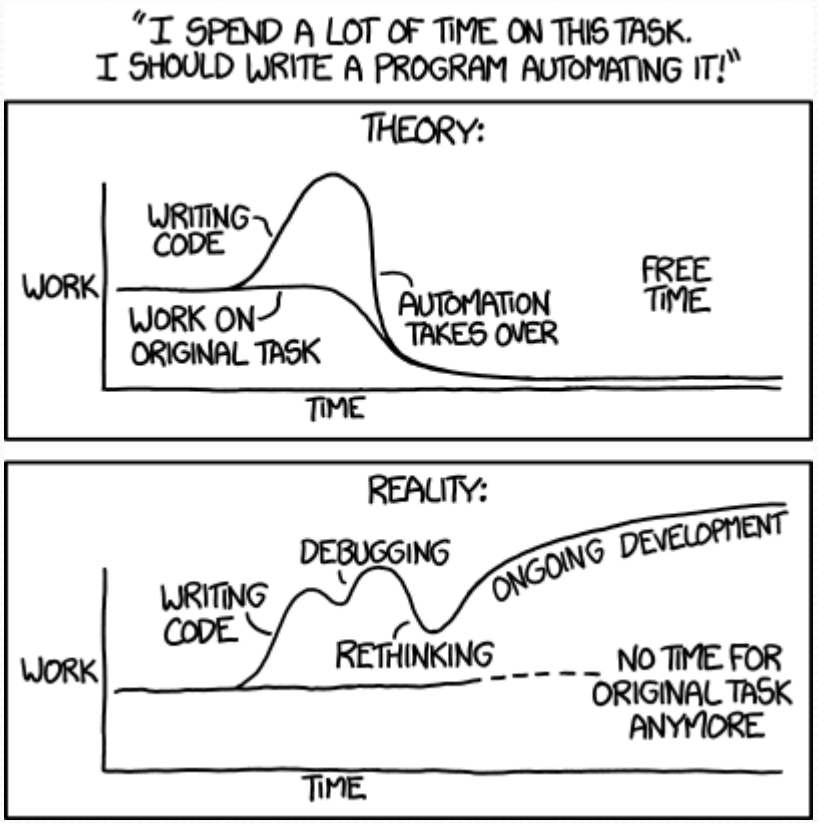
CS 115, Autumn 2021

Lecture 33: advanced editors; GUIs

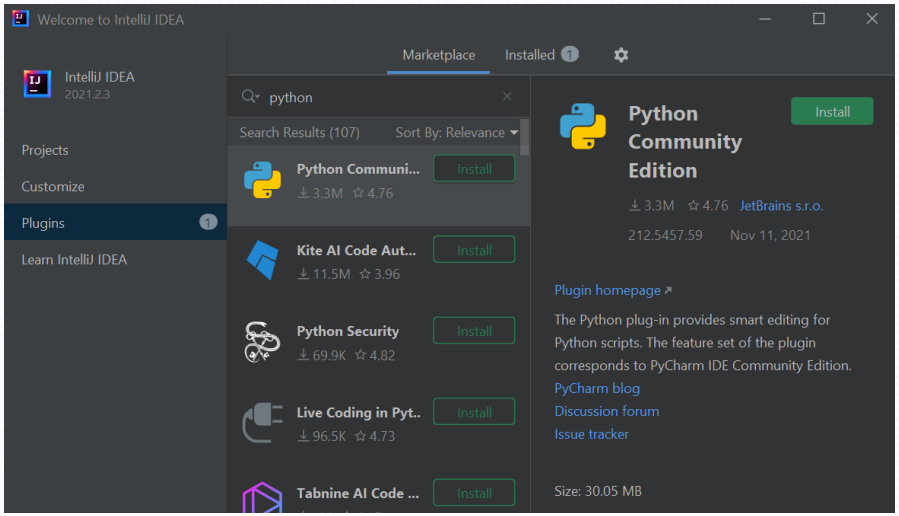


Thanks to Marty Stepp and Stuart Reges for parts of these slides

Using advanced editors

- Some editors come with the ability to run Python by default
  - With many you need to install a plugin
    - Search for "python" to find one

