

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:
- Docs landing page:
 - <https://nicegui.io/#features>
- Please listen to episode 78 (Aug 22 2026) of “the programming podcast”
 - There are links to youtube and spotify on the page when you scroll to the Aug 22 episode
 - <https://www.dthompsondev.com/podcast>
- We will discuss in class in one week.
- <https://medium.com/codefarm-java-ecosystem/engineering-interview-best-practices-what-actually-gets-you-hired-in-the-age-of-ai-087cf54ce037>

Programmers are Lazy

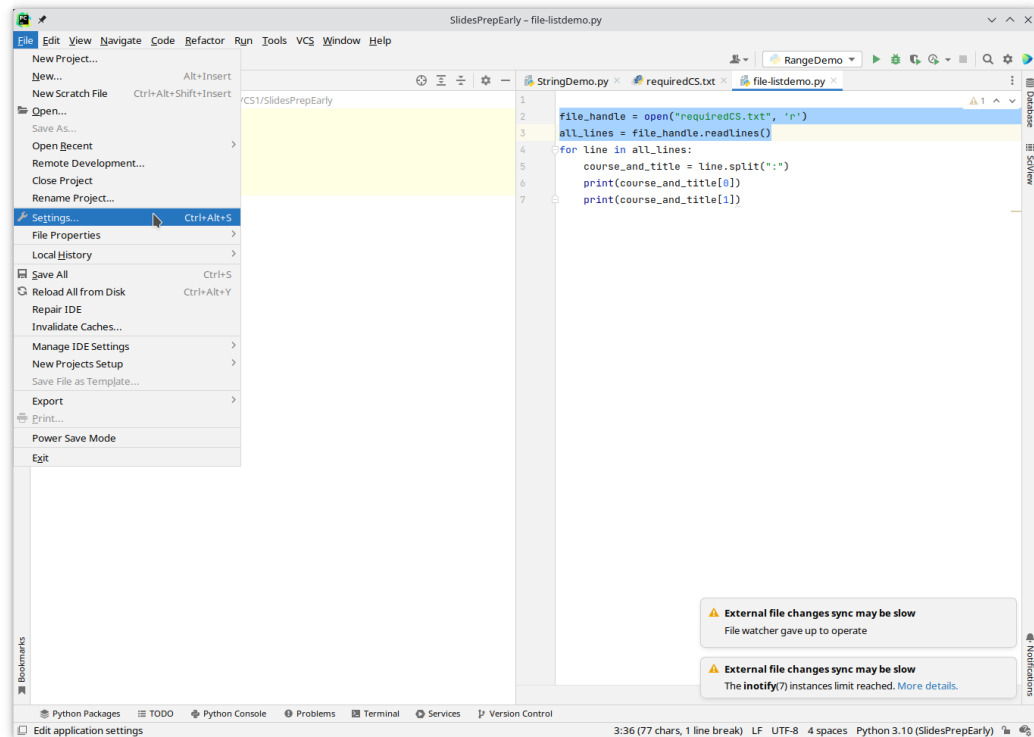


- “programmers are lazy”
 - Joke stereotype of programmers
 - Really “programmers don’t want to reinvent the wheel”
 - If someone else already did some of the work, bring it into your program and use it.
 - Done in python using the “**import**” statement.
 - In most python both “standard library” and 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



