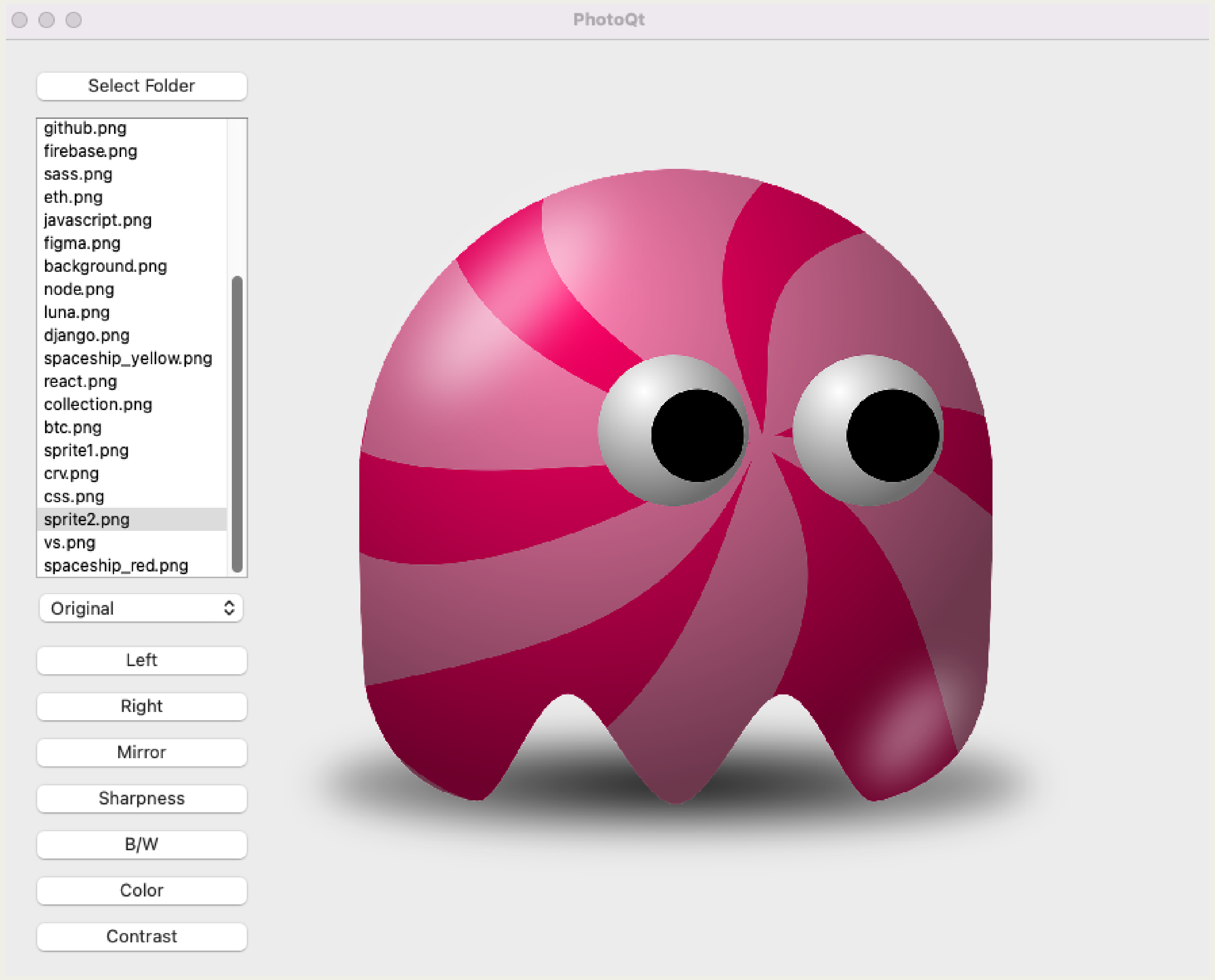
IMAGE EDITING APP

Combine multiple modules to build an interactive photo editing app with Python

ZERO TO KNOWING

App Overview:

What Widgets do you see?



