Announcements

- Homework
  - Homework 1 all graded! Check GitHub!
  - Homework 2 due on Friday night
  - After today should have everything you need to complete the assignment
- Polling: `rembold-class.ddns.net`

Homework Comments

- All HW1 has been graded. Comments and scores uploaded to GitHub.
- Most common issues:
  - Not following instructions about what to print (usually not that big of a deal when I check myself, but can mask other errors)
  - Having "holes" in blocks of conditional statements.
  - Comments are optional. . . until they cost you points.
  - Just change the "Assignment status:" line in the README file please! Not the entire filename!
- That said, most people did quite well

Review Question

Suppose you have the string: `x = "consternation"` and you'd like to just extract and print the word `"nation"`. Which expression below will not give you the string `"nation"`?

A) `x[7:len(x)]`

B) `x[7:]`

C) `x[-6:len(x)]`

D) `x[-6:-1]`

Solution: `x[-6:-1]`

Can't change a string's colors

- Strings are what we call immutable: they can not be modified in place
- You can "look" at different parts of the string, but you can not "change" those parts
  - Phased differently, you can't reassign the various pieces of a string

```
s = "Cats!"

s[0] = "R"          # This will error!!
```

- You can of course still reassign `s` to some new string object

```
s = "R" + s[1:]
```

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle
  - Matplotlib

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle
  - Matplotlib
  - Tkinter/Graphics.py

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle
  - Matplotlib
  - Tkinter/Graphics.py
  - PyQt

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle
  - Matplotlib
  - Tkinter/Graphics.py
  - PyQt
  - PyGame

Enter the Arcade

- There comes a time when reading and entering text on a terminal doesn't cut it.
  - Maybe you need more complicated input
  - Maybe you need a more complicated interface than pure text can manage
  - Maybe you have output that can not be shown in text
- Standard Python really only deals with a terminal interface
- Lots of outside "extensions" give Python a more visual input/output
  - Turtle
  - Matplotlib
  - Tkinter/Graphics.py
  - PyQt
  - PyGame
  - Arcade

Setup Costs

- Arcade is not part of the Python "standard library"
  - Also doesn't come packaged by default with Anaconda
  - You will need to install it separately, which we will cover in lab today
- Will need to `import` it at the top of your file to use its special commands
  - Importing is how you gain the powers of "outside" libraries in Python
  - We'll discuss in more detail in Ch 4 in terms of how to import your own files and break up your code.
  - After import, will always have that library's name at the start of every command from that library:

```
import fish

fish.make_a_fish()
fish.make_fish_swim()
```

The Basics

- At its simplest, `arcade` does the following:
  - Open a window:

    ```
    arcade.open_window(<width>, <height>, <name>)
    ```

  - Draw things to the window:

    ```
    arcade.draw_circle_filled(0,0,10, arcade.color.RED)
    ```

  - Keep the window open so we can see it:

    ```
    arcade.run()
    ```

What to draw?

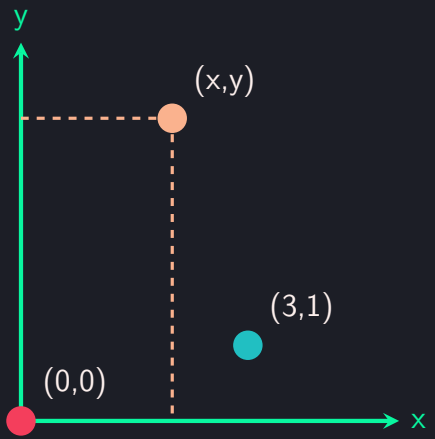
- You have a wide variety of primitive objects you can draw
  - Circles
  - Rectangles
  - Lines
  - Arcs
  - Polygons
- Most have filled/outline options available

```
arcade.draw_circle_filled()
arcade.draw_circle_outline()
```

- All start with `arcade.draw_<what to draw>`

Where to draw?

- Uses a standard x-y coordinate system like in math
- Only uses the first quadrant, so everything positive
- (0,0) is in the lower left corner
  - This is different from many computer graphics programs, which put (0,0) in the upper left corner.



How to draw?

- All drawing commands you make must be between two extra commands:

```
arcade.start_render()

<draw all your stuff here!>

arcade.finish_render()
```

- If you want to change the default background of your window, it goes before the `arcade.start_render()`:

```
arcade.set_background_color()
```

I demand rainbows

You can pick colors in a variety of ways:

- Using arcade's built in colors (there are 1000. Go wild!)

```
arcade.color.ANTIQUE_FUCHSIA
```

- Using arcade's css colors (only 147 here!)

```
arcade.csscolor.SILVER
```

- Using RGB codes (16.7 million possibilities here...)
  - 0 is the min, 255 the max
  - Come as a set of 3 (or 4) with comma's separating them
  - Fourth option will give transparency

```
red = (255,0,0)
trans_dark_green = (0,100,0,100)
```