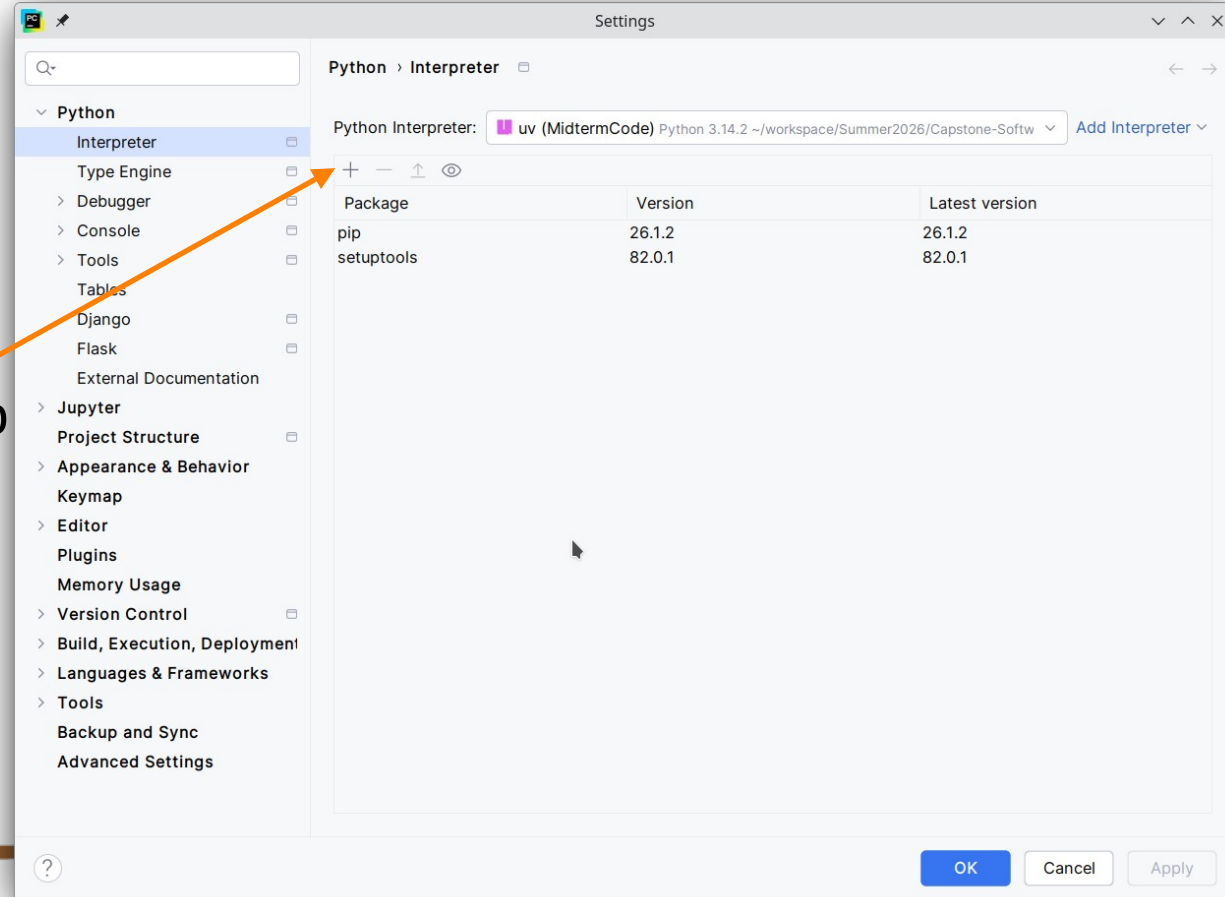
- You could do this with a command line
 - But we'll look at it with pycharm
- We need to add a library
 - On Windows/Linux choose <file><settings>
 - On Mac choose <pycharm><preferences>