Let's take a look at the App we will be building:

We can:

-**Select** an **image** from the list

-Apply **filters**

-**Edit** our photos and

-**Save** the edited version

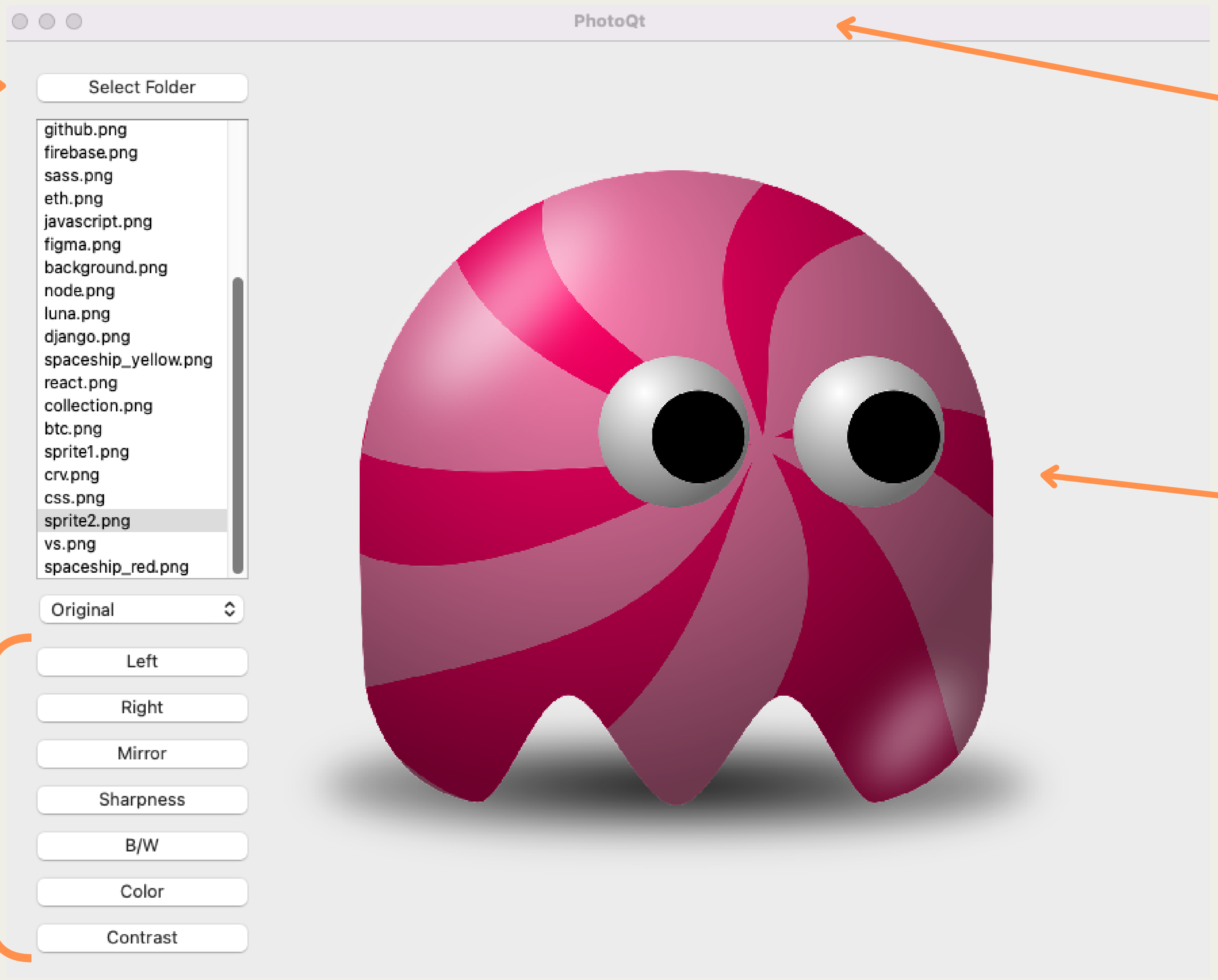
App Overview:

QPushButton

QListWidget

QComboBox

QPushButton

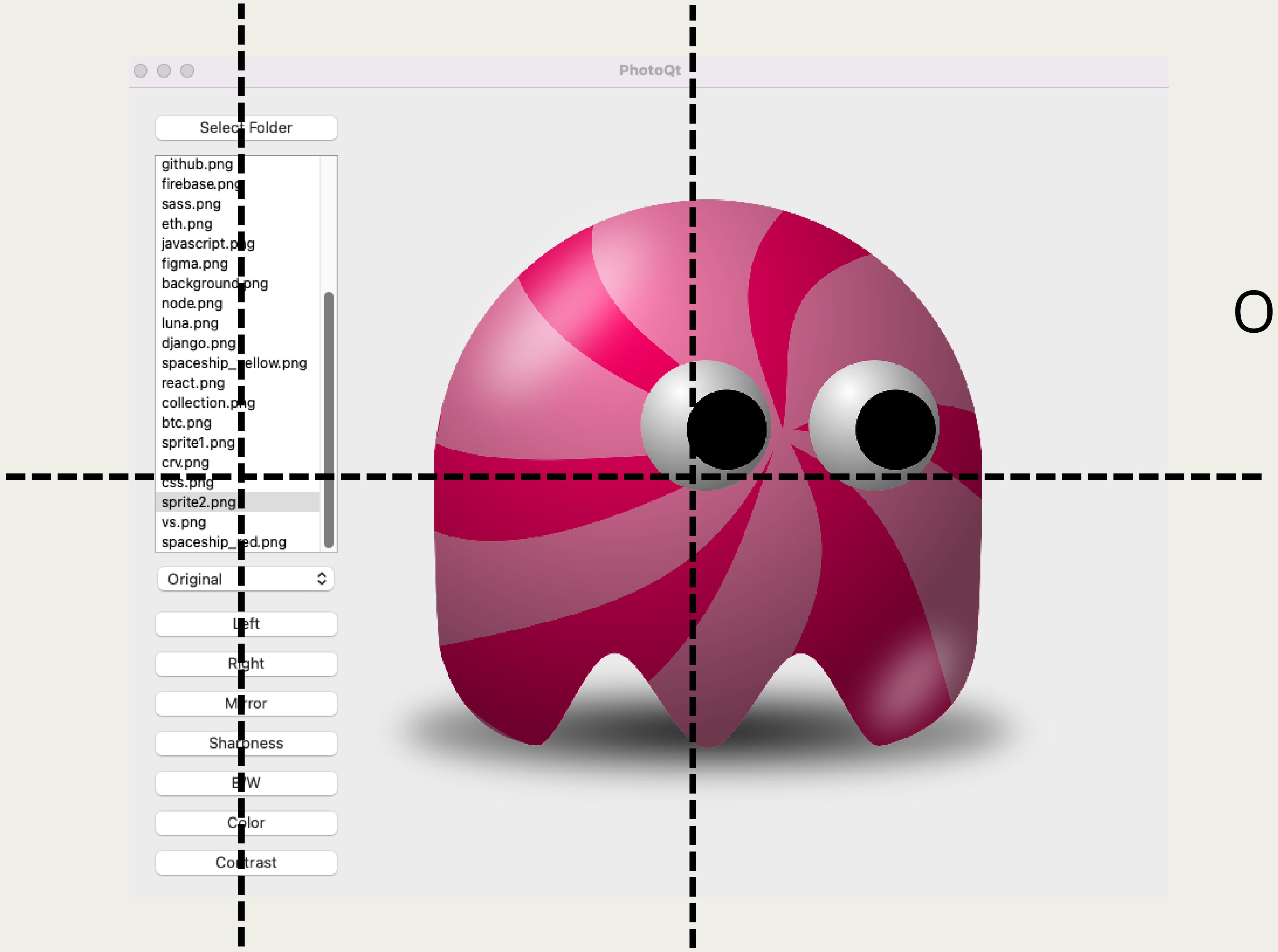


QWidget

QLabel

App Design:

ZERO TO KNOWING



Our design is nothing complicated.

