PREAMBLE: Graphics Package

Throughout the this course, we will often make use of the Zelle graphics package. This package (written by John Zelle) tells Python how to make basic shapes appear on the screen.

You can download the file graphics.py from Canvas.

Put this file in the same folder (on your desktop) where you saved the Python program that you wrote. (If you put this somewhere else, Python won't be able to find it -- Python is easily confused.)

Type the following Python program

```
from graphics import *

win = GraphWin("My Program", 500, 400)
```

The first line tells Python that you want to use the Zelle graphics package (graphics.py)
The second line tells Python to make a graphics window on your screen and call it "My Program".
(The 500 and the 600 are how wide and how tall you want the window. You can change these numbers)

PREAMBLE: Time

The following line imports the time package. This is a common packages that are included whenever you install Python.

```
from time import *
```

From the time package, we will make use of the sleep function. The sleep function causes Python to wait for some number of seconds.

```
from time import *
from graphics import *

win = GraphWin("My Program", 500, 400)

my_circle = Circle( Point(200,200), 100)
my_circle.draw(win)
sleep(3)
my_circle.setFill("red")
```

The above program waits for 3 seconds before changing the color of the circle.

Note: If you want things to happen more quickly you can sleep for half a second instead.

```
from time import *
from graphics import *

win = GraphWin("My Program", 500, 400)

my_circle = Circle( Point(200,200), 100)
my_circle.draw(win)
sleep(0.5)
my_circle.setFill("red")
```

PREAMBLE: Randomness

The following line imports the time package. This is a common packages that are included whenever you install Python.

```
from random import *
```

From the time package, we will make use of the randint function. The randint function returns a random integer in a specified range.

```
from random import *
from graphics import *

win = GraphWin("My Program", 500, 400)

radius = randint(50,200)
my_circle = Circle( Point(200,200), radius)
my_circle.setFill("red")
my_circle.draw(win)
```

This program makes a circle whose radius is a random number between 50 and 200. Try running the program several times and observe that you get different size circles.

Note: For the mathematically interested, the numbers returned by the randint function aren't truly random, they are actually "pseudorandom". I am happy to talk outside of class about techical differences between random and pseudorandom numbers, but for our purposes, Python random numbers are "random enough".

PREAMBLE: While Loops

The if statement in Python causes some instructions to execute once if a condition is true. The while loop causes instructions to execute many times as long as the condition is true.

When you run a while loop, Python will check the condition at the start of the loop. If the condition is true, Python will start the loop. When Python gets to the bottom of the loop, it will check the condition again. If the condition is still true, Python will do the loop again. This continues until the condition becomes false.

```
from graphics import *
from time import *
win = GraphWin("My Program", 500, 400)

while win.checkMouse() == None:
    print("Hello")
    sleep(3)
```

The checkMouse function returns a Point if the mouse has been clicked recently and otherwise returns the special value None. (Note that unlike the getMouse function, checkMouse doesn't wait for a mouse click. If the mouse hasn't been clicked recently it just returns None and moves along with the rest of the program.)

This function keeps printing "Hello" until the user clicks the graphics window. (Recall: If you happen to be using Python 2 in the lab, don't put parentheses around "Hello" in the print statement.)

Often you will want to set a variable inside your loop and then check the variable in the while statement at the top of the loop.

```
number = 0
while number < 5:
        number = int( input( "Enter a Number: ")
```

The above program keeps asking the user for a number as long as the user enters numbers that are less than 5. (That is, the loop stops when the user enters a number that is NOT less than 5.)

It is important to give your variable a starting value (in this case, I picked 0). Otherwise, Python will be confused at the top of the loop because it doesn't know about any variable named number. This is often called "initializing" your variable.

The following program does exactly the same thing as the program on the previous page (it just does it in a slightly different way):

```
from graphics import *
from time import *
win = GraphWin("My Program", 500, 400)

mouse_point = None
while mouse_point  == None:
        print("Hello")
        sleep(3)
        mouse_point = win.checkMouse()
```

PREAMBLE: Responding to Keypress

In class, we have discussed using the input function which waits for the user to type something on the keyboard and hit enter.

Sometimes you don't want to wait for the user to act (you just want to move on if nothing has been pressed). The Zelle graphics package provides a checkKey() function -- similar to the checkMouse() function -- that allows you to see if the user has pressed a key.

If win is a graphicsWindow object, then win.checkKey() returns the most recent key pressed by the user, as a string (in quotes). So it returns things like "w" or "d" or "s". If no key has been pressed, then it returns the special empty string "".

The following code keeps making circles until the user presses a key.

```
from graphics import *
from random import *

win = GraphWin("My Program", 600, 600)

key = win.checkKey()
while key == "":
        rand_x = randint(100, 500)
        rand_y = randint(100, 500)
        rand_point = Point( rand_x, rand_y)

        my_circle = Circle( rand_point, 100)
        my_circle.setFill("blue")
        my_cirlce.draw(win)
        key = win.checkKey()
```

This function can tell how far apart two circles are:

```
from math import sqrt
def circleDistance( cir1, cir2):
    center1 = cir1.getCenter()
    center2 = cir2.getCenter()
    x1 = center1.getX()
    y1 = center1.getY()
    x2 = center2.getX()
    y2 = center2.getY()
    dist = sqrt( (x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) )
    return dist
```

You can use this function (along with an if statement) to check if the moving circle gets close to (i.e., comes in contact with) another circle. If the distance between two circles is less than the sum of the radius(es) then the circles have collided.

1) For example, at the start of your program, give the target circle a random velocity. You should do this by picking a random "horizontal speed" and a random "vertical speed". Each time through your main loop, move the target circle to a new position based on its "horizontal velocity" and its "vertical velocity". (You can use whatever variable names you want.)

   For example:

   ```
   target_circle.move(x_velocity, y_velocity)
   ```

   You should allow the horizontal and vertical velocity to be negative. (So the target circle doesn't always move down and right.)

   When the target circle hits the edge of the window, give it a new random velocity (to send it flying back towards the middle of the window).

   Your program should still stop once the first circle (controlled by the user) hits the target circle.