

# SOFTWARE: APPS

(PROGRAMS THAT DO THINGS)

OT 699 – WEEK 3

CHRIS LAINE, PHD.

# OBJECTIVES THIS WEEK

- 1) Basic concepts for App design
- 2) Build a fully functional graphical user interface in Python!

# BASIC CONCEPTS

What is an App?

A game, a word processor, even a website with more than simple text

Due to the popularity of the App store, “App” has started to mean “on my phone”, although this is not a distinction made in the app development community

What an App usually isn’t?

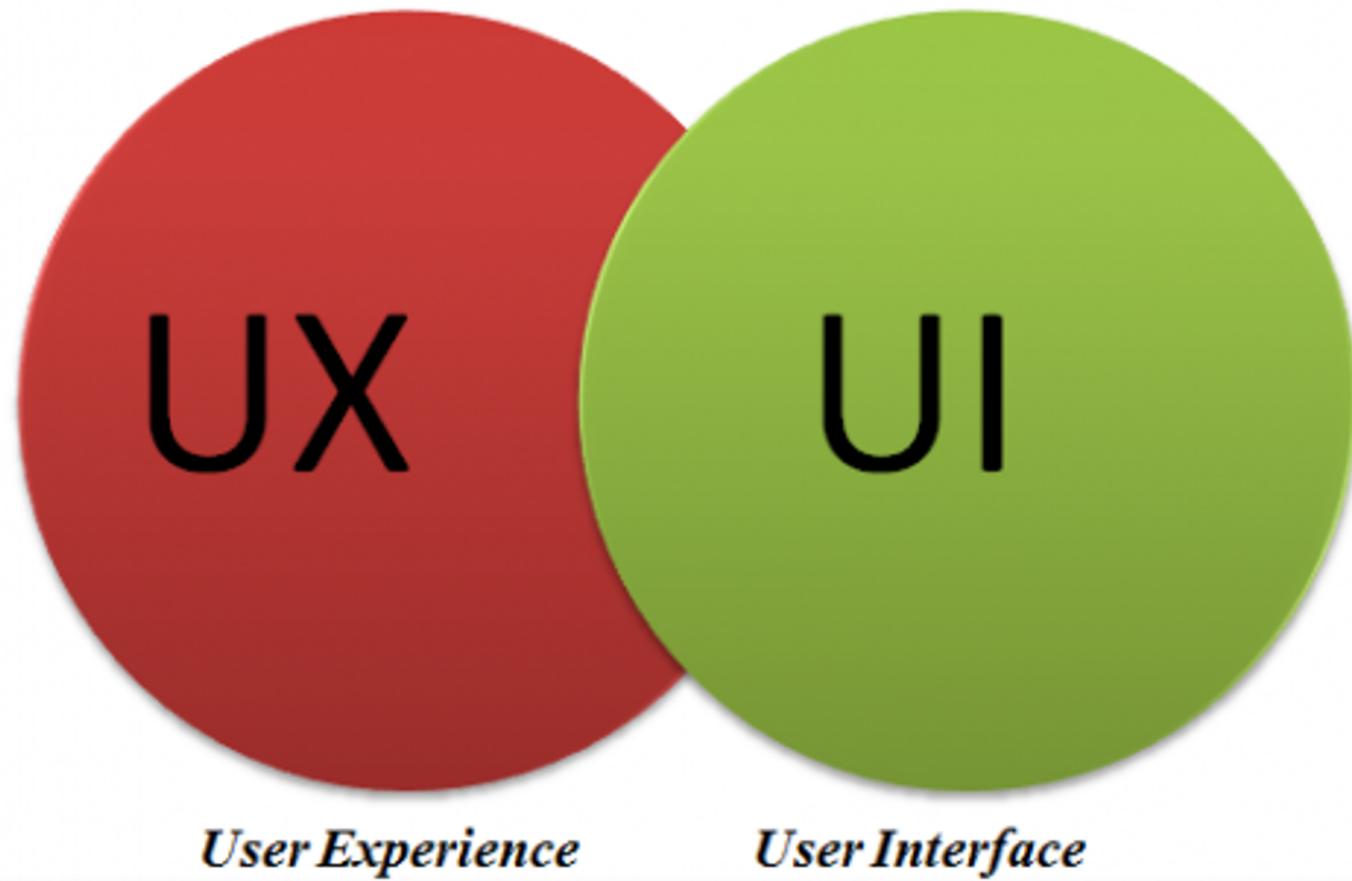
Command line code that you write yourself to do some math on data you collect

	Desktop	Web	Mobile	Mobile Web
<b>Facebook</b>	None	Open through browser on computer (Chrome, Safari, FireFox, IE, etc)	Open through icon downloaded in Google Play, iTunes, etc.	Open through browser on device at <a href="http://m.facebook.com">http://m.facebook.com</a>
<b>Photo Editing</b>	iPhoto, Paint, Microsoft Office Picture Manager	Sites like Flickr and PicMonkey—launched by computer	Instant Retro Photo, PicSay (Android); PicStitch, Be Funky Photo Editor (Apple)	Flickr, etc, launched through mobile browser
<b>Solitaire</b>	Comes stock on Microsoft in “Accessories”—doesn’t need Internet	worldofsolitaire.com; games.com; solitaire-cardgame.com	The Solitaire Games (Apple); Solitaire Free Pack (Android)	worldofsolitaire.com; games.com; solitaire-cardgame.com opened in mobile browser

# CONSIDERATIONS BEFORE YOU TRY MAKING AN APP:

- Who is it for
- What does it do
- Why is this needed
- How and where should a user interact with it
- Limitations of device hardware or software
- Who will build it? How fast? And for how much?

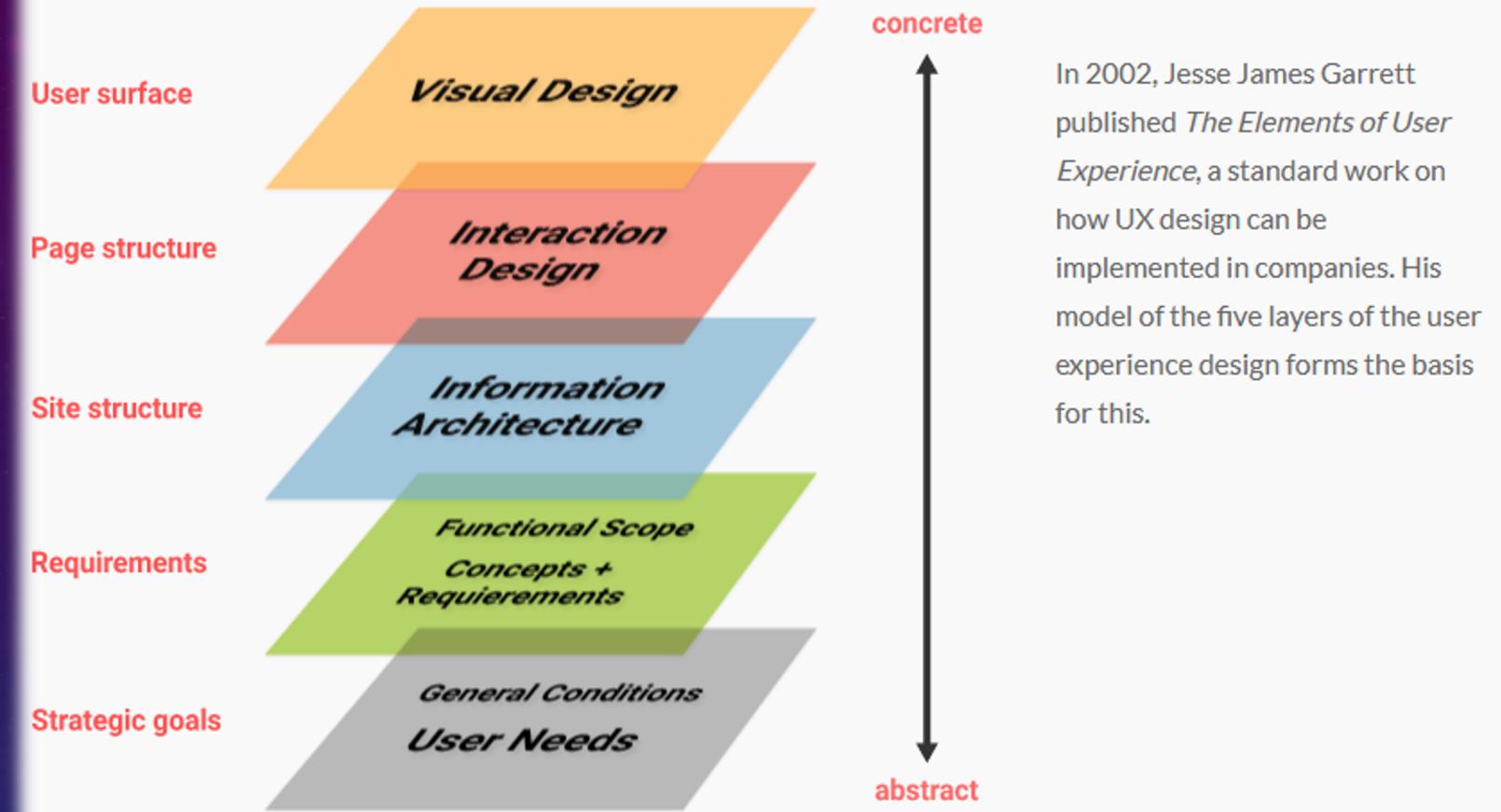
# *Know the Difference*



User Stories  
Usability Testing  
User Research  
Personas

Layout  
Visual Design  
Branding

\* GUI = graphical user interface. Not all interfaces are graphical.