We have a **master_row**

We have **two Columns**

QVBoxLayout
QHBoxLayout

Working with Multiple Modules in One App

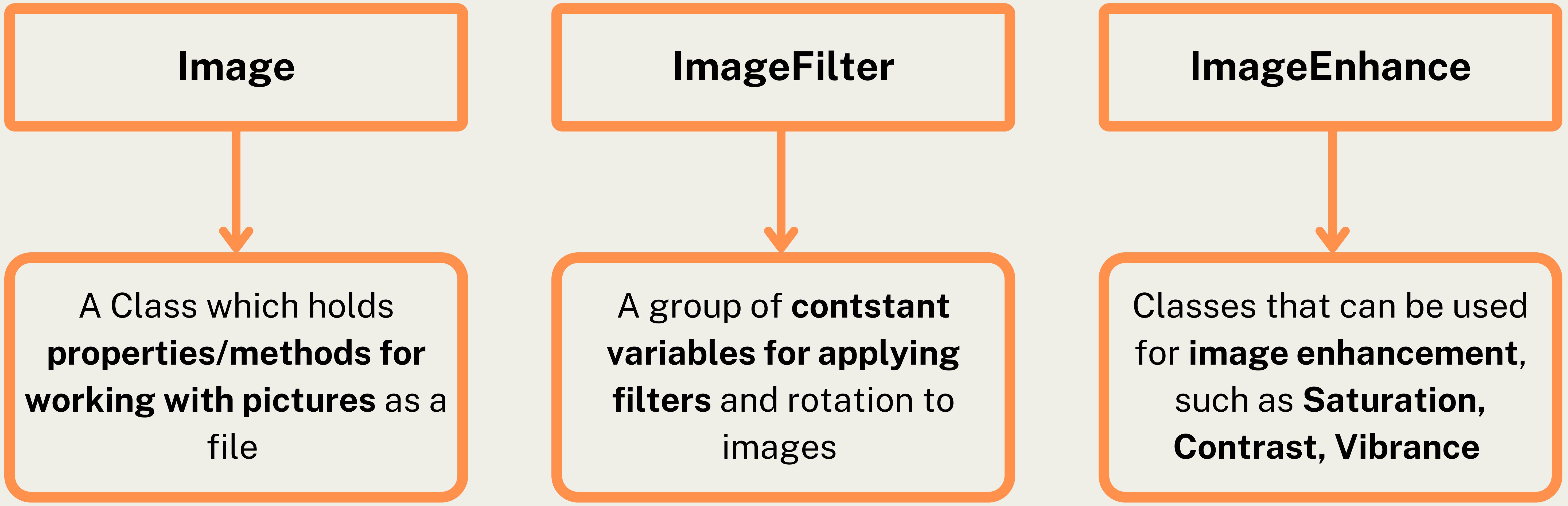
PyQt & PIL ~ Python Image Library

ZERO TO KNOWING

PIL Library

Python Image Library - Allows us to **work with real photos in Python**. Giving us the ability to edit a photo similar to basic photoshop

***Notice PIL is similar to PyQt in the aspect that PIL has **many modules within the main module** itself. We can **only import what we need**



Image

A Class which holds **properties/methods for working with pictures** as a file

ImageFilter

A group of **contstant variables for applying filters** and rotation to images

ImageEnhance

Classes that can be used for **image enhancement**, such as **Saturation, Contrast, Vibrance**