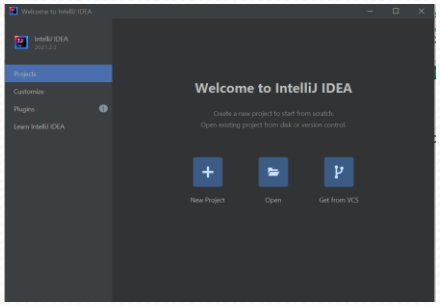
VSCode



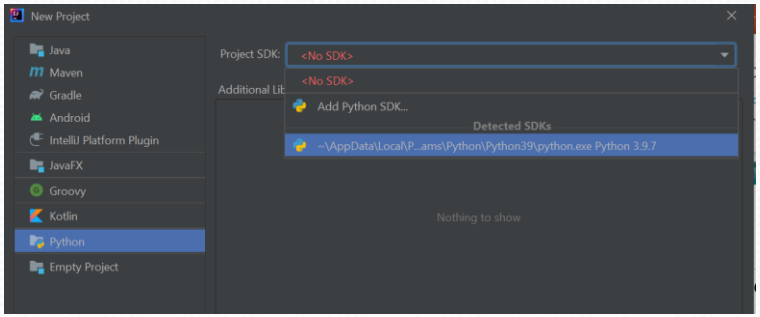
IntelliJ

Using advanced editors

- You need to create a project to be able to run your code in many advanced editors
  - Select "New Project"

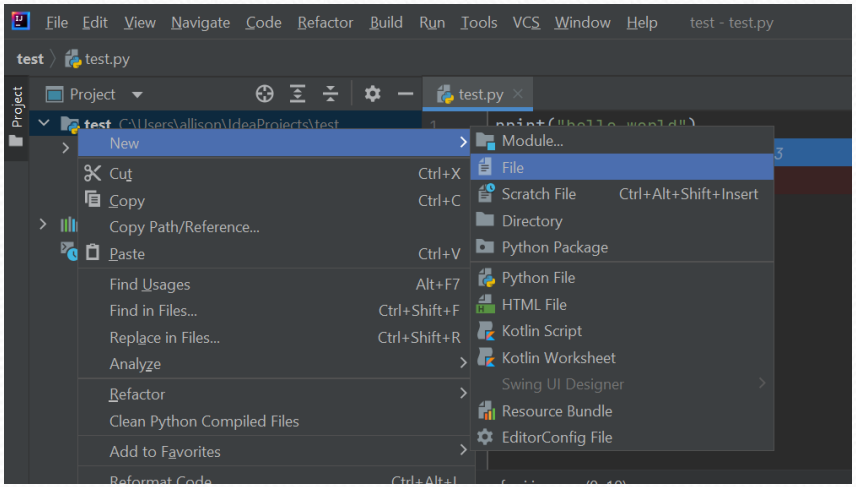


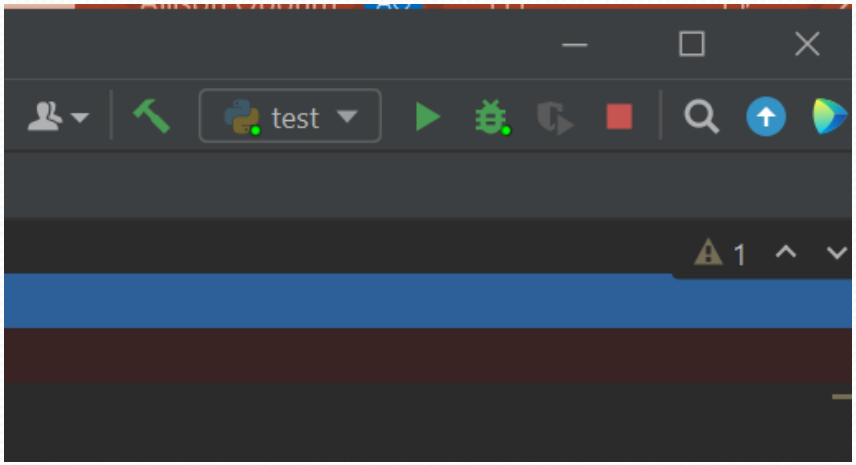
  - Select "Python", click the arrow next to Project SDK. You should see an option at the bottom under "Detected SDKs". Select this.
  - Select any name for your project



Using advanced editors

- Once you have a project you will need to create a file to write your code in.
  - Right click on the folder with the same name as your project, select "New" and then select "File".
  - Run your code by pressing the green arrow/triangle in the upper right





Graphical User Interfaces

Why GUIs?

