# Zero to Knowing

**BUILDING APPS**

**IN PY**THON

WITH PY**QT**

ZERO TO KNOWING

## WHAT IS PYQT?

ZERO TO KNOWING

A **module that connects the Qt C++ framework and  Python**

Allows us to **create graphical user interfaces** (GUIs).  It has a **wide range of functionalities** such as thread management, support for SQL databases, an embedded web browser, and an extensive collection of GUI widgets.   There is so **much more**!

These features make **Qt** a comprehensive framework for building a wide range of applications beyond just GUIs.

## PyQt5 vs PyQt6

**No Major Difference** Bettween the Two!

We focus on PyQt5 as these two versions of the Framework are so **smiliar** and there are easy work arounds

By learning and understanding PyQt5, you'll be able to **work with older PyQt Apps as well as new PyQt Apps**

## Why do we use PyQt?

- Cross-platform: PyQt allows your programs to run on different operating systems
- Ready-made GUI widgets: Provides pre-built **graphical user interface** (GUI) elements. (buttons, menus, etc)
- Extensive functionality: A wide range of features beyond GUI components, including networking, databases, multimedia, and graphics.
- Visual design: Integrates with, enabling you to visually design your user interface by dragging and dropping elements.
- Language simplicity: PyQt is based on Python, which has a straightforward syntax and is known for being beginner-friendly.

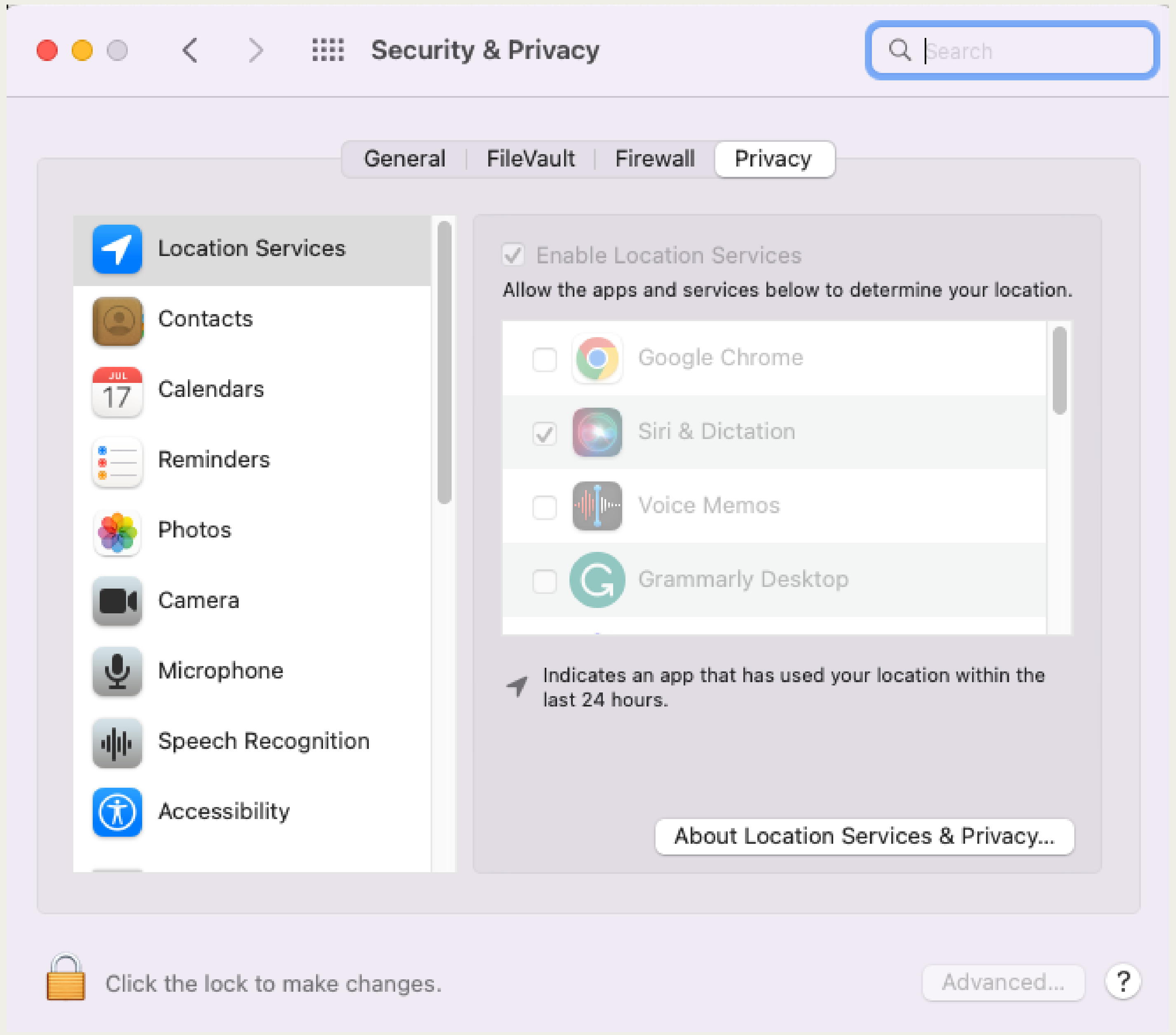
## Windowed Applications

### The Foundations of your first App

ZERO TO KNOWING

# Windowed Applications

Two current and Popular examples of Windowed Applications

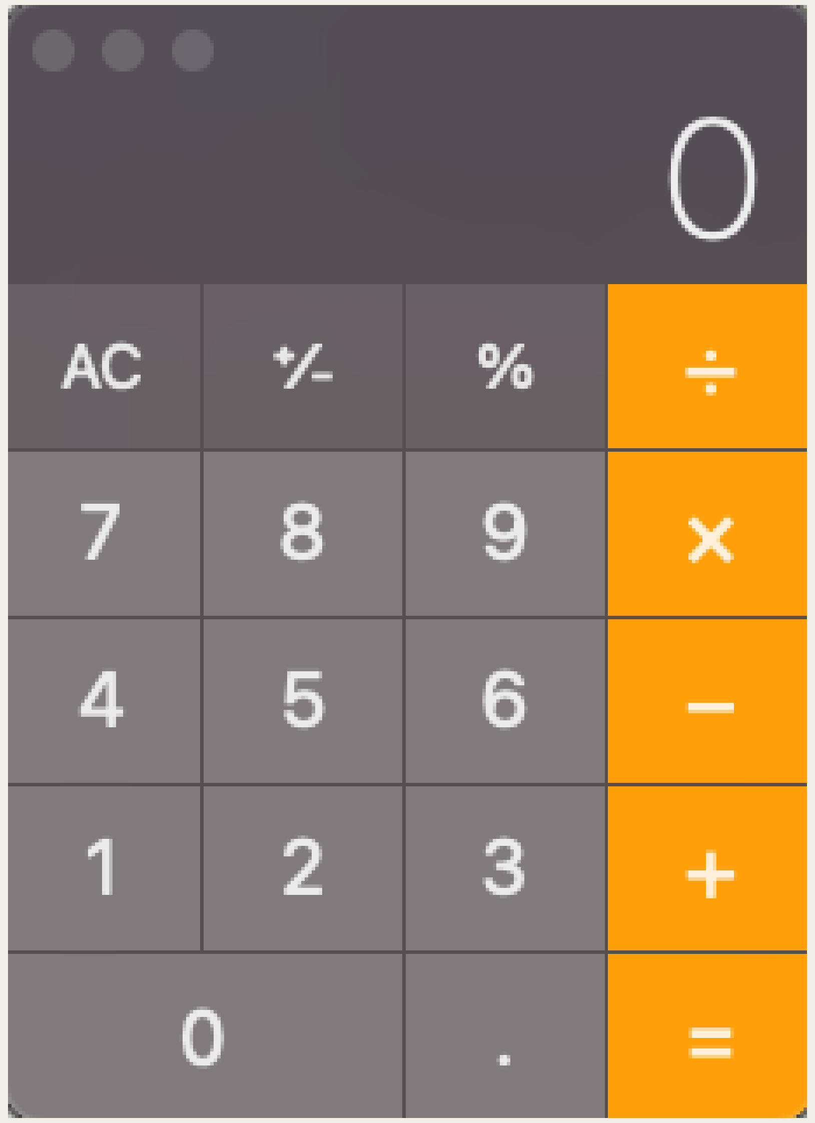


☑ Enable Location Services
Allow the apps and services below to determine your location.

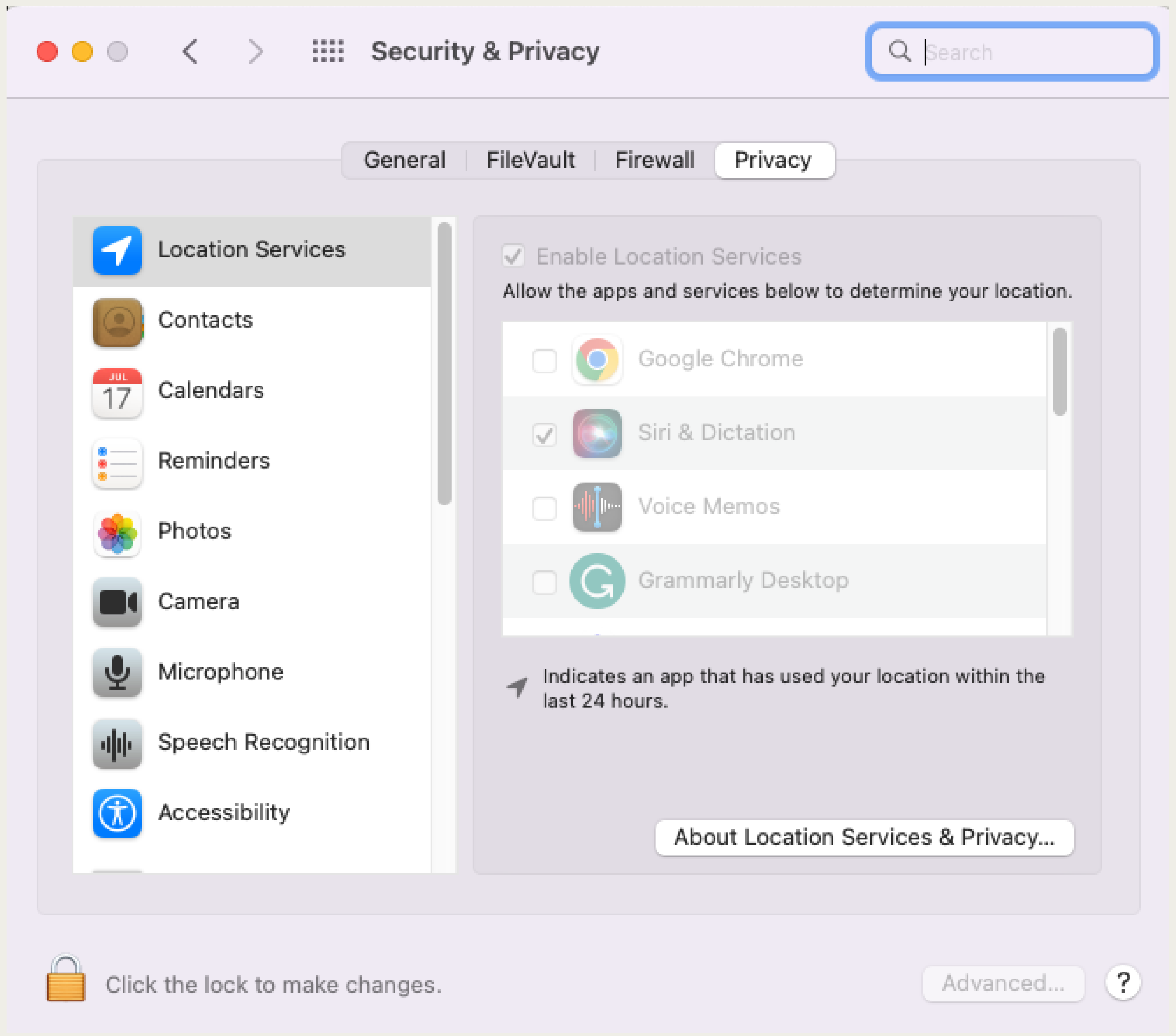
| | |
|---|---|
| ☐ | Google Chrome |
| ☑ | Siri & Dictation |
| ☐ | Voice Memos |
| ☐ | Grammarly Desktop |

Indicates an app that has used your location within the last 24 hours.

