GSOC Raspberry Pi - PHOTOBOOTH project

Last updated June 3, 2020

Congratulations on setting up the Raspberry Pi !!!

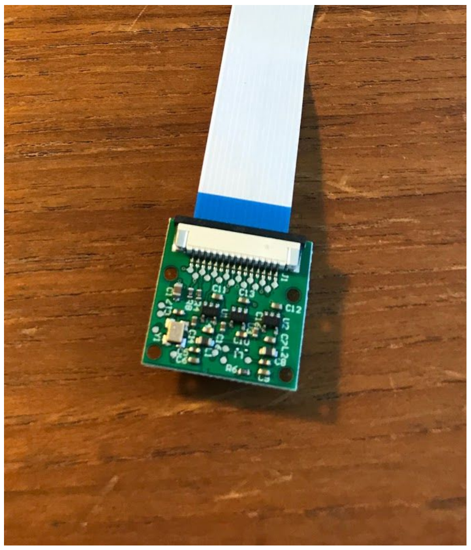
Let's work on a fun project! We'll learn how to add a camera to the Raspberry Pi, and test the camera, and then write a Python photobooth program to control the camera, along with a Python slideshow program to display our photobooth pictures.

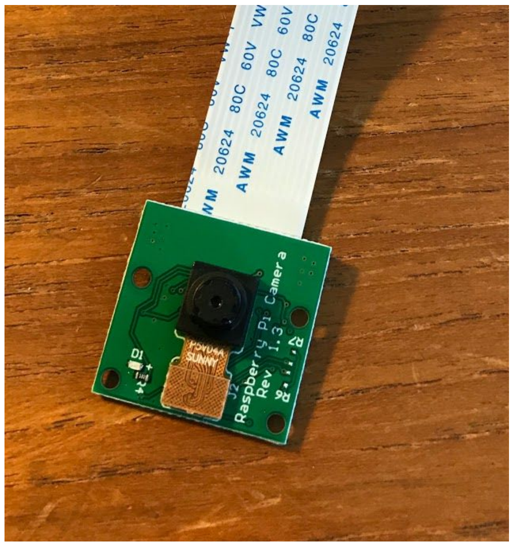
After setting up your Raspberry Pi, go ahead and pull the small white box labeled CAMERA from within the GSOC Raspberry Pi Box. That's the only "hardware" we will need for this project. You'll also need some tape (scotch or blue tape) or alternatively a little bit of playdough to attach the camera board to the monitor.

STEP #1 Attach the camera to the Raspberry PI

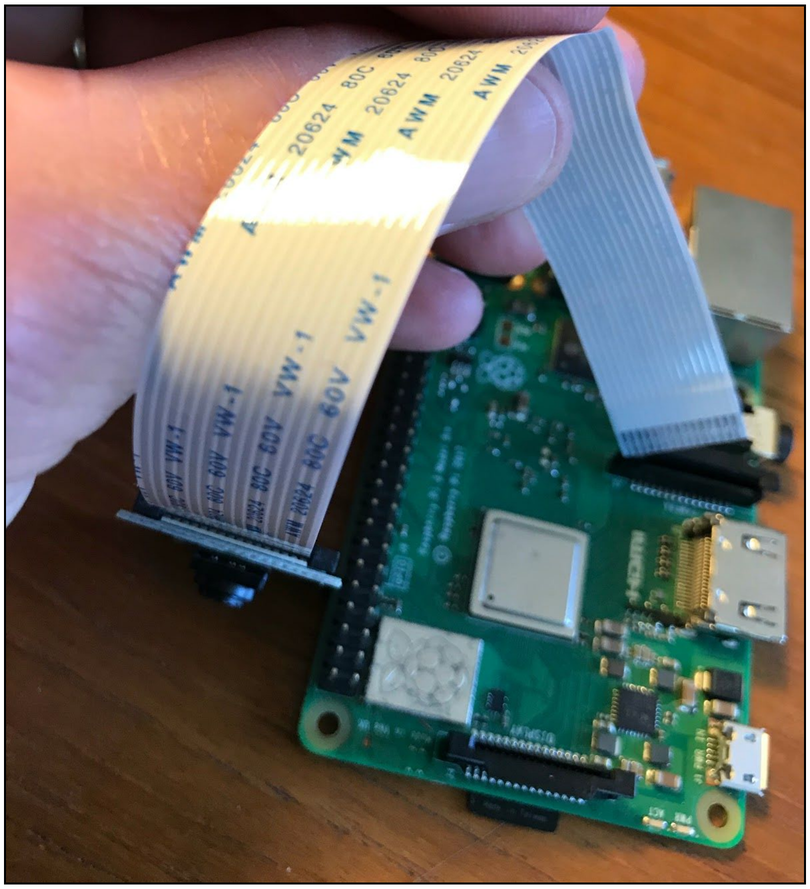
IMPORTANT! You need to power down (i.e. be sure the micro USB power cable is OUT of the Raspberry Pi) before attaching the camera!

Each end of the camera ribbon cable is "blue" on one side and has a shiny silver connector on the other side. The left photo shows the blue side connected to the "circuit side" of the camera module. The right photo shows the other side of the camera module, which is the lens of the camera.

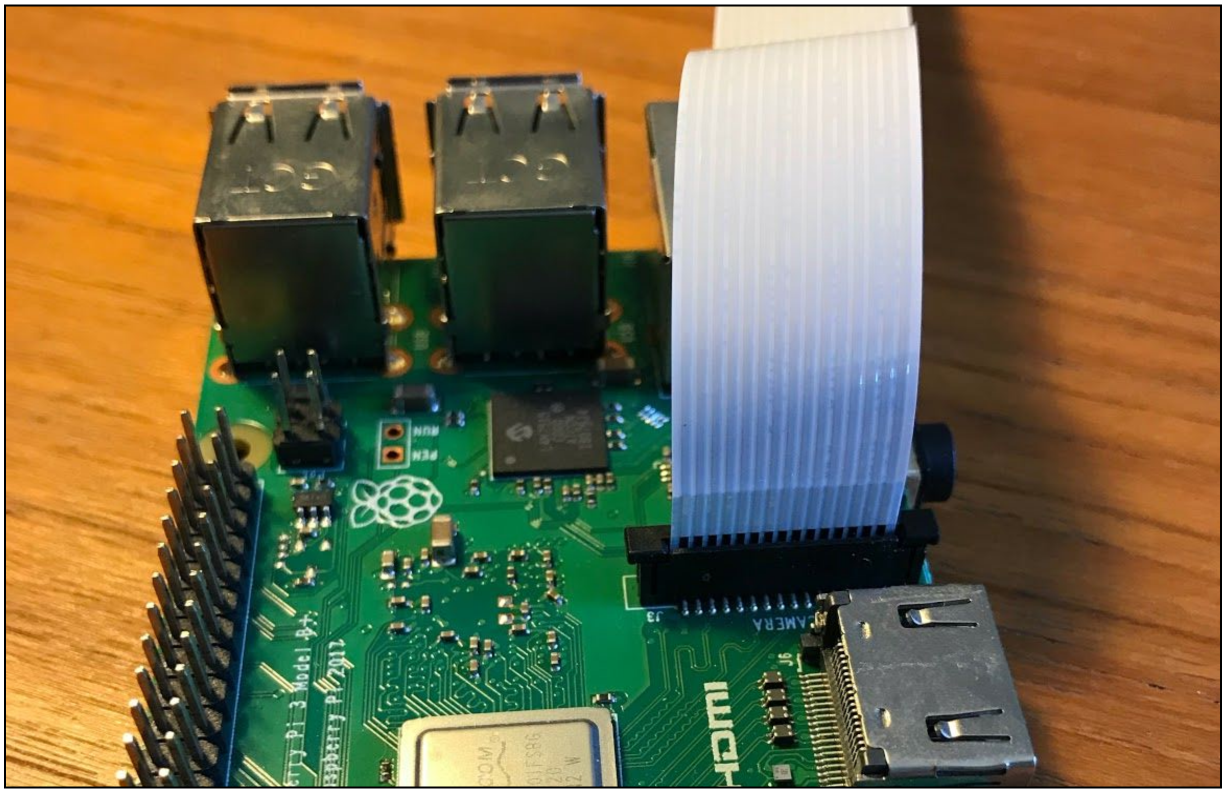




There are two black connector ports on the Raspberry Pi board, one on the side opposite of the USB ports (labeled DISPLAY) and the other next to the HDMI port (labeled CAMERA). Gently pull up the edges of the CAMERA port's black plastic clip, and slide the camera ribbon cable into the port as shown below so that the shiny side faces the Raspberry Pi's HDMI port.



Press down gently on the black plastic clip to lock the camera ribbon cable in place. Lightly tug up on the ribbon cable to be sure it is secure. You should be able to see a tiny bit of the silver connectors sticking out above the black connector port as shown below.



