Welcome!

We’ll get started shortly …

CS 49 Section

Week 7

Surajit A Bose

Agenda

- Logistics and check-ins
- Review of lecture concepts
  - Drawing shapes
  - Centering shapes
- Section Problems:
  - Random Circles

Logistics

How to get hold of me / get help

- The class forum or section forum, 24 hr turnaround
    - Feel free not only to ask, but also to answer questions there!
- Surajit's office hours:
    - Tuesdays 1p–2p on Zoom
    - By appointment on Zoom or on campus
- Lane's office hours
- Canvas inbox for Lane or Surajit
- Email bosesurajit@fhda.edu, 24 hr turnaround
- Online or in-person tutoring via the STEM center (Room 4213)
- The section GitHub repo has lecture and section slides and solutions

Check In

Any questions about:

- Boolean expressions
- The comparators: **`==`**, **`<`**, **`>`**, **`<=`**, **`>=`**, **`!=`**
- The logical operators: **`not`**, **`and`**, **`or`**
- Problems from the homework or extra credit
- Anything else?

Please take the Zoom poll!

Lecture Review: Graphics

Graphics

- Used to draw shapes on the screen
- The standard Python library for graphics is **`Tkinter`**
- For this class, we'll be using a purpose-built subset called **`graphics`**
  - **`graphics`** enables drawing lines, rectangles, and ovals
  - Need to include this import statement at the top of the code:

    ```
    from graphics import Canvas
    ```

Canvas

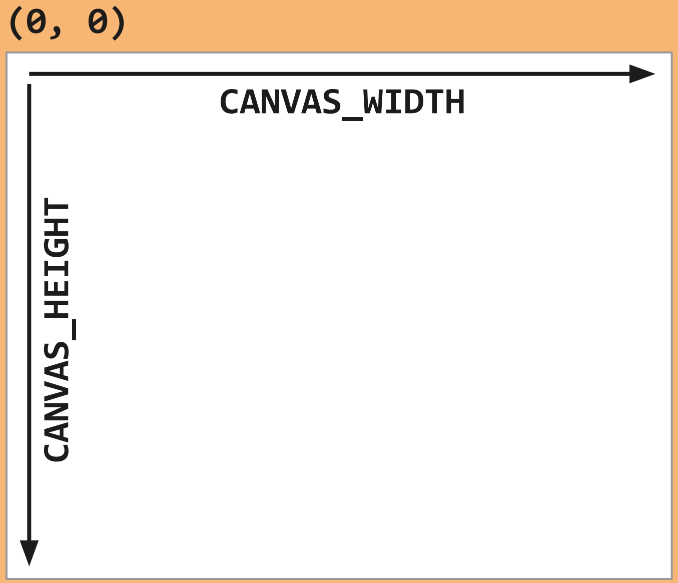
- The **Canvas** is the background on which to draw shapes
- The canvas must be created by specifying a width and height
- Typically these are specified as constants:

```
from graphics import Canvas

CANVAS_WIDTH = 400
CANVAS_HEIGHT = 300

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT)
```

Any Questions?

Canvas

- Positions on the canvas are determined by a coordinate system
- `(0, 0)` is the top left of the canvas
- The x-axis goes from **`0`** to **`CANVAS_WIDTH`**
- The y-axis goes from **`0`** to **`CANVAS_HEIGHT`**
- The bottom left of the canvas is **`(CANVAS_WIDTH, CANVAS_HEIGHT)`**



Canvas

- Drawing a line requires specifying two sets of coordinates:
    - The **`(x, y)`** of the top left point
    - The **`(x, y)`** of the bottom right point
    - **`canvas.create_line(x1, y1, x2, y2)`**
- Drawing a rectangle requires specifying coordinates for its diagonal:
    - **`canvas.create_rectangle(left_x, top_y, right_x, bottom_y)`**
- Drawing an oval requires specifying coordinates for the diagonal of the rectangle that will constitute the oval's bounding box:
    - **`canvas.create_oval(left_x, top_y, right_x, bottom_y)`**

Canvas

- To draw a square or a circle, the difference between **`left_x`** and **`right_x`** should equal the difference between **`top_y`** and **`bottom_y`**
- For any shape, there can be an optional fifth argument that specifies the color: **`'red'`**, **`'blue'`**, **`'green'`**, **`'purple'`**, etc.
- If no color is specified, the default is **`'black'`**
- Sample code for a colored square and circle:

```
canvas.create_rectangle(25, 85, 45, 105, 'fuchsia')
canvas.create_oval(94, 119, 124, 149, 'yellow')
```

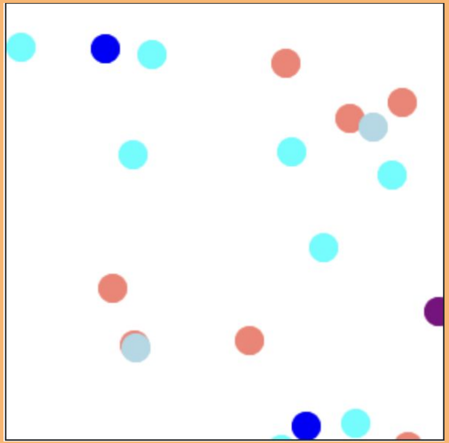
Any Questions?

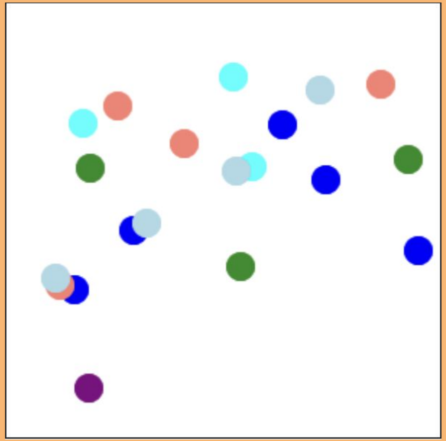
Section problem: Random Circles

https://codeinplace.stanford.edu/foothill-cs49/ide/a/randomcircles

Random Circles

- Given two integers *n* and *m*, draw *n* circles of size *m*
- The circles should be at random positions on the canvas
- The circles should be of random colors





Random Circles

- The canvas dimensions, number of circles, and circle size are constants
- To get a random color, use **`random_color()`** as the fifth argument to **`canvas.create_oval()`**
- Stepwise refinement:
    - Draw one circle at a fixed position
    - Randomize the position
    - Repeat as many times as specified
    - Optional: Make the circles fit in the canvas
    - Optional: Put the code to draw the random circles in a separate function

Any Questions?

That's all, folks!

Next up: Functions!