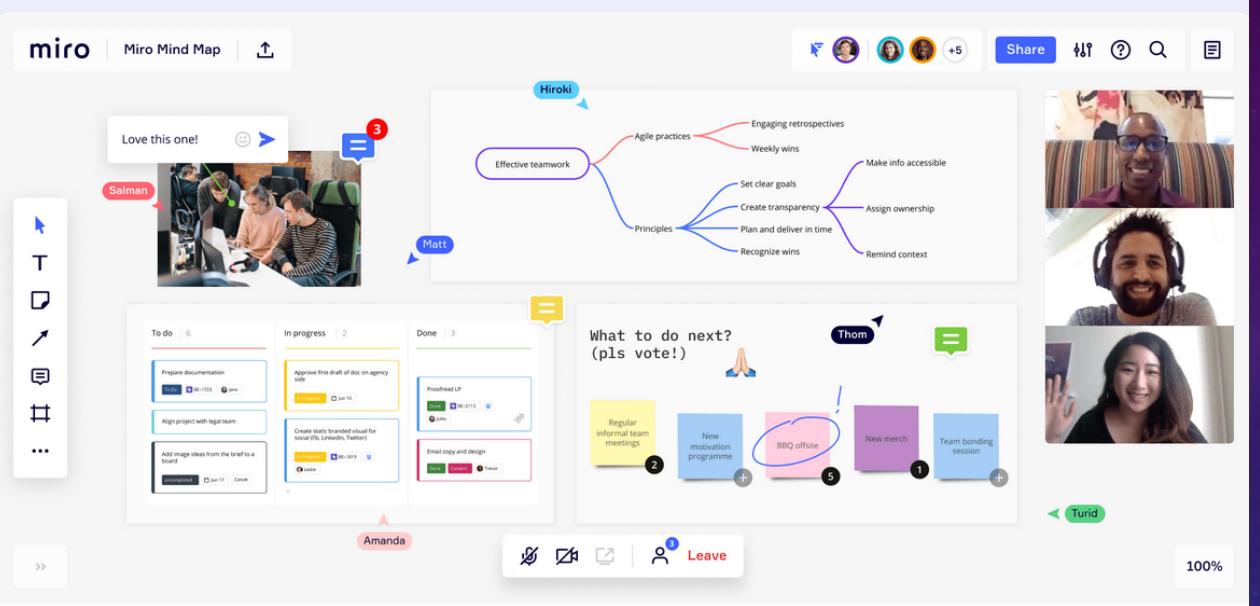


In 2002, Jesse James Garrett published *The Elements of User Experience*, a standard work on how UX design can be implemented in companies. His model of the five layers of the user experience design forms the basis for this.

Nearly all app development in industry is an iterative process that starts with abstract, high level thinking about the user.

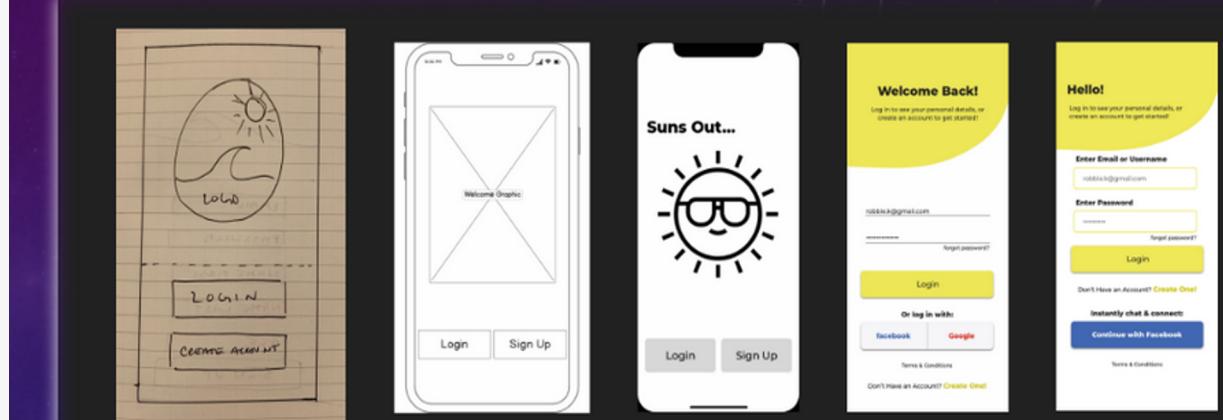
This process is full of sketches, sticky notes, polls, and interviews. The end is essentially a layout design of your app... something that can be handed off to a software developer.

# FROM UX TO UI



Brainstorming software, various online whiteboard sites, trello (simple) or **miro** (e.g. <https://miro.com/>) are valuable for team efforts.

Help to link subgroups during design phase (e.g. developers with user experience specialists)



**Wireframe stage:** usually starts as a sketch and then you can use dedicated wireframing software or graphics design software of your choosing... This is where UI begins to depart from UX.

**Software:** Balsamiq, Framer, Figma, SmartDraw Webflow, Sketch.

Short 4 minute video: <https://balsamiq.com/learn/>

## UX designer vs UI designer vs Front-end Developer roles comparison

	UX designer	UI designer	Front-end dev
Skills and knowledge	Needs marketing, technological and psychological knowledge	Needs graphic design skills, should know some specifics of front-end	Needs web development skills
Responsibilities	Navigates the design process	Designs the visual of the user interface	Develops part of the web application
What they need to start working?	Works on the basis of the research of users needs	Works on the basis of requirements	Works on the basis of requirements and design
Stage of development	Designs the user journey	Prepares the graphic layout according to the user journey	Writes code that connects the layout and the user journey
The role in UX design	Conducts user research, gathers feedback, works closely with the (internal or external) customer and end-user	Should be part of the UX design team	May play a role in the development team and/or in the UX design team
Do they work on development?	Is present before, throughout and after the development process	Usually, ends job with hand-over to developers	Should not start development before getting requirements (and the design)

For web app design, UI is often still conceptual.

The actual code making is split between:

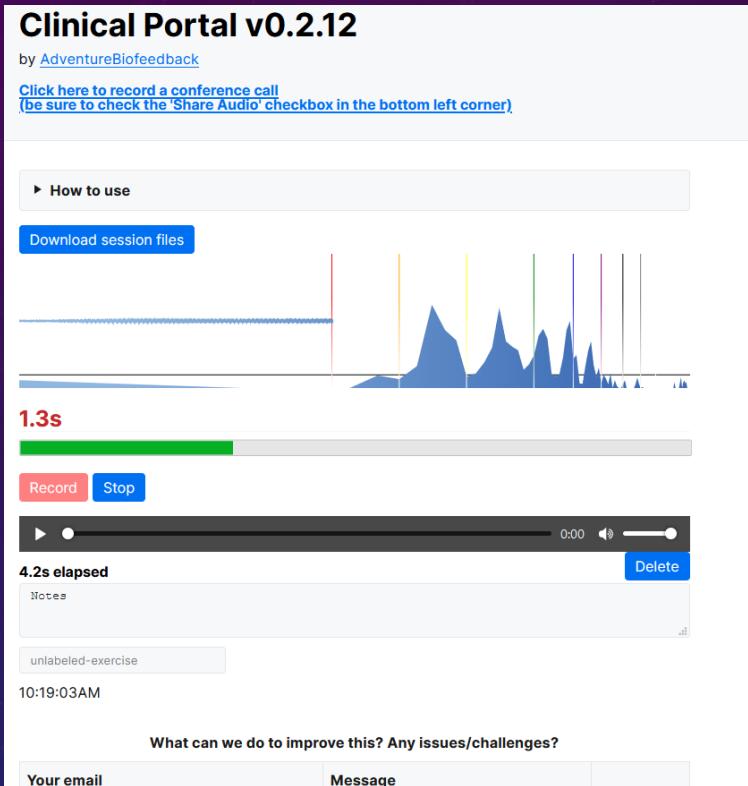
**Front End:** the normal code for your website. Software: **JavaScript, CSS, HTML**

**Back End:** if your website needs to communicate with a cloud server or database, do new fancy functions, etc.

Software: **Java, PHP, Node.js, Python**

# Example of a web app (old beta test from adventurebiofeedback.com):

<https://www.adventurebiofeedback.com/slp/>



Fully Front End.

html, CSS, and JavaScript were the primary languages.

We ran into issues going across platforms and browsers, especially trying to collect data from the computer's speaker (the other person on a zoom call) rather than the microphone.

We'll eventually have to hire developers, and do most of our real data processing using an AWS cloud server (Back End) running python scripts and sending data back to the web page.