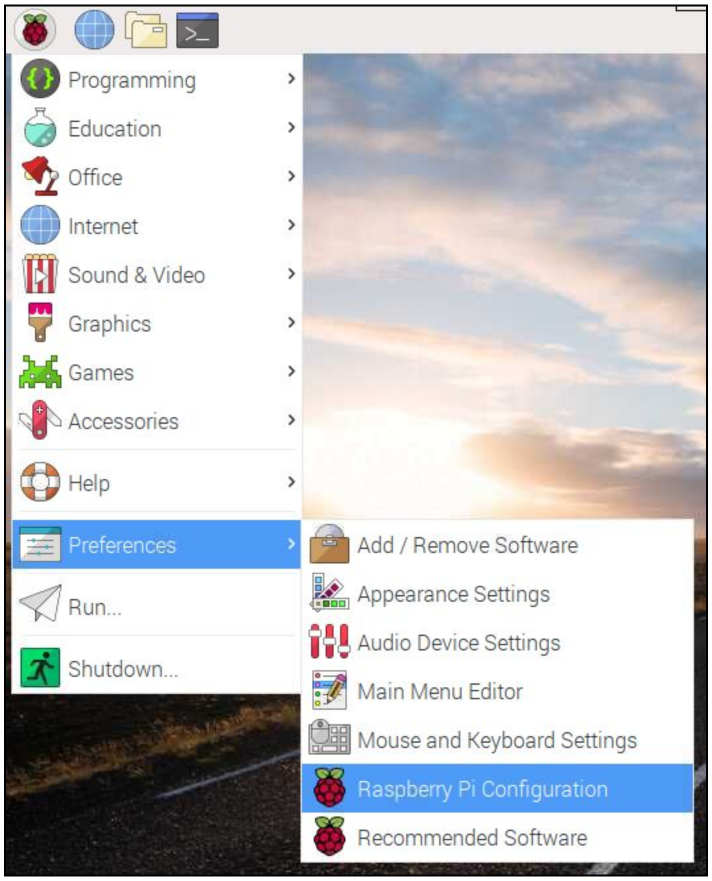
STEP #2 Carefully position the Raspberry Pi camera and micro USB cable

Grab your tape (or playdough) and attach the camera board to the monitor (or something else) so the black camera lens is facing the direction you wish to take photos. You can now go ahead and attach the micro USB power cable to your Raspberry Pi as well at this point and power up the Raspberry Pi.

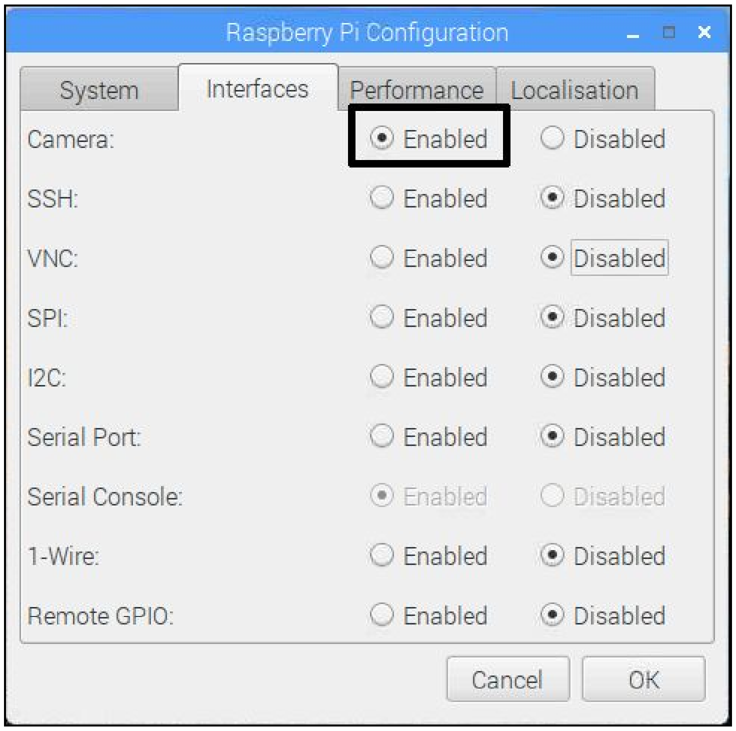
You should notice a small red LED on the camera board flash momentarily as the Raspberry Pi powers up. This confirms that the camera board is connected to the Raspberry Pi and receiving power.

STEP #3 configure the Raspberry Pi for the camera (and then reboot)

You'll need to verify that this Raspberry Pi has been configured to use the camera. Click on the Raspberry in the upper left corner of the screen to open the system menu, then select the menu items Preferences and Raspberry Pi Configuration menu items as shown below.



Next, click on the INTERFACES tab at the top of the Raspberry Pi Configuration dialog, and be sure the Camera interface radio button is set to Enabled as shown below.

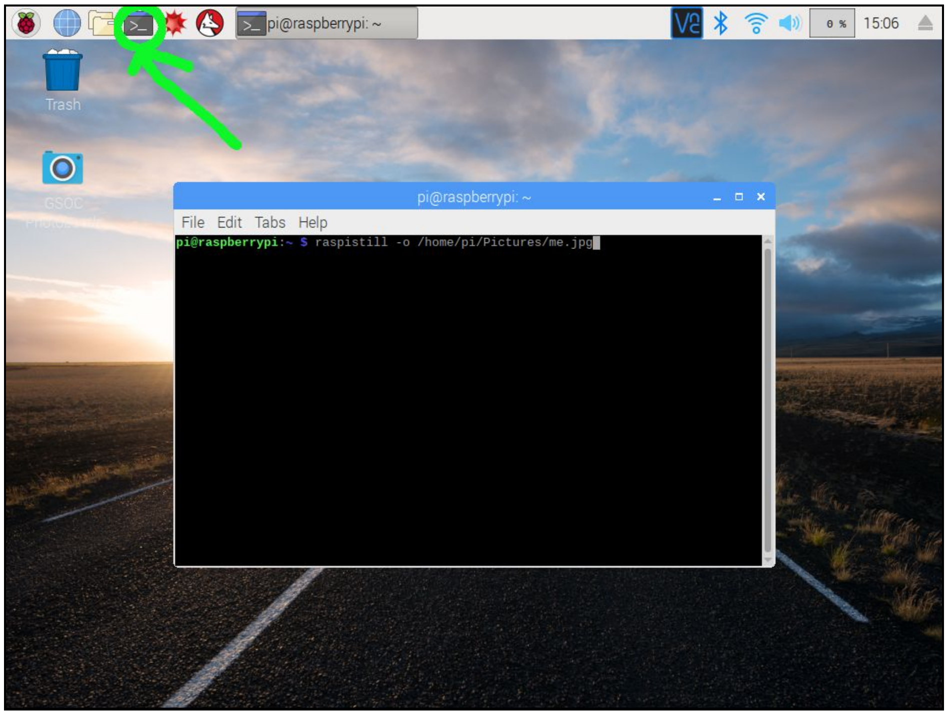


Camera:	[✓ Enabled]	Disabled
SSH:	Enabled	[✓ Disabled]
VNC:	Enabled	[✓ Disabled]
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial Port:	Enabled	[✓ Disabled]
Serial Console:	[✓ Enabled]	Disabled
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]

You will need to select OK to exit this dialog. If you had to change the Camera interface to Enabled, you will additionally need to REBOOT the Raspberry Pi before continuing. Select SHUTDOWN from the system menu and then REBOOT.

STEP #4 use the raspistill program to take your picture!

Click on the terminal icon (little black window) on top left of your Raspberry Pi screen to open a terminal window.



```
pi@raspberrypi:~ $ raspistill -o /home/pi/Pictures/me.jpg
```

Type the following command at the terminal prompt:

```
raspistill -o /home/pi/Pictures/me.jpg
```

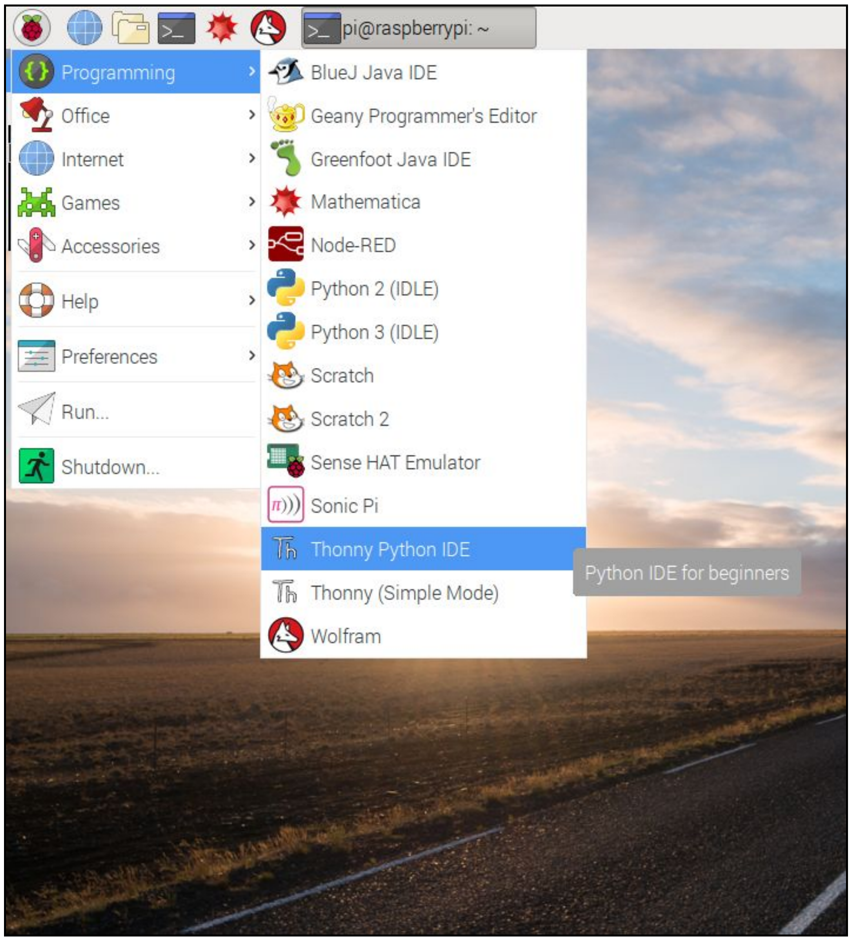
… and then press the enter key to execute the command. The red LED on the camera board will light up for a few seconds, after which a picture will be taken! The picture is saved to the file ME.JPG within the Raspberry Pi's /home/pi/Pictures folder. Use the folders tool on the Raspberry Pi's toolbar (to the left of the terminal tool) to navigate to the /home/pi/Pictures folder and click on the file ME.JPG to see your picture!

