

# 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?

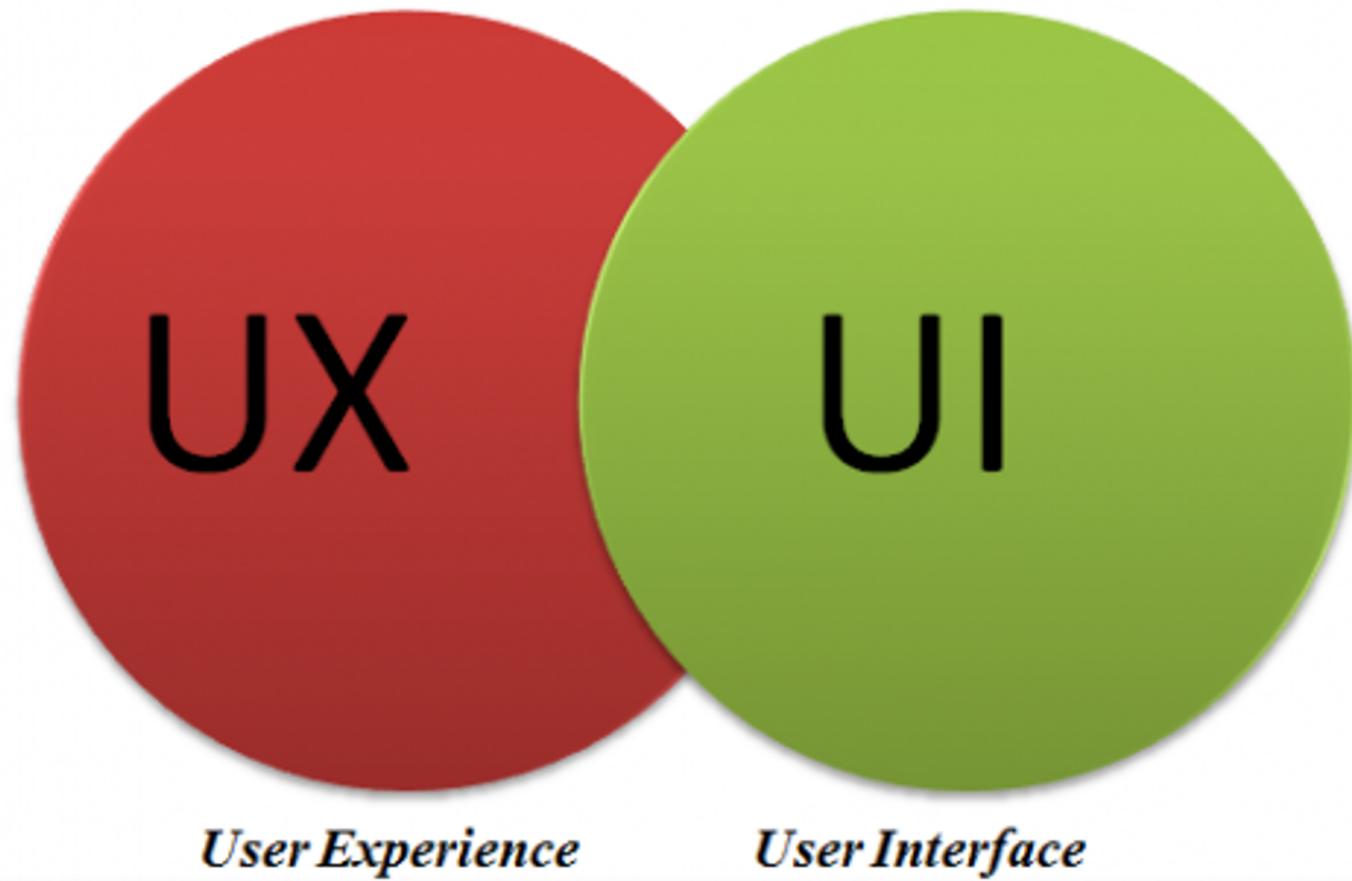
Command line code that you write yourself to do some math on data you collect