Project Interpreter



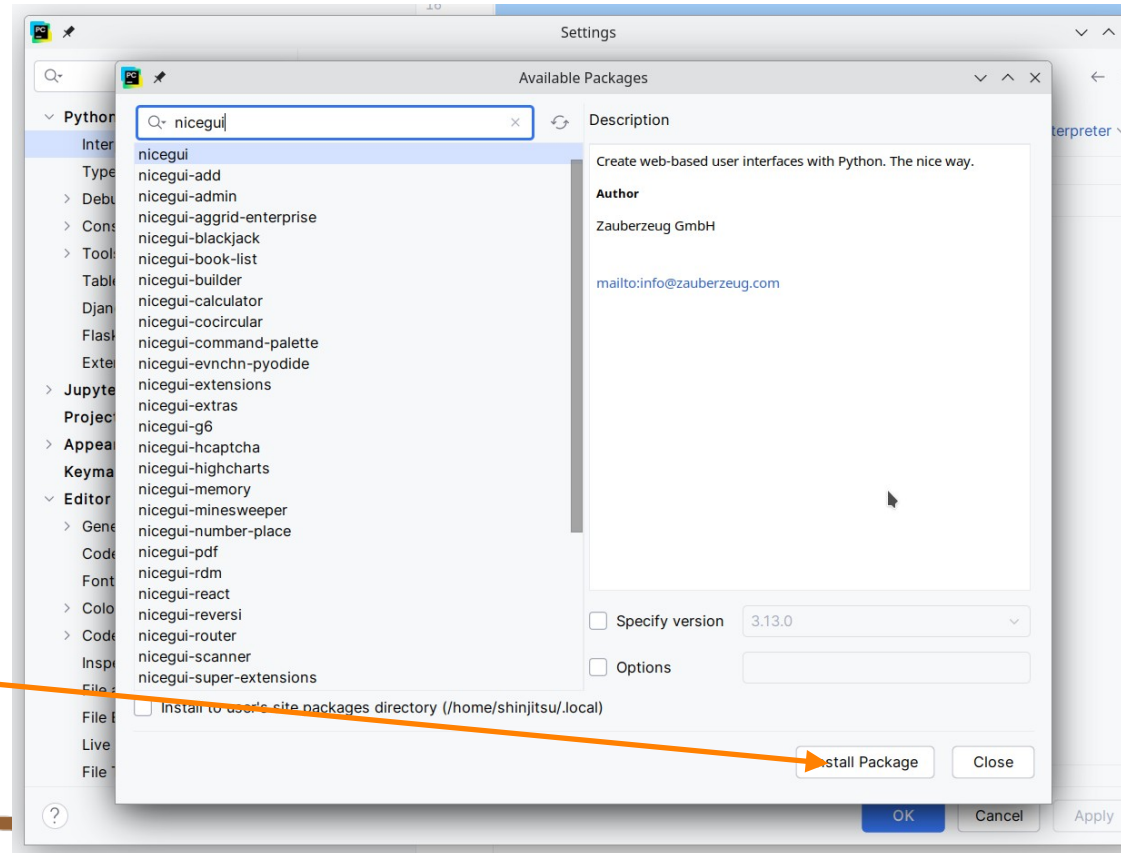
- In the Settings dialog
 - At the top left is a Python option
 - select it.
 - The choose interpreter
 - Then press the “+” button to add a library
 - The location of the “+” button changes based on system



Available Packages



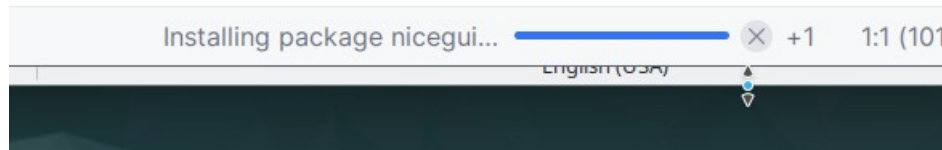
- This brings up an Available Packages dialog.
 - There are 100K+
 - In the search at the top type the name of the package/library you want to install.
 - Today we will install the nicegui library by Zaubergezeug
 - When the correct library is selected, press the install package button



Hopefully success



- The install will take a **few**** seconds
- In late 2025 version of UI (Still the default in mid-2026)
 - press “install package”
 - then close the settings
 - watch the progress bar at bottom of pycharm
 - Hopefully no errors
- ** 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 nicegui

– Or

from nicegui import ui

– Or

import nicegui as gui

NiceGui



- Last fall I used DearPyGUI
 - But it would fail with errors on a small number of Windows 11 machines
 - (3 students out of 50)
- Enter NiceGUI
 - Last December (2025) Zauberzeug GmbH, a german robotics company released a big overhaul of NiceGUI
 - Germans made a big deal of the punny pronunciation
 - Nice-Guy
 - Web based UI, so it should work for all students.
 - So we'll use this library this time

