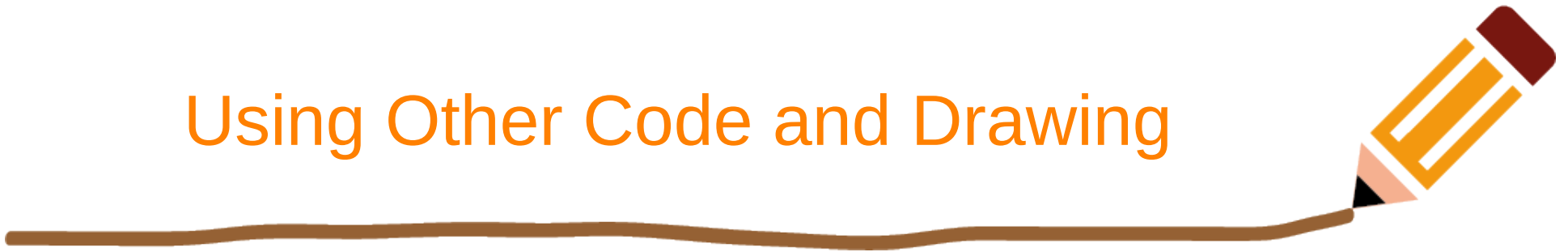
Using Other Code and Drawing



Reading

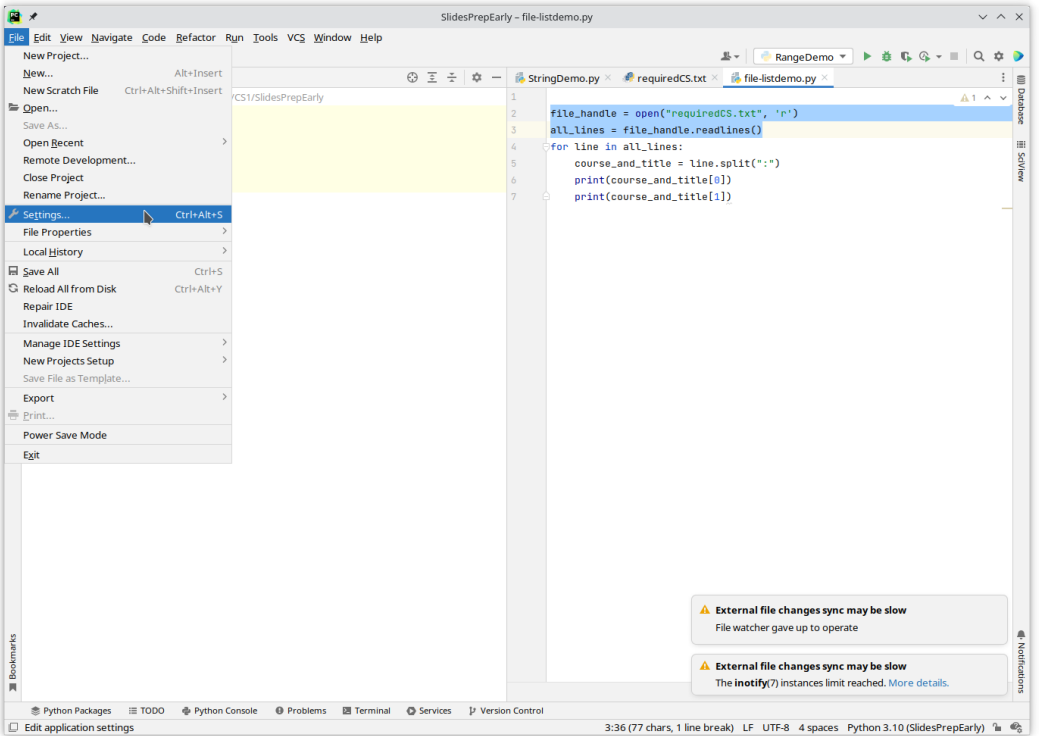
- We are going off the book for this section. So no reading, you can use the official docs as well as our slides:
-
-

Programmers are Lazy

- “programmers are lazy”
  - Joke stereotype of programmers
  - Really “programmers don’t want to reinvent the wheel”
  - If someone else already did some of the work, bring it into your program and use it.
  - Done in python using the “import” statement.
  - In most python both “standard library” and additional libraries available
    - **Really** easy to get those extra libraries today.
  - We are going to use an extra library.
  - Lets look at the steps