About Location Services & Privacy...
Click the lock to make changes.
Advanced...
?



# Windowed Applications



✓ Enable Location Services
Allow the apps and services below to determine your location.

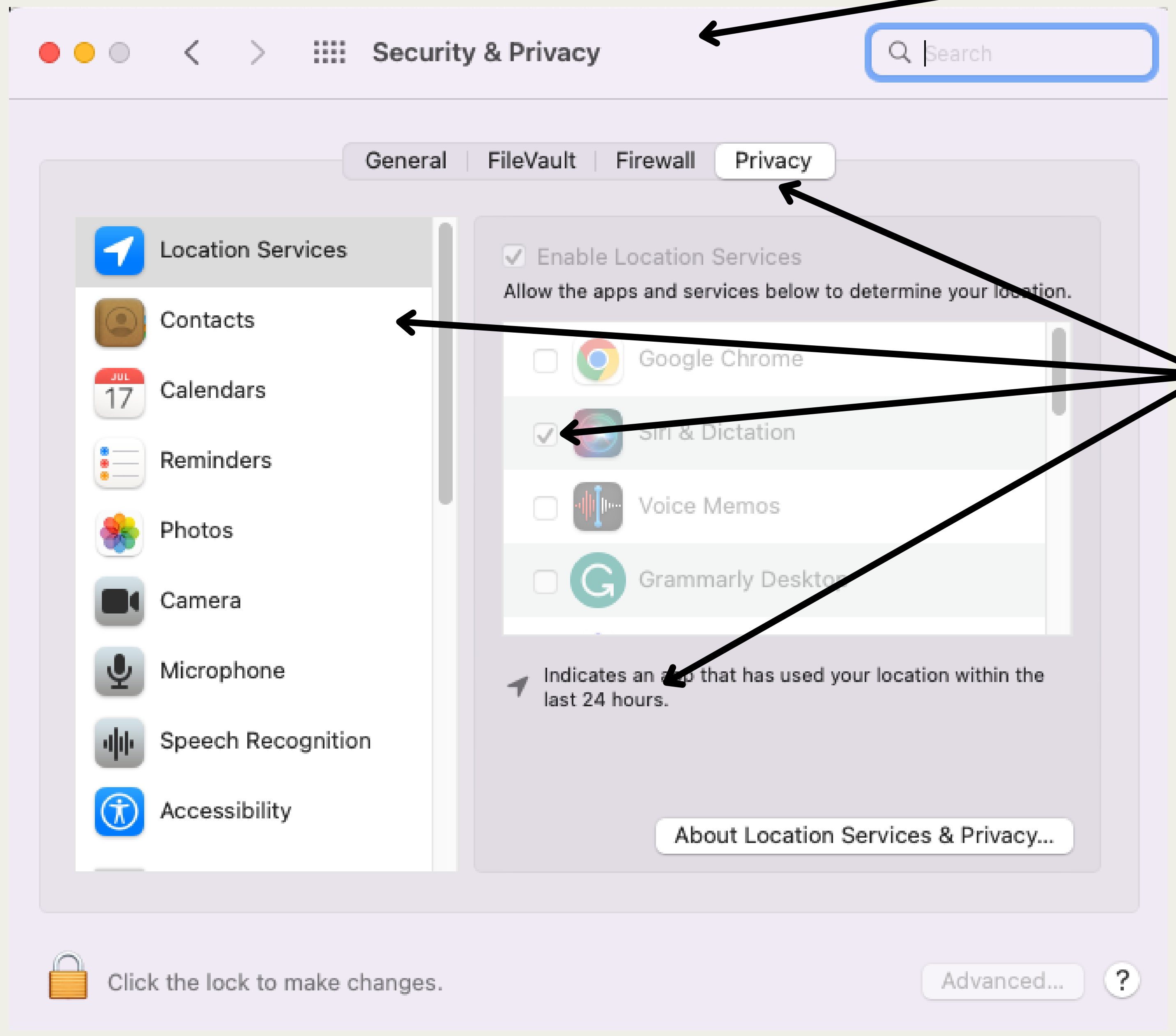
Indicates an app that has used your location within the last 24 hours.

Click the lock to make changes.

Main **Window**

App **Widgets** (Objects)

# Windowed Applications



Security & Privacy

Search

General | FileVault | Firewall | Privacy

- Location Services
- Contacts
- Calendars
- Reminders
- Photos
- Camera
- Microphone
- Speech Recognition
- Accessibility

✓ Enable Location Services

Allow the apps and services below to determine your location.

- Google Chrome
- ✓ Siri & Dictation
- Voice Memos
- Grammarly Desktop

Indicates an app that has used your location within the last 24 hours.

About Location Services & Privacy...

Click the lock to make changes.

Advanced...

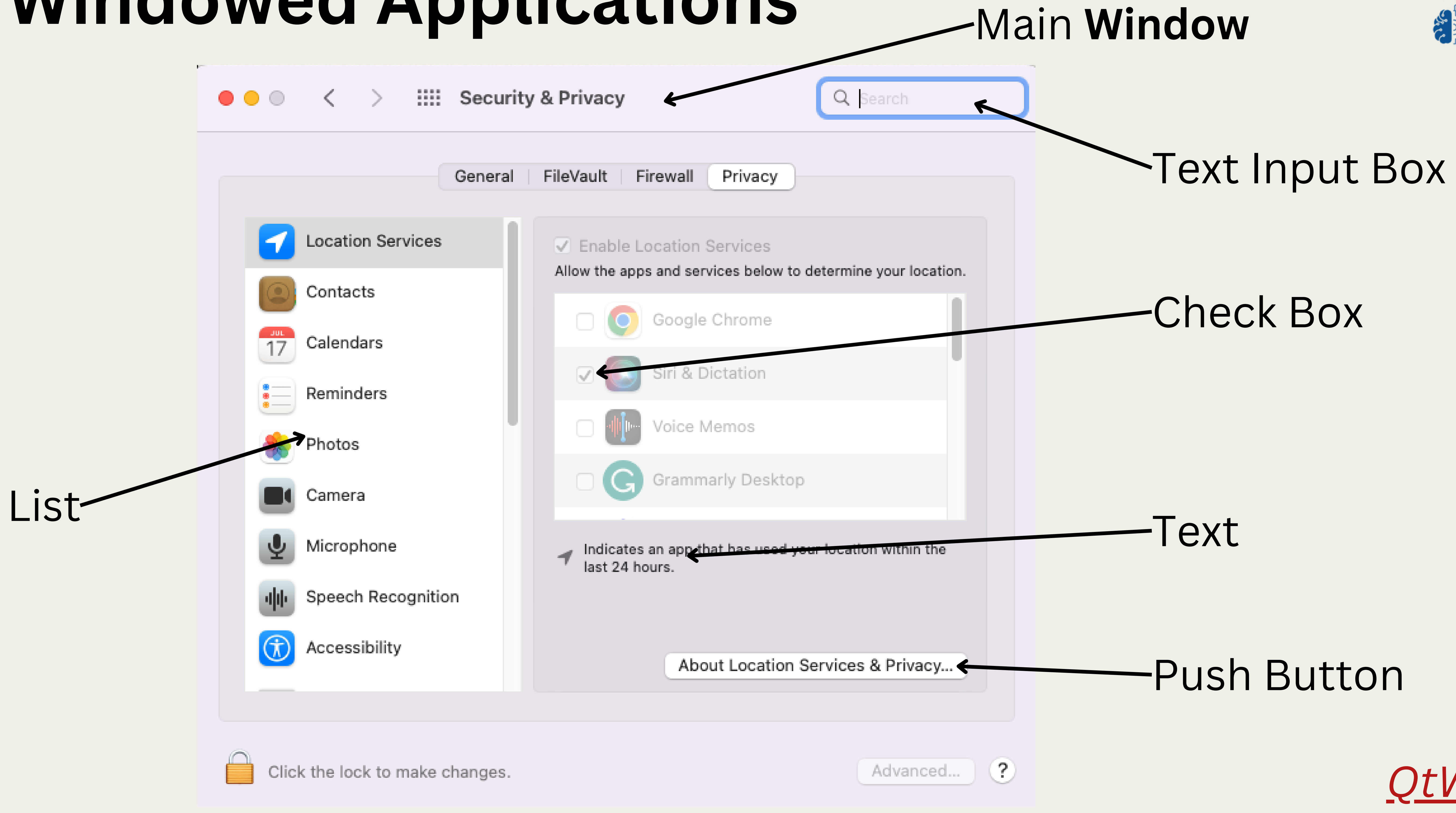
Main **Window**

App **Widgets** (Objects)

These work together to **create the Main App Interface** (GUI)

# Windowed Applications

ZERO TO KNOWING



☑ Enable Location Services

Allow the apps and services below to determine your location.