You can take pictures one at a time this way, but it's way more fun (and more interesting) to make your own PHOTOBOOTH program.

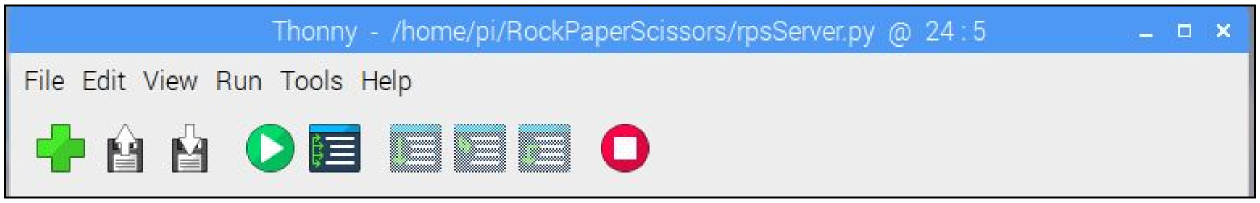
BUILD YOUR OWN PHOTOBOOTH with PYTHON (recommended for Cadettes +)

STEP #1 open up the Thonny Python IDE

IDE = "Integrated Development Environment". Click on the Thonny Python IDE menu item as shown below.



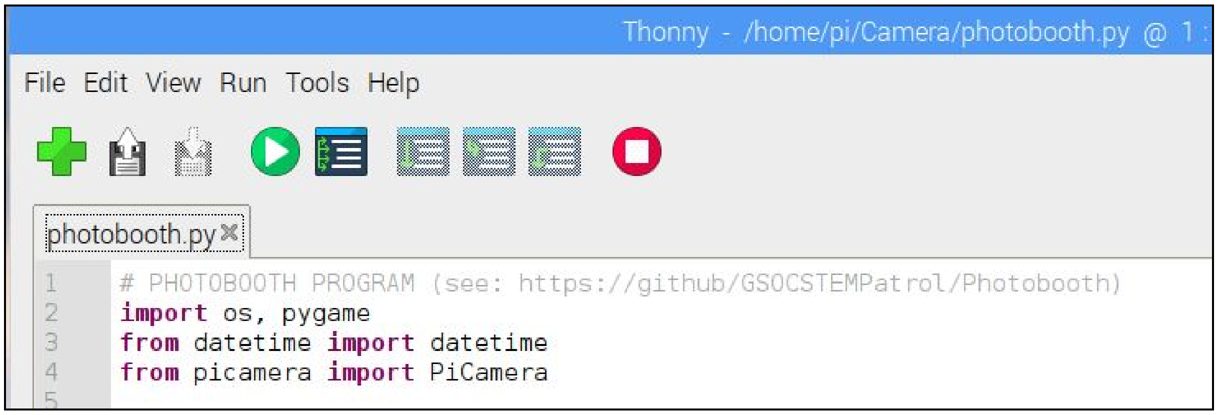
Click on the Thonny IDE's NEW (the green plus sign) button as shown to create a new python coding file:



Thonny - /home/pi/RockPaperScissors/rpsServer.py @ 24 : 5
File Edit View Run Tools Help

Now click on the Thonny IDE's SAVE button (third icon, looks like a diskette with a DOWN arrow on top) and save your program file as "photoboothFL" **where F is the first character of your**

first name and L is the first character of your last name. The name photoboothFL.py will appear in the Thonny IDE window tab as shown in the following picture.

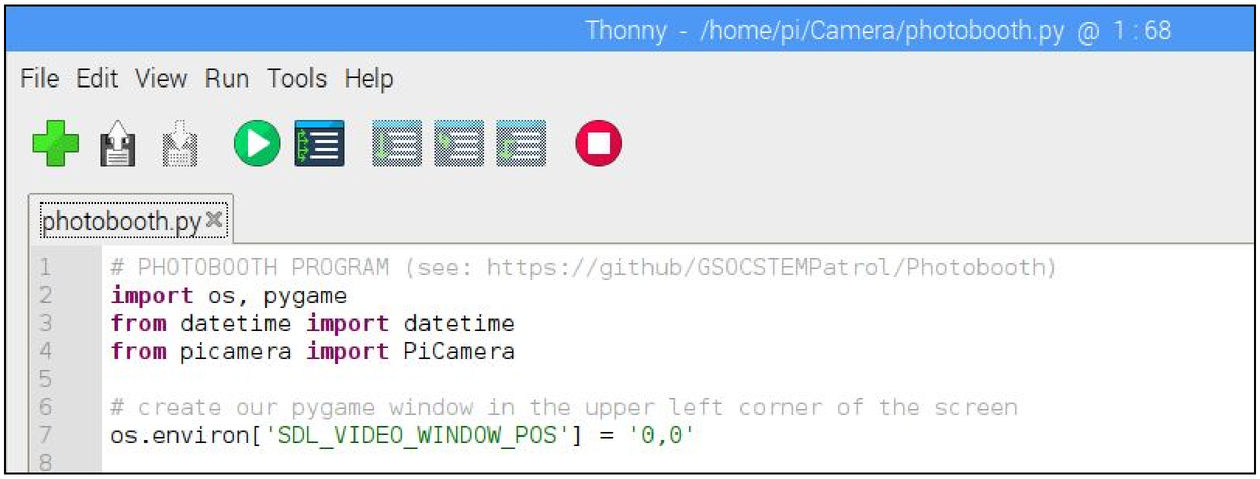

Thonny - /home/pi/Camera/photobooth.py @ 1 :
File Edit View Run Tools Help
photobooth.py

```
# PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
import os, pygame
from datetime import datetime
from picamera import PiCamera
```

Python allows you to leverage the work of others (or yourself) by including modules, or other pieces of code, within your program. Almost every python program starts out this way!

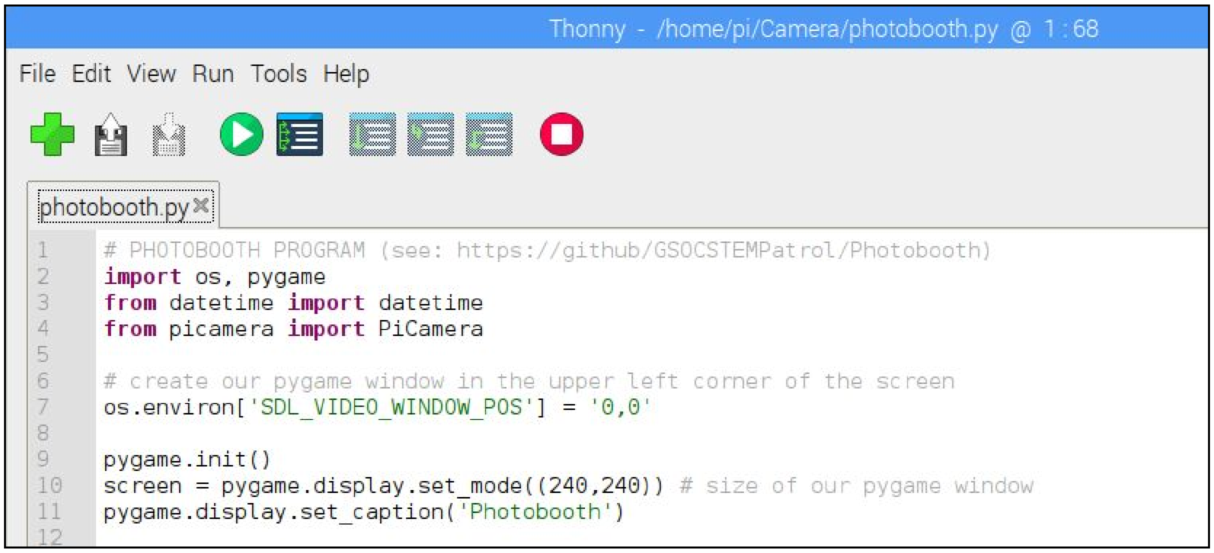
Carefully copy the four lines shown above into your Thonny IDE editor window.

