a for- or while-loop):

```
if age < 16:
        print "I can't drive"
```

The key to understanding an if-statement is to know that the body will execute only when the **if condition** evaluates to **True** and that it will not execute when the **if condition** evaluates to **False**. In the above statement, "I can't drive" will only be printed when the value of the variable **age** is less than 16. When the value of **age** is greater or equal to 16, the if-statement does nothing.

Write the following program and save it in file **cond1.py**:

```
# Lab 5: cond1.py
# If statements and your age
# Written by: Your Name
age = 0
while age < 20:
        print  "I am ", age
        if age < 16:
                print "I can't drive"
        if age < 18:
                print "I can't vote"
        print    # prints an empty line
        age = age+1

print "end of program"
```

Now run the program. What does it do?

- Change the first line to **age = 7**. Run the program again. What was different?

[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.

- Modify the program to print “Now you can drive” only when you are 16 and “Now you can vote” only when you are 18. In all other situations, nothing is printed. Remember that == is the syntax for “equal to”. Save the new program in file **cond2.py**.

<u>If-then-else and a remark about nested statements</u>

Python (as well as other languages) provide more than the simple *“if the condition is true, then do the following”* construction described. A natural extension is *“if the condition is true, then take actions 1, else take actions 2”*. Below is an example on how to write an if-then-else statement.

```
age = 17

if age < 16:
   print "I can't drive"
else:
   print "I can drive"
```

Notice the indentation: the word **else** lines up with the **if**, indicating that statements indented under it are executed when the condition is false. The body of an **else-statement** is executed when the body of the preceding **if-statement** did not execute. The statement above will always print something; what depends on the value of age.

Before we give another if-example, a comment on nested statements. When a loop contains another loop, we refer to it as a nested loop. Nested statememts can create interesting code*: a while-loop can contain a for-loop which can contain another while-loop which can contain an if-then-else statement and the else-part can contain a while loop*. Still with me? Not surprisingly, heavily nested code gets complex and can be challenging to debug. You will not see heavily nested code in this class, but you want to realize the potential and complexity hidden in nested code.

Load program **cond3.py** which contains a nested if-statement. It uses the age and the body mass index (bmi) to determine the heart disease risk. The risk does not only depend on the bmi, but also on the age (at age 45, the classification changes). A single if-then-else statement is not enough to determine the risk of a given age and bmi pair. Program **cond3.py** shows one of the possible solutions for identifying the correct risk.

```
# Lab 5: cond3.py
# Using if-then-else and nested if's
# Written by: Your instructor
age = 55
bmi = 33
print "age =", age, "body mass index =", bmi

if age < 45:
   if bmi < 22.0:
      print "risk of heart disease is low"
   else:
      print "risk of heart disease is medium"
else:
   if bmi < 22.0:
      print "risk of heart disease is medium"
   else:
      print "risk of heart disease is high"
```

Run the program with different pairs of age and bmi. Pay attention to the indentation of the if-statement within an if-statement.

If-statements and VPython

If-statements will be very useful in physical simulations. In the example below, we develop a program in which a ball bounces off two opposing walls (it will do so in an infinite loop so you can watch it for as long as you want).

Take a look at program **while_loop4_modified.py:** a box named **wallR** is positioned to the right of sphere **ball** and box **wallL** is positioned to the left of the ball. The goal is to have the ball bounce between the two walls indifintely.

In order to have a ball bounce off a wall, we need to detect when the wall has been reached (an if-statement will do this) and we need to change its velocity to move into the opposite direction. This means changing the sign of **vball.x**. How do we change the sign of a quantity? We simply negate it (to change the sign of a, write the assignment statement **a = –a**).

```
# Lab 5: cond5.py
# sphere bouncing off two walls
# Written by: Your Name
from __future__ import division
from visual import *

ball = sphere(pos=vector(0,0,0), radius=1, color=color.red)
wallR = box(pos=(8,0,0),size=(2,12,8),color=color.green)
wallL = box(pos=(-8,0,0),size=(2,12,8),color=color.green)        # define a right and a left wall

vball = vector(1,0,0) #vector vball represents the velocity of ball
deltat = 0.1

while 1 == 1:                # the infinite loop lets the ball bounce off the walls
   rate(10)

   if ball.pos.x + ball.radius >= wallR.x - wallR.length/2:
      vball.x = -vball.x

   ball.pos = ball.pos + vball*deltat                # upadate the postion of the ball
```

Run the new program. The ball should bounce off the right wall, but go through the left wall. If the ball is not bouncing correctly, take a careful look at the Boolean condition.

Change **if ball.pos.x + ball.radius >= wallR.x - wallR.length/2:** to use == rather than >= and run it.

Question: *Why does it do that?* Add these lines of code after the if-statement, but NOT in the if-statement.

```
    print "% .10 f " % (ball.pos.x + ball.radius)        #displays the value using ten decimal places
    print "% .10 f " % (wallR.x - wallR.length/2)
    print                                                # prints a blank line
```

Modify the program so that it will now bounce between the two walls.

Have your instructor check your work.

Challenge

- Add a top and a bottom wall in the form of two box objects **wallT** and **wallB.**
- Change the value of **vball.y**. Start with a small value like 0.1.
- Add statements to the program that make the ball not only bounce off the left and right wall, but also the top and bottom.
- If this was a breeze, make the ball move three-dimensionally! Add a back wall (you are now looking into an open box), and have the ball bounce off the back and an imaginary force field in the front of the box.
- Now if you are really ambitious, add another ball so you now have two balls bounce off the walls, and then off each other!

Copy your program for the challenge and print it out. Include the answer to the single question below that.