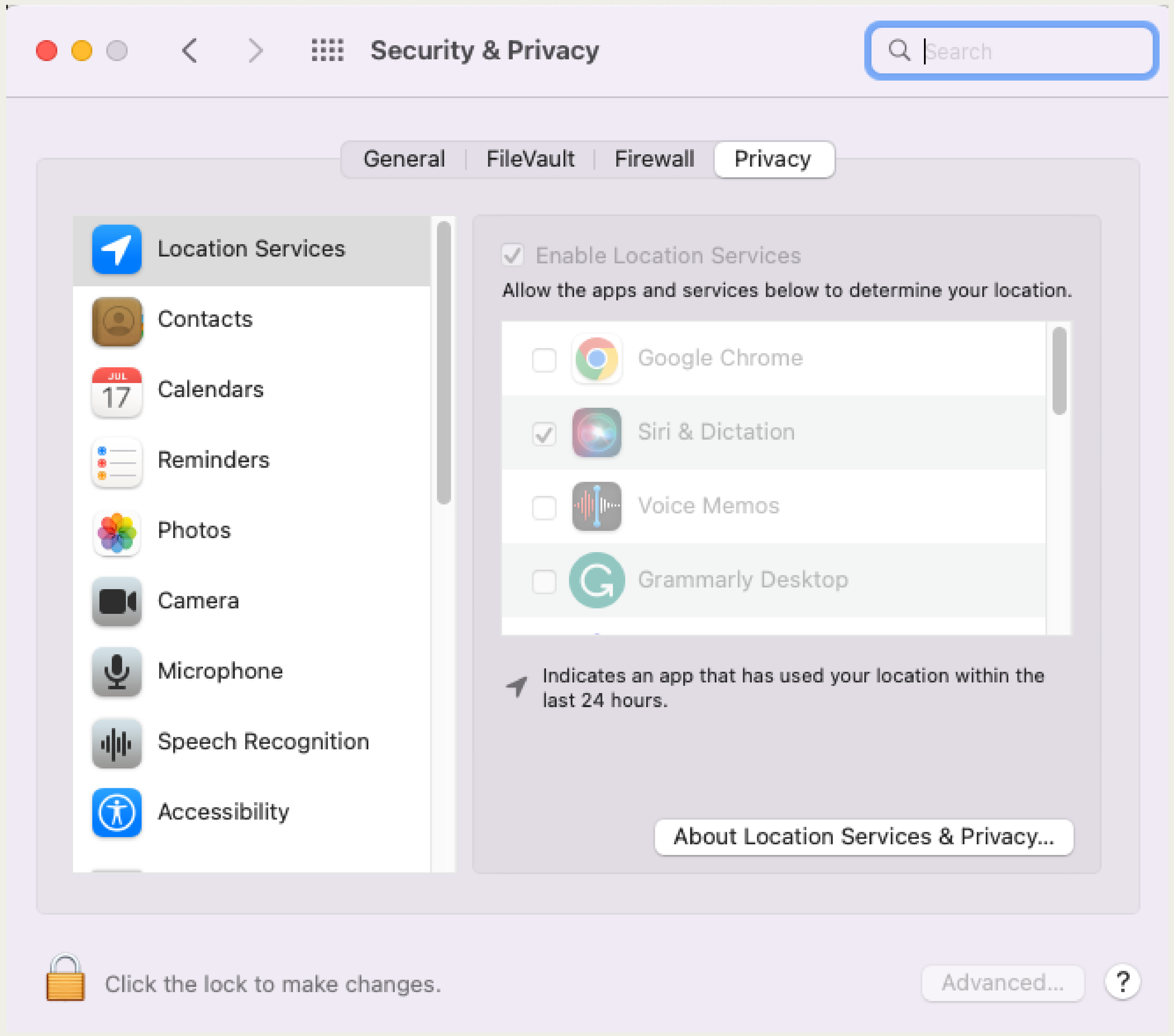
Indicates an app that has used your location within the last 24 hours.

Click the lock to make changes.

*QtWidgets*

# Designing the Layout

ZERO TO KNOWING



Security & Privacy

General | FileVault | Firewall | Privacy

- Location Services
- Contacts
- Calendars
- Reminders
- Photos
- Camera
- Microphone
- Speech Recognition
- Accessibility

✓ Enable Location Services

Allow the apps and services below to determine your location.

- [ ] Google Chrome
- [✓] Siri & Dictation
- [ ] Voice Memos
- [ ] Grammarly Desktop

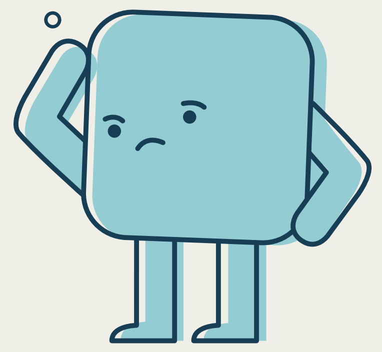
Indicates an app that has used your location within the last 24 hours.

About Location Services & Privacy...

Click the lock to make changes.

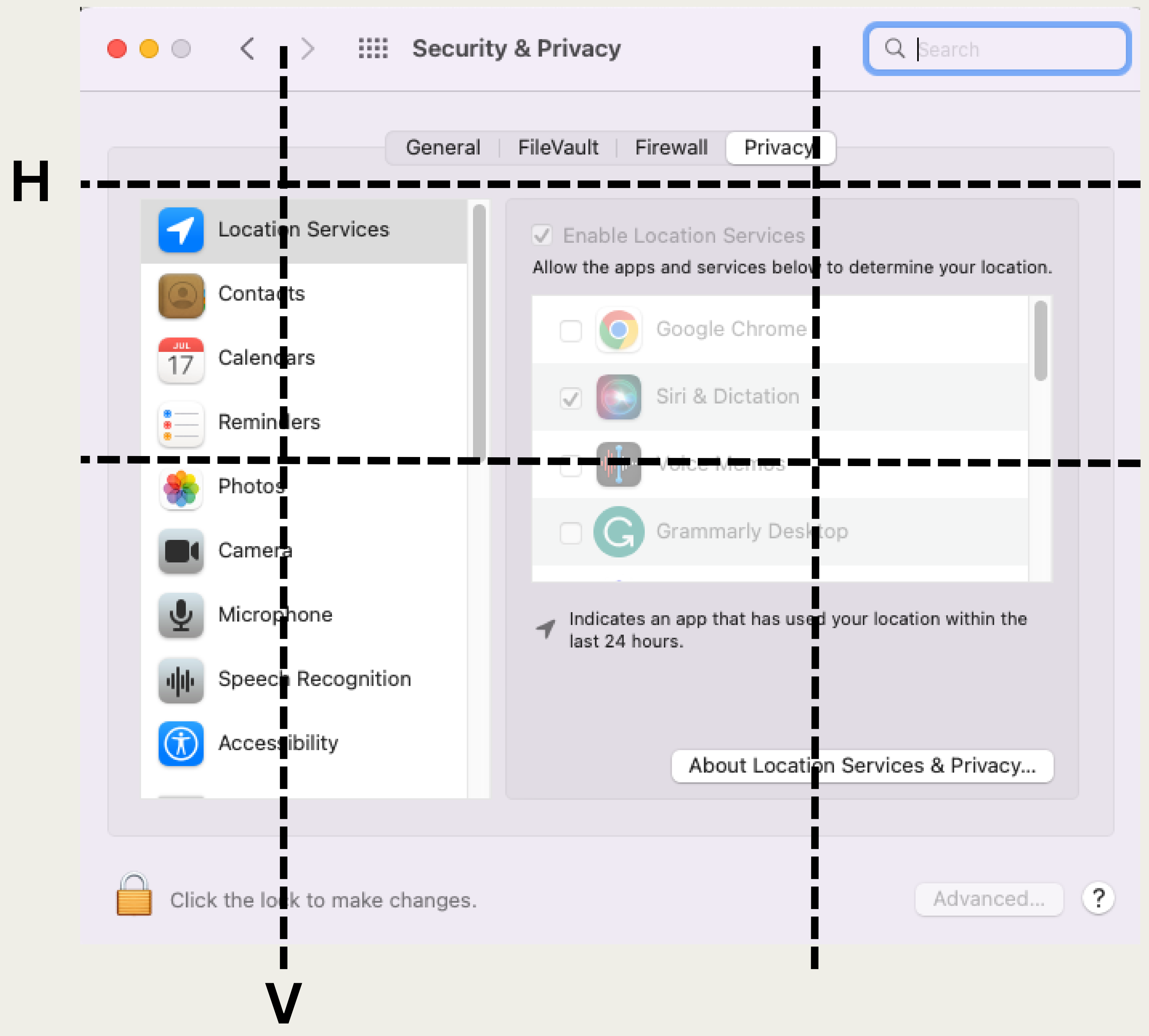
Advanced... ?

## Rows & Columns



How could be design this App so that everything is held in either a Row or a Column?

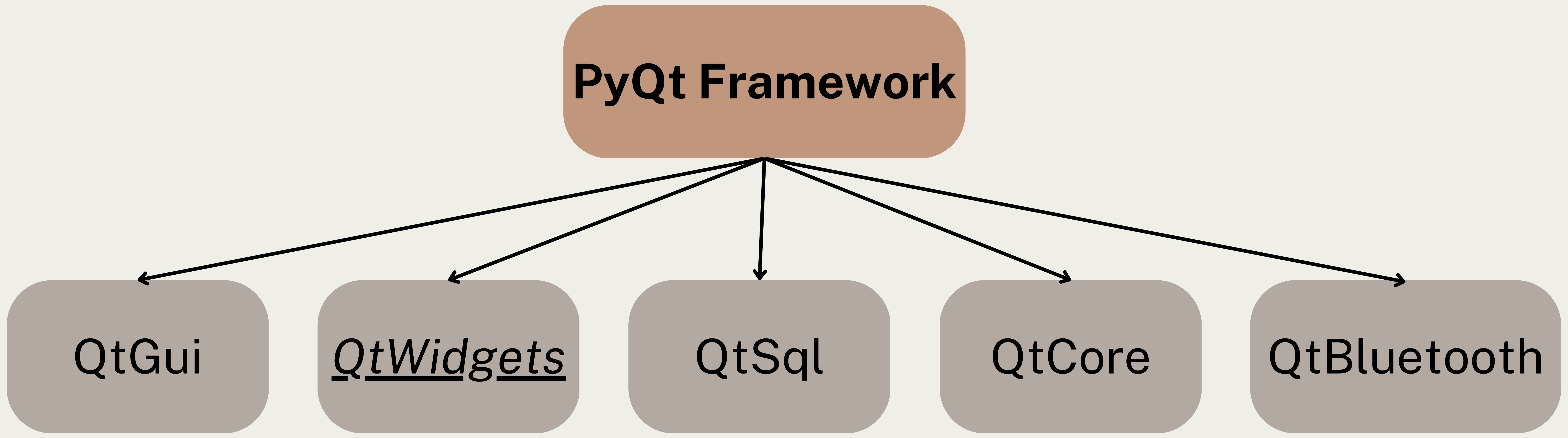
# Designing the Layout



We use Layouts in PyQt to build out our design.

Widgets (objects) are automatically aligned **vertically** and/or **horizontally**

## How to use PyQt



PyQt Framework has extensive modules that are all apart of it. Allowing you to import and use the specific modules you need for your project

# How to use PyQt

**PyQt Framework**

- **QtGui**: Classes for system integration, event handling, 2D graphics, basic imaging, fonts
- **QtWidgets**: provides a set of UI elements to create classic desktop-style user interfaces
- **QtSql**: Provides a driver layer, SQL API layer, and a user interface layer for SQL databases
- **QtCore**: A part of Qt's essential modules. The module adds features to C++:
- **QtBluetooth**: enables connectivity between Bluetooth enabled devices

## The Code Burger

1. All **Imports**
2. Main App Objects and **Settings**
3. **Create all Widgets** needed in App
4. **Design** your Layout, add your widgets to the screen
5. **Set the final layout** to the Main window
6. **Show** and **Execute** your app

## Class Flashback

```
class App():
    def __init__(self, users, storage, username):
        self.users = users
        self.storage = storage
        self.username = username

    def login(self):
        if self.username == "owner" and self.users >= 1:
            print("Welcome: ", self.username)
        else:
            print("You are not a user!")

    def increase_capacity(self, number):
        self.storage += number
        print("Updated:", self.storage)

admin = App( 100, 256, "owner" )

admin.login()
admin.increase_capacity( 64 )
```

ZERO TO KNOWING

**OOP**

**Pause the Video!**

Try to understand what is happening

This should look familiar

Do you remember how to work with Objects and Classes in Python?