What can a pro developer do vs. someone who knows a bit of programming?  
Example  
<https://musiclab.chromeexperiments.com/Spectrogram/>

# WEB APPS- WHERE TO START

- Might be best to start with **WordPress** or **Wix** to get a foot in the door.
  - Then you can add little gadgets to your website (e.g. Java Applets or IFTTT (if this then that) to start turning a text+ picture website into a kind of user interface.
- Along the way you'll learn some internet jargon too.... Like how a web client (e.g. a browser) communicates via an FTP or HTTP protocol with a web server that hosts your HTML code and other website resources, how an extra step might be added if you want to use another server for data storage or analysis, etc.
- Overview of what modern web development entails:  
<https://www.youtube.com/watch?v=VfGW0Qiy2I0>

So, if web apps are a bit complicated.... are you stuck with mockups that don't do anything? NO!

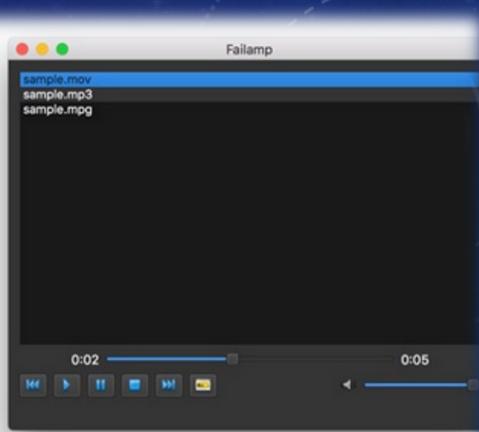
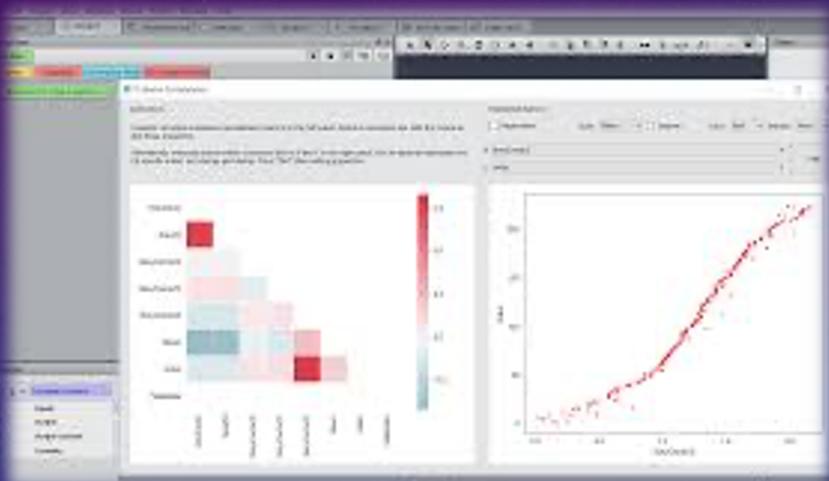
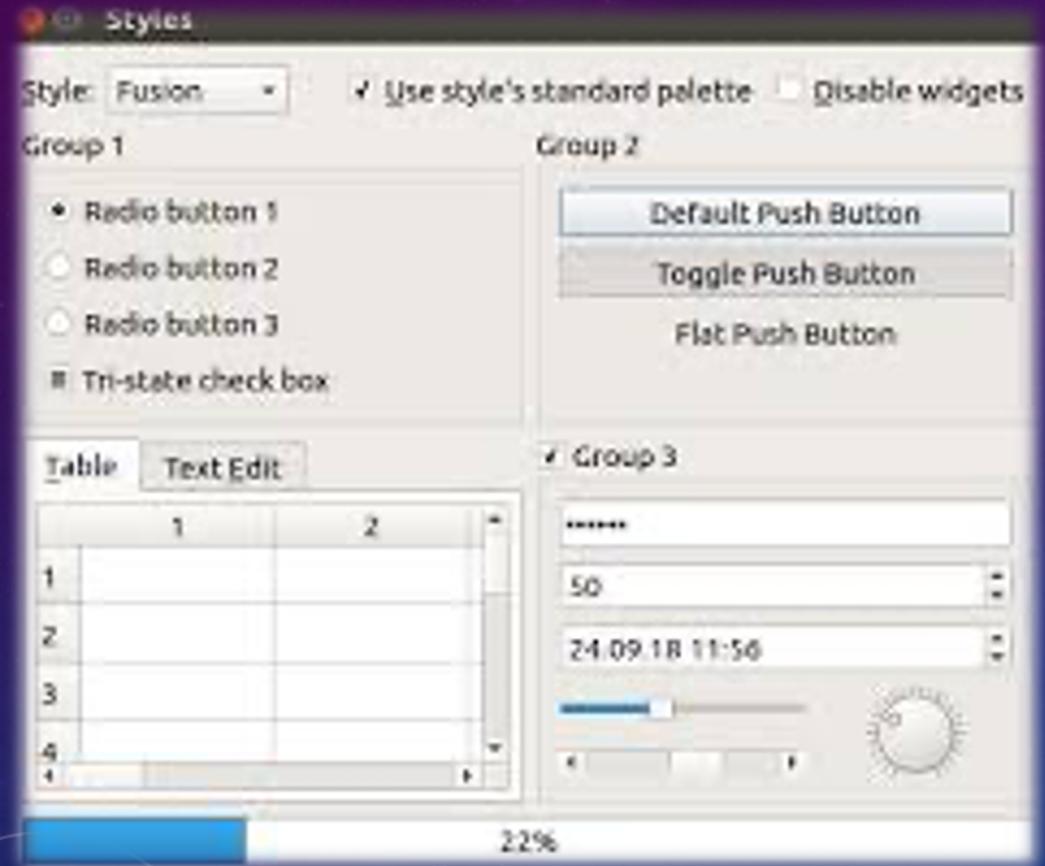
Development of desktop apps (which also work on mobile devices that use windows or android) use general purpose programming languages

For hacking/innovation, it may be better to start by building something, then if it seems really good, hire a developer who can put it in the App store

Also, .... computers haven't been replaced by phones yet... people still use computer-based programs even if the word "App" seems to imply iPhone lately.

# Python can be used to make apps.

# Many are able to work on all operating systems



PyQt5

Kivy

Tkinter

PySide

PySimpleGUI

BeeWare

Flask

Flexx

Django

Python can create stand alone applications with graphical interfaces  
Combined with standard python programming, this allows you to be creative  
and develop your own applications. On the left are some possible tools for  
developing GUIs and apps.

**PyQt5** is the largest, most well documented, with the easiest learning curve  
It is mainly for Windows/Linux/Mac, but is increasingly Android friendly

It also has a “designer” interface within Anaconda so that you only need to  
code the functions that run when you push buttons, rather than coding the  
visuals.

\*Some occasional bugs appear when using Mac. These are typically solvable  
with some wrestling and trial & error, but if you have a choice, use a PC!

# PYQT5 DESIGNER + DEVELOPMENT IN PYTHON (SPYDER)

PyQt5 is a library (just a set of functions) that you can import into your favorite IDE (e.g. Spyder) at the start of a program.

Using these functions, you can get pop-up windows, buttons, text boxes, etc. and interacting with these “widgets” can trigger other python functions to run (e.g. printing something, doing a computation, etc.)

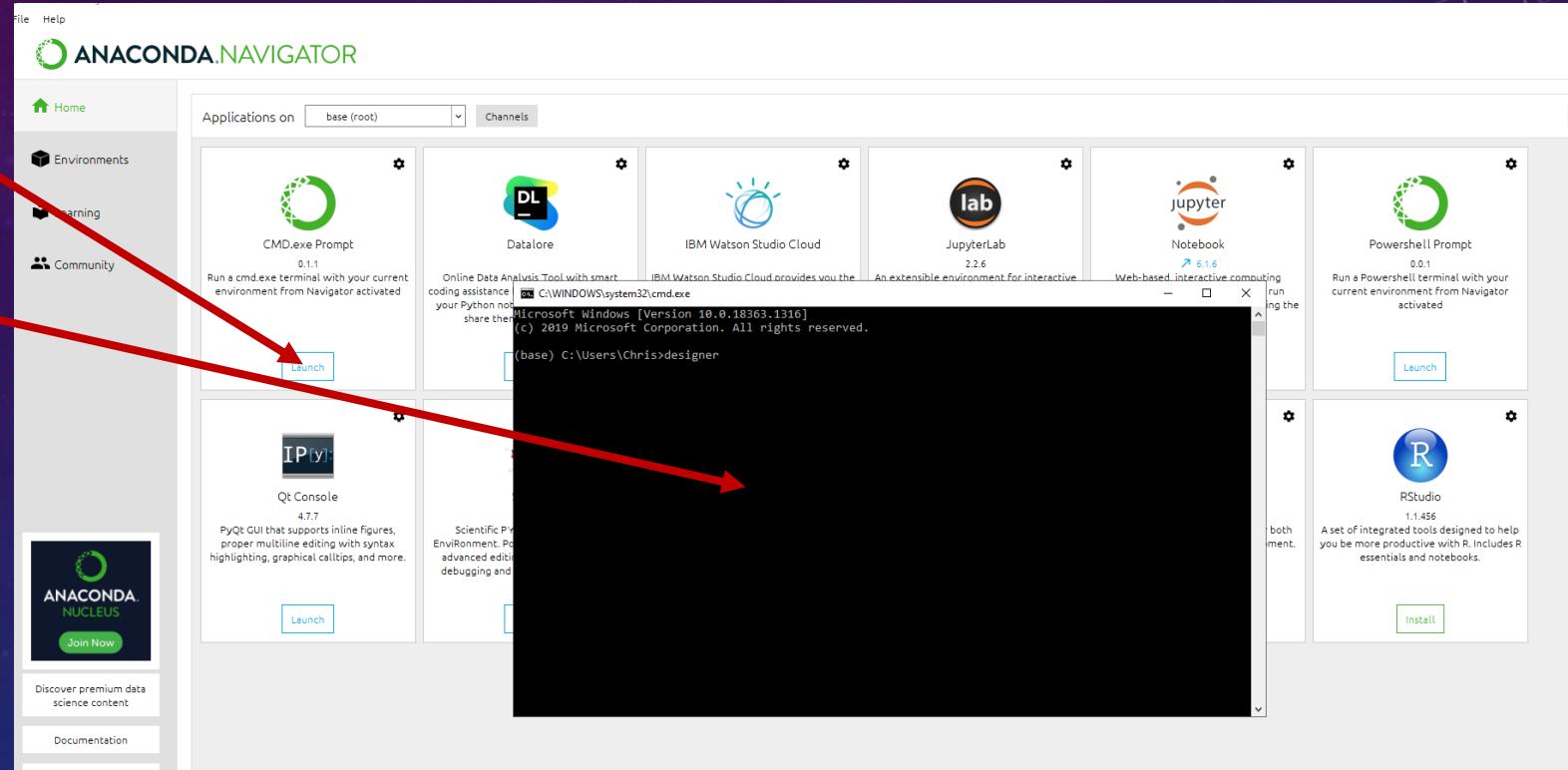
Many tutorials exist, for example a calculator made from absolute scratch is here:  
<https://realpython.com/python-pyqt-gui-calculator/>