	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**

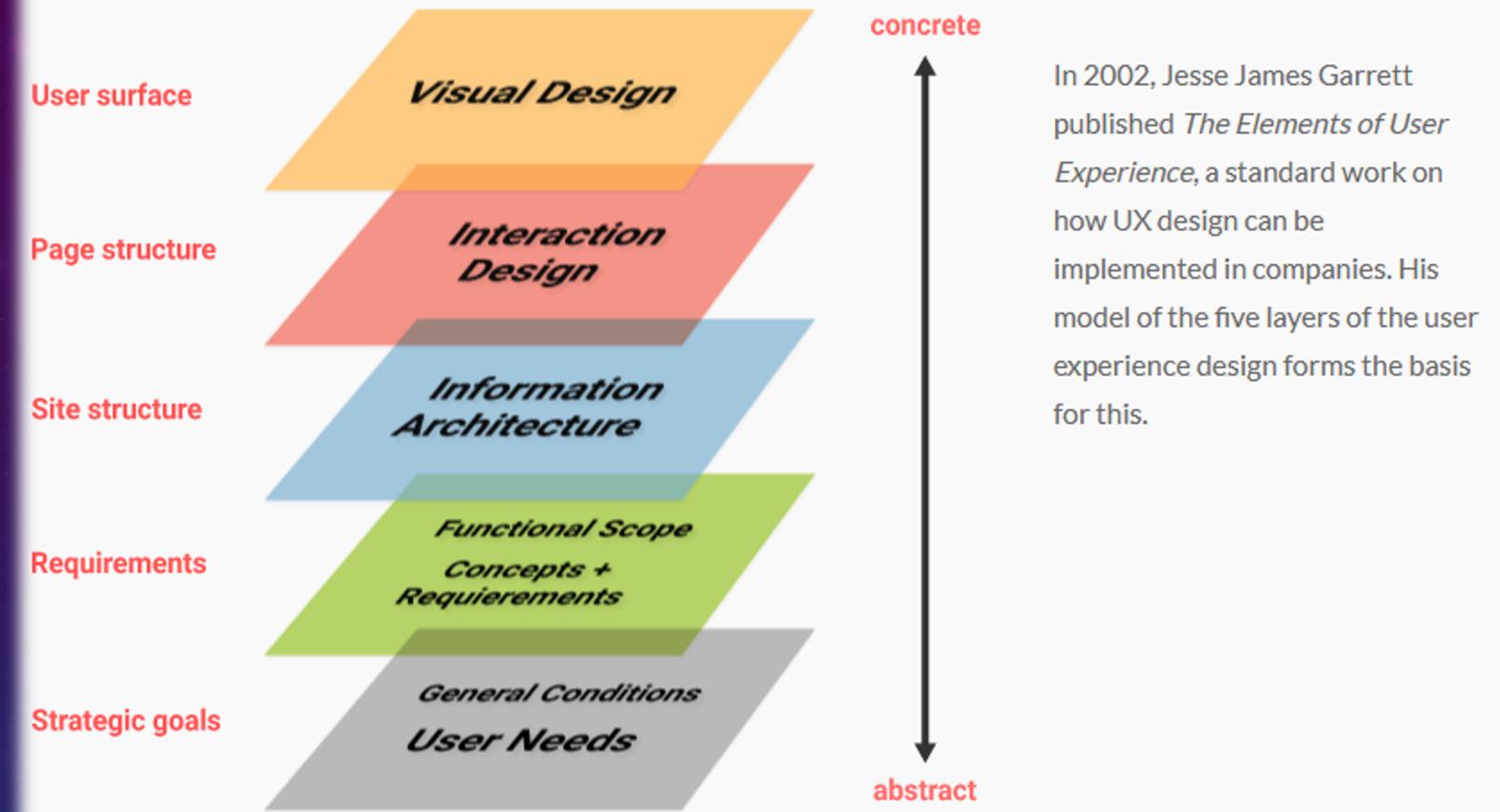
**Personas**

**Layout**

**Visual Design**

**Branding**

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

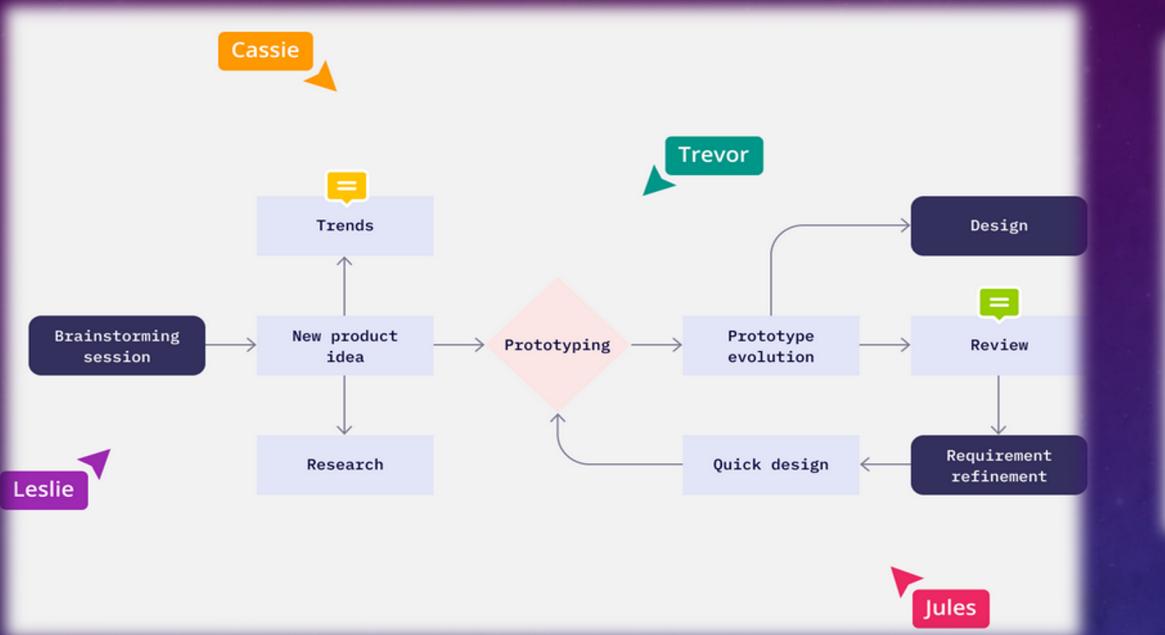


In 2002, Jesse James Garrett published *The Elements of User Experience*, a standard work on how UX design can be implemented in companies. His model of the five layers of the user experience design forms the basis for this.

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 or miro (e.g. <https://miro.com/>) are valuable for team efforts PPT is still used.



**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, Webflow, Sketch

## 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**

# 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, how your HTML and other website resources are hosted, if you have a static or dynamic website, 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 in PPT that don't do anything? NO!

