Looking at Games: Pong

Getting Started

Up until this point in the course, you've been doing all of your assignments in CS50 IDE. This problem, however, cannot be completed in an online IDE as it will employ graphics that the IDE cannot render. You will need to install the following software onto your computer: python3 (https://www.python.org/downloads), which usually includes the tkinter library (if not included this must also be installed) and download a simple python text editor like Atom (https://atom.io).

You can then use Atom as your text editor and your mac or windows terminal as your terminal window.

The reference for graphics.py is available at: http://mcsp.wartburg.edu/zelle/python/graphics/graphics.pdf

---

Backstory

Pong, is a groundbreaking electronic game released in 1972 by the American game manufacturer Atari, Inc. One of the earliest video games, *Pong* became wildly popular and helped launch the video game industry. The original *Pong* consisted of two paddles that players used to volley a small ball back and forth across a screen.

The German-born American television engineer Ralph Baer laid the groundwork for *Pong* in 1958 when he proposed making simple video games that people could play on their home television sets.
The Magnavox Odyssey, known as the first console video game system, was released in 1972 and offered a game of table tennis, or Ping-Pong. Atari founder Nolan Bushnell created *Pong*, his version of this concept, as an arcade game. A small company at the time, Atari began manufacturing the games in an old roller skating rink, and by 1972 the company had sold more than 8,000 *Pong* arcade machines. In 1975 Atari turned *Pong* into a console system game. After striking an exclusive deal with Sears, Roebuck and Company, *Pong* was soon in the homes of many American families. *Pong*'s popularity declined in the 1980s as video games temporarily went out of style, but it had already secured its place in history as the most popular arcade game up to that time.

—Encyclopedia Britannica

Your Challenge

Your challenge for this problem is to read through the distribution code, write comments for each TODO and complete the implementation of *Pong* so that a player wins when they are the first to reach 10 points. After this you may further customize the game by adding your own unique feature.

Whereas all of your C programs to date have only had "command-line interfaces" (CLIs), this one will have a graphical user interface (GUI), not unlike Scratch! You'll be building Pong atop graphics.py, which is similar in spirit to the CS50 Library but includes a simple object oriented graphics library.

Let's take a look at what you can do with graphics.py by way of some code examples. In your terminal you will download the distribution code by typing in:

```
git clone https://github.com/cs50nestm/pong.git
```

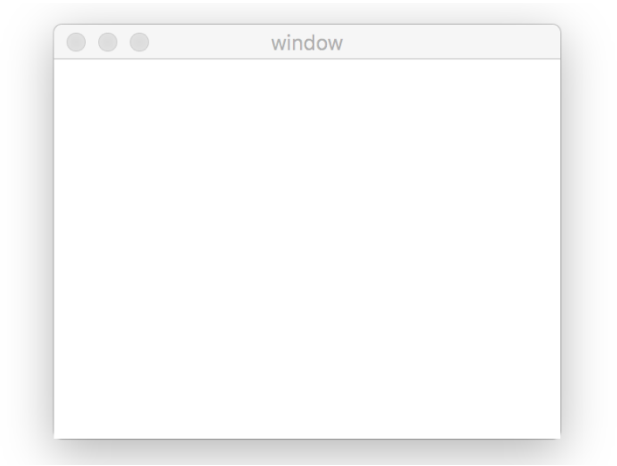
Then in your terminal type:

```
cd pong
```

If you then execute `ls`, among the files you see should be `bounce.py` and `window.py`. In your terminal window, go ahead and execute window.py as follows:

```
python3 window.py
```

A window quite like the one below should appear and then disappear after about 5 seconds.



Neat, right? Go ahead and open window.py in Atom.

The first thing you will notice is:

```
from graphics import *
```

which will look for a python file named graphics in the current directory, find graphics.py, and upload all (* means all here) of that code. You should already have graphic.py installed on your computer, or residing in the same folder as your python files. This program is what gives your python programs graphics capabilities.

Next you see:

```
win = GraphWin("window", 320, 240)
```

which creates or "instantiates" a new window, with a width of 320 and a height of 240.

Finally, you will see:

```
time.sleep(5)
win.close()
exit(0)
```

which is why your window stays open for 5 seconds and then closes. Finally, your program ends with:

```
exit(0)
```

which is the equivalent of return 0 in languages like C. This is a conventional way of saying exit the program and everything was okay. If a number other than 0 is returned, it typically means that an error occurred.

Now let's look at a program with a window and a circle with the circle bouncing back and forth. Go ahead and open bounce.py.