Hello NiceGui



- Here is the “hello world” equivalent for NiceGUI programs
- Lets try it.
- Notice that windowed programs require more code
 - Though NiceGUI tries to minimize this
- When we run this in pycharm – we get links to view in the web
 - If you have only one network interface you won't have all of these

```
from nicegui import ui  
  
ui.label('Hello NiceGUI!')  
  
ui.run()
```



```
/home/shinjitsu/.local/bin/uv run /home/shinjitsu/workspace/Fall2026/CS1/NiceGUIInitialInvestigation/.venv/bin/python /home  
NiceGUI ready to go on http://localhost:8080, http://172.17.0.1:8080, http://172.18.0.1:8080, and http://192.168.1.215:8080
```

Hello NiceGUI



- Here is the slightly more complicated version
 - We can change look/feel of NiceGUI components using CSS
 - Tailwind-css
 - Eg <https://tailwindcss.com/docs/colors>
 - Or plain css
 - Since these are web components
- Lets look at all of the parts.

```
from nicegui import ui
```

```
hello_label = ui.label('Hello NiceGUI!')  
hello_label.classes("text-rose-600")  
hello_label.style("font-size: 200%")
```

```
second_label = ui.label('This is a second label')  
second_label.classes("text-blue-800")  
second_label.style("font-size: 18px")
```

```
ui.run()
```

Hello NiceGUI



- Lets look at all of the parts.
- Two labels
 - So two lines of text, one on top of each other
 - First one is has text a medium dark rose, and text is twice as large as default
 - Relative text size
 - Second one has fairly dark blue text with 18 point font
 - Fixed text size

```
from nicegui import ui
```

```
hello_label = ui.label('Hello NiceGUI!')  
hello_label.classes("text-rose-600")  
hello_label.style("font-size: 200%")
```

```
second_label = ui.label('This is a second label')  
second_label.classes("text-blue-800")  
second_label.style("font-size: 18px")
```

```
ui.run()
```

Drawing



- We'll return to graphical user interfaces later in the semester
 - hopefully
- But for now, lets draw.
- Drawing 'canvas' in NiceGUI is called
 - interactive_image
- Use it and draw on it
 - Explanation on next

```
from nicegui import ui
```

```
drawing = ui.interactive_image(size=(1000, 1000),  
                                cross=False,  
                                sanitize=True)  
drawing.classes('w-lg bg-gray-200')  
ui.run()
```

Drawing



- Create Image

- Size needs to be tuple
 - Width and height
- cross
 - If true will put horizontal and vertical line cross centered on mouse
 - If false, no lines
- sanitize
 - Usually should be true – when true will make sure there are no security injections

```
from nicegui import ui
```

```
drawing = ui.interactive_image(size=(1000, 1000),  
                               cross=False,  
                               sanitize=True)  
drawing.classes('w-lg bg-gray-200')  
ui.run()
```

Background tiling:

w-lg
w-64 are common

Then add color
bg-<tailwind color>

- Classes

- Pass a string with tailwind classes

Drawing



- To draw something useful, put it in the content using `set_content`
 - Set content takes a string with one or more SVG (image format) shape definitions
 - c.f. https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorials/SVG_from_scratch/Basic_shapes
 - Let's look then try it out.

```
from nicegui import ui

drawing = ui.interactive_image(size=(1000, 1000),
                                cross=False,
                                sanitize=True)
drawing.classes('w-lg bg-gray-200')
drawing.set_content('<circle cx="500" cy="500" r="50" fill="orange" />')
ui.run()
```

Drawing Coordinates.