What is a **Method/Property**?

# Class Flashback

ZERO TO KNOWING

```
class App():
    def __init__(self, users, storage, username):
        self.users = users
        self.storage = storage
        self.username = username

    def login(self):
        if self.username == "owner" and self.users >= 1:
            print("Welcome: ", self.username)
        else:
            print("You are not a user!")

    def increase_capacity(self, number):
        self.storage += number
        print("Updated:", self.storage)

admin = App(100, 256, "owner")

admin.login()
admin.increase_capacity( 64 )
```

**A Class has properties and methods which relate to that class or an object of that Class**

We create an Object of the App class

We pass 3 Arguments to our Class

These **Arguments are** then **given to** our **__init__** method as our parameters

We use the **parameters** to the **value of our properties**

We can use these **properties around our class** similar to a normal **variable in a function**

A **Method** is a Function in a Class
A **Property** is a Variable in a Class

A method/property **must be linked** to an Object to work

**QApplication, QPushButton, QLabel**

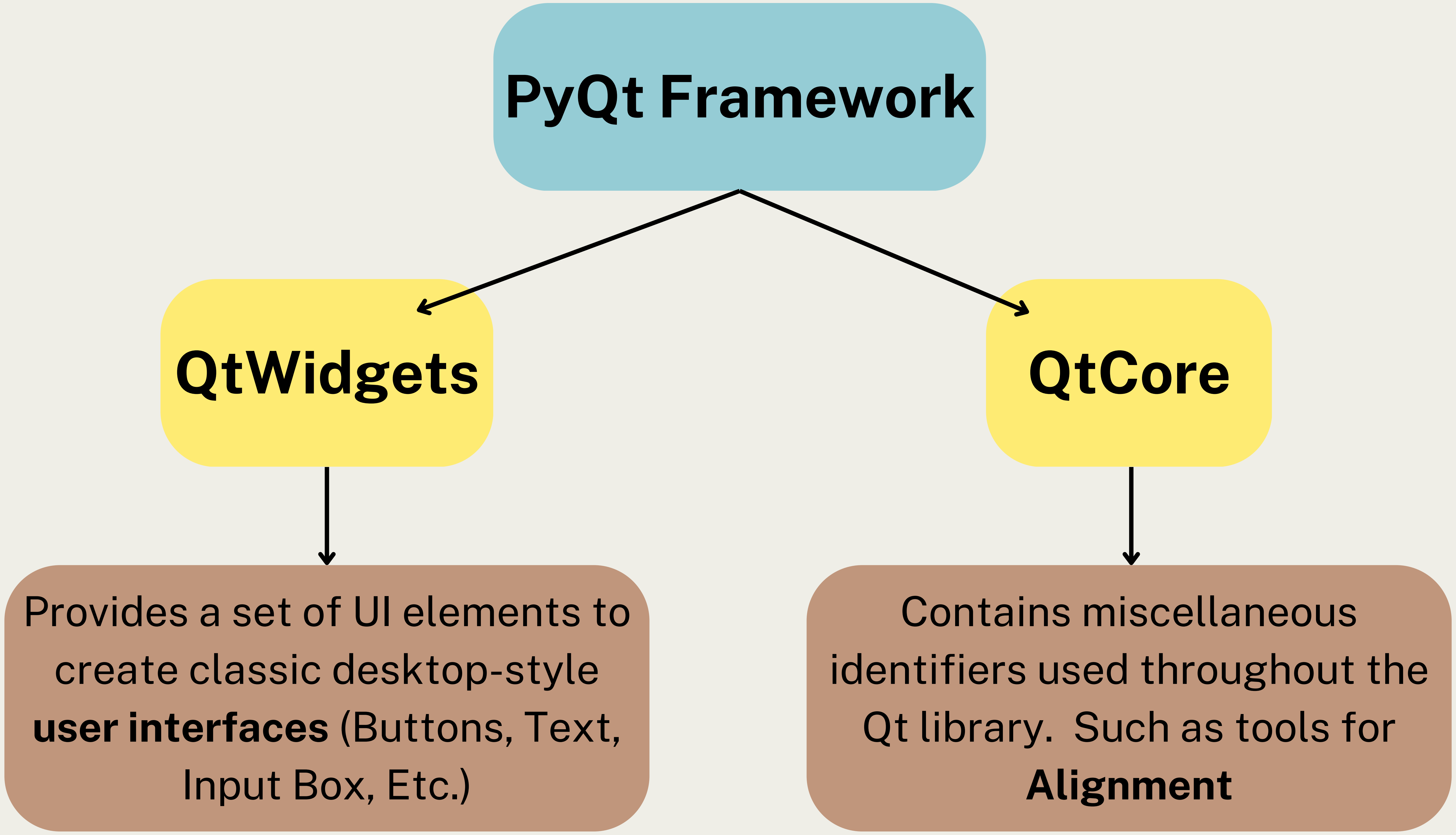
# Connecting to PyQt

ZERO TO KNOWING

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QVBoxLayout
```

| **PyQt Classes** | **What it does** |
|---|---|
| QApplication | Important Object - Allows us to **create** and execute our **app** |
| QWidget | Important Object - Allows us to **create a Main Window** |
| QLabel | This is simply a **Text** (string) object |
| QPushButton | This is a click/submit style **button** object |
| QVBoxLayout | This allows us to use **Vertical Alignment** |

# PyQt Modules -> QtWidgets and Qt:



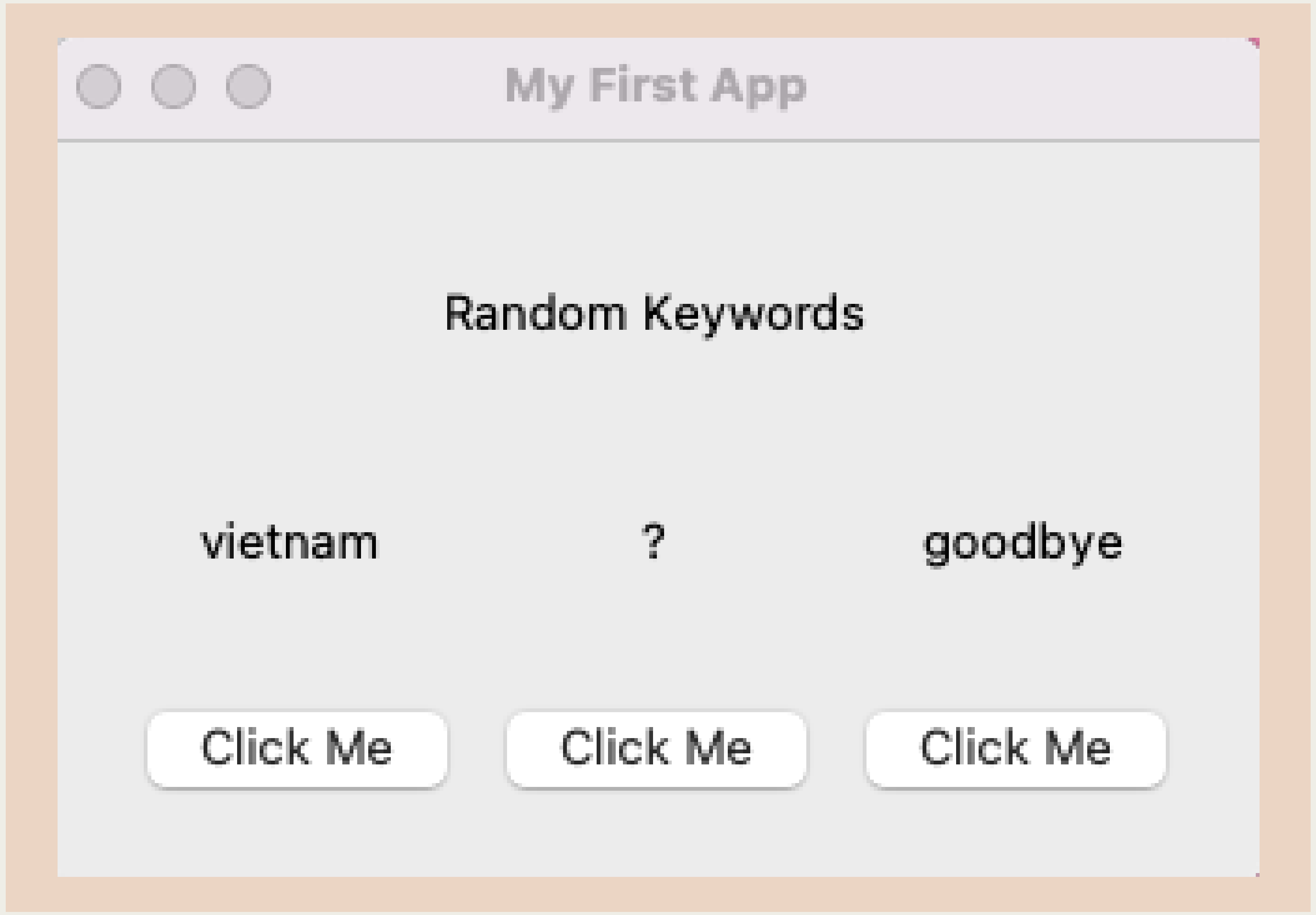
Provides a set of UI elements to create classic desktop-style **user interfaces** (Buttons, Text, Input Box, Etc.)

Contains miscellaneous identifiers used throughout the Qt library.  Such as tools for **Alignment**

# Main PyQt Methods

| **PyQt Method** | **What it does** |
|---|---|
| .addWidget() | Allows you to **add an object to the layout** |
| .setText() | **Change the text** of an existing object |
| .addLayout() | Used to **add Layouts** together |
| .setLayout() | Used to **set the final design** to the main window |