# NOW WE'RE GOING TO MAKE A GRAPHICAL USER INTERFACE IN PYTHON

- Follow along/take notes... The steps shown in the recording are summarized in the following slides.
- You should make the program described herein for yourself so that it runs prior to class on Thursday.

# THE DESIGNER: MAKES THE UI MOSTLY CODE-LESS

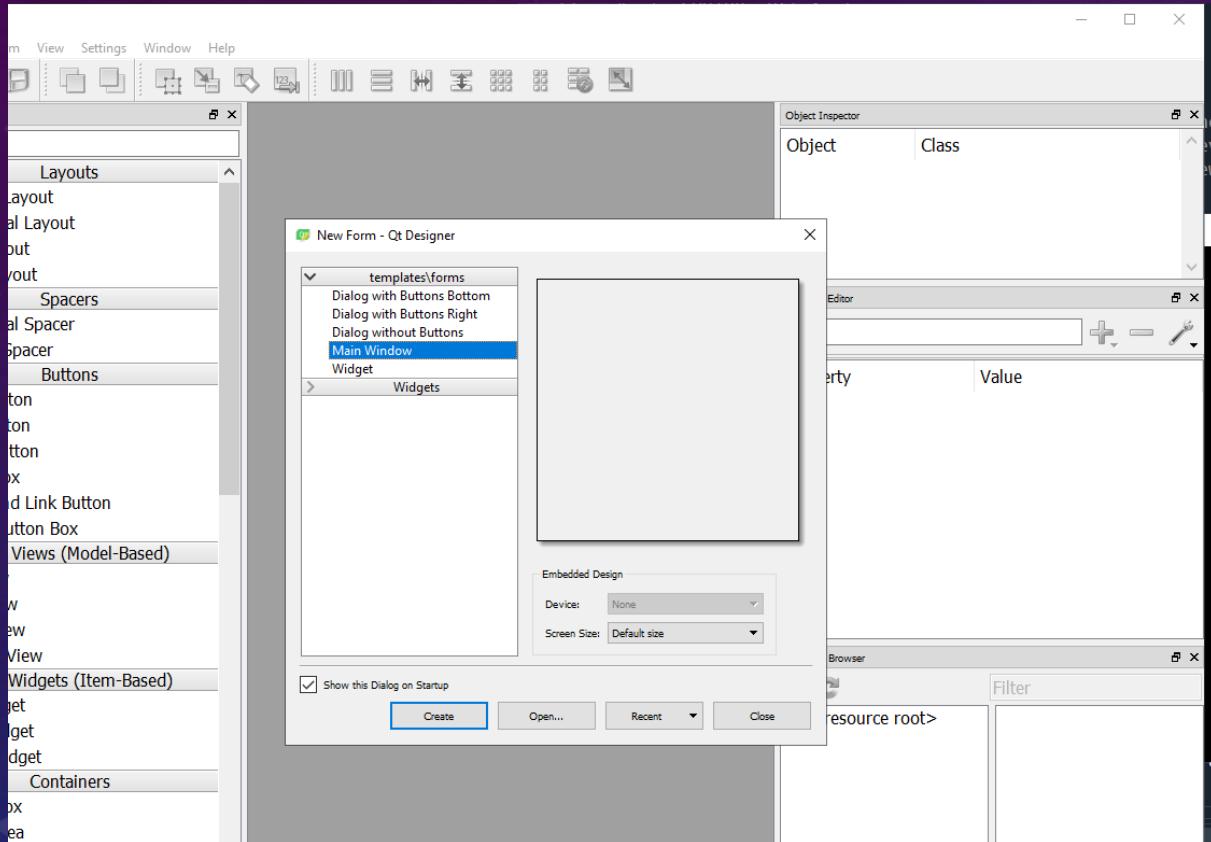
- Go to Anaconda Navigator
- Launch the prompt
- Type “designer” in the command line (no quotes)
- Hit enter



For Mac:

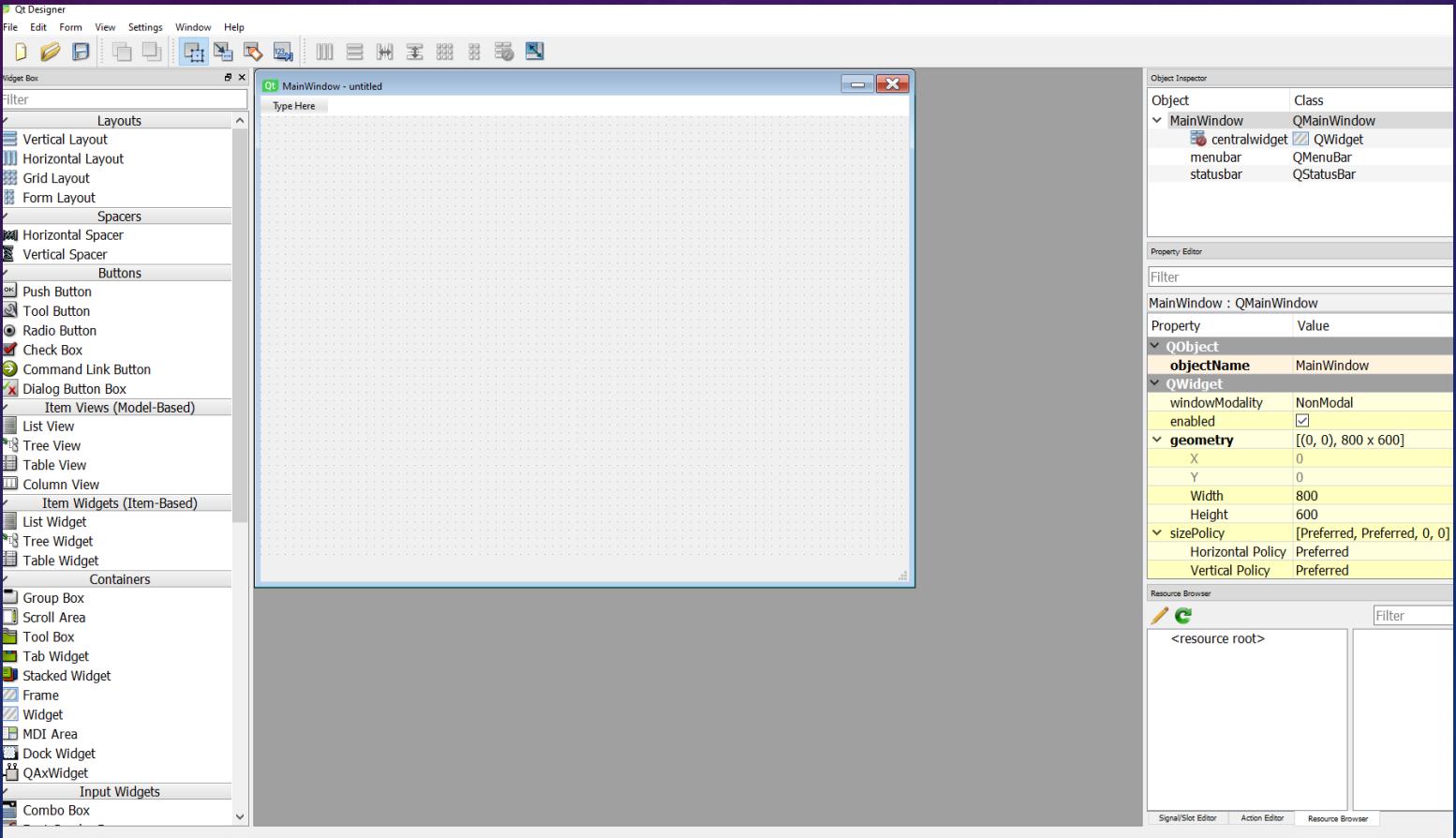
Open Terminal (the regular Mac Terminal. From Launchpad, type Terminal, you should see something like the terminal above, but a Mac version): now type  
“open –a Designer” in the command line, with no quotes. Hit enter.