The first line is simply a comment to help us identify the program. Next, we import the “os” (operating system) module, the pygame module which we’ll use to create a countdown window, the datetime module which we’ll use to create the filenames for the pictures we take, and finally the picamera module, which contains the methods (i.e. “code”) we need to control our Raspberry Pi camera.


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

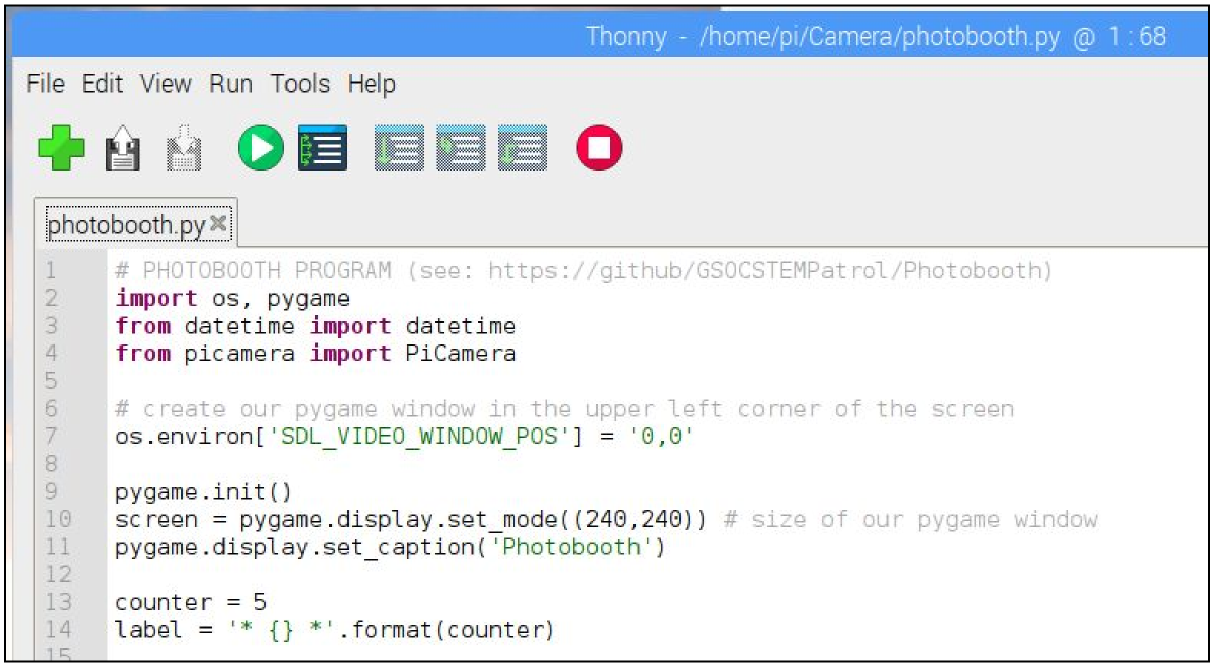
```
1  # PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2  import os, pygame
3  from datetime import datetime
4  from picamera import PiCamera
5
6  # create our pygame window in the upper left corner of the screen
7  os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
8
```

Within python, any text following the # symbol is a non-executable COMMENT. On line 6, we’ve left a comment to ourselves to remind us that the code on line 7 (os.environ) is used to specify where our pygame window is drawn on our screen; in this case, the upper left most corner of our screen (position 0,0 is the upper left corner of the screen).


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

```
# PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
import os, pygame
from datetime import datetime
from picamera import PiCamera

# create our pygame window in the upper left corner of the screen
os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'

pygame.init()
screen = pygame.display.set_mode((240,240)) # size of our pygame window
pygame.display.set_caption('Photobooth')
```

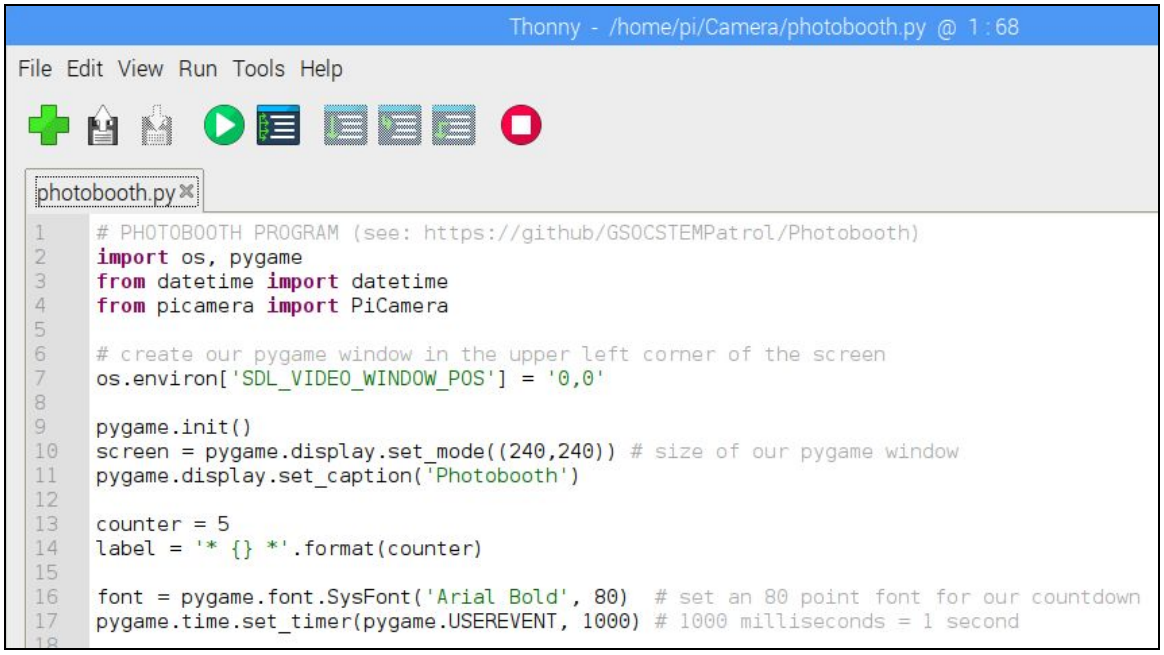
On line 9 we initialize the pygame system. We then set the size of our pygame window (240 x 240 pixels) on line 10 and we add a title to our pygame window's title bar on line 11.


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

```
1   # PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2   import os, pygame
3   from datetime import datetime
4   from picamera import PiCamera
5
6   # create our pygame window in the upper left corner of the screen
7   os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
8
9   pygame.init()
10  screen = pygame.display.set_mode((240,240)) # size of our pygame window
11  pygame.display.set_caption('Photobooth')
12
13  counter = 5
14  label = '* {} *'.format(counter)
```

Now we create two new variables for our own use. On line 13, variable *counter* holds our count down value (5, 4, 3, 2, 1 …). On line 14, variable *label* is a string representation of our *counter* numerical variable that we "dress up" by adding asterisks alongside our numerical *counter* variable's value. Within line 14 the { } symbols represent a variable that is set to the value of variable *counter* and displayed between two asterisks.

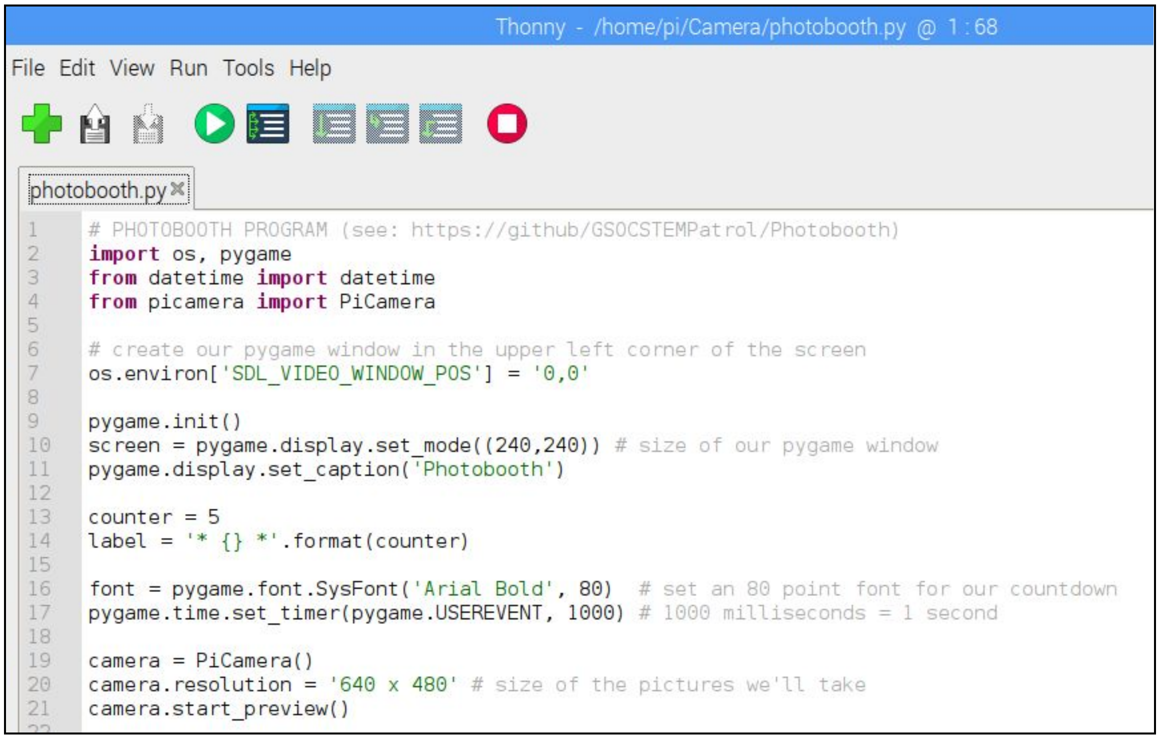
Don't forget to click the SAVE button every time you add a few new lines of code!