| .show() / .hide() | Allows you to **show or hide an object** |

## You're first application:



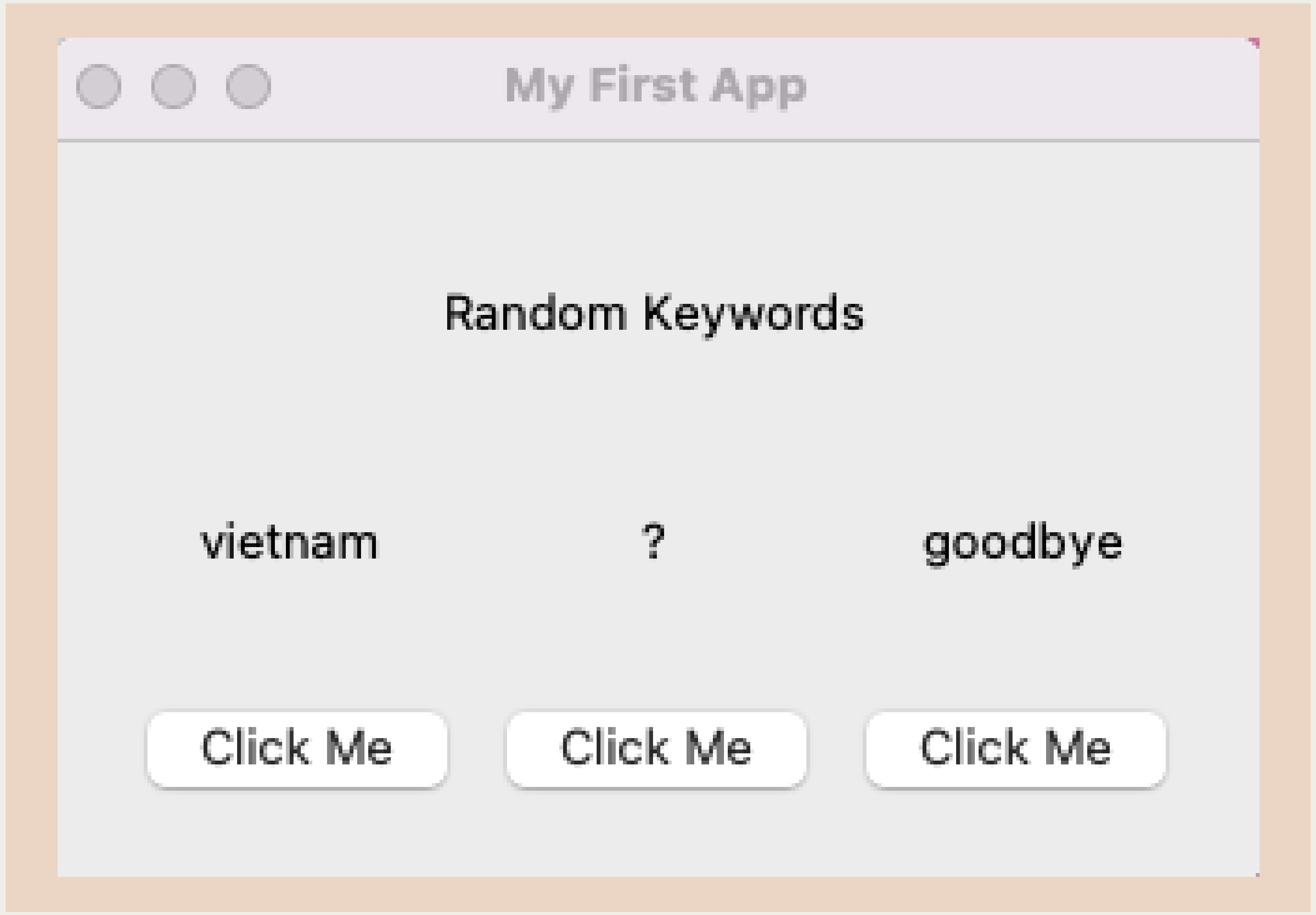
Random Keywords

vietnam ? goodbye

**Your first App!**

Let's get you comfortable with the **foundations** of PyQt and understanding the **basic concepts**!

## You're first application:



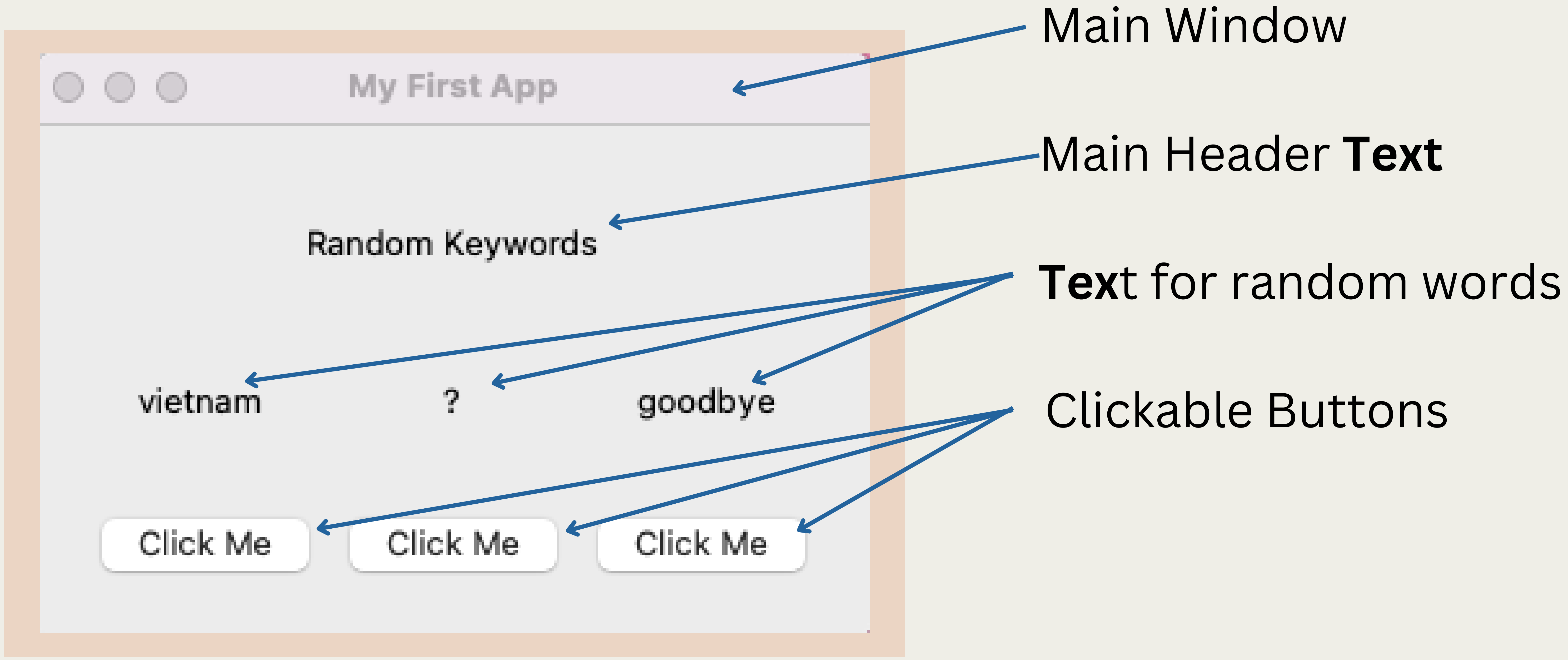
Main Window

Main Header Text

Text for random words

Clickable Buttons

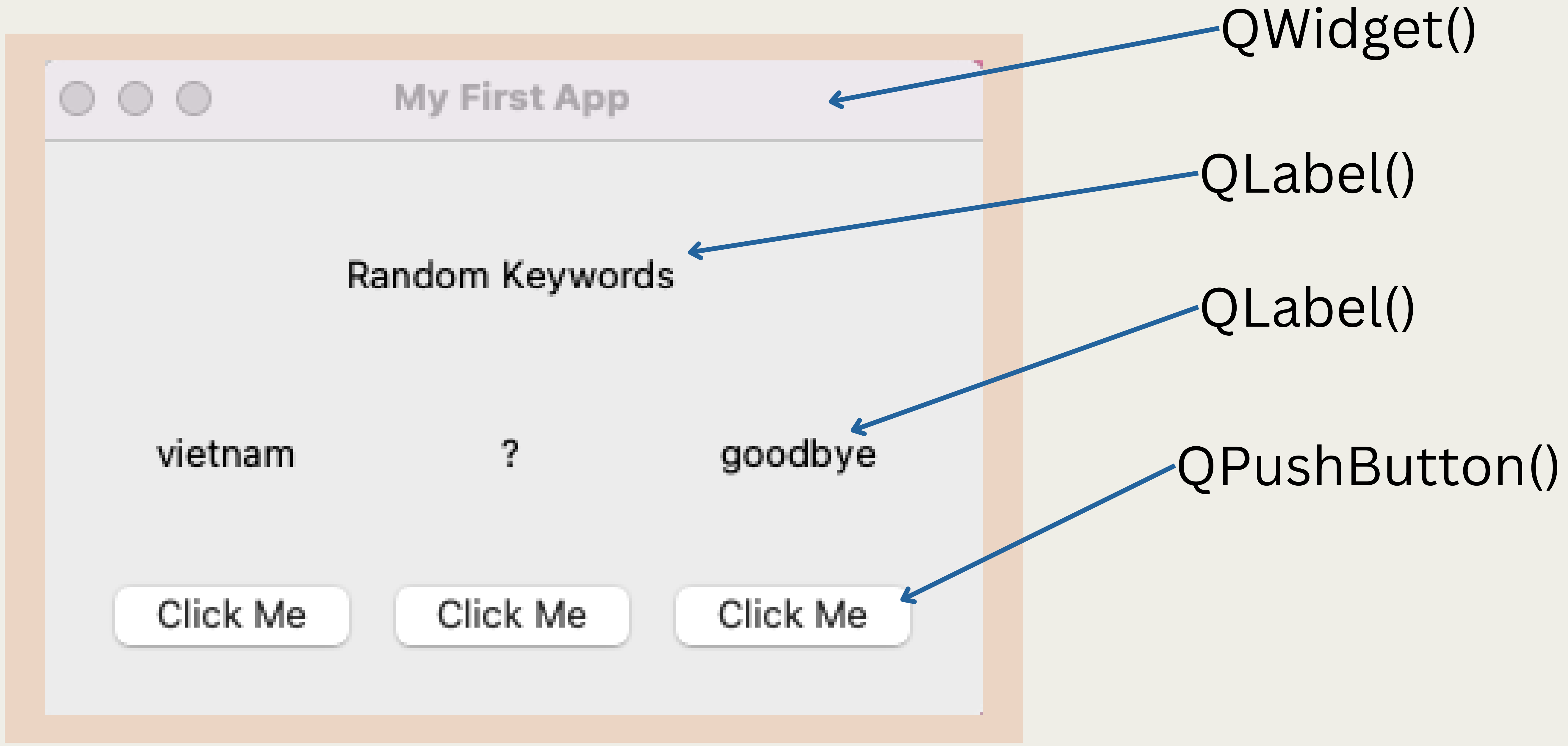
## You're first application:



Can you name all the Classes we will need?

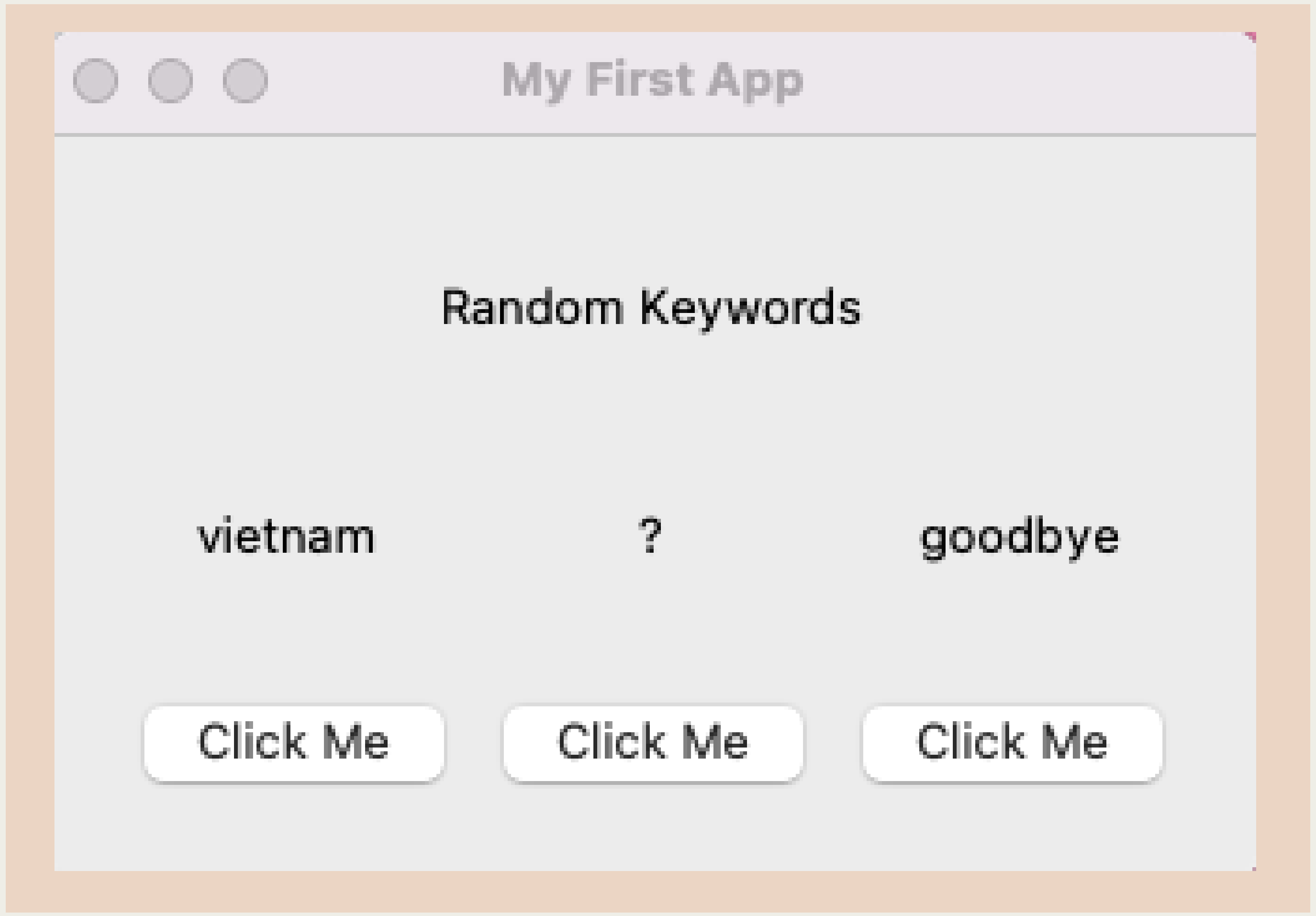
## You're first application:

ZERO TO KNOWING



What about our Design?  How could we do this?

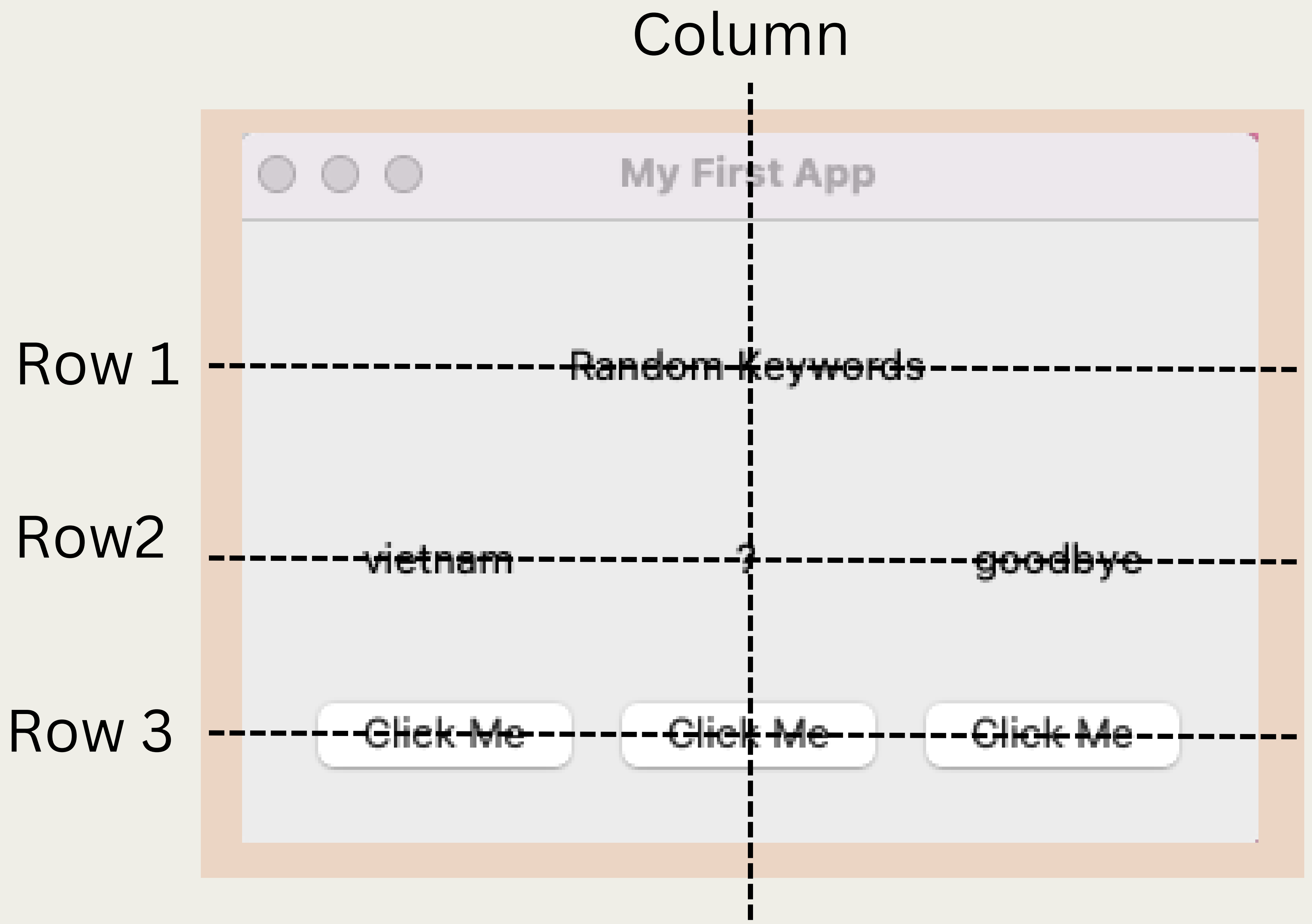
## Application Design



The app
(Base layers).
These 3 layers are
into
our which is
used as our

What about our Design? How could we do this?

## Application Design



The app **starts with 3 Rows** (Base layers). These 3 layers are **stacked like a cake** into our **column** which is used as our **final design**

What about our Design? How could we do this?

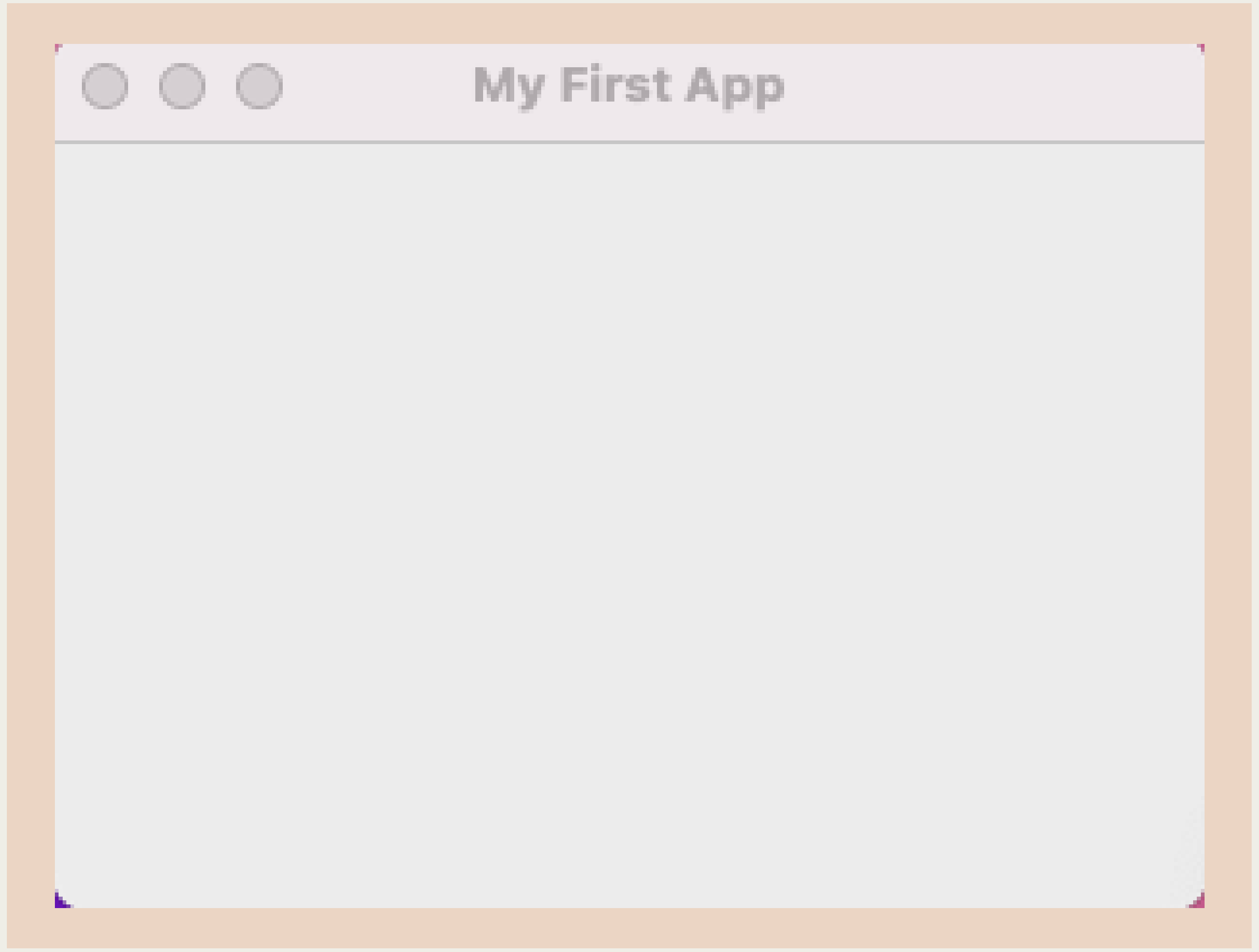
## Initial Code Setup:

```
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QPushButton,
QVBoxLayout, QHBoxLayout

#App Settings
app = QApplication([])
main_window = QWidget()
main_window.setWindowTitle("My First App")
main_window.resize(300,200)

#Create all Object/Widgets below here

main_window.show()
app.exec_()
```



## Designing our App

Learning how to set the layout and design the app

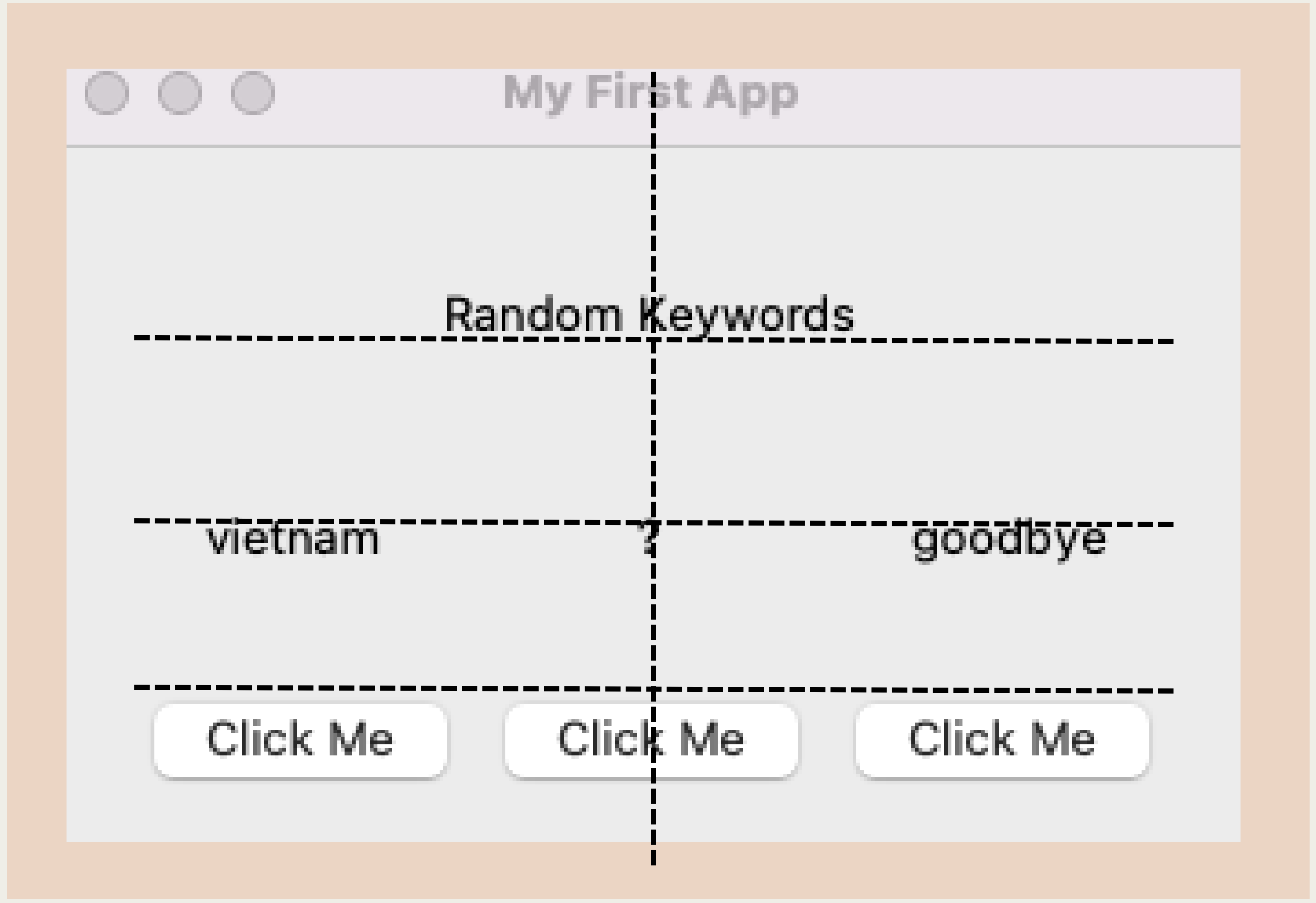
ZERO TO KNOWING

## DESIGNING OUR APP

How can we achieve this look?

PyQt Layout Tools:

- QVBoxLayout()
- QHBoxLayout()



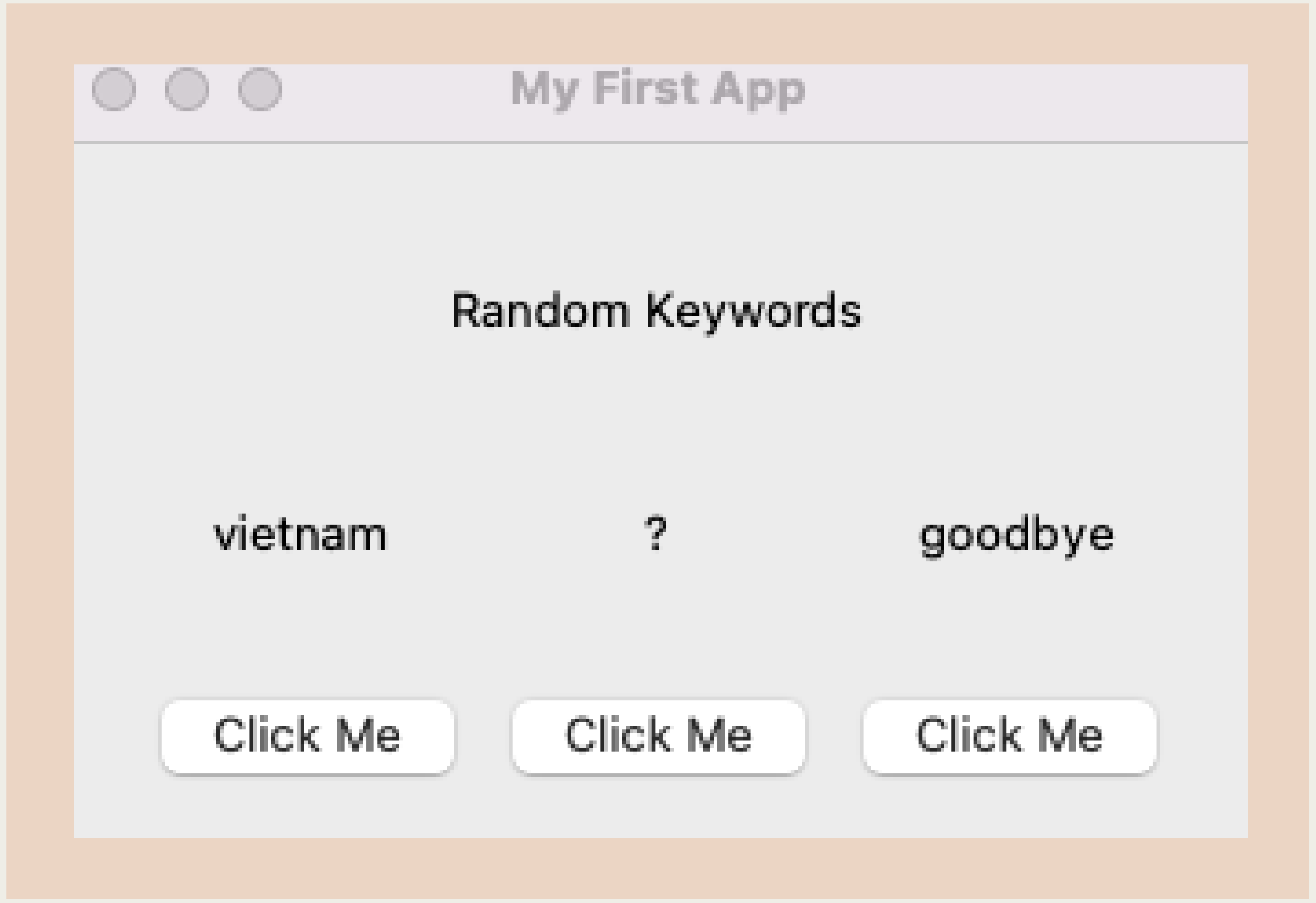
## DESIGNING OUR APP

ZERO TO KNOWING

```
#Create all Object/Widgets below here
title_text = QLabel("Random Keywords")
text1 = QLabel("?")
text2 ...
text3 ...

button1 = QPushButton("Click Me")
button2 ...
button3...

#Design our app
master_layout = QVBoxLayout()
row1 = QHBoxLayout()
```



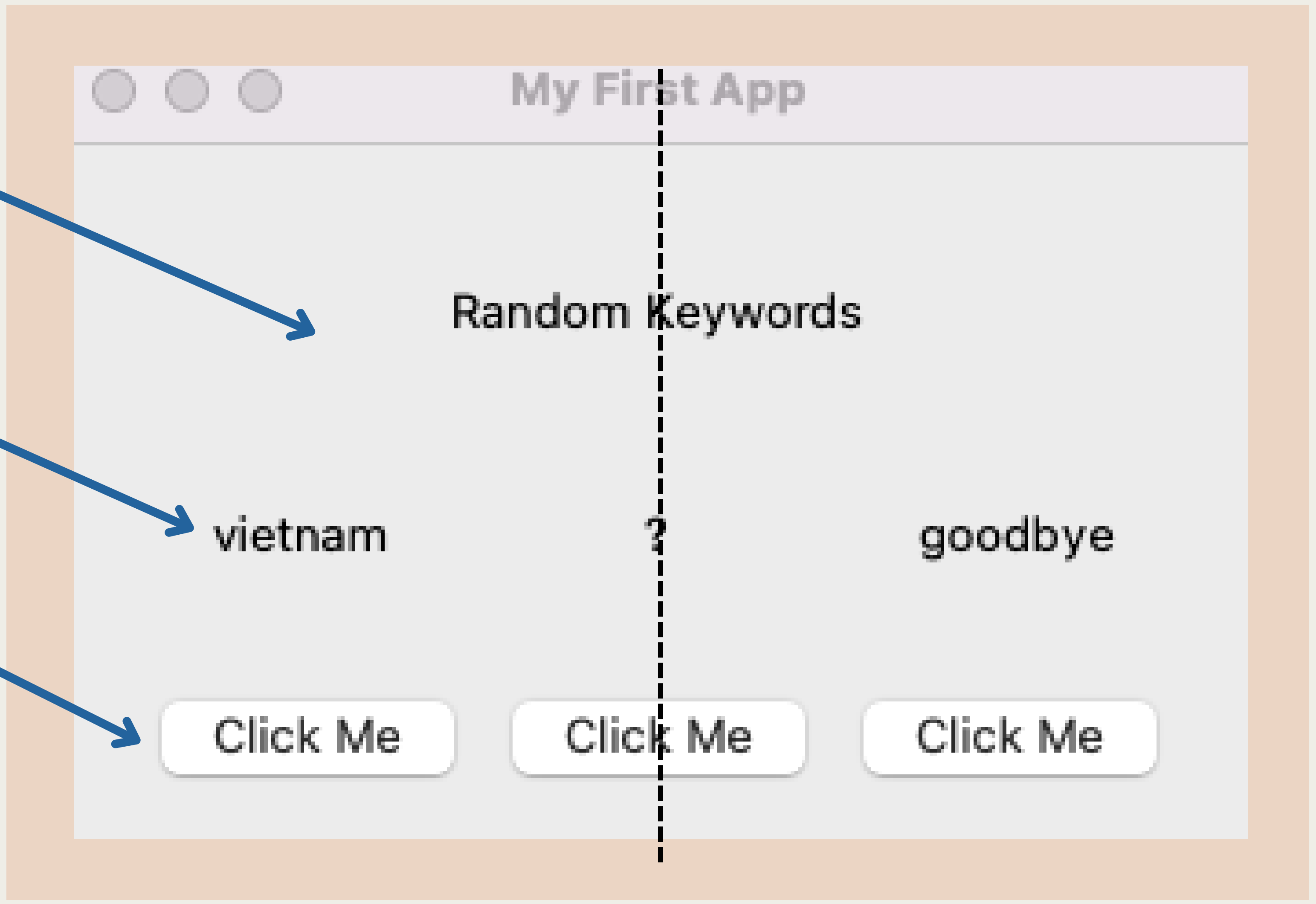
## DESIGNING OUR APP

ZERO TO KNOWING

```
#Create all Object/Widgets below here
title_text = QLabel("Random Keywords")
text1 = QLabel("?")
text2 ...
text3 ...

button1 = QPushButton("Click Me")
button2 ...
button3...