PIL Library

Some of the code we will use

Code Structure	**What it does**
`from PIL import Image, ImageFilter, ImageEnhance`	Structure to **import the Classes** we will use in this app
`black_white = pic.convert("L")`	**Converts** an image called **pic into a black and white picture**, stored in the variable black_white
`sharp = pic.filter(ImageFilter.SHARPEN)`	**Sharpens an image called pic** and stores it in a variable called sharp
`saturation = ImageEnhance.Color(pic).enhance(1.2)`	**Adds saturation** to an image called pic **by 1.2x**
`mirror = pic.transpose(Image.Flip_LEFT_RIGHT)`	**Mirrors an image** called pic
`edited_picture.save("edit.jpg")`	The **save** method allows us to **save the newly editied photo**

Basic Code Structure

ZERO TO KNOWING

```
from PIL import Image, ImageFilter, ImageEnhance

with Image.open("selfie.jpg") as pic :
    pic.show()

    saturate = ImageEnhance.Color( pic )
    saturate = saturate.enhance(1.2)
    saturate.show()

    black_white = pic.convert("L")
    black_white.show()


    mirror = pic.transpose(Image.FLIP_LEFT_RIGHT)
    mirror.show()
```

Import needed Classes from PIL

Using the python open function to open a photo names "selfie.jpg" and giving it a nickname, pic

Adding Saturation to the image then showing the results

Converting the Image to a Black & White picture then showing

Mirroring the already enhanced picture then showing the results

Did your pictures save?

Basic Code Structure

ZERO TO KNOWING

```
from PIL import Image, ImageFilter, ImageEnhance

with Image.open("selfie.jpg") as pic :
    pic.show()

     saturate = ImageEnhance.Color( pic )
     saturate = saturate.enhance(1.2)
     saturate.show()

     black_white = pic.convert("L")
     black_white.save("gray_pic.jpg")
     black_white.show()


    mirror = pic.transpose(Image.FLIP_LEFT_RIGHT)
    mirror.show()
```

We created a **nickname**, **pic**. We can now use "pic" as a **variable.**

pic holds the value of selfie.jpg

Remember to add **.save()** the photos you want to save!

Spend the next 10 minutes to experiment with PIL

Implementing PIL in our PhotoQt App

Creating a Class based Application

Create a File Path to an Image

ZERO TO KNOWING

- We need to **create a Function** to **filter** through a <u>list of files and a list of extensions</u>. **For every File**, **If the File has one of the correct extensions** we want to **add it to a new list**

  Hint ->
  ```
  if filename.endswith(extension):
          results_list.append( filename )
  ```

- **Create a Function** that **gets the current working directory** (work_directory = QFileDialog.getExistingDirectory()). This function needs to **call the filter function** we just made and **add the filenames to our QListWidget** on our App screen

  Hint-> `filenames = filter(os.listdir(work_directory), extensions)`

Example Path: /Users/josh/Desktop/folder_one/main.py

Creating our Class

```
class Editor():
    def __init__(self):
        self.pic = None
        self.original_pic = None
        self.file = None
        self.save_folder = "edits/"

    def loadImage(self, file):
        self.file = file
        fullname = os.path.join(work_directory, file)
        self.pic = Image.open(fullname)
        self.original_pic = self.pic.copy()
```

We define **4 Properties** in our __init__ method, **Assigning each Value**

We give them the **Value of None**, because they **will have a value as the program runs**

This method is responsible for loading our images.