GUI: Graphical User Interface

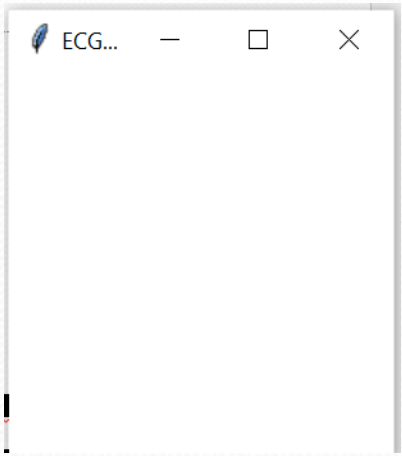
- So we can write programs where the user doesn't have to interact with the console
- Use already created components instead of drawing our own
- Try event-driven programming

Creating your own GUI

- Create a window with `make_window`
  - `make_window()` – creates and returns a window with a white background and a default title of ECGUI
  - `make_window(`**`title`**`, `**`background_color`**`)` – creates and returns a window with the title on the top bar and with the specified background color
- You will need `ECGUI.py` in the same folder as your code

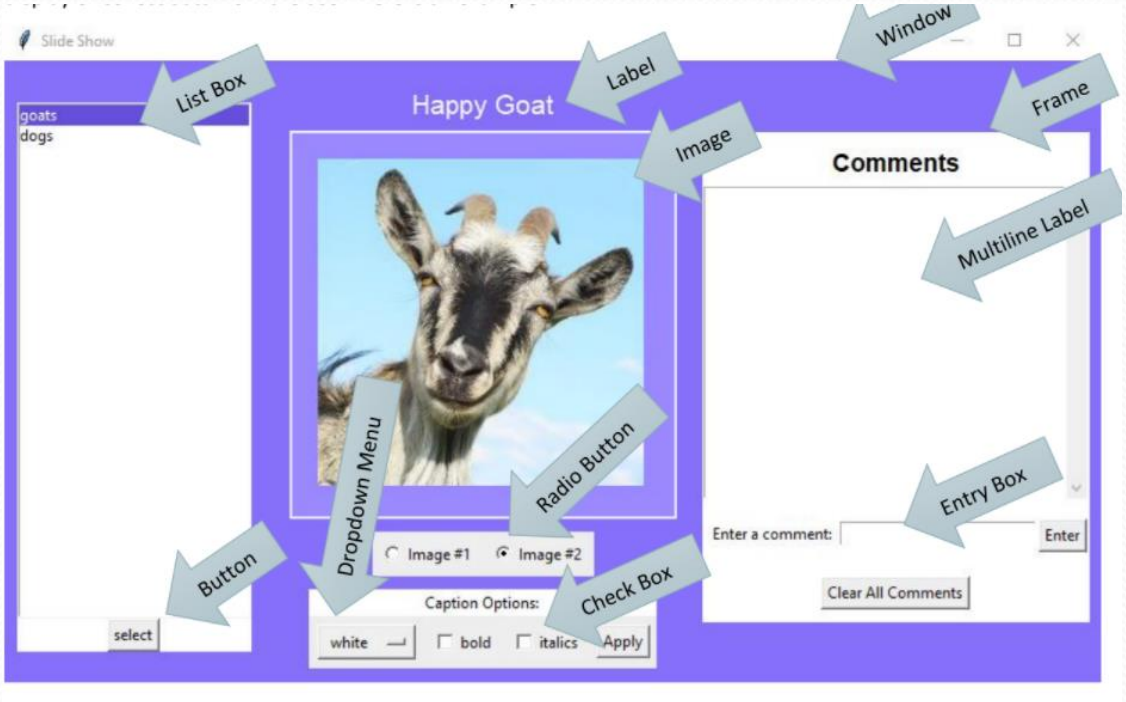
Example:

```
from ECGUI import *
window = make_window()
```



Adding components to your GUI

- You can add the following elements to your GUI:
  - `button`
  - `label`
  - `multiline_label`
  - `image`
  - `entry_box`
  - `radio_buttons`
  - `checkbox`
  - `dropdown`
  - `list_box`
  - `message_box`



- `canvas` – `drawing_panel` but added to a window with other things

Buttons

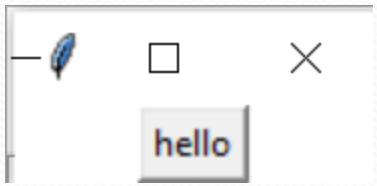
- **Button** – an element the user can click to make a particular action occur

  - `add_button(`**window**`,` **text**`)` – adds a button with the passed in text on it to the passed in window. Returns the button.

Example:

```
from ECGUI import *

window = make_window()
button = add_button(window, "hello")
```



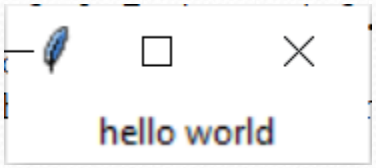
Labels

- **Label** – an element for displaying text (usually non-interactive) on the interface

  - `add_label(`**window**`,` **text**`)` – adds a label with the passed in text in it to the passed in window. Returns the label.