#Design our app
master_layout = QVBoxLayout()
row1 = QHBoxLayout()
```



## DESIGNING OUR APP

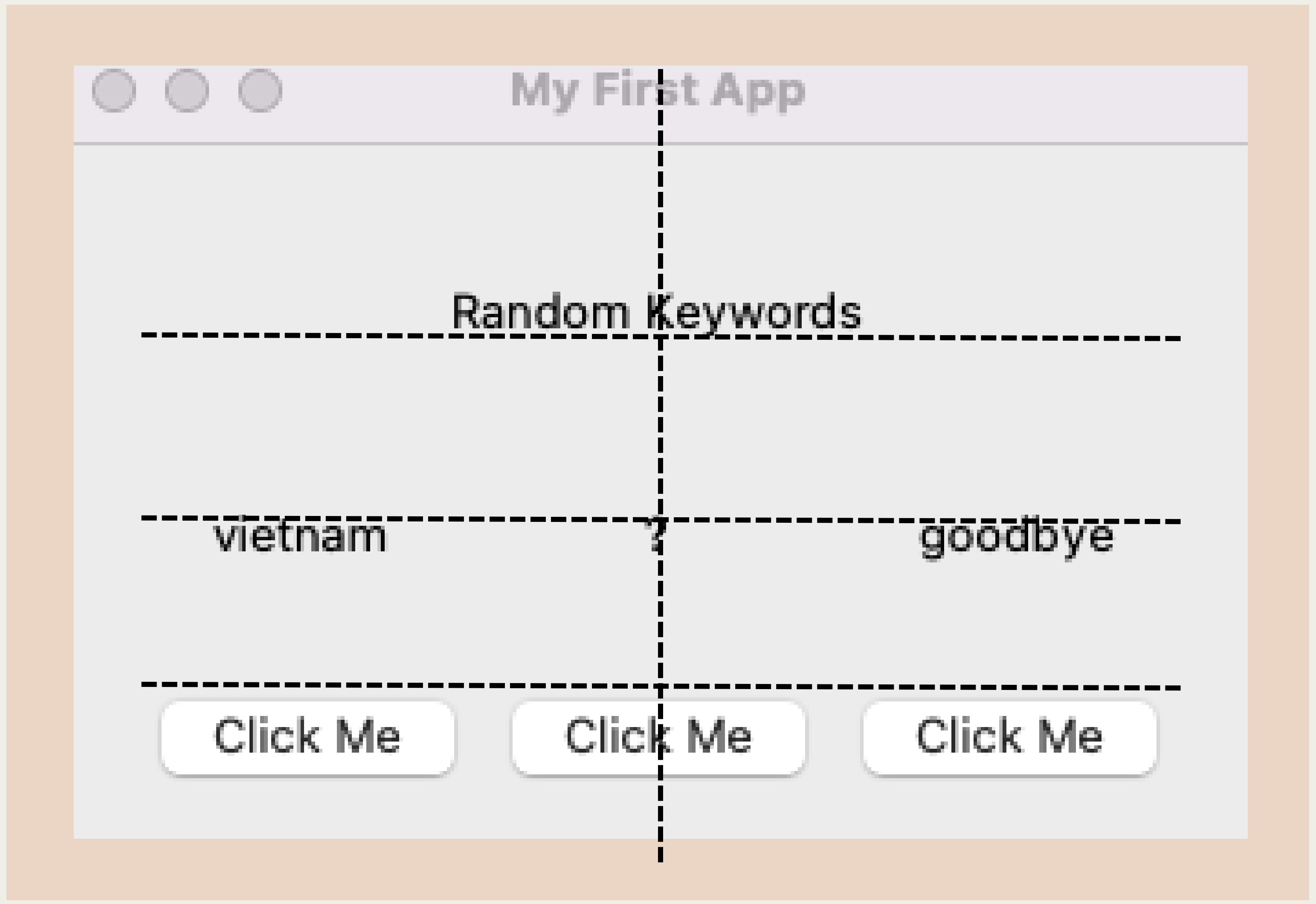
ZERO TO KNOWING

```
#Design our app
master_layout = QVBoxLayout()
row1 = QHBoxLayout()
row2...
row3 ...

row1.addWidget(title_text, alignment=Qt.AlignCenter)

row2.addWidget(text1, alignment=Qt.AlignCenter)
row2.addWidget...

row3.addWidget(button1)
row3.addWidget...
```



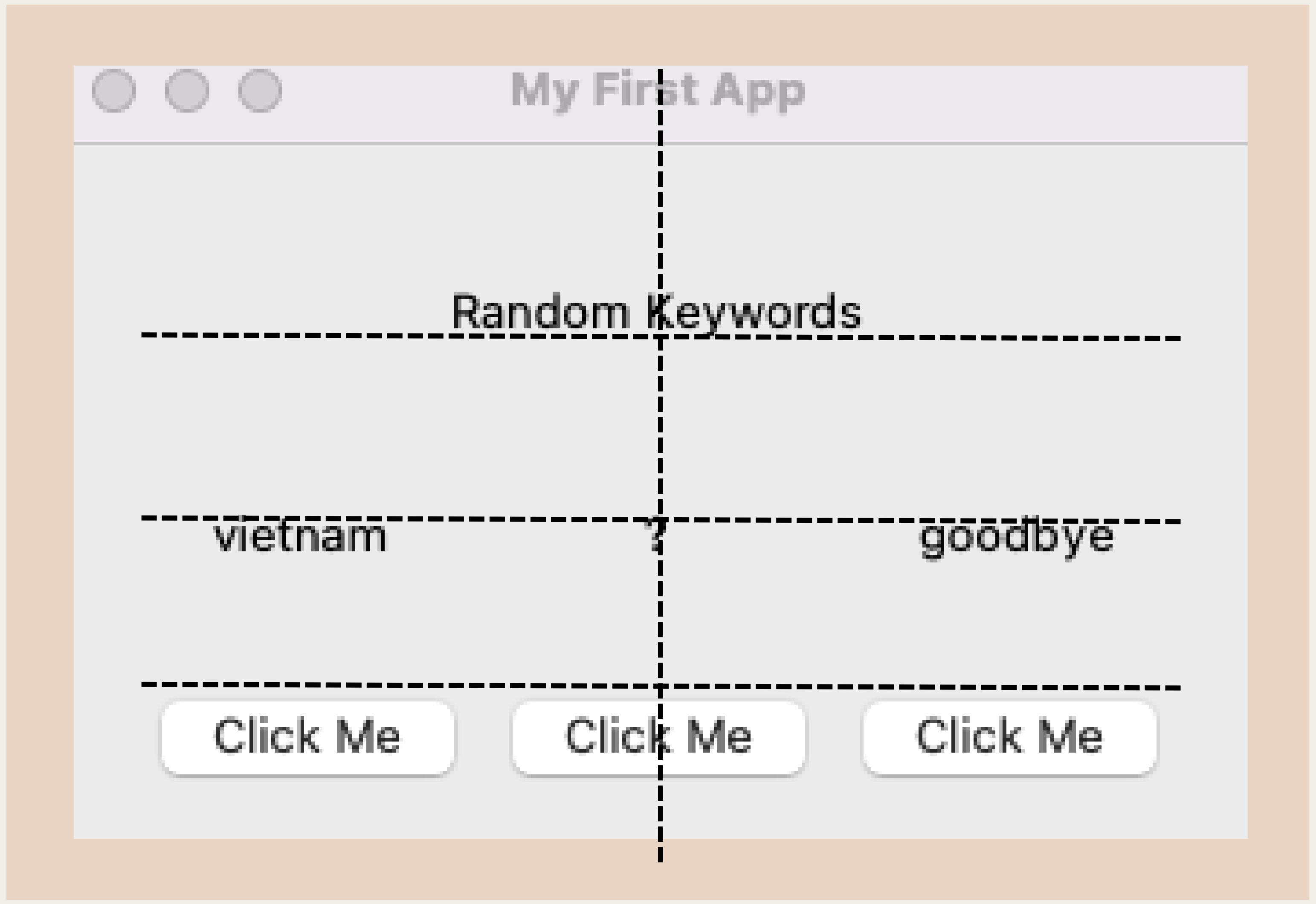
## DESIGNING OUR APP

```
#Design our app
master_layout = QVBoxLayout()
row1 = QHBoxLayout()
row2 = QHBoxLayout()
row3 = QHBoxLayout()

#Previous Code Here


master_layout.addLayout(row1)
master_layout.addLayout(row2)
master_layout.addLayout(row3)

main_window.setLayout(master_layout)
```



## DESIGNING OUR APP

ZERO TO KNOWING

```
#Design our app
master_layout = QVBoxLayout()
row1 = QHBoxLayout()
row2 = QHBoxLayout()
row3 = QHBoxLayout()

#Previous Code Here

master_layout.addLayout(row1)
master_layout.addLayout(row2)
master_layout.addLayout(row3)

main_window.setLayout(master_layout)
```

Creating a **master layout, everything is added to this**

Creating **3 rows**, we will add our objects/widgets to each row

Here we **add our rows to our master layout**, in the order we want them to appear

We **use.addLayout method** to **add "design layouts"** together, **don't use .addWidget**

We **take our master layout** and **set it as the final design to** our **main_window**

## The Code Burger

1. All **Imports**
2. Main App Objects and **Settings**
3. **Create all Widgets** needed in App
4. **Design** your Layout, add your widgets to the screen
5. **Set the final layout** to the Main window
6. **Show** and **Execute** your app

## Displaying a Random word

ZERO TO KNOWING

```
#Getting a Random Word from a List
from random import choice
my_words= ["Hello", "Goodbye", "Test", "Python", "PyQt", "Code"]

def display_word():
    word = choice( my_words )
    text1.setText( word )
```

The **value of** the variable **word** is a **random word** from the list

choice -> This **takes a list** as an argument and **randomly selects one of the elements**

# Event Handing in PyQt

```
#Linking a Button to a Function
button = QPushButton("Click Me")

def test_function():
    print("This button is working!")
```

Event Type -> Click Event

```
button.clicked.connect(test_function)
```

When this button   is Clicked   I want to connect   To this Function

# Event Handing in PyQt

```python
#Linking a Button to a Function
button = QPushButton("Click Me")

def test_function():
    print("This button is working!")
```

Event Type -> Click Event

```python
button.clicked.connect(test_function)
```

When this **button** is **Clicked** I want to **connect** To this **Function**

(Literal Translation)