# QUICK TOUR OF DESIGNER



The “main Window” should be highlighted  
Click “Create”

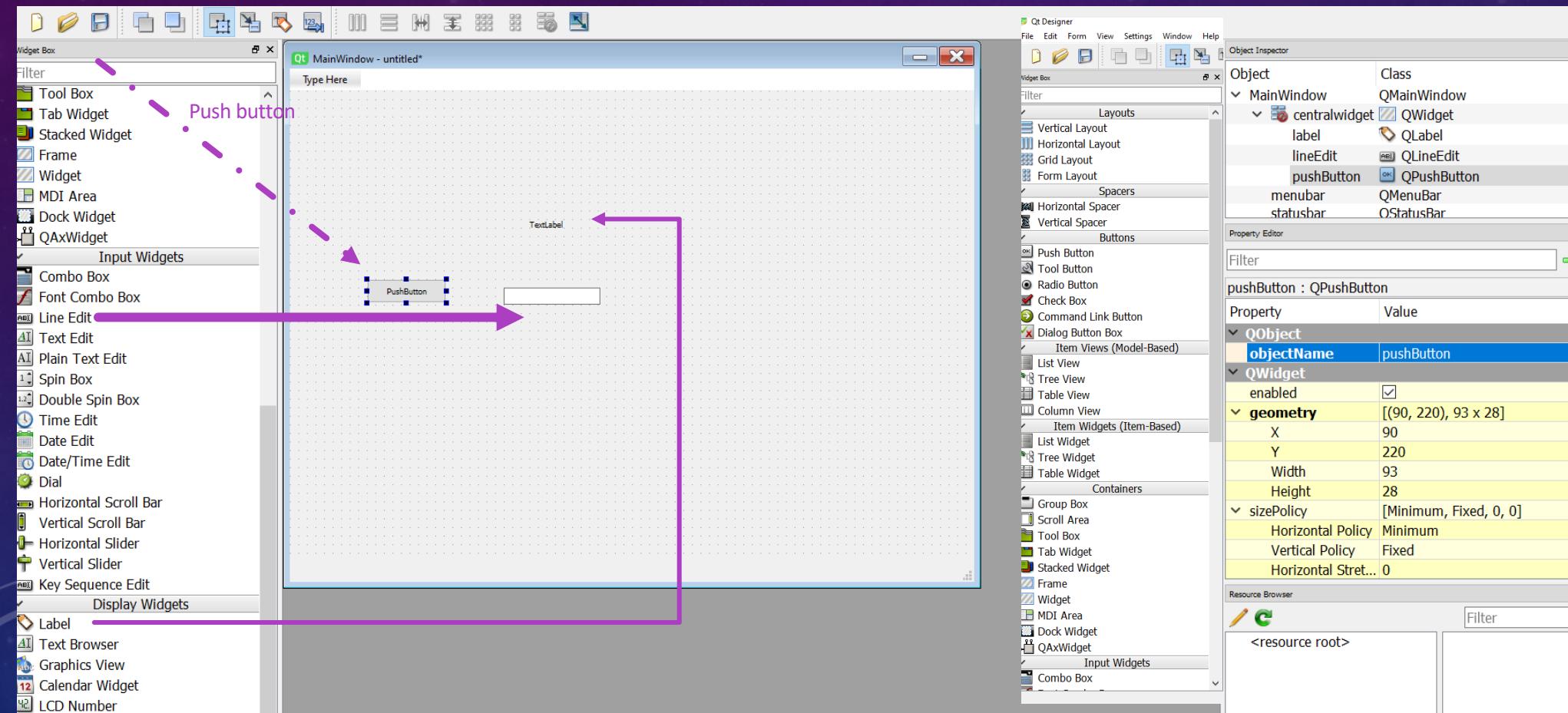
**Left:**  
**Widgets**  
To drag and drop  
Onto the main window



**Right:** properties of  
Anything in the main  
Window that is highlighted

# ADDING WIDGETS

Drag a “PushButton”, a “Line Edit”, and a “Label” from the left into the main window



Note,  
When you select the  
PushButton, you can  
look over on the right  
where the yellow is,  
and you will see  
“objectName” is  
“pushButton”.

# MANIPULATING WIDGET PROPERTIES

- 1) Resize the widgets so they are big enough to see, physically drag their corners like any other window
- 2) Change font and text size by scrolling down under where “objectName” was.. You’ll see a “font” option. Click in the cell to the right of “font” to make it readable
- 3) Do that for all 3 widgets.
- 4) Change the “TextLabel” widget’s text by selecting it, scrolling down beyond “font” to where it says “Text” in bold. Change the text to “count button presses”
- 5) Select the empty text box and give it a starting value of 0. Scroll down to where it says “text” and add a 0 to the “value” column next to it.
- 6) Change the “objectName” (first property) as follows
  - 1) PushButton -> add\_one
  - 2) lineEdit -> current\_count
- 7) Change the PushButton’s text to say “counter”

Qt MainWindow - untitled\*

Type Here

Count button presses

counter

0

inputMethodHints ImhNone

QAbstractButton

text counter

icon

iconSize 20 x 20

shortcut

checkable

Resource Browser

current\_count : QLineEdit

Property	Value
QObject	
objectName	current_count
QWidget	
enabled	<input checked="" type="checkbox"/>
geometry	[(320, 170), 111 x 71]
X	320
Y	170
Width	111
Height	71

Property

Value

QObject

objectName add\_one

QWidget

enabled

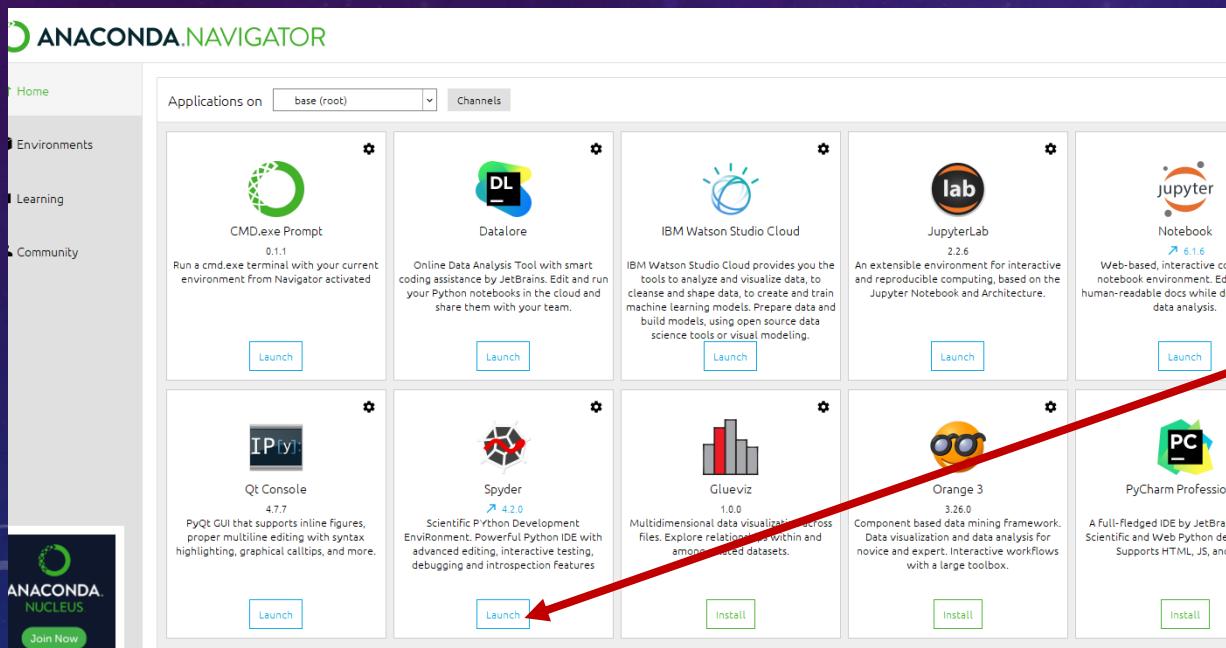
geometry [(70, 130), 161 x 121]

X 70

Y 130

FILE → SAVE AS → DemoGui Into your OT699 folder

It doesn't have to be OT699 folder necessarily, just the same folder from which you will run your spyder programs.