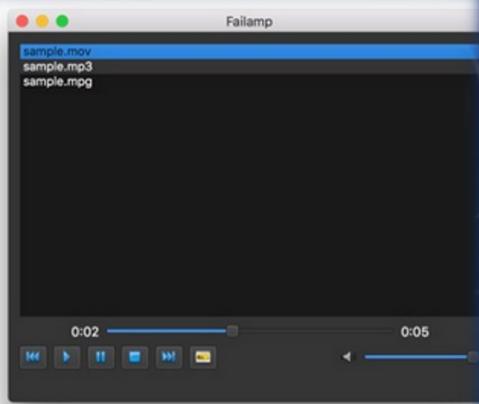
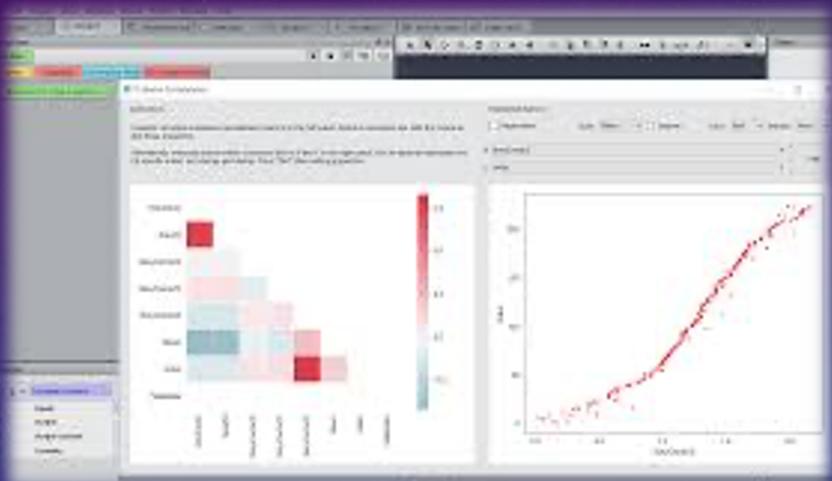
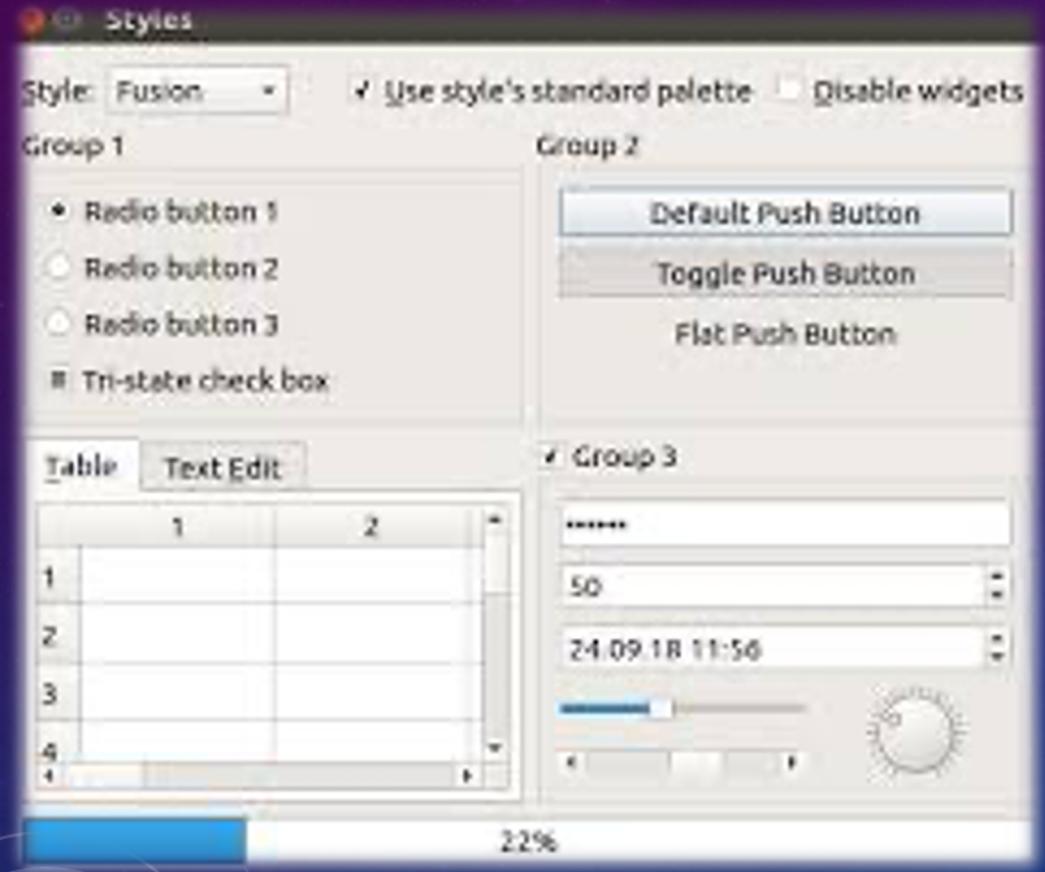
Development of desktop apps (which can also work on mobile devices that use windows or android) use general purpose programming languages

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)

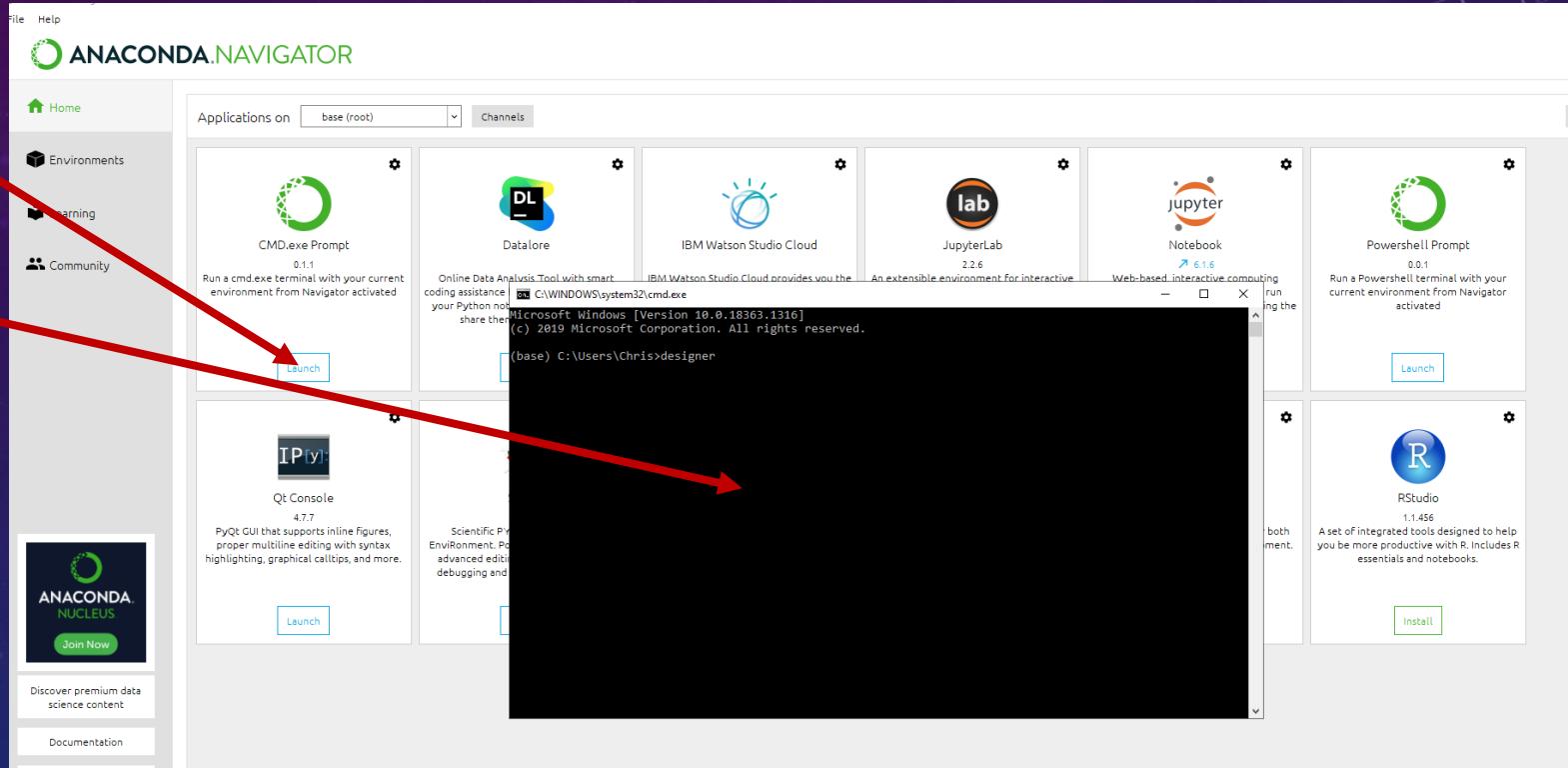
PyQt5 is a library (just a set of functions) that you can import into your favorite IDE (e.g. Spyder) at the start of a program.