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

```
1  # PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2  import os, pygame
3  from datetime import datetime
4  from picamera import PiCamera
5
6  # create our pygame window in the upper left corner of the screen
7  os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
8
9  pygame.init()
10 screen = pygame.display.set_mode((240,240)) # size of our pygame window
11 pygame.display.set_caption('Photobooth')
12
13 counter = 5
14 label = '* {} *'.format(counter)
15
16 font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
17 pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second
```

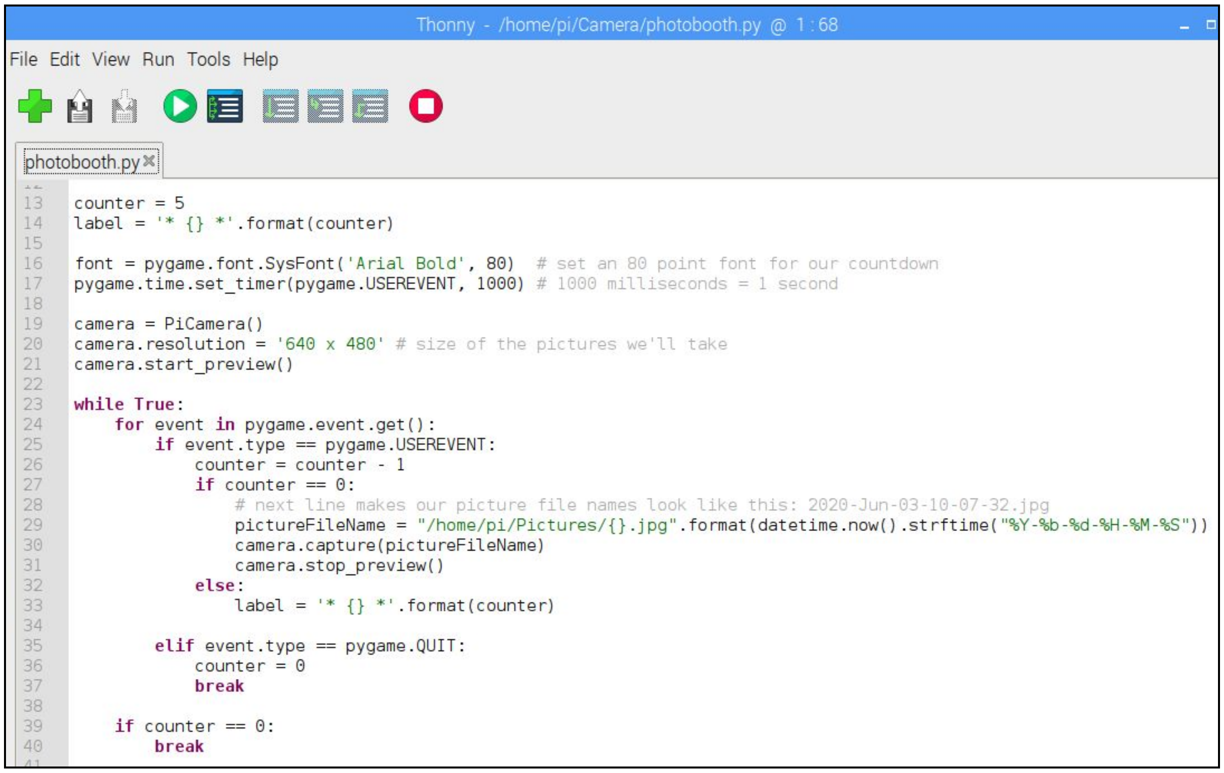
On line 16 we set the font we'll use within our pygame window. On line 17 we create an EVENT that fires off every second (1000 milliseconds is equal to 1 second).

Windowed programs (e.g. Microsoft Windows, the Mac operating system, etc.) use what is called EVENT-based programming; an event loop runs continuously and your program “reacts” to events like the movement of the mouse, mouse clicks, menu selections, and so forth. On line 17 we created our own event that “fires” every second, which we can then “listen” for inside our pygame window's event loop.


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

```
1  # PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2  import os, pygame
3  from datetime import datetime
4  from picamera import PiCamera
5
6  # create our pygame window in the upper left corner of the screen
7  os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
8
9  pygame.init()
10 screen = pygame.display.set_mode((240,240)) # size of our pygame window
11 pygame.display.set_caption('Photobooth')
12
13 counter = 5
14 label = '* {} *'.format(counter)
15
16 font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
17 pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second
18
19 camera = PiCamera()
20 camera.resolution = '640 x 480' # size of the pictures we'll take
21 camera.start_preview()
```

Line 19 creates a camera object variable; we'll use this object to control the Raspberry Pi camera. On line 20 we set the resolution of the pictures we will take, and then on line 21 we start up the camera by putting it into "preview mode".


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

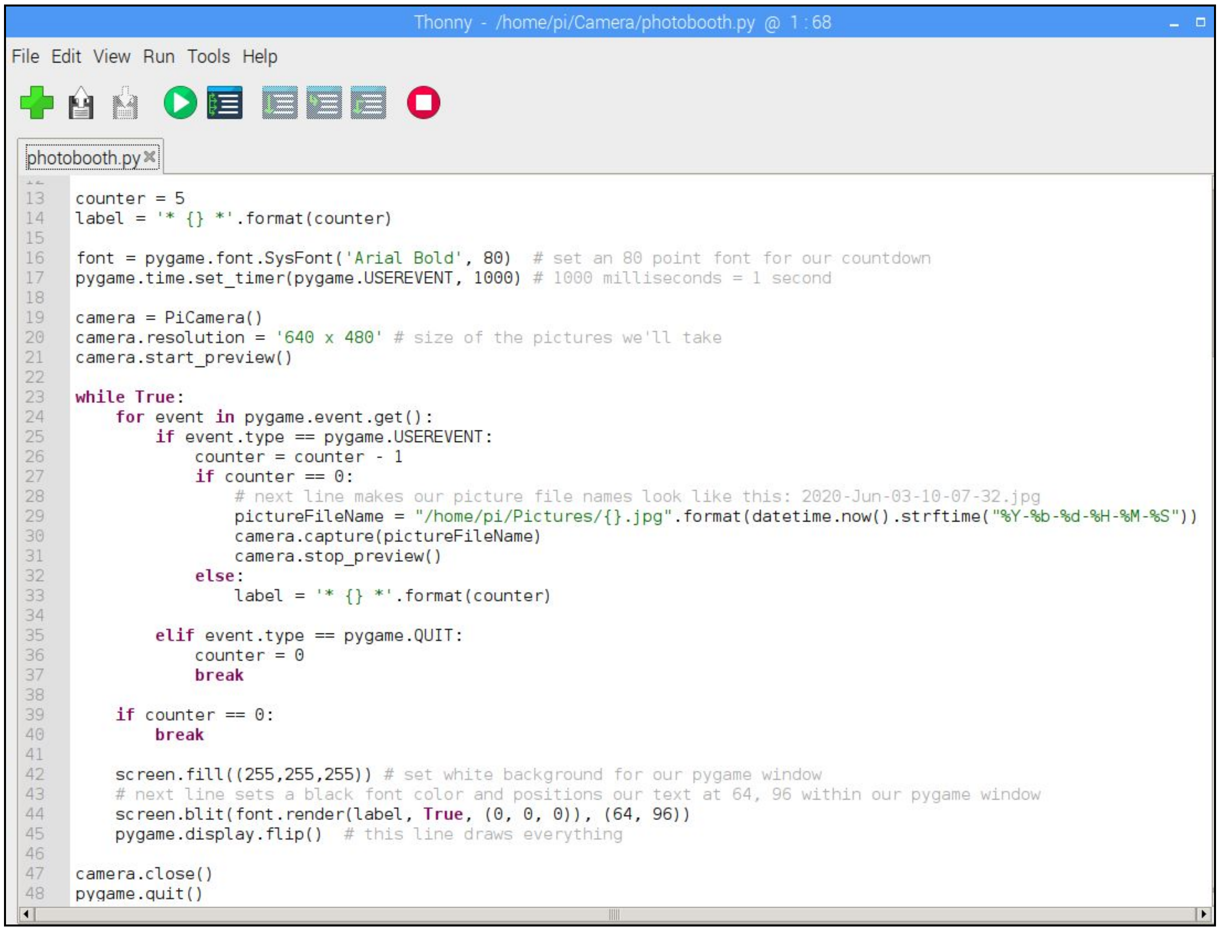
```
13  counter = 5
14  label = '* {} *'.format(counter)
15
16  font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
17  pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second
18
19  camera = PiCamera()
20  camera.resolution = '640 x 480' # size of the pictures we'll take
21  camera.start_preview()
22
23  while True:
24      for event in pygame.event.get():
25          if event.type == pygame.USEREVENT:
26              counter = counter - 1
27              if counter == 0:
28                  # next line makes our picture file names look like this: 2020-Jun-03-10-07-32.jpg
29                  pictureFileName = "/home/pi/Pictures/{}.jpg".format(datetime.now().strftime("%Y-%b-%d-%H-%M-%S"))
30                  camera.capture(pictureFileName)
31                  camera.stop_preview()
32              else:
33                  label = '* {} *'.format(counter)
34
35          elif event.type == pygame.QUIT:
36              counter = 0
37              break
38
39      if counter == 0:
40          break
```