We **give this a file** as an argument. We then **join together** the **current directory** and our **file**

Finally, now that **we have a Path**, we can **Open the Picture**

```
/Users/josh/work_directory/file
/Users/josh/Desktop/pictures_folder/selfie.jpg
```

Key Methods

```
def saveImage(self):
    path = os.path.join(work_directory, self.save_folder)
    if not(os.path.exists( path ) or os.path.isdir( path )):
        os.mkdir( path )
    fullname = os.path.join(path, self.file)
    self.image.save(fullname)

def showImage(self, path):
    picture_box.hide()
    image = QPixmap( path )
    w, h = picture_box.width(), picture_box.height()
    image = image.scaled(w, h, Qt.KeepAspectRatio)
    picture_box.setPixmap( image )
    picture_box.show()
```

When we want to **Save an Image**, we must first **check if** that **path already exisits or is the current directory**

If it does not exist we can **save** that new picture

This path is acutally the image we want to load into the App

We have an Image, we need to bring it into our PyQt App

We use **QPixmap from QtGui** giving it the **path** (image)

We **scale the image** to fit the **width and height** of our screen

Finally, **set the image** to the screen with **.setPixmap()**

/Users/josh/work_directory/save_folder/file

/Users/josh/Desktop/pictures_folder/editied/gray_selfie.jpg

Methods of Editing

ZERO TO KNOWING

```
def do_color(self):
    self.image = ImageEnhance.Color(self.image).enhance(1.2)
    self.saveImage()
    image_path = os.path.join(work_directory, self.save_folder, self.file)
    self.showImage( image_path )
```

We have worked with these PIL values and classes before

Instead of using "pic" we can **now use self.image** within all these editing

These 3 lines of Code will be the **same for every edit method** you create in this App.

1. We always want to **save our image** -> self.saveImage()
2. We always want to create a **new image_path** from our new image
3. Allowing us to **display the new image** on our screen

Can you program the other methods yourself?

Event Handling

```
def displayImage():
    if file_list.currentRow() >= 0:
        filename = file_list.currentItem().text()
        main.loadImage( filename )
        main.showImage( os.path.join( work_directory, main.filename ))


main = Editor()

btn_folder.clicked.connect( chooseDirectoryAndDisplayFiles )

file_list.currentRowChanged.connect( displayImage )

btn_color.clicked.connect(main.do_color)
```

This function is responsible for **loading in the image** of the **filename** that was clicked in out **QListWidget**

Basics of Lambda

Creating Anonymous Functions in Python

ZERO TO KNOWING

Lambda Explained

Lambda is a special feature that allows you to create **anonymous functions** on the go.

An **anonymous function** is a special kind of function that **doesn't have a name**. It's a bit like a secret helper that **can perform a specific task** for you **without** needing a **name**

Normally, we give functions a name so we can use them later, But an **anonymous functions**, we don't give it a name.

We create it right where we need it and **use it immediately!**

Lambda Code

```
cube = lambda  x : x ** 3
result = cube(4)
print( result )
```

```
multi_numbers = lambda x, y: x * y
res = multi_numbers(3, 5)
print(result)
```

Lambda Code

ZERO TO KNOWING

```
cube = lambda  x : x ** 3
result = cube(4)
print( result )
```

We have a **Lambda** function -> cube

It takes a **single Parameter** -> x

It **calculates** the cube of x

```
multi_numbers = lambda x, y: x * y
res = multi_numbers(3, 5)
print(result)
```

Lambda Code

```python
cube = lambda x: x ** 3
result = cube(4)
print( result )
```

We have a **Lambda** function -> cube

It takes a **single Parameter** -> x

It **calculates** the cube of x

```python
multi_numbers = lambda x, y: x * y
res = multi_numbers(3, 5)
print(result)
```

Imagine you have a math problem to solve, like **multiplying two numbers** together

Instead of writing a separate named function, you can **create an anonymous function**

To **solve that specific problem** for you

Lambda Code

ZERO TO KNOWING

```
cube = lambda x: x ** 3
result = cube(4)
print( result )
```

```
Output in Terminal
64
```

We have a **Lambda** function -> cube

It takes a **single Parameter** -> x

It **calculates** the cube of x

```
multi_numbers = lambda x, y: x * y
res = multi_numbers(3, 5)
print(result)
```

```
Output in Terminal
15
```

Imagine you have a math problem to solve, like **multiplying two numbers** together