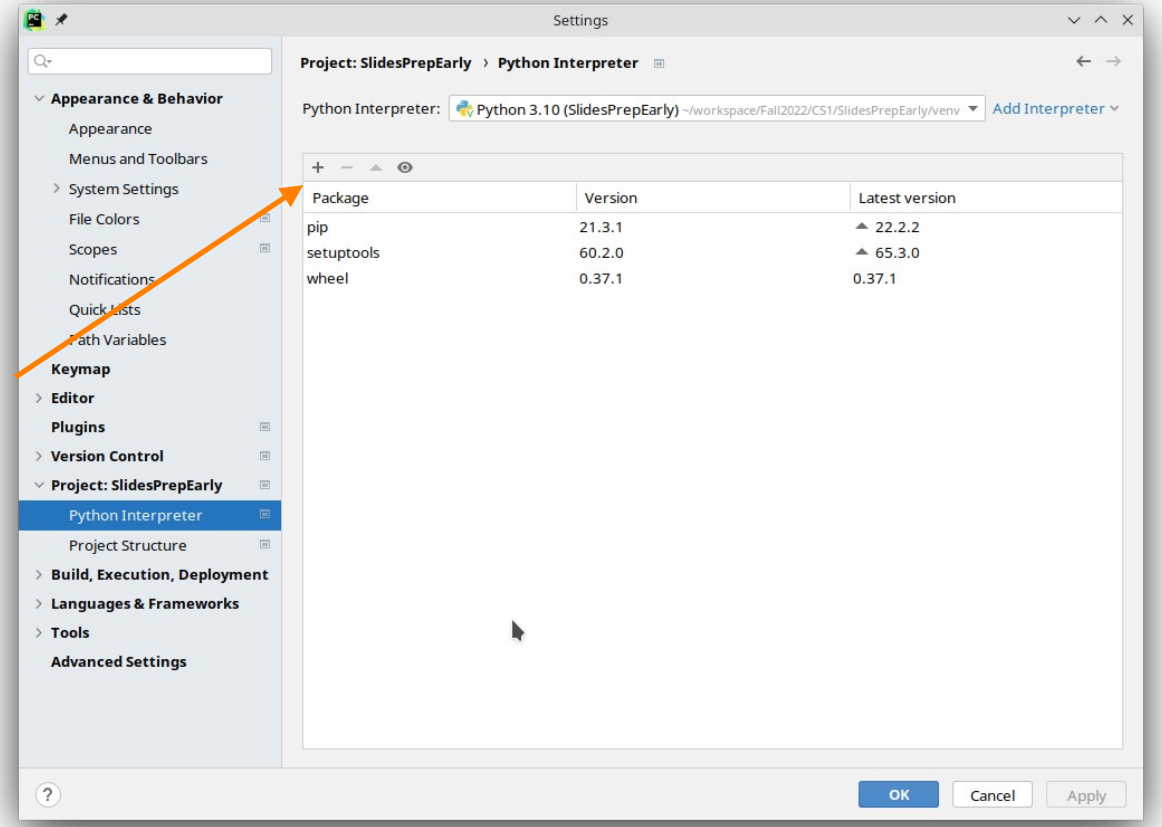
Adding a library to a project

- You could do this with a command line
  - But we'll look at it with pycharm
- We need to add a library
  - On Windows/Linux choose <file><settings>
  - On Mac choose <pycharm><preferences>