On line 23 we create an "endless loop" by stating that anything we add inside of this loop will run forever, or until we purposely break out of the loop. In python, we indent the lines of code within a loop to signify that the code runs "inside of the loop". Note that lines 24 through 40 are indented beneath the "while True" loop.

Lines 24 through 37 represent the pygame "event loop". Line 23 loops a new variable we name *event* over a list of the events that the pygame window is receiving. This list of events is called the event queue. As an illustration, if you were to move your mouse while our countdown window was open, a "mouse move" event would be added to the pygame window event queue. We need only to concern ourselves with our one-second timer event; on line 25 we specifically look to see if the most recent event in the event queue was of the type of our one-second timer USEREVENT.

If we do receive a USEREVENT, then on line 26 we reduce the value of our variable *counter* (initially set to a value of 5) by a value of one. Once the value of *counter* is equal to zero (line 27), we then create a filename for our picture based upon the current date and time, and we take the picture with the code found on lines 30 and 31. Alternatively, if the value of *counter* is greater than zero, then we simply update our *label* variable to the current value of *counter*. We will display the updated *label* variable in our window. Note that we have also indented lines 28 - 31 beneath our conditional statement "if counter == 0" found on line 27, which is python's convention.

Outside of our event loop (lines 39-40), if the *counter* variable is equal to zero (meaning we know we took the picture) then we add the means to break out of our endless “while True” loop using python’s break command, thereby ending our program.


Thonny - /home/pi/Camera/photobooth.py @ 1 : 68
File Edit View Run Tools Help
photobooth.py

```
13  counter = 5
14  label = '* {} *'.format(counter)
15
16  font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
17  pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second
18
19  camera = PiCamera()
20  camera.resolution = '640 x 480' # size of the pictures we'll take
21  camera.start_preview()
22
23  while True:
24      for event in pygame.event.get():
25          if event.type == pygame.USEREVENT:
26              counter = counter - 1
27              if counter == 0:
28                  # next line makes our picture file names look like this: 2020-Jun-03-10-07-32.jpg
29                  pictureFileName = "/home/pi/Pictures/{}.jpg".format(datetime.now().strftime("%Y-%b-%d-%H-%M-%S"))
30                  camera.capture(pictureFileName)
31                  camera.stop_preview()
32              else:
33                  label = '* {} *'.format(counter)
34
35          elif event.type == pygame.QUIT:
36              counter = 0
37              break
38
39      if counter == 0:
40          break
41
42      screen.fill((255,255,255)) # set white background for our pygame window
43      # next line sets a black font color and positions our text at 64, 96 within our pygame window
44      screen.blit(font.render(label, True, (0, 0, 0)), (64, 96))
45      pygame.display.flip()  # this line draws everything
46
47  camera.close()
48  pygame.quit()
```

With lines 42 - 45 (still within our “while True” loop) we fill our pygame’s window background color to white (line 42), and update the screen to include our label value (i.e. what the current counter variable’s count is set to). On line 45 we display (or update) our pygame window on our Raspberry Pi’s screen.

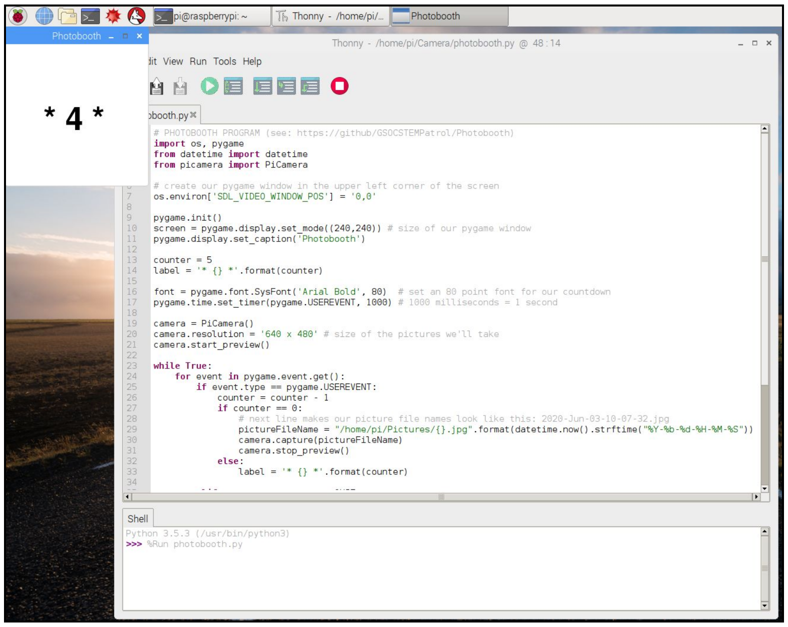
Once we break out of the “while True” loop we do a few cleanup tasks in lines 47 and 48; closing the camera and as well the pygame engine before our program ends.

STEP #2 RUN (and debug) your new photobooth program!

Copying someone else’s code is a great way of learning how to program! Once you get the code running and understand it a bit, you can make some changes to it, like changing the window or font sizes, for example, to learn and make the code your own.

In reality, it’s nearly IMPOSSIBLE to copy 40+ lines of unfamiliar code without making an error! Don't be surprised (in fact you can EXPECT) that your code won’t run the first time you try it. Don’t give up - learning to program requires patience and persistence!

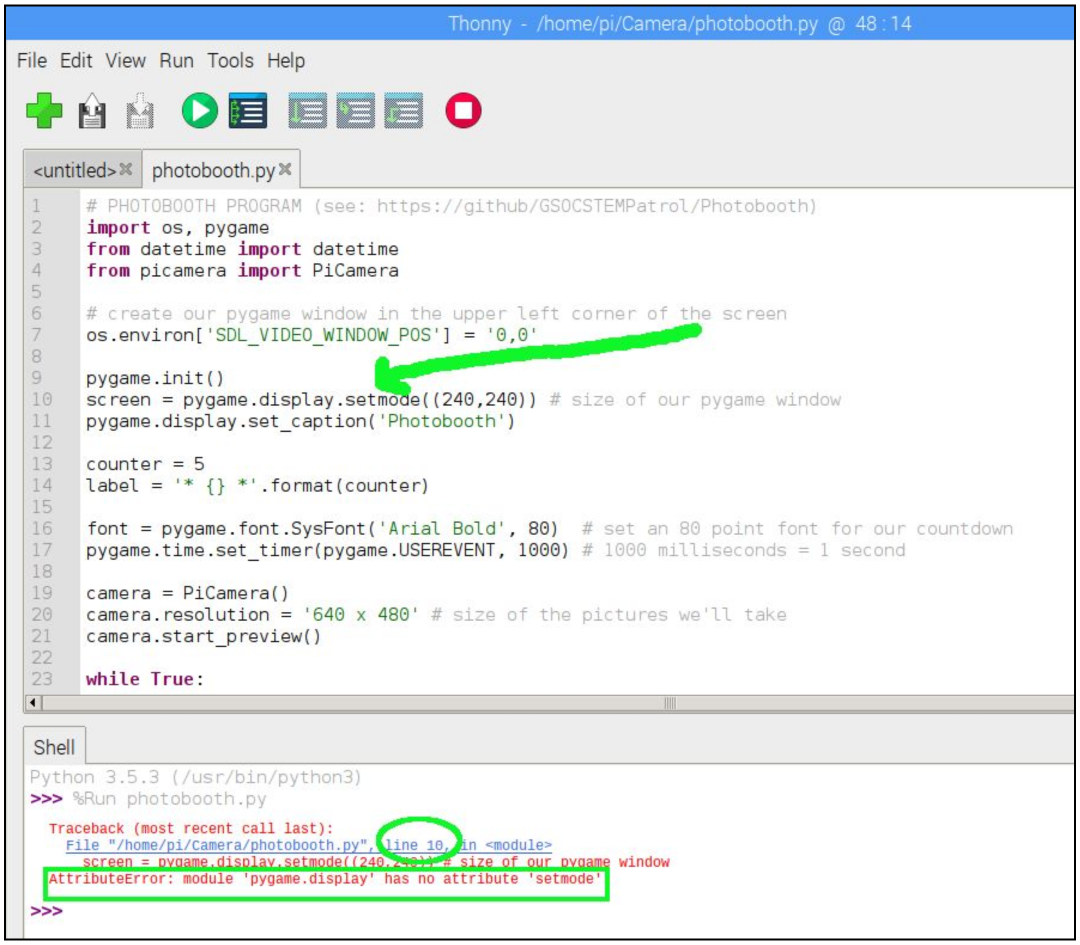
With that being said, let’s go ahead and click on the Thonny IDE’s RUN (Green button with white arrow) and try to run our code. One of two things will happen: (1) a small window will appear in the upper left corner of our Raspberry Pi’s screen, counting down to zero as shown here:



```
# PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
import os, pygame
from datetime import datetime
from picamera import PiCamera

# create our pygame window in the upper left corner of the screen
os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'

pygame.init()
screen = pygame.display.set_mode((240,240)) # size of our pygame window
pygame.display.set_caption('Photobooth')

counter = 5
label = '* {} *'.format(counter)

font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second

camera = PiCamera()
camera.resolution = '640 x 480' # size of the pictures we'll take
camera.start_preview()

while True:
    for event in pygame.event.get():
        if event.type == pygame.USEREVENT:
            counter = counter - 1
            if counter == 0:
                # next line makes our picture file names look like this: 2020-Jun-03-10-07-32.jpg
                pictureFileName = "/home/pi/Pictures/{}.jpg".format(datetime.now().strftime("%Y-%b-%d-%H-%M-%S"))
                camera.capture(pictureFileName)
                camera.stop_preview()
            else:
                label = '* {} *'.format(counter)
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Run photobooth.py
```