Instead of writing a separate named function, you can **create an anonymous function**

To **solve that specific problem** for you

Using Lambda in our App

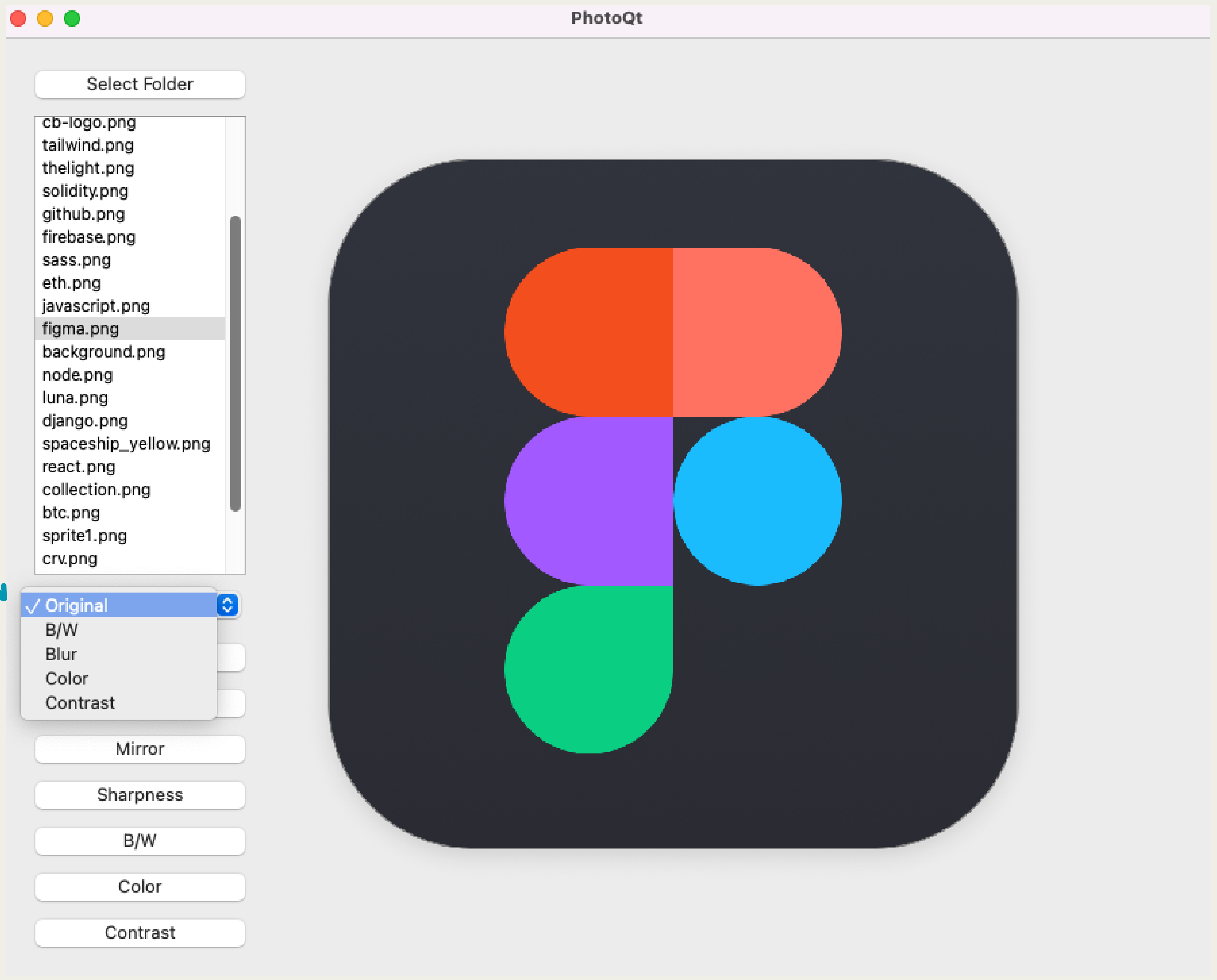
Our mission with **Lambda**

Create a Lambda (anonymous) **function for each filter in** our **QComboBox** (B/W, Blur, Color, Contrast)