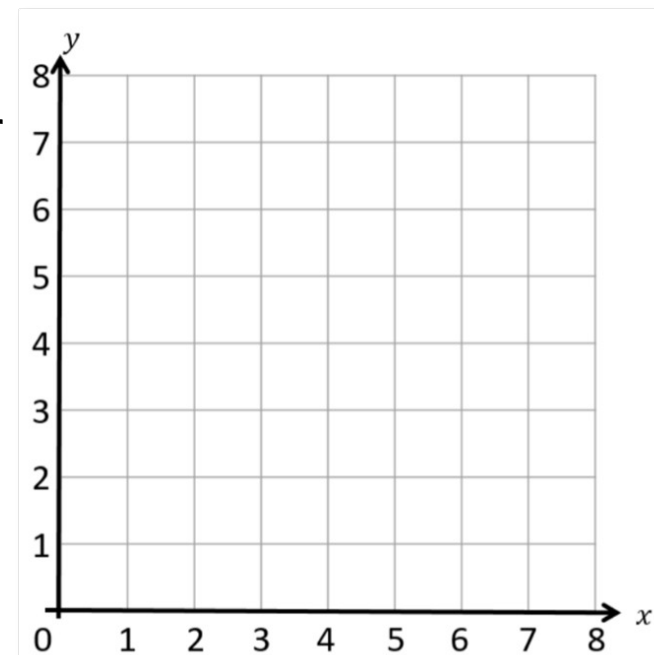


- In Cartesian Coordinate System (that you learned in middle school math)
 - Where is the origin in the most common quadrant?
 - Let's draw on the board?

Drawing Coordinates.



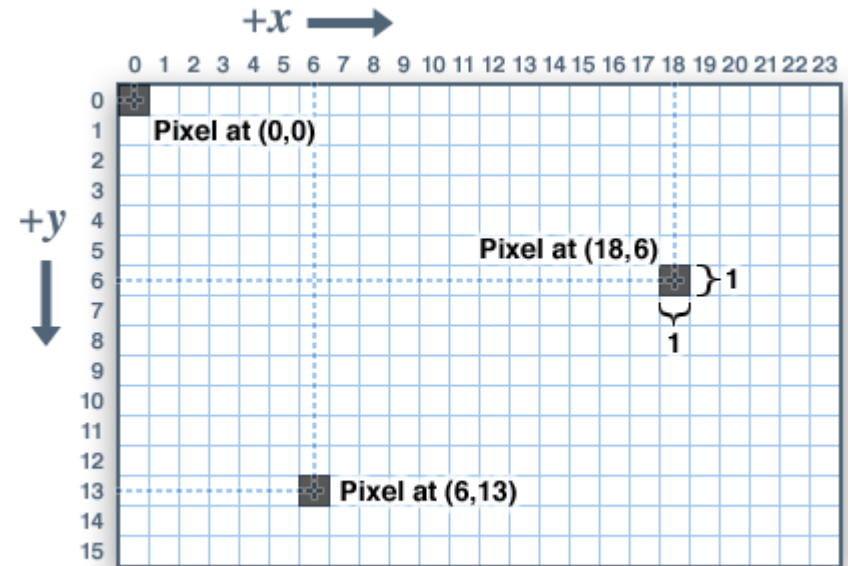
- In Cartesian Coordinate System (that you learned in middle school math)
 - Where is the origin in the most common quadrant?
 - Let's draw on the board?
 - This is the coordinate system we are most familiar with



Drawing Coordinates.



- In computers:
 - Coordinates on screens start with the upper left corner of the screen
 - Most window libraries (including NiceGUI) use these same coordinate systems
 - Draw on the board.
 - Let's move that circle to have a center-y of 200 and see this in action.



Drawing Lines



- To draw a line in NiceGUI, use the SVG line string
 - I recommend triple quoted strings!
- String format:
 - `<line x1= "int value" y1="int value" x2="int value" y2="int value" stroke="some color name" stroke-width="int value"/>`
 - The x1, y1 is the first point in the line
 - X2, y2 is the end point of the line
 - Stroke is optional, but will be background color (invisible) if not included
 - Stroke-width is optional and will default to one pixel
 - **Be very careful about the internal double-quotes** (don't leave one out like I did)

String join()