First, you'll see that a 320 by 240-pixel window is instantiated. Then a point is defined with (x, y) coordinates of (160, 120) which is used as the center of a circle with a radius of 20 pixels, which is instantiated on the next line.

```
center = Point(160, 120)
ball = Circle(center, 20)
ball.setFill("BLACK")
ball.draw(win)
```

Finally, the fill of the ball is set to black, and it is drawn in the window.

Next, a variable velocity, which will represent the speed the ball moves, is set to 2 pixels per unit of time.

Now to make the circle bounce back and forth, we'll create an infinite loop, and inside the loop for every unit of time, we're going to move the circle 2 pixels. Then we're going to have to check if the ball touches the right edge of the window or the left edge of the window, because if it does we need to make it bounce.

So first we make the ball move along the x-axis, but not at all along the y-axis.

```
    ball.move(velocity, 0)
```

Then we get the x-coordinate of the center of the circle so we know where it is after it moves.

```
    centerBall = ball.getCenter()
    xBall = centerBall.getX()
```

We add the radius of the ball to the x value of the center to find the coordinates of the right edge. If this value exceeds the width of the window, we know the ball has reached the right edge. We can make the ball "bounce" so to speak, by reversing the velocity when it reaches the right edge.

```
    if xBall + 20 > 320:
        velocity = -velocity
```

We do the same to check if the ball reaches the left edge of the window:

```
    if xBall - 20 < 0:
        velocity = -velocity
```

Finally, we check for a mouse click, which tells us to close the window, break out of the loop, and end the program.

```
    if win.checkMouse():
        win.close()
        break
    exit(0)
```

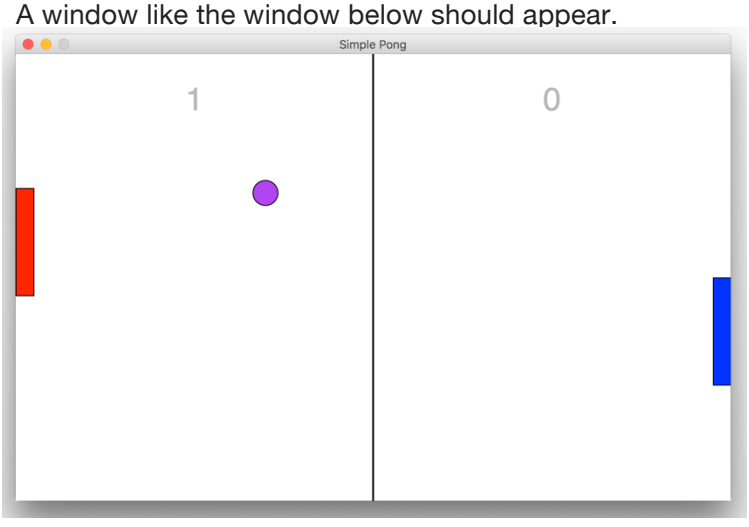
---

pong

Okay, let's see what the distribution code for `pong` itself does. Go ahead and execute:

```
    python3 pong.py
```

A window like the window below should appear.



The game itself is completed, but there is no way to win!

But, first, a tour!

Open up pong.py with Atom and take a moment to scroll through it to get a sense of what lies ahead.

- Atop the file you will find two import statements.
- Next up are some constants (which are really just variables in Python), values that you don't need to change, but because the code we've written (and that you'll write) needs to know these values in a few places, we've factored them out as constants so that we or you could, theoretically, change them in one convenient location. By contrast, hard-coding the same number (pejoratively known as a "magic number") into your code in multiple places is considered bad practice, since you'd have to remember to change it, potentially, in all of those places.
- Next, we instantiate a window.
- Then, you'll see something you need to create a comment for (TODO) followed by a main function.
- In the main function, you'll see more TODO's where you need to add additional comments.
- Please note that I modified the graphics.py module to allow for multi-user play. So, line 59, keys = win.keysPressed() returns an array-like structure where keys["w"] is true if the w is pressed, keys["s"] is true if the s key is pressed, etc. So, this game uses the w and s keys to move the left paddle and the o and l keys to move the right paddle.

Read through the code, play the game and complete the comments until you understand what all parts of the code is doing. When you've completed this, add a new function to determine if a player wins the game and change their score to "YOU WIN"! Then go ahead personalize your implementation to make it uniquely yours!

Because this game expects a human to play, no `check50` for this one! Best to invite some friends to find bugs!