After saving, launch  
spyder from anaconda  
navigator

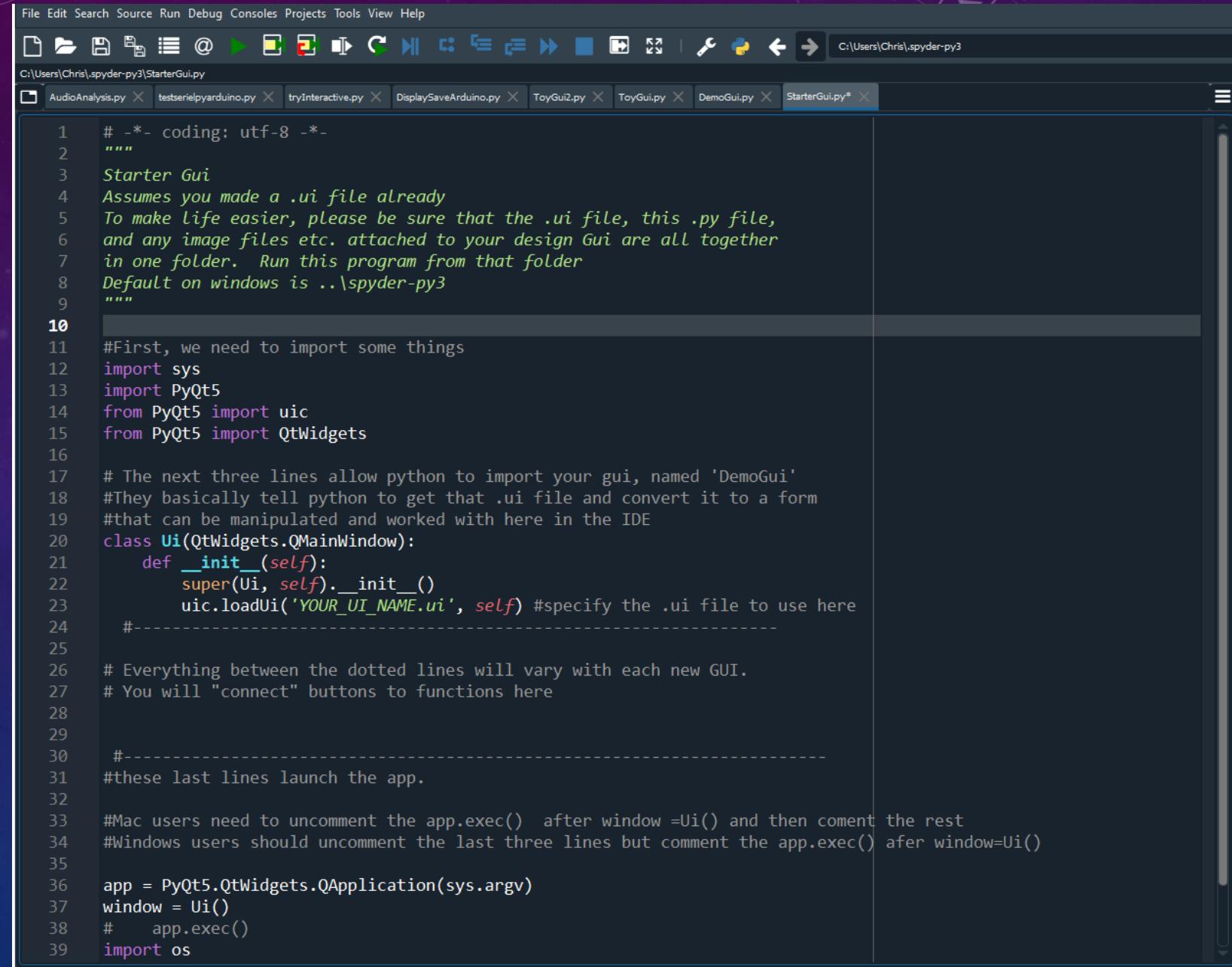
# START THE CODE

This code to the right is the start of your GUI.

Other than the name of the UI file, which is labeled 'your UI\_NAME.ui' as a placeholder, this .py starter code will be the same for any GUI you want to make.

You will have this file to work with.  
Please note the Mac-specific instructions at the bottom.

You will fill in the middle yourselves, following the instructions in the lecture.  
A written synopsis is also provided in the next slides.



The screenshot shows the Spyder Python IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The top status bar shows the path C:\Users\Chris\spyder-py3\StarterGui.py and the version C:\Users\Chris\spyder-py3. The code editor displays the 'StarterGui.py' file with the following content:

```
1 # -*- coding: utf-8 -*-
"""
3 Starter Gui
4 Assumes you made a .ui file already
5 To make life easier, please be sure that the .ui file, this .py file,
6 and any image files etc. attached to your design Gui are all together
7 in one folder. Run this program from that folder
8 Default on windows is ..\spyder-py3
"""

10 #First, we need to import some things
11 import sys
12 import PyQt5
13 from PyQt5 import uic
14 from PyQt5 import QtWidgets
15
16 # The next three lines allow python to import your gui, named 'DemoGui'
17 #They basically tell python to get that .ui file and convert it to a form
18 #that can be manipulated and worked with here in the IDE
19 class Ui(QtWidgets.QMainWindow):
20     def __init__(self):
21         super(Ui, self).__init__()
22         uic.loadUi('YOUR_UI_NAME.ui', self) #specify the .ui file to use here
23         #-----
24
25         # Everything between the dotted lines will vary with each new GUI.
26         # You will "connect" buttons to functions here
27
28         #-----
29
30         #these last lines launch the app.
31
32         #Mac users need to uncomment the app.exec() after window =Ui() and then coment the rest
33         #Windows users should uncomment the last three lines but comment the app.exec() afer window=Ui()
34
35         app = PyQt5.QtWidgets.QApplication(sys.argv)
36         window = Ui()
37         # app.exec()
38         import os
```

# WRITTEN STEPS: THE MAIN PROGRAM LOGIC:

*First make sure that you have a .ui file named ‘DemoGui’ in the same folder as this template .py file you are using. Note that if you click the “open” icon in the top left of spyder, you may have to select “all files” instead of “supported text files” to confirm that the .ui file is there.*

*In the blank part of the code, you will connect buttons to functions. Write:*

```
self.add_one.clicked.connect(self.counter_function)
```

*And since that’s the only button, we can tell the GUI to show itself. Write:*

```
self.show()
```

*The next step is to define the function specified above, called “counter\_function”. Write:*

```
Def counter_function(self)
```

*If the objectName of your lineEdit text box is called “current\_count”, then to extract that you could write:*

```
self.current_count.text()
```

*But, writing that command gives you text, you want a number to do math with, so you can convert this text to whole number using the int() command like this: int(self.current\_count.text())*

*Now you can create a variable that is always the number in the counter text box +1. write:*

```
newcount= Int(self.current_count.text())+1
```

*And then you can make the variable newcount be the new text in the counter box. Write:*

```
self.current_count.setText(str(newcount))
```

Note that newcount was converted back to a string using str() first before putting it in the text box

# COMPLETE THE CODE

To to File, Save As, and save this as DemoGui

This will put a .py file called DemoGui in your OT699 window.

Confirm now that the .py file is in the same folder as the .ui file we created earlier.

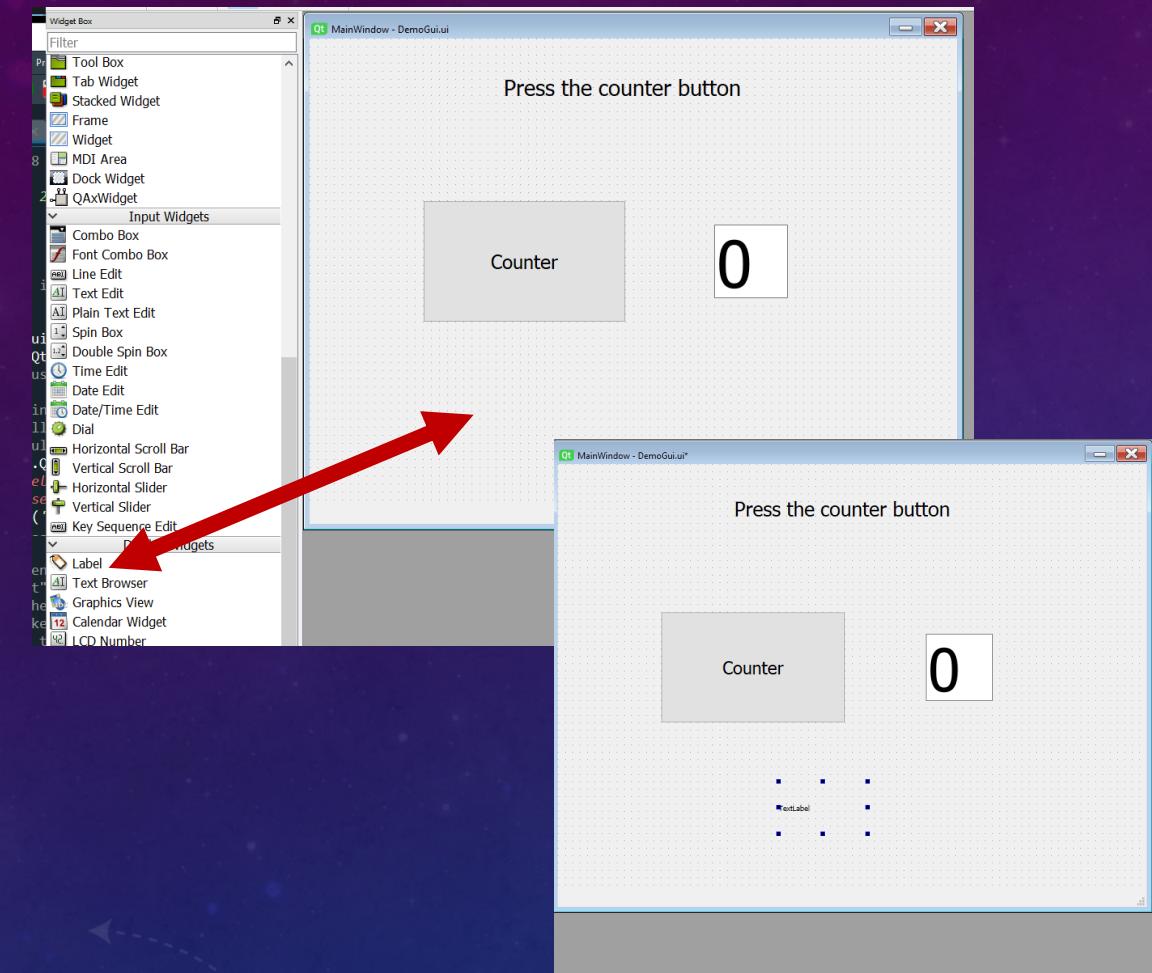
Now try running this code by clicking the green triangle.