- Remember strings are objects
 - So we can ask them to do stuff for us
- join () is a 'function' we can call on a string
 - Put a list of strings in the ()
 - It will return a new longer string
 - Eg:

```
all_shapes = []
shape1 = '<circle cx="500" cy="500" r="50" fill="orange" />'
shape2 = ""<line x1="200" y1="0" x2="200" y2=1000 stroke="blue" stroke-width="5"/> ""
all_shapes.append(shape1)
all_shapes.append(shape2)
shapes_to_draw = "\n".join(all_shapes)
```
- This will create a long string with the two original strings separated by newline
- Let's try it

Let's put it together



- Let's blend this with last time
- Write a program that will
 - Pop up a window/drawing area 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.
 - You will likely need (to do it efficiently)
 - A for loop
 - String.join()
 - F-strings
 - And of course NiceGUI stuff.

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
 - NiceGUI uses the first.

Rectangles



- Drawing rectangles

- Use rect svg string (again use triple quoted strings by default)

```
"""<rect x="int value" y="int value" width="int value" height="int value"/>"""
```

```
"""<rect x="int val" y="int val" width="int val" height="int val" fill="color name"/>"""
```

- The x and y are the upper left hand corner of the rectangle
- Width and height are just that
- fill is optional, if omitted, defaults to black (on linux/firefox)
- stroke-width is also optional – usually only useful if fill is set to transparent.

Rounded Rectangle



- You can create a rectangle with rounded corners by adding an rx and an ry to the description string
 - Represents the x and y radius of the curved 'corners'
 - Eg:
`<rect x="60" y="10" rx="10" ry="10" width="30" height="30" stroke="black" fill="green" stroke-width="5"/>`
 - Note that it is possible to have a different fill color than 'stroke' (outside edge) color
 -

Drawing circles



- We drew a circle once before, but now let's look closer and understand
 - Circle needs cx and cy (center x and y) and a radius (r)
 - It can also take stroke, stroke-width and fill
 - Eg:
 - `<circle cx="25" cy="75" r="20" stroke="red" fill="transparent" stroke-width="5"/>`

Ellipse



- Drawing ellipse using NiceGUI/SVG
 - Sort of a combination of circle and rounded rectangle
 - Provide the center x and y, and then x and y radius of the 'corners'
 -
 - Eg:
 - `<ellipse cx="75" cy="75" rx="20" ry="5" stroke="red" fill="transparent" stroke-width="5"/>`
 - The radius of the corners is also the width and height of the 'bounding box' of ellipse

polyline



- In SVG, a polyline is a series of connected line segments.
 - Need the usual optional stroke and stroke-width
 - Also must have points, can be separated by commas or 'white space'
 - Every pair of numbers will be an x,y point pair
 - Eg
- ```
<polyline points="600 110 650 120 700 115 750 130 800 125 850 140 900 135 950 150 1000 145" stroke="orange" fill="transparent" stroke-width="5"/>
```
- If you have an incomplete final point, and sanitize is true, then the extra number is ignored.

# Polygon



- A polygon, is like a polyline, but the last point will automatically be connected to the first (and then can fill)
  - Eg
- ```
<polygon points="50, 160 55, 180 70, 180 60, 190 65, 205 50, 195 35, 205 40, 190 30, 180 45, 180" fill="green", stroke-width="4"/>
```
- Let's try it out.

Now lets build a face



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

Drawing Text



- Can also draw text on these NiceGUI interactive images
- Add text to content string
- EG:

```
<text x="100" y="900" font-size="14" fill="black" writing-mode="vertical-rl" >We will draw this</text>
```

- The x and y are the point to begin drawing the text.
- Font-size is what it says
- Fill will be font color
- writing-mode defaults to horizontal, to do vertical in English vertical-rl

—

Assignment



- Next Project?
- And lets talk about the podcast
- And perhaps this article
- <https://medium.com/codefarm-java-ecosystem/engineering-interview-best-practices-what-actually-gets-you-hired-in-the-age-of-ai-087cf54ce037><https://medium.com/codefarm-java-ecosystem/engineering-interview-best-practices-what-actually-gets-you-hired-in-the-age-of-ai-087cf54ce037>
-