… or (2) an error message will appear in a window beneath our code.


Thonny - /home/pi/Camera/photobooth.py @ 48 : 14
File Edit View Run Tools Help
<untitled>
photobooth.py
Shell

```
# PHOTOBOOTH PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
import os, pygame
from datetime import datetime
from picamera import PiCamera

# create our pygame window in the upper left corner of the screen
os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'

pygame.init()
screen = pygame.display.setmode((240,240)) # size of our pygame window
pygame.display.set_caption('Photobooth')

counter = 5
label = '* {} *'.format(counter)

font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second

camera = PiCamera()
camera.resolution = '640 x 480' # size of the pictures we'll take
camera.start_preview()

while True:
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Run photobooth.py

  Traceback (most recent call last):
    File "/home/pi/Camera/photobooth.py", line 10, in <module>
      screen = pygame.display.setmode((240,240)) # size of our pygame window
  AttributeError: module 'pygame.display' has no attribute 'setmode'
>>>
```

Ninety nine percent of the time the error will pertain to syntax that was not copied exactly. Usually a line number is provided within the error message that will help you identify the line (or lines) to reexamine within your code. In the example above we typed in “setmode” in line 10 but we should have typed “set_mode”. This might not be the error you had, but yours will likely be very similar (and don’t be surprised if you have more than one error!).

Click on the Thonny IDE’s STOP button (red button with white square) to stop your code from running. Correct your code and save it, then click on the RUN button once again to try again!

Once your code works, you can click on the STOP/RUN button a number of times and create some “photobooth” works of art! The photos that you are creating will appear within the Raspberry Pi’s /home/pi/Pictures folder; you can use the Raspberry Pi File Manager tool to view them.

If the photobooth program was fun, then you’ll certainly want to move on to the next project and create a **SLIDESHOW program using python** that will create a slideshow of all of the pictures you took with the photobooth on that day. Now that you have some coding experience, this next python program will be even easier!

The entire Photobooth coding listing follows for your convenience:

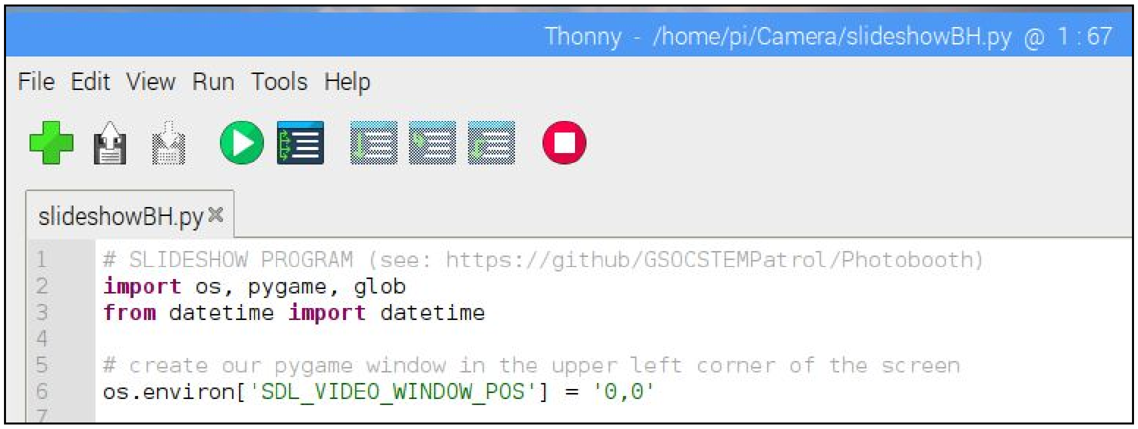
```
# PHOTOBOOTH PROGRAM (see: http://github/GSOCSTEMPatrol/Photobooth)
import os, pygame
from datetime import datetime
from picamera import PiCamera

# create our pygame window in the upper left corner of the screen
os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'

pygame.init()
screen = pygame.display.set_mode((240,240)) # size of our pygame window
pygame.display.set_caption('Photobooth')

counter = 5
label = '* {} *'.format(counter)

font = pygame.font.SysFont('Arial Bold', 80)  # set an 80 point font for our countdown
pygame.time.set_timer(pygame.USEREVENT, 1000) # 1000 milliseconds = 1 second

camera = PiCamera()
camera.resolution = '640 x 480' # size of the pictures we'll take
camera.start_preview()

while True:
    for event in pygame.event.get():
        if event.type == pygame.USEREVENT:
            counter = counter - 1
            if counter == 0:
                # next line makes our picture file names look like this:
2020-Jun-03-10-07-32.jpg
                pictureFileName =
"/home/pi/Pictures/{}.jpg".format(datetime.now().strftime("%Y-%b-%d-%H-%M-%S"))
                camera.capture(pictureFileName)
                camera.stop_preview()
            else:
                label = '* {} *'.format(counter)

        elif event.type == pygame.QUIT:
            counter = 0
            break

    if counter == 0:
        break

    screen.fill((255,255,255)) # set white background for our pygame window
    # next line sets a black font color and positions our text at 64, 96 within our pygame
window
    screen.blit(font.render(label, True, (0, 0, 0)), (64, 96))
    pygame.display.flip()  # this line draws everything

camera.close()
pygame.quit()
```

BUILD YOUR OWN SLIDESHOW with PYTHON (recommended for Cadettes +)

<u>STEP #1 open up the Thonny Python IDE</u>

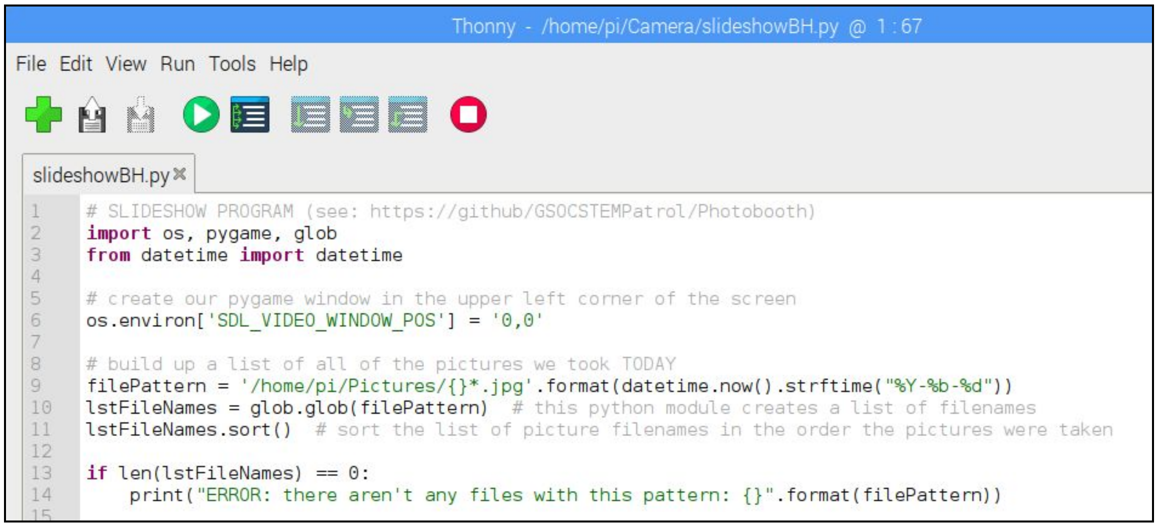
Refer to the PHOTOBOOTH activity documentation if you have forgotten how to open the Thonny IDE programming environment and create a new program. Save the program as “slideshowFL” **where F is the first character of your first name and L is the first character of your last name**.