The screenshot shows the Spyder IDE interface. On the left, the code editor displays a Python script named `DemoGui.py`. A red arrow points from the top of the code editor towards the toolbar. The toolbar contains several icons: a play button, a stop button, a step forward button, a step backward button, and a refresh button. Another red arrow points from the bottom of the toolbar towards the right side of the screen. On the right, a graphical user interface window titled "MainWindow" is displayed. The window has a title bar with tabs for "ui.py", "DisplaySaveArduino.py", "ToyGui2.py", "FirstGui.py", and "DemoGui.py". Below the title bar is a toolbar with icons for file operations like save, open, and search. The main area of the window contains a label "counter" next to a text input field containing the value "0". Above the counter, the text "Count button presses" is displayed. At the bottom of the window, there is a status bar showing the path "C:/Users/Chris/.spyder" and some other information. The overall background of the image is dark blue with abstract circular patterns.

```
ui.py X DisplaySaveArduino.py X ToyGui2.py X FirstGui.py X DemoGui.py X
MainWindow
a pri
ython
get t
orked
:
, sel
iine
o fun
' and
subpar
butto
name,
as de
connect
nection
button
within
t is i
an int
current_
current_count again, as text (aka a 'string' or str)
text(str(newcount))
ue and the current time (using the time.time() function)
```

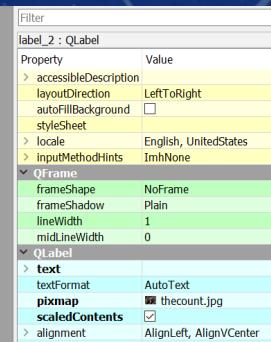
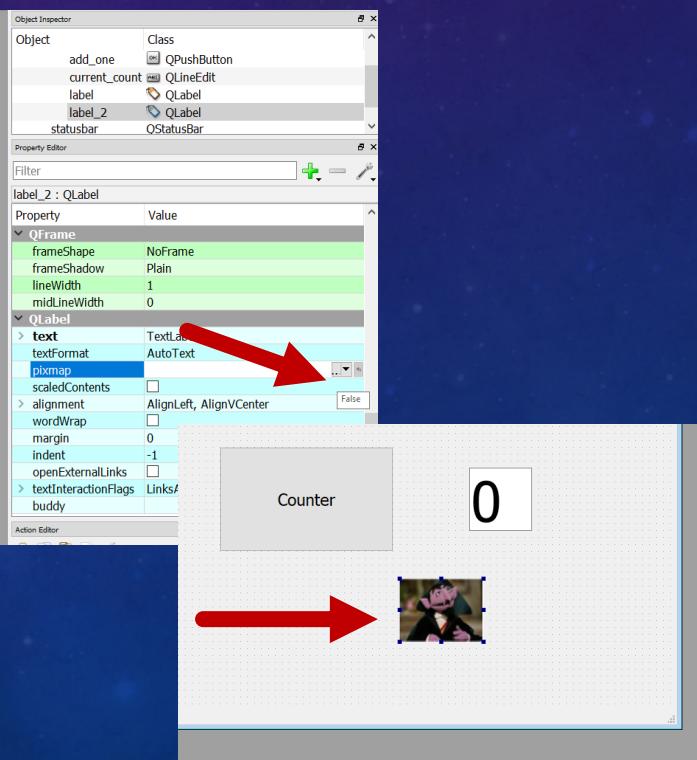
# NOW WE CAN SOME VISUALS

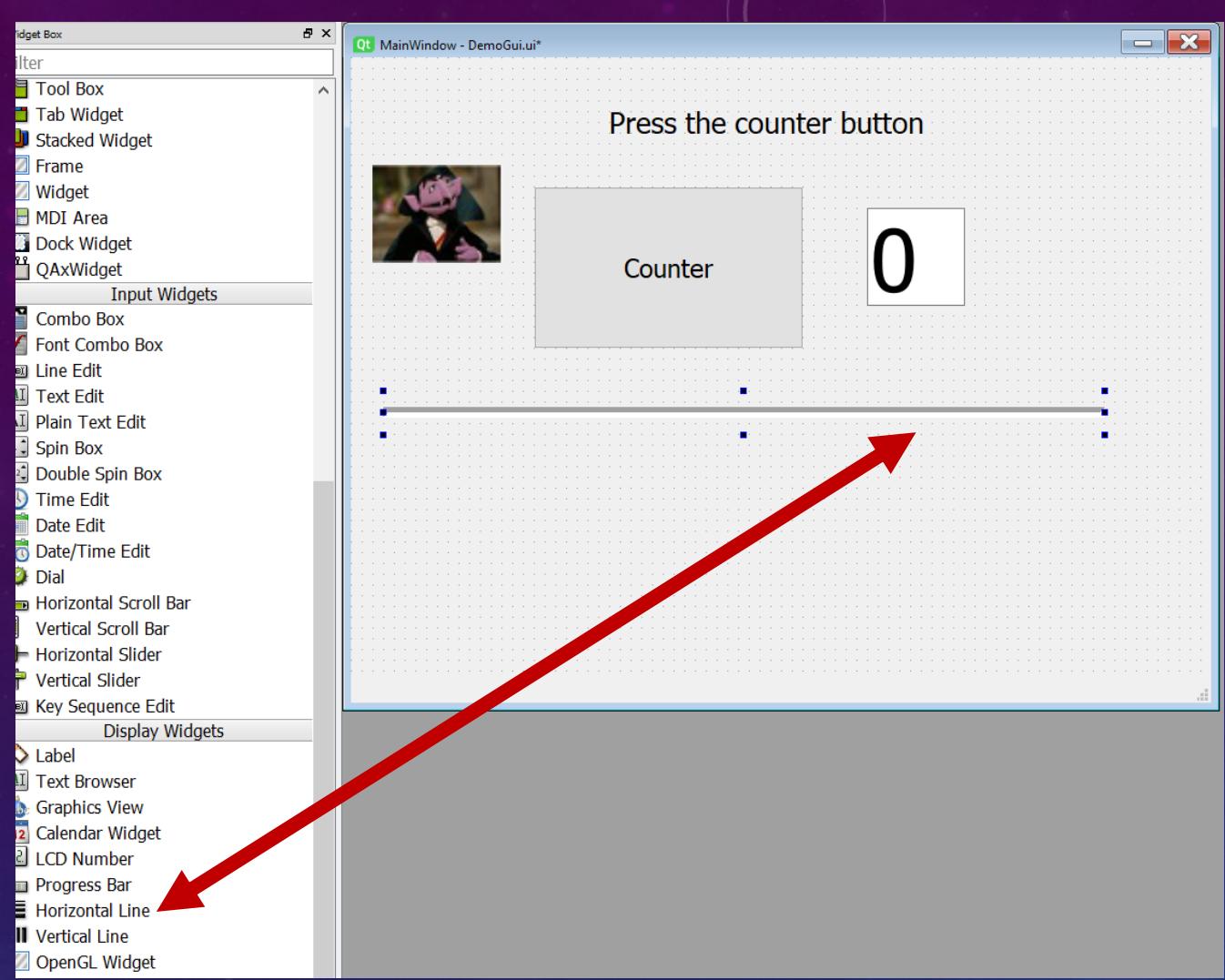
You can add images to your GUI!  
It won't even change the code.



Drag a “Label” onto the main window

Then go to “pixmap” under QLabel  
There is a down arrow on the right of the empty text  
box that lets you select a file  
Just make sure that the image is stored in the same  
folder as the .ui file before running the .py code





Horizontal/Vertical lines

Can be added from the “Display” widgets

And “linewidth” can be adjusted from the property editor

# THAT'S AN APP

The more python you know, the more you can do when various buttons are pushed.