How could we do this with a Dictionary?

dictionary[key] = value

dictionary **key unlocks** a **value**



Using Lambda in our App

Create an "**Instant**" Anonymous Lambda **Function to be executed instantly**

The function is **called Lambda**

The Function takes **One Parameter** - **image**

```
filter_mapping = {
        "B/W" : lambda image: image.convert( "L" ),

          Key                 Value

        "Blur" : lambda image: image.filter( ImageFilter.BLUR),
        "Color" : lambda image: ImageEnhance.Color( image ).enhance( 2 ),
        "Contrast" : lambda image: ImageEnhance.Contrast( image ).enhance( 2 )
        # You can keep adding more here }
```

Using Lambda in our App

ZERO TO KNOWING

```
def apply_filter( self, filter_name ):
    if filter_name == "Original":
        self.image = self.original_image.copy()
    else:
        filter_mapping = {
            "B/W" : lambda image: image.convert( "L" ),
            "Blur" : lambda image: image.filter( ImageFilter.BLUR),
            "Color" : lambda image: ImageEnhance.Color( image ).enhance( 2 ),
            "Contrast" : lambda image: ImageEnhance.Contrast( image ).enhance( 2 )
            # You can keep adding more here
        }
        filter_function = filter_mapping.get( filter_name )
        if filter_function:
            self.image = filter_function( self.image )
            self.saveImage()
            image_path = os.path.join(work_directory, self.save_folder, self.filename)
            self.showImage( image_path )
        pass
```

Given a Name from QComboBox

Use the **original copy** of our image

#Add the final 3 lines of code, same as the methods we made for the editing tools

Using Lambda in our App

ZERO TO KNOWING

```
def apply_filter( self, filter_name ):
    if filter_name == "Original":
        self.image = self.original_image.copy()
    else:
        filter_mapping = {
            "B/W" : lambda image: image.convert( "L" ),
            "Blur" : lambda image: image.filter( ImageFilter.BLUR),
            "Color" : lambda image: ImageEnhance.Color( image ).enhance( 2 ),
            "Contrast" : lambda image: ImageEnhance.Contrast( image ).enhance( 2 )
            # You can keep adding more here
        }
       filter_function = filter_mapping.get( filter_name )
       if filter_function:
           self.image = filter_function( self.image )
           self.saveImage()
           image_path = os.path.join(work_directory, self.save_folder, self.filename)
           self.showImage( image_path )
       pass
```

Given a Name from QComboBox

Use the **original copy** of our image

Insert our Dictionary

The **value of filter_function** is **whatever** our **Lambda** function **returns**

if Lambda returned something, then **show what Lambda returned**

#Add the final 3 lines of code, same as the methods we made for the editing tools

Lambda Challenge

How can you make your App more efficient with Lambda?

Lambda Project Challenge

ZERO TO KNOWING

CONVERT THIS

```
def do_bw(self):
    self.image = self.image.convert("L")
    self.saveImage()
    image_path = os.path.join(work_directory, self.save_folder, self.filename)
    self.showImage( image_path )

def do_flip(self):
    self.image = self.image.transpose(Image.FLIP_LEFT_RIGHT)
    self.saveImage()
    image_path = os.path.join(work_directory, self.save_folder, self.filename)
    self.showImage( image_path )

def do_color(self):
    self.image = ImageEnhance.Color(self.image).enhance(1.2)
    self.saveImage()
    image_path = os.path.join(work_directory, self.save_folder, self.filename)
    self.showImage( image_path )
```

TO THIS

Can you replace all of our button methods?

How can we do this using Lambda?

Create a new method - transformImage

This method will hold a Dictionary with all our Lambda function