As with our PHOTOBOOTH program, we start by including a few external modules into our program as shown below. In our slideshow program, we don’t need the picamera module as we won’t be taking pictures. We do however, need a new external module (glob) to help us read the filenames from a folder.   Additionally, just like in our photobooth program, on line 6 we set an environment variable (using the imported os module) to specify where our pygame window is drawn on our Raspberry Pi’s screen.


Thonny - /home/pi/Camera/slideshowBH.py @ 1 : 67
File Edit View Run Tools Help
slideshowBH.py

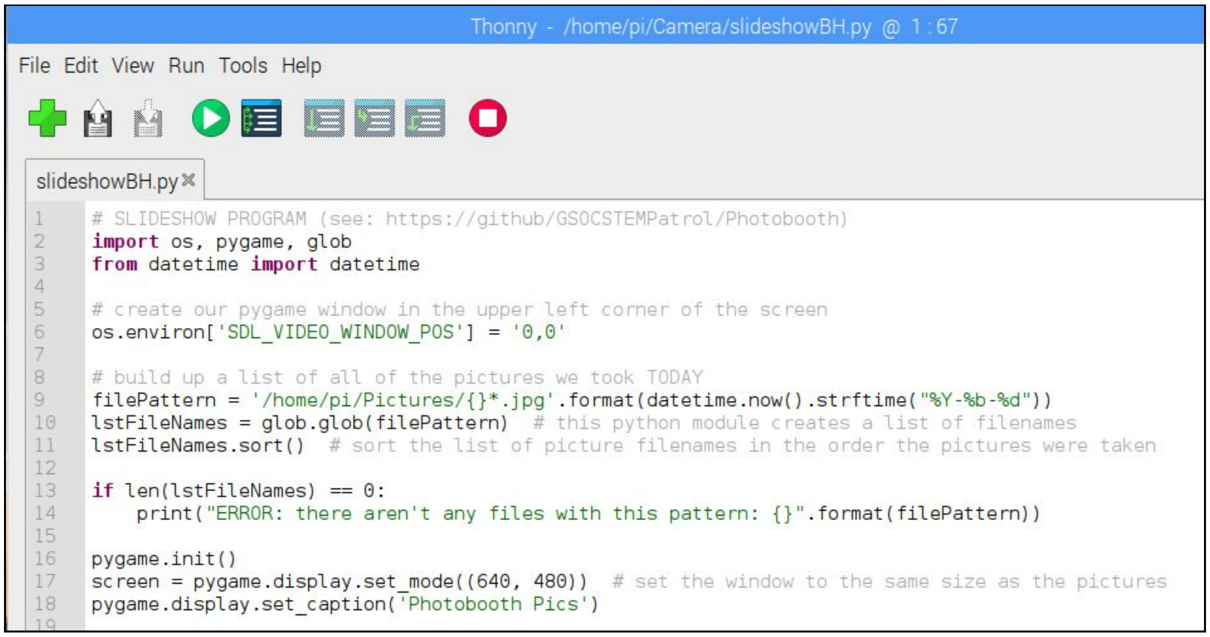
```
1  # SLIDESHOW PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2  import os, pygame, glob
3  from datetime import datetime
4
5  # create our pygame window in the upper left corner of the screen
6  os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
```

Referring to the following picture, on lines 9 - 11, we use the glob module to search for all of the files beginning with today’s date (example: %Y-%b-%d = 2020-Jun-3). We then return the filenames that match this “file pattern” into a variable we name *lstFileNames* (“list of file names”). On line 11 we sort this list by filename, and since a timestamp is part of our filename, the list will be sorted in the order that the pictures were taken. Lines 13 and 14 allow us to report an error if there weren’t any pictures taken today (meaning, we didn’t use the photobooth program before the slideshow program).


Thonny - /home/pi/Camera/slideshowBH.py @ 1 : 67
File Edit View Run Tools Help
slideshowBH.py

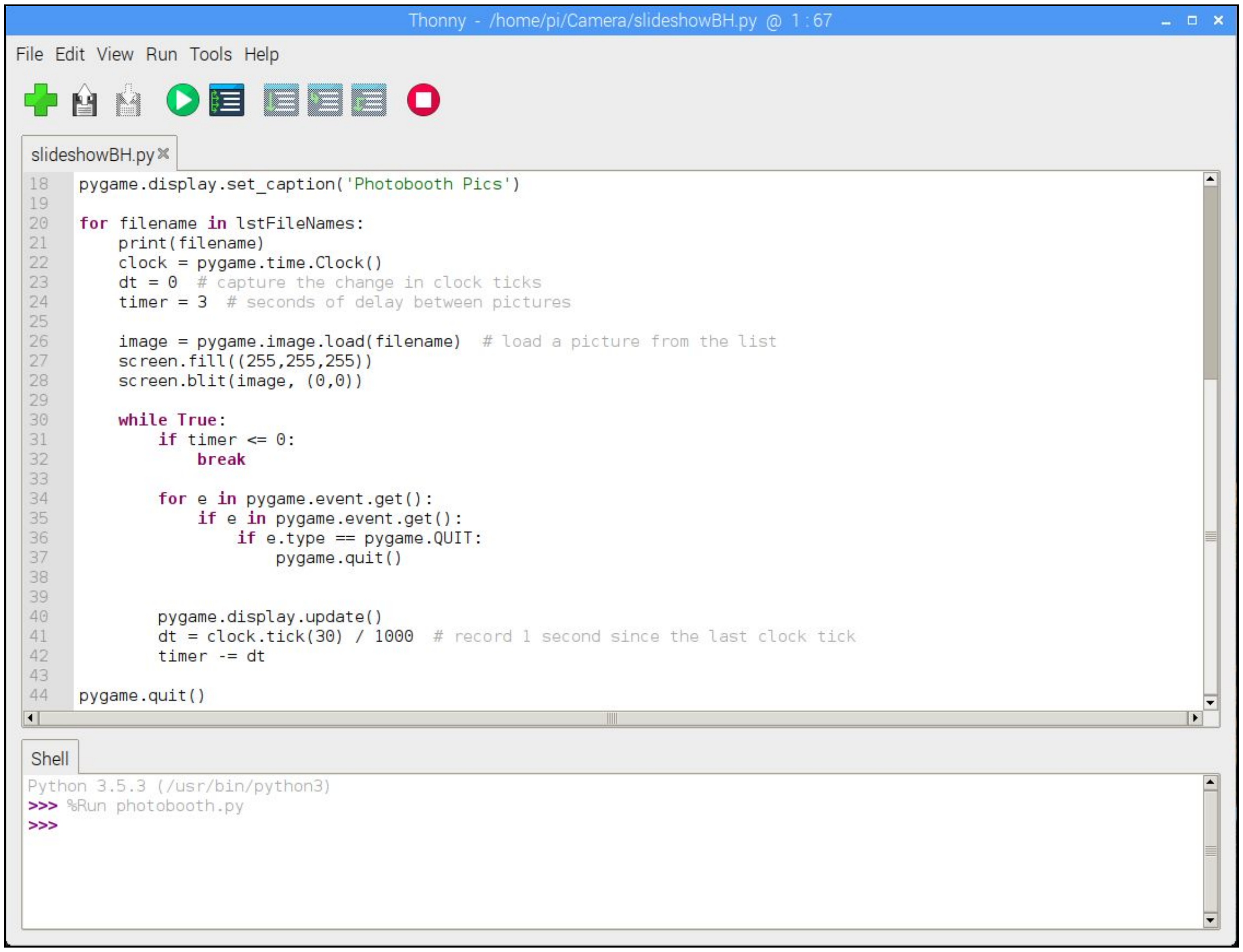
```
1   # SLIDESHOW PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2   import os, pygame, glob
3   from datetime import datetime
4
5   # create our pygame window in the upper left corner of the screen
6   os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
7
8   # build up a list of all of the pictures we took TODAY
9   filePattern = '/home/pi/Pictures/{}*.jpg'.format(datetime.now().strftime("%Y-%b-%d"))
10  lstFileNames = glob.glob(filePattern)  # this python module creates a list of filenames
11  lstFileNames.sort()  # sort the list of picture filenames in the order the pictures were taken
12
13  if len(lstFileNames) == 0:
14      print("ERROR: there aren't any files with this pattern: {}".format(filePattern))
```

Within lines 16 - 17 we initialize the pygame system, then set our pygame screen size to 640 x 480 pixels, which is the same size of the pictures that we took with the photobooth program. On line 18 we set a caption for the pygame window that will display our slideshow.


Thonny - /home/pi/Camera/slideshowBH.py @ 1 : 67
File Edit View Run Tools Help
slideshowBH.py

```
1   # SLIDESHOW PROGRAM (see: https://github/GSOCSTEMPatrol/Photobooth)
2   import os, pygame, glob
3   