Now just a few notes on moving forward:



Categories ▾

[Library](#) [Videos](#) [Q/A](#) [eBooks](#)

ENH



## LEARN PYQT programming language

### PyQt Tutorial

- [PyQt - Home](#)
- [PyQt - Introduction](#)
- [PyQt - Hello World](#)
- [PyQt - Major Classes](#)
- [PyQt - Using Qt Designer](#)
- [PyQt - Signals and Slots](#)
- [PyQt - Layout Management](#)
- [PyQt - Basic Widgets](#)
- [PyQt - QDialog Class](#)
- [PyQt - QMessageBox](#)
- [PyQt - Multiple Document Interface](#)
- [PyQt - Drag and Drop](#)
- [PyQt - Database Handling](#)
- [PyQt - Drawing API](#)
- [PyQt - BrushStyle Constants](#)
- [PyQt - QClipboard](#)

## PyQt - Basic Widgets

Advertisements

[Previous Page](#)[Next Page](#)

Here is the list of Widgets which we will discuss one by one in this chapter.

Sr.No	Widgets & Description
1	<a href="#">QLabel</a> A QLabel object acts as a placeholder to display non-editable text or image, or a movie of animated GIF. It can also be used as a mnemonic key for other widgets.
2	<a href="#">QLineEdit</a> QLineEdit object is the most commonly used input field. It provides a box in which one line of text can be entered. In order to enter multi-line text, QTextEdit object is required.
3	<a href="#">QPushButton</a> In PyQt API, the QPushButton class object presents a button which when clicked can be programmed to invoke a certain function.
4	<a href="#">QRadioButton</a> A QRadioButton class object presents a selectable button with a text label. The user can select one of many options presented on the form. This class is derived from QAbstractButton class.

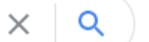
Keep in mind when using the internet:

There isn't one way to create a GUI. Sources of information can sometimes present code that solves the same problem in different ways. That's normal, you may need to try a few alternatives.

The StarterGui I've provided should be a quick way to learn about PyQt5 and how to make interactive graphical programs, but many tutorials start from scratch without using designer, so there will be some differences in how the windows are labeled.

To learn more about how to set up and interact with different widgets, the tutorial on the left should be very helpful. This is where to find commands like "setText()"

pyqt5 documentation pdf



All Images Videos News Shopping More

Settings Tools

About 255,000 results (0.44 seconds)

[www.tutorialspoint.com › pyqt › pyqt\\_tutorial](http://www.tutorialspoint.com/pyqt/pyqt_tutorial.pdf) ▾ PDF

## Download PyQt Tutorial (PDF Version) - TutorialsPoint

PyQt is a blend of Python programming language and the Qt library. This introductory tutorial will assist you in creating ... MULTIPLE DOCUMENT INTERFACE .

[readthedocs.org › projects › downloads › pdf › latest](http://readthedocs.org/projects/downloads/pdf/latest.pdf) ▾ PDF

## Python Qt tutorial Documentation - Read the Docs

Jun 11, 2018 — Python libraries to use Qt from Python (PyQt and PySide), but rather than picking one of these, this tutorial makes use of the QtPy package ...

[doc.bccnsoft.com › docs › PyQt5](http://doc.bccnsoft.com/docs/PyQt5) ▾

## PyQt5 Reference Guide — PyQt 5.7 Reference Guide

PyQt5 Reference Guide · Introduction License · Platform Specific Issues OS X · Deprecated Features and Behaviours · Incompatibilities with Earlier Versions PyQt ...  
Introduction · Installing PyQt5 · PyQt5 Class Reference · Python Module Index

[doc.bccnsoft.com › docs › PyQt5 › introduction](http://doc.bccnsoft.com/docs/PyQt5/introduction) ▾

## Introduction — PyQt 5.7 Reference Guide

PyQt5 is a set of Python bindings for v5 of the Qt application framework from The Qt Company. ... version, current development previews, and the latest version of this documentation. ... It also enables the generation of PostScript and PDF files.

[codeby.net › attachments › pyqt5tutorial-pdf](http://codeby.net/attachments/pyqt5tutorial-pdf.pdf) ▾ PDF

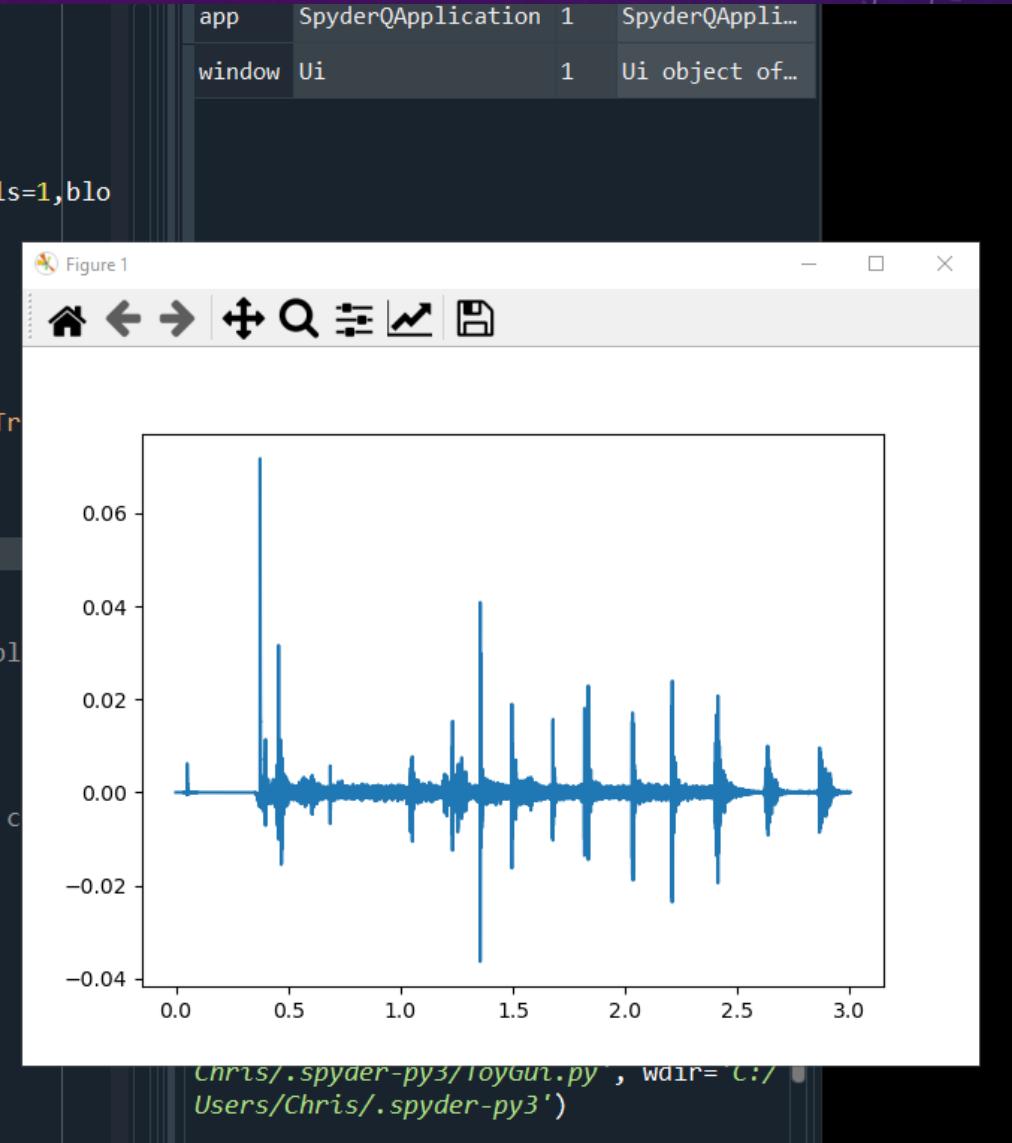
## PyQt5 Tutorial Documentation - Codeby.net

Mar 1, 2016 — PyQt5 Tutorial Documentation, Release 1.0. 3.3 Example. Below is an example of a Window: #!/usr/bin/env python3 from PyQt5.QtCore import \*

Lots of resources for documentation

# EXAMPLE NEXT STEPS: THE POSSIBILITIES ARE ENDLESS

```
RecordPressed(self):  
    # This is executed when the button is pressed  
    self.Record.setChecked(True)  
    # self.PlayBack.setChecked(False)  
    S=float(self.duration.text()) #duration was also an object name  
    self.myrecord=S  
    self.Record.  
  
    PlayBackPresses(self):  
        # This is executed when the button is pressed  
        self.PlayBack.setChecked(True)  
        # self.show()  
        sd.playrec(self.myrecord, S)  
        self.PlayBack.  
  
    Plot_wavePresses(self):  
        # This is executed when the button is pressed  
        S=float(self.duration.text())  
        timepoints=round(S*44100)  
        figWindow=plt.figure()  
        figAxes=figWindow.add_subplot(111)  
        figData,=figAxes.plot([0]*timepoints,'k')  
        # plt.plot(ti  
        # figData can be updated with figData.set_ydata(newdata) rather than c  
  
        me__ == "__main__":  
            app = QtWidgets.QApplication(sys.argv)  
            ui = Ui()  
            os  
            not os.environ.get('SPY_UMR_ENABLED'):ui.exec_()
```



GOOD LUCK!