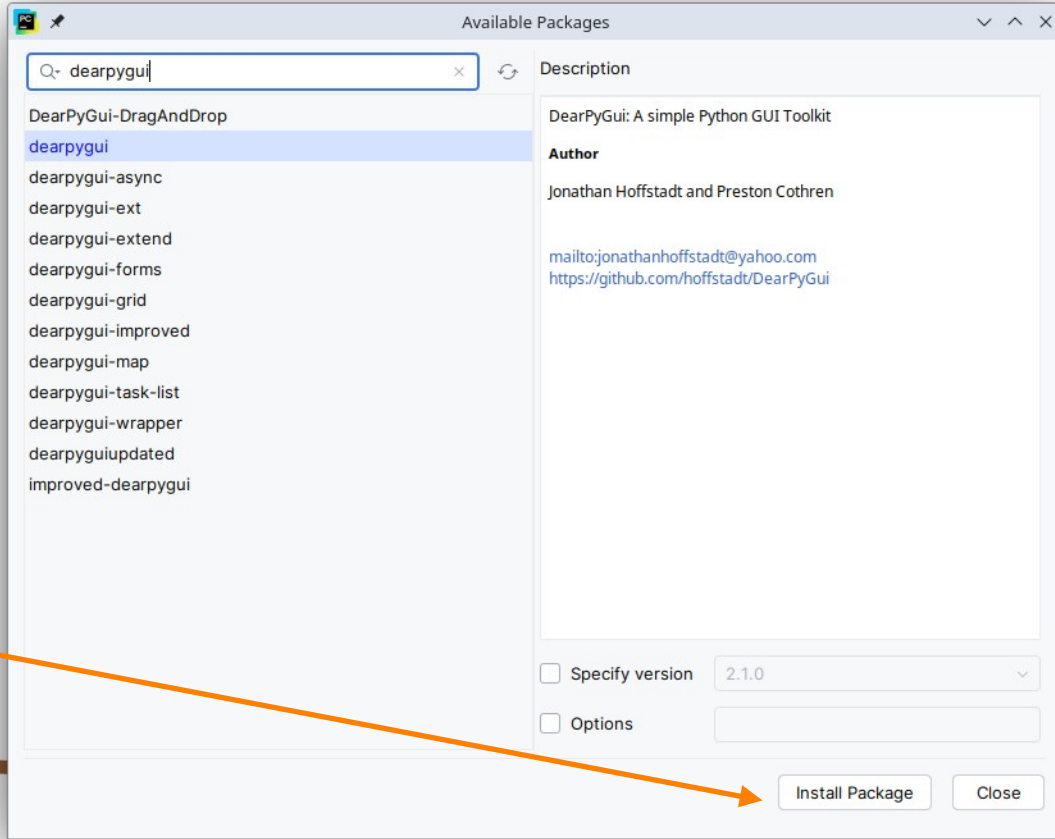
Project Interpreter

- In the Settings dialog
  - Open the project item to find the python interpreter option
  - And select it.
  - Then press the “+” button to add a library
  - The location of the “+” button changes based on system