Example:

```
from ECGUI import *
window = make_window()
button = add_label(window, "hello world")
```



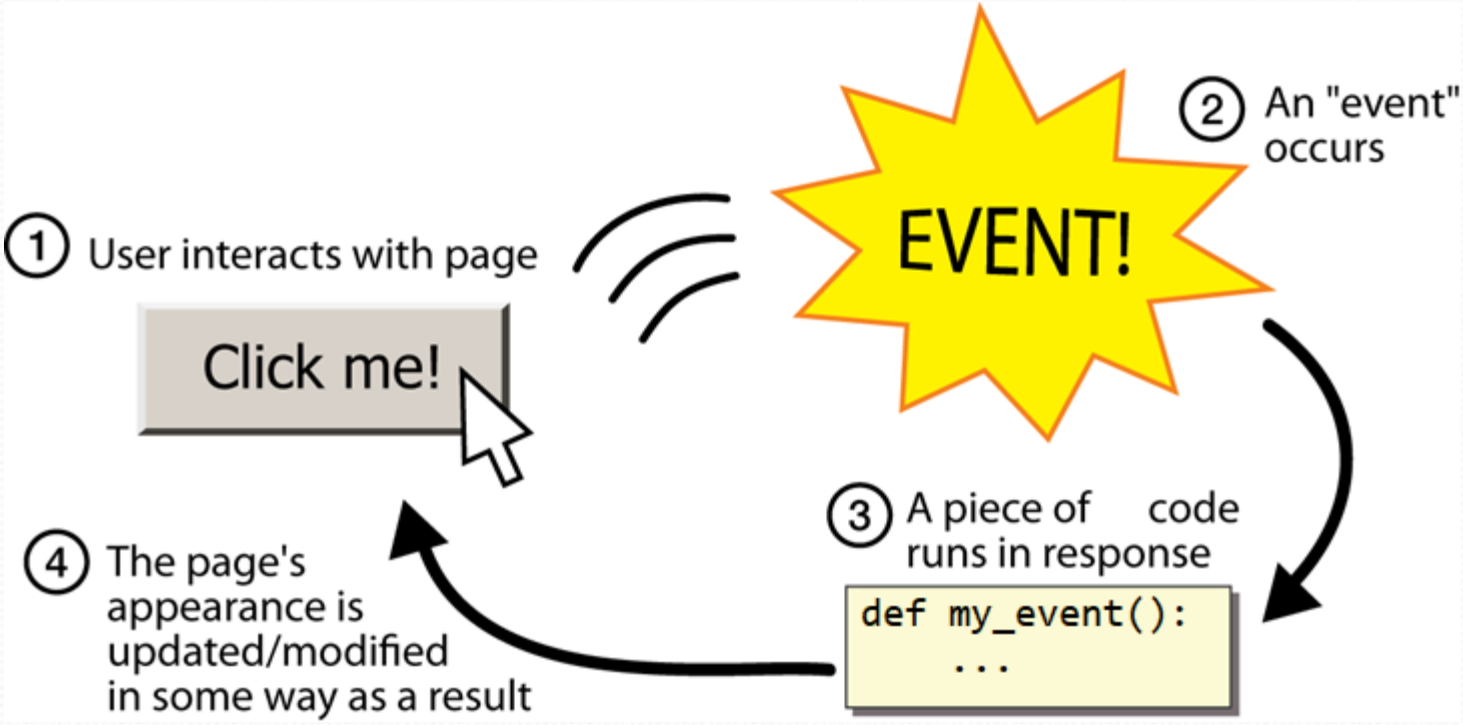
Making our GUI react

- Right now we can add components to our GUI but they don't do anything.
  - How can we make the text change when we click the button?
  - How can we find out when the button will be clicked?

Types of Programming

- So far, we have been writing *procedural programs*
  - Programs that start at the beginning of main and execute one statement after the other until they run out of statements
- We need a different style of programming for creating our own GUIs

Event-driven programming



- We don't know what the user will do first so we can't start at the beginning of `main` and execute to the end - we must respond to user actions called **events**

- **event-driven programming**: writing programs driven by user events

Adding events

- In order for your program to be able to respond to events it has to be listening.
  - Tell your GUI to keep listening by including the following instruction at the end of your program:

```
window.mainloop()
```

Replace "window" with the name of your window variable

Adding events

- Set a button to react when it is clicked by setting its command:

  button_variable`['command'] =` **function_name**

  Make sure not to include () after your function name!
  - That will call the function right now. You want it to be called when the action happens

Example:

```
from ECGUI import *
def print_something():
    print("hello world!")
window = make_window()
hello_button = add_button(window, "hello")
hello_button['command'] = print_something
window.mainloop()
```

How can we change the GUI?

- `change_label(label, message='_SAME_', fg_color='', bg_color='',font_family='', font_size=12, bold=False, italics=False)`