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.)

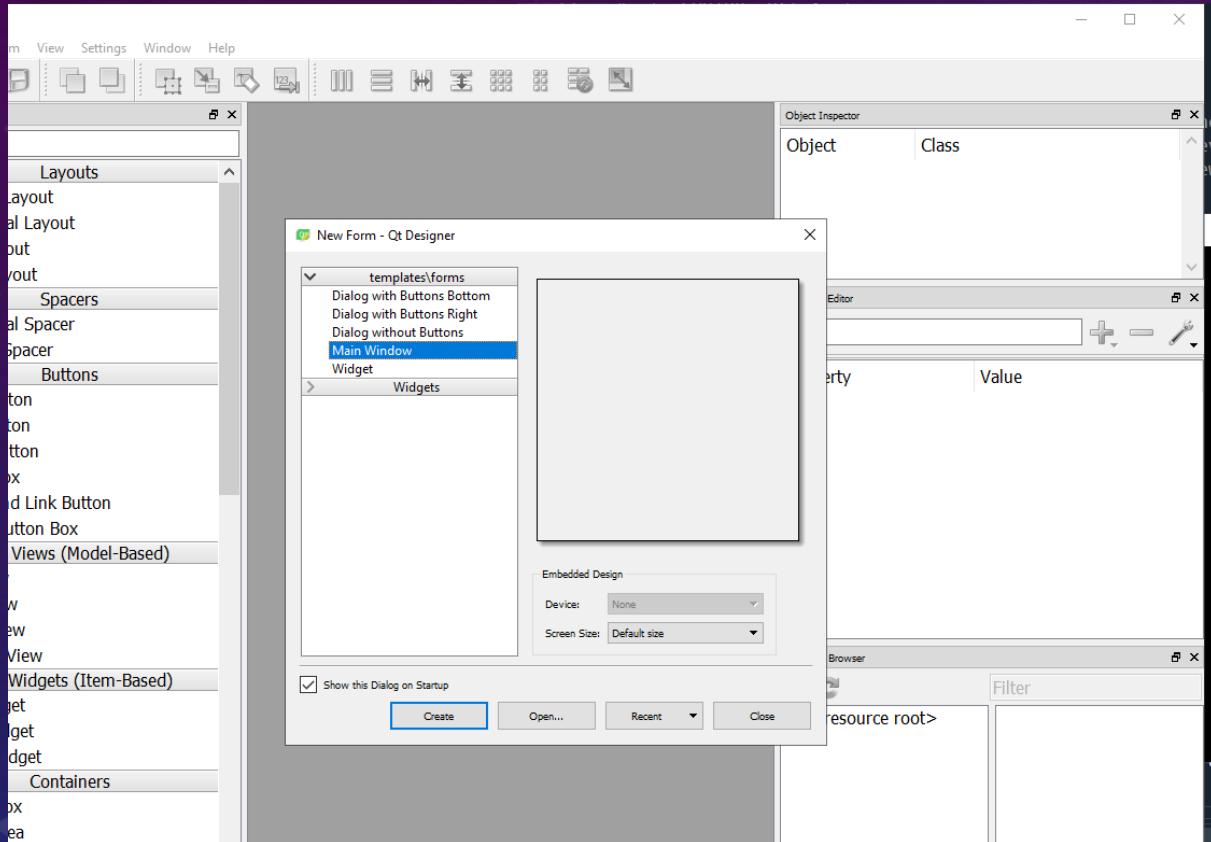
Many tutorials exist, for example a calculator made from absolute scratch is here:  
<https://realpython.com/python-pyqt-gui-calculator/>

# 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

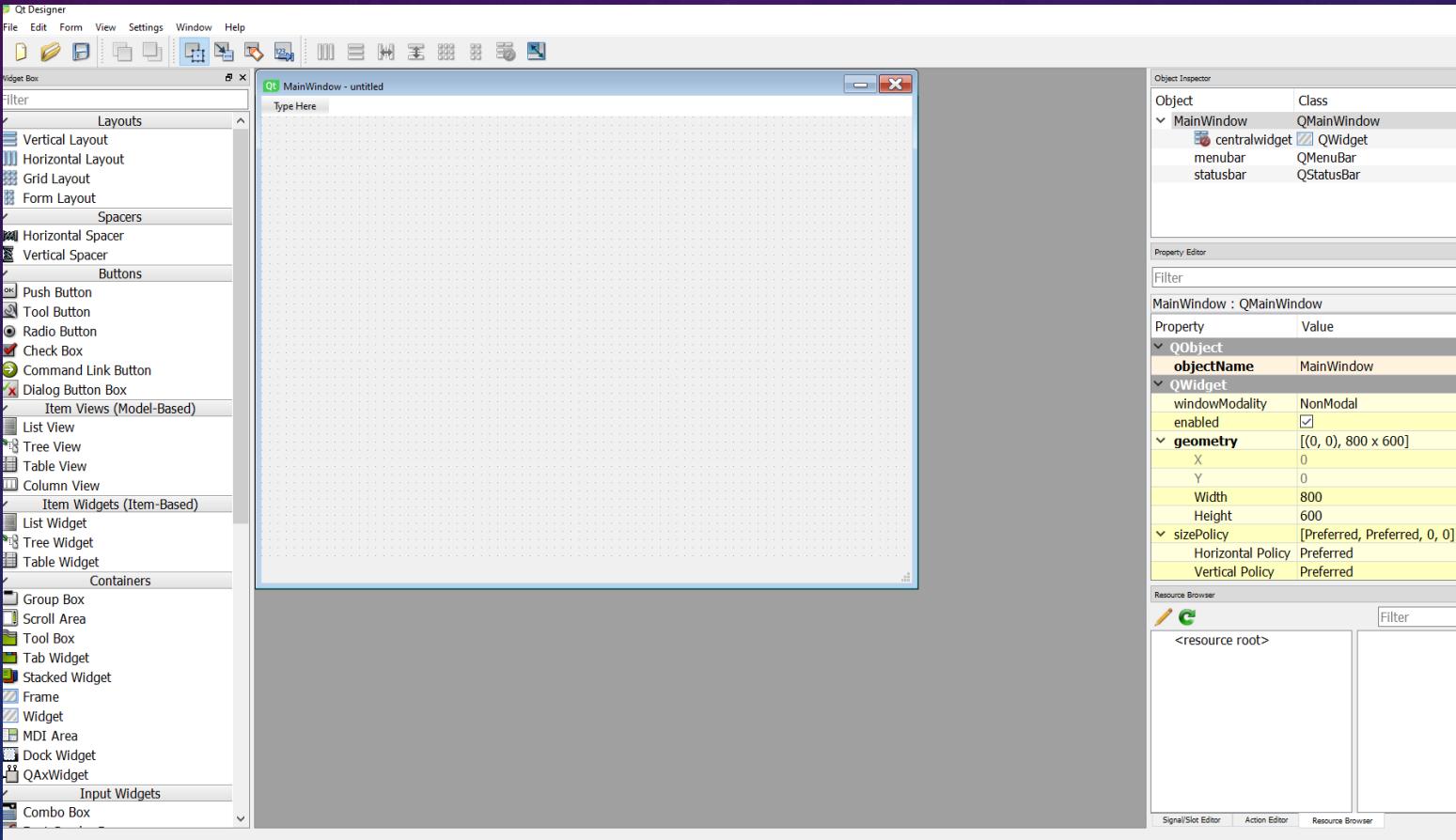


# QUICK TOUR OF DESIGNER



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

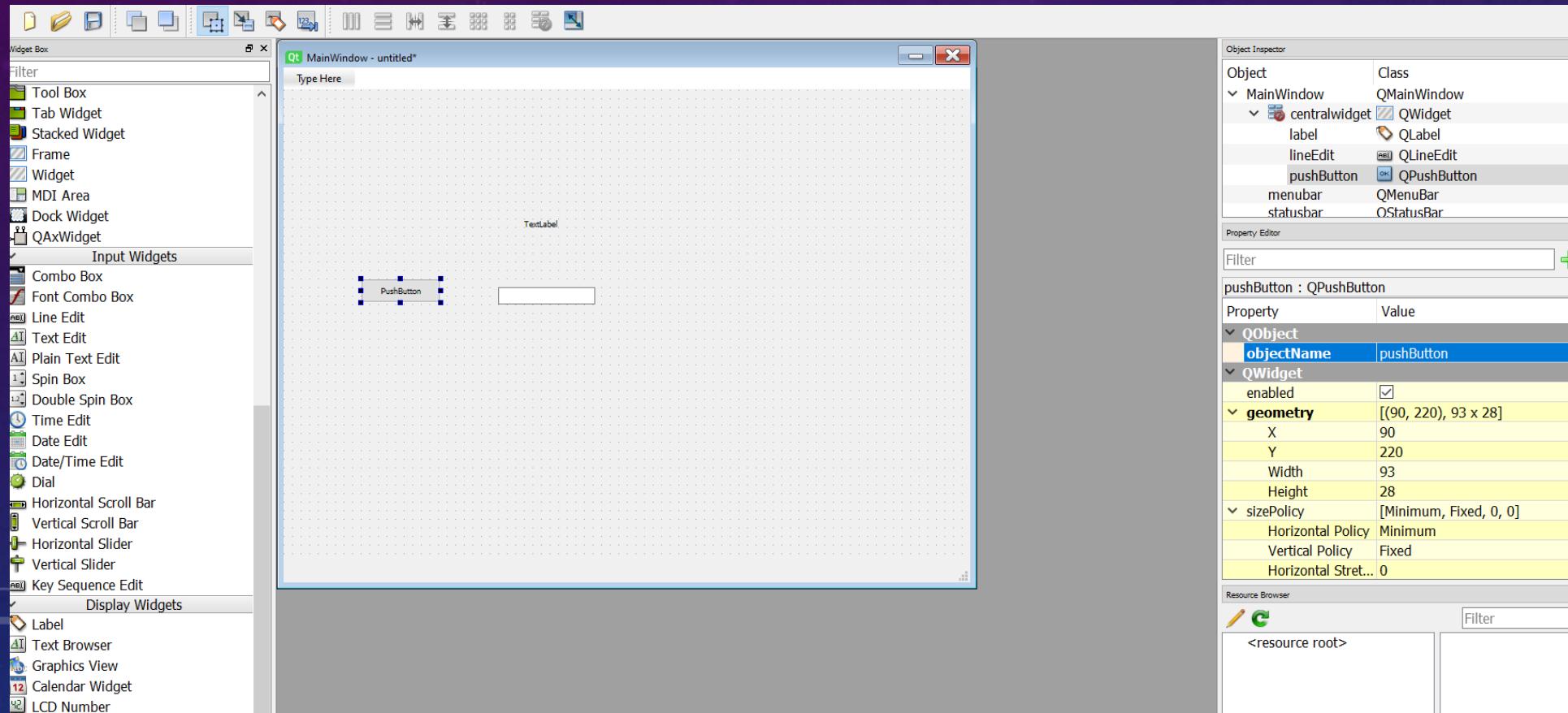
**Left:**  
**Widgets**  
To drag and drop  
Onto the main window



**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

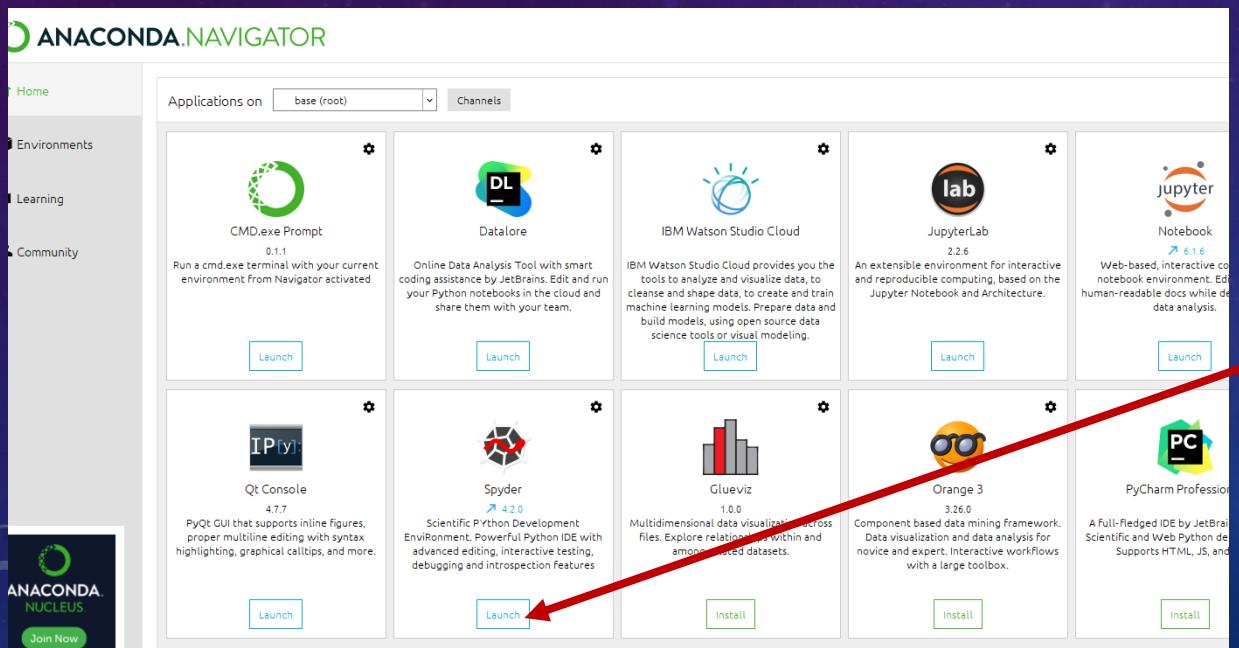
enabled

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

X 70

Y 130

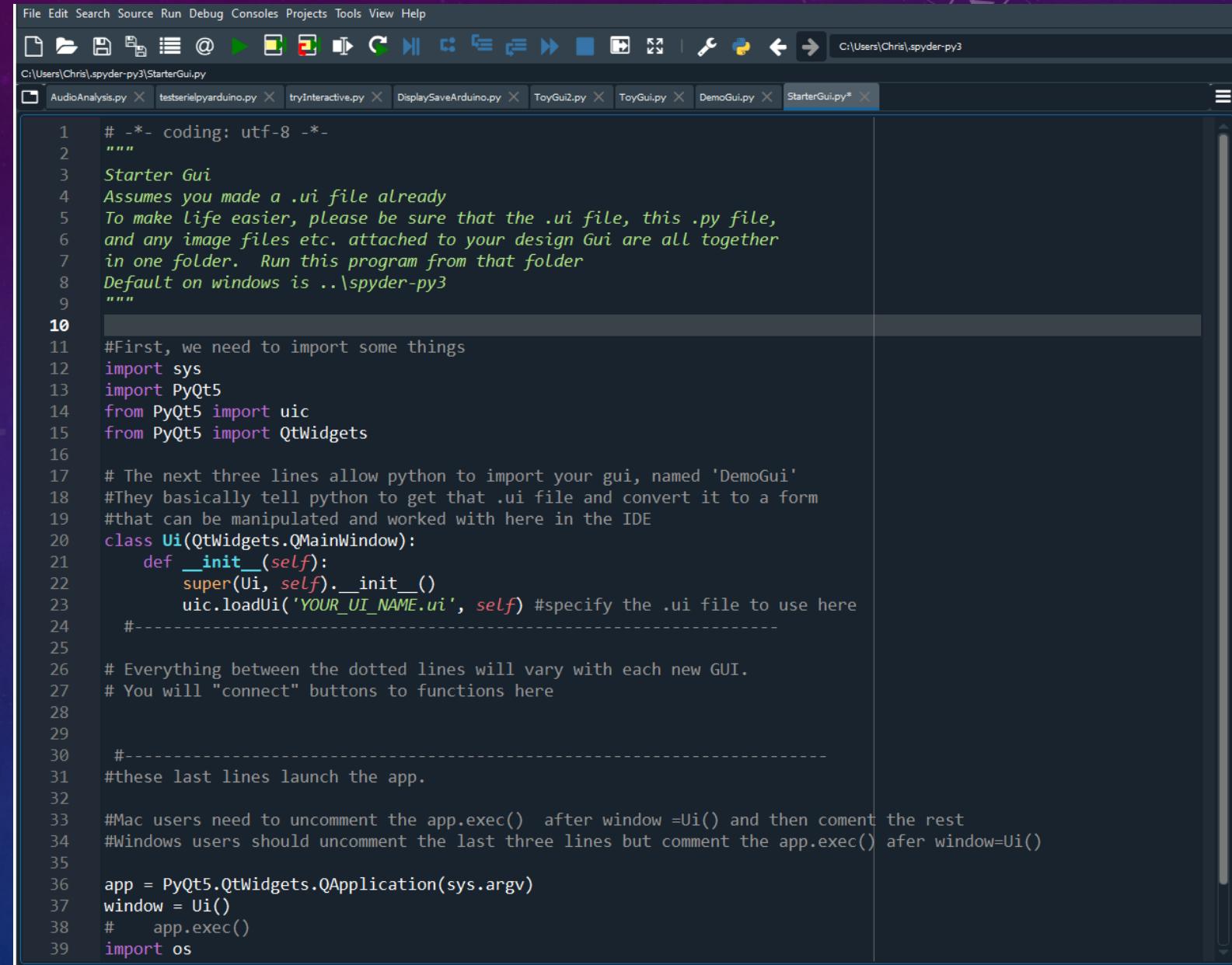
FILE → SAVEAS → DemoGui Into your .spyder-py3 folder



After saving, launch  
spyder from anaconda  
navigator

# START THE CODE

This code to the right is the start of your GUI. Other than for the “import time” statement at the top, And the name of the UI file in line 21 (‘DemoGui’), this won’t need to change if You make a different UI file.



The screenshot shows the Spyder IDE interface with the following details:

- File Menu:** File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Copy, Paste, Find, and Run.
- Path Bar:** C:\Users\Chris\spyder-py3\StarterGui.py
- Tab Bar:** Shows multiple open files: AudioAnalysis.py, testserielpyarduino.py, tryInteractive.py, DisplaySaveArduino.py, ToyGui2.py, ToyGui.py, DemoGui.py, and StarterGui.py\*.
- Code Editor:** Displays the Python code for the Starter Gui. The code uses PyQt5 to load a UI file and run the application.

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

#First, we need to import some things
import sys
import PyQt5
from PyQt5 import uic
from PyQt5 import QtWidgets

# The next three lines allow python to import your gui, named 'DemoGui'
#They basically tell python to get that .ui file and convert it to a form
#that can be manipulated and worked with here in the IDE
class Ui(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui, self).__init__()
        uic.loadUi('YOUR_UI_NAME.ui', self) #specify the .ui file to use here
    ----
# Everything between the dotted lines will vary with each new GUI.
# You will "connect" buttons to functions here

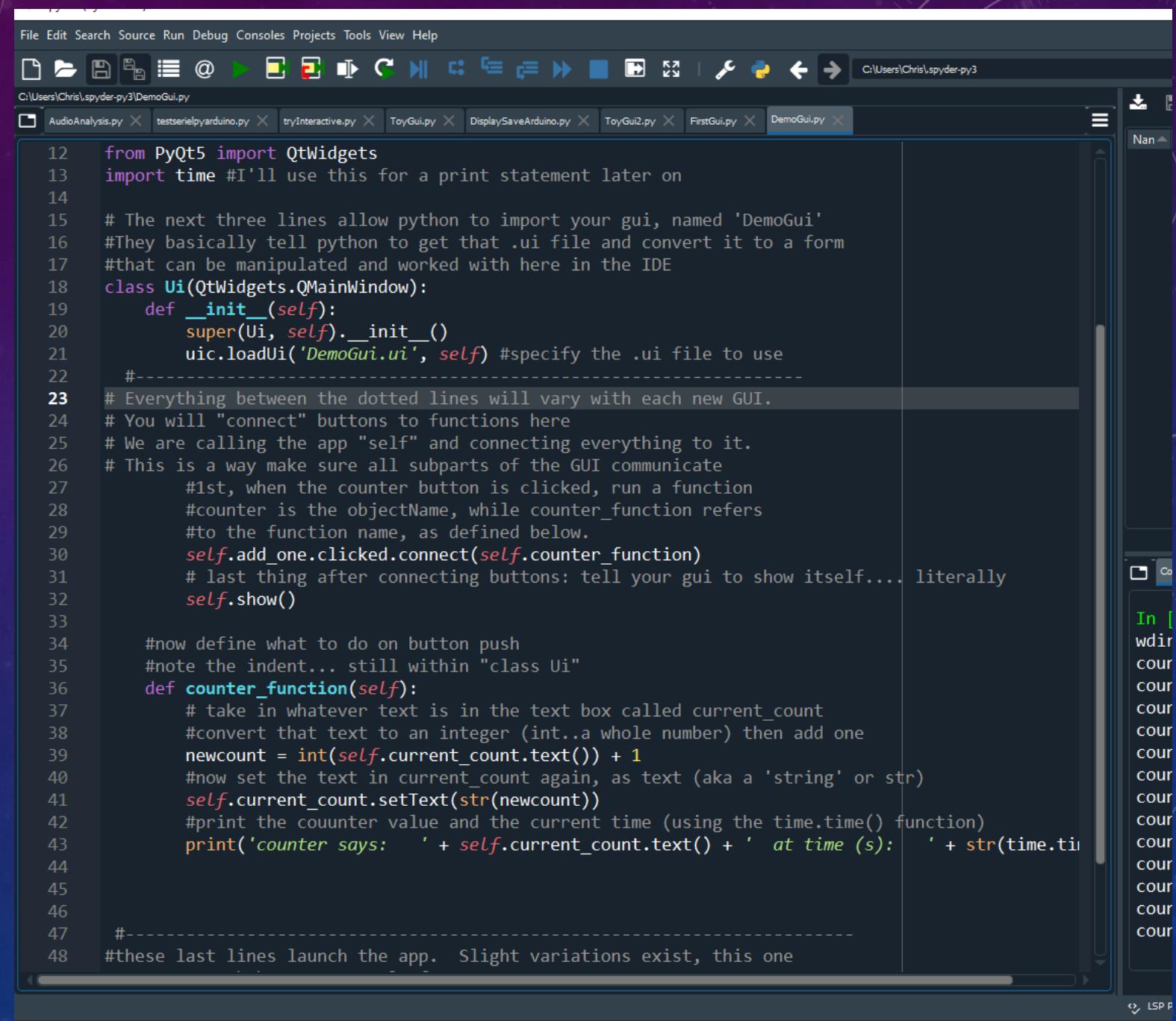
    -----
#these last lines launch the app.

##Mac users need to uncomment the app.exec() after window =Ui() and then coment the rest
##Windows users should uncomment the last three lines but comment the app.exec() afer window=Ui()

app = PyQt5.QtWidgets.QApplication(sys.argv)
window = Ui()
# app.exec()
import os
```

# COMPLETE CODE

You have this code.  
It's called DemoGui.py  
Cleverly (or no) named to be the same as  
DemoGui.ui



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 contains various icons for file operations like Open, Save, Copy, Paste, and Run. The top status bar shows the path C:\Users\Chris\spyder-py3 and the current file name DemoGui.py. The main code editor window displays the following Python code:

```
12 from PyQt5 import QtWidgets
13 import time #I'll use this for a print statement later on
14
15 # The next three lines allow python to import your gui, named 'DemoGui'
16 #They basically tell python to get that .ui file and convert it to a form
17 #that can be manipulated and worked with here in the IDE
18 class Ui(QtWidgets.QMainWindow):
19     def __init__(self):
20         super(Ui, self).__init__()
21         uic.loadUi('DemoGui.ui', self) #specify the .ui file to use
22     #
23     # Everything between the dotted lines will vary with each new GUI.
24     # You will "connect" buttons to functions here
25     # We are calling the app "self" and connecting everything to it.
26     # This is a way make sure all subparts of the GUI communicate
27     #1st, when the counter button is clicked, run a function
28     #counter is the objectName, while counter_function refers
29     #to the function name, as defined below.
30     self.add_one.clicked.connect(self.counter_function)
31     # last thing after connecting buttons: tell your gui to show itself.... literally
32     self.show()
33
34     #now define what to do on button push
35     #note the indent... still within "class Ui"
36     def counter_function(self):
37         # take in whatever text is in the text box called current_count
38         #convert that text to an integer (int..a whole number) then add one
39         newcount = int(self.current_count.text()) + 1
40         #now set the text in current_count again, as text (aka a 'string' or str)
41         self.current_count.setText(str(newcount))
42         #print the counter value and the current time (using the time.time() function)
43         print('counter says: ' + self.current_count.text() + ' at time (s): ' + str(time.time()))
44
45
46
47     #
48     #these last lines launch the app. Slight variations exist, this one
```

# THE MAIN PROGRAM LOGIC:

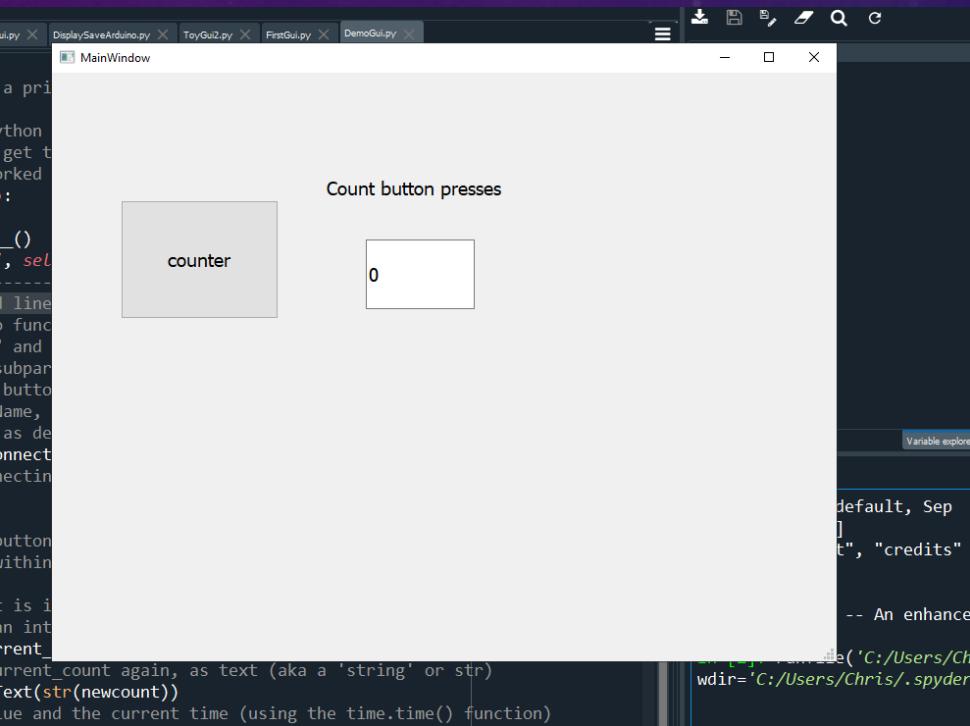
If the `objectName` of your `lineEdit` text box is called “`current_count`”, then to get the text out upon a button press, then in the function definition for that button press, write:  
`self.current_count.text()`

That’s a string, so to make it a number say:  
`Int(self.current_count.text())`

Now it is an integer. You can `+1` and call it something else like `Newcount`:  
`Newcount= Int(self.current_count.text())+1`

Now you can make `Newcount` be the new text in the GUI:  
`self.current_count.setText(str(Newcount))`

Note that `Newcount` was converted back to a string first since it is a text box



# THAT'S AN APP

From these beginnings, more can be done

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

# THIS CAN SCALE UP: 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.setCheckState(Qt.Checked)
    self.Record.setEnabled(False)
    self.PlayBack.setEnabled(True)
    self.Record.clicked.connect(self.Recorded)
    self.PlayBack.clicked.connect(self.PlayBackPress)

PlayBackPress(self):
    # This is executed when the button is pressed
    self.PlayBack.setChecked(True)
    self.show()
    sd.playrec(self.myrecord,S)
    self.PlayBack.setCheckState(Qt.Checked)
    self.PlayBack.setEnabled(False)
    self.Record.setEnabled(True)
    self.Record.clicked.connect(self.Recorded)
    self.PlayBack.clicked.connect(self.PlayBackPress)

Plot_wavePress(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)
    plt.plot(ti...
```



count button presses

0



Record a short memo

Record Audio

3

Duration (s)

Play back clip

Plot the sound wave