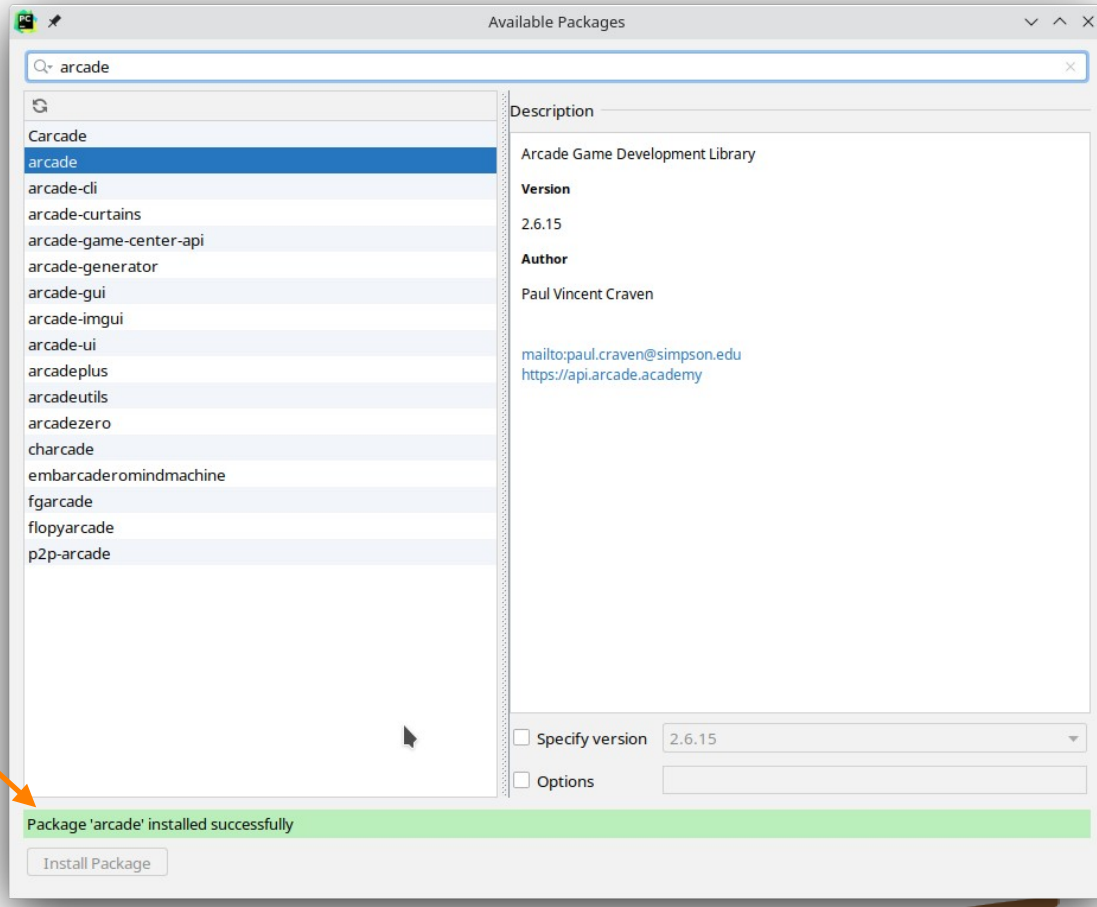
Available Packages

- This brings up an Available Packages dialog.
  - There are 100K+
  - In the search at the top type type the name of the package/library you want to install.
  - Today we will install the dearPyGui library by Jonathon Hoffstadt at. al.
  - When the correct library is selected, press the install package button



Hopefully success

- The install will take a few** seconds
- Then hopefully you will see the green “Package installed successfully” bar
-



- ** your value of “few” will depend on your computer and network speed

Python and Packages

- Python calls all of its modules/libraries/etc “Packages”
- Each Package contains code written for a particular related purpose.
- Need to tell python to use code in packages.
  - Protect project from ‘code bloat’
- Some code is so common that it is always available.
  - print/input/etc
- But for most, need to tell python to use it

import

- The way that you tell python to use another module is the ‘import’ statement.
- Two forms
  - import <package>
  - from <package> import <thing>
- Must be in python file above where you use the thing
  - Technically anywhere
- BUT!!!! always put them at the top of the file

```
import dearpygui.dearpygui
```

- Or

```
from dearpygui.dearpygui import create_context
```

- Or

```
import dearpygui.dearpygui as gui
```

Hello dearPyGui

- Here is the “hello world” equivalent for dearPyGui programs
- Lets try it.
- Notice that windowed programs require more code
- Code highlighted is ‘boilerplate’
- Needs to be in (pretty much) any dearPyGui project

```
import dearpygui.dearpygui as gui

gui.create_context()
gui.create_viewport(title='Hello World', width=600,
height=300)
gui.setup_dearpygui()
gui.show_viewport()
gui.start_dearpygui()
gui.destroy_context()
```

Hello dearPyGui

- Here is the slightly more complicated version
- Lets look at all of the parts.

```
import dearpygui.dearpygui as gui

gui.create_context()
gui.create_viewport(title='Hello World', width=600,
height=300)
with gui.window(label='Hello World', width=600,
height=300):
    gui.add_text('Hello, world!')
gui.setup_dearpygui()
gui.show_viewport()
gui.start_dearpygui()
gui.destroy_context()
```

Drawing

- We'll return to graphical user interfaces later in the semester
- But for now, lets draw.
- Go get Comp151Colors.py from the class website and put it in your project
- Will stop highlighting boiler plate now
- Lets put this in pycharm and review it.

```
import dearpygui.dearpygui as gui_graphics
import comp151Colors

gui_graphics.create_context()
with gui_graphics.window(pos=[0,0]) as window:
    with gui_graphics.drawlist(width=500,
height=500) as draw_list:
        gui_graphics.draw_circle((300,300), 50,
color=comp151Colors.OLIVE,
fill=comp151Colors.MAROON)
gui_graphics.create_viewport(title='Drawing
Example', width=500, height=500, x_pos=0,
y_pos=0)
gui_graphics.setup_dearpygui()
gui_graphics.show_viewport()
gui_graphics.start_dearpygui()
gui_graphics.destroy_context()
```

Drawing

- To draw more, put all of the drawing in the code block for the drawlist

```
import dearpygui.dearpygui as gui_graphics
import comp151Colors

gui_graphics.create_context()
with gui_graphics.window(pos=[0,0]) as window:
    with gui_graphics.drawlist(width=500, height=500) as draw_list:
        gui_graphics.draw_circle((300,300), 50, color=comp151Colors.OLIVE,
fill=comp151Colors.MAROON)
gui_graphics.create_viewport(title='Drawing Example', width=500, height=500, x_pos=0, y_pos=0)
gui_graphics.setup_dearpygui()
gui_graphics.show_viewport()
gui_graphics.start_dearpygui()
gui_graphics.destroy_context()
```

Drawing Coordinates.

- In Cartesian Coordinate System (that you learned in middle school math)
  - Where in the origin in the most common quadrant?
  - Let’s draw on the board?

Drawing Coordinates.

- In Cartesian Coordinate System (that you learned in middle school math)
  - Where in the origin in the most common quadrant?
  - Let’s draw on the board?
  - This is the coordinate system we are most familiar with

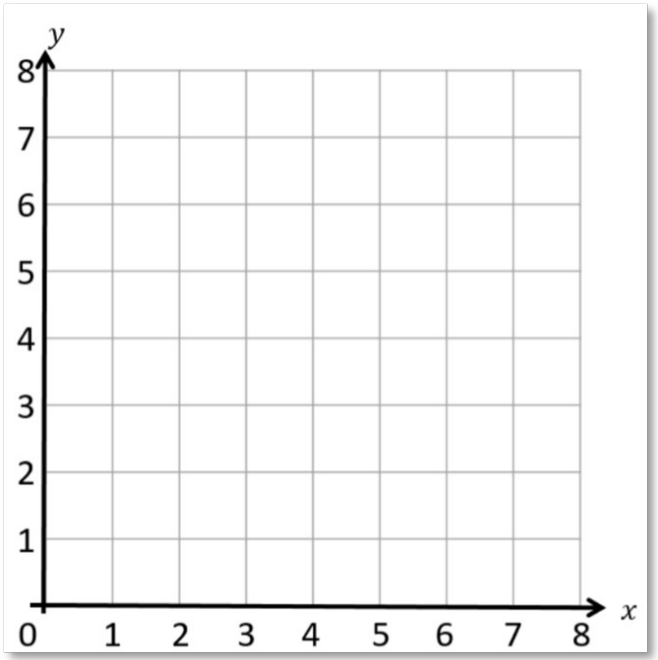


Image credit:https://whiteroseeducation.com/latest-news/how-teach-coordinates-maths

Drawing Coordinates.

- In computers:
  - Coordinates on screens start with the upper left corner of the screen
  - Most window libraries (including DearPyGui) use these same coordinate systems
    - Draw on the board.

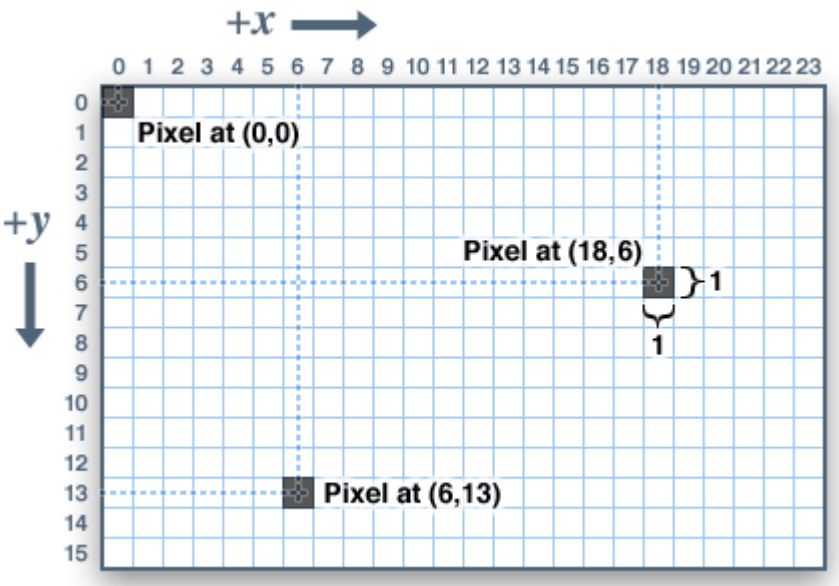


Image credit:https://learn.adafruit.com/assets/1264

Drawing Lines

- Functions in dearPyGui for drawing lines
  - draw_line():
    - Two required parameters:
      - First one is start point in the line (x, y)
      - Second one is end point of the line (x, y)
    - Optional parameters:
      - color
        - Color for the line
      - thickness
        - How many pixels wide is the line
    - Example:

```
draw_line((20,20), (200,200), color=comp151Colors.SEA_GREEN, thickness=4)
```

Let's put it together

- Let's blend this with last time
- Write a program that will
  - Pop up a window that a thousand pixels wide.
  - Using a for loop, draw vertical lines every 100 pixels
  - Make sure they are a contrasting color from the window color.

Dearpygui.draw_arrow

- draw_arrow
  - For now three things in the the parentheses
    - Required:
      - Start point
      - End point
    - The color for the line(s)
    - The thickness of the line
  - The arrowhead is on the start point!

```
gui_graphics.draw_arrow((50,100), (300, 150),
color=comp151Colors.YELLOW, thickness=5)
```

Rectangles

- Rectangles are our next primitive to draw.
  - How might you define a rectangle on a coordinate plane?
  - Lucky volunteer?

Rectangles

- Rectangles are our next primitive to draw.
  - How might you define a rectangle on a coordinate plane?
  - Likely answers:
    - One corner and then width and height
    - Top left and bottom right corners
  - Dearpygui uses the second.

Rectangles

- draw_rectangle
  - Mostly same parameters (stuff in the parenthesis) as line
  - draw_line():
    - Two required parameters:
      - First one is start point in the line (x, y)
      - Second one is end point of the line (x, y)
    - Optional parameters:
      - color
        - Color for the outline of the rectangle
      - thickness
        - How many pixels wide is the line
      - Fill
        - Color to fill the rectangle

```
gui_graphics.draw_rectangle((500,100), (600, 350), thickness=3, color=comp151Colors.WHEAT,fill=comp151Colors.CYAN )
```

Drawing circles

- draw_circle
- Requires
  - Center point (x,y)
  - Radius (in pixels)
- optional
  - Color (for outline)
  - Fill (color for rest of the circle)
  - Thickness optionally can specify line width other than one

- Examples

```
gui_graphics.draw_circle((500,500), 300, thickness=8,
color=comp151Colors.CYAN)
gui_graphics.draw_circle((50,50), 30,
fill=comp151Colors.ORANGE )
```

Triangles

- draw_triangle
  - Required:
    - Three points for the three vertices of the triangle
  - Optional
    - Thickness of the outline
    - Color of the outline
    - Fill color of the triangle

```
gui_graphics.draw_triangle((100, 600), (200, 600), (150, 500), color=comp151Colors.SENNA, thickness=6, fill=comp151Colors.MAROON)
```

Ellipse

- draw_ellipse
  - Required:
    - Two points, upper left and lower right of bounding box
  - Optional
    - Usual thickness, fill and color for outline

```
gui_graphics.draw_ellipse((500, 500), (750,
600), thickness=3,
color=comp151Colors.PURPLE,
fill=comp151Colors.TEAL)
```

Polygon

- draw_polygon
- Draw arbitrary polygon
  - Required:
    - A list of points
    - Each will be a vertex.
    - Make the first and last point identical to make the shape complete
  - Optional
    - Start with same three: fill, color and thickness

```
gui_graphics.draw_polygon([(50, 10), (100, 50), (120, 100), (300, 300), (200, 320), (75, 75), (50, 10)], color=comp151Colors.YELLOW, thickness=7)
```

polyline

- draw_polyline
  - Draw a connected series of line segments
  - Required
    - List of points in the line
  - Optional
    - Line color and line thickness
    - No fill since this is a line.

```
gui_graphics.draw_polygon([(500, 100), (560, 150), (700, 75), (600, 400)], color=comp151Colors.WHITE, thickness=3)
```

Now lets build a face

- Lets put it all together in a coherent whole and build a face or even an emoji

Drawing Text

- draw_text
  - Required:
    - Point for top left of text
    - Actual text to draw
  - Optional
    - Color
    - size

```
gui_graphics.draw_text((300, 500), "DEMO TEXT",
color=comp151Colors.YELLOW, size=32)
```

Assignment

- Next Project?
- And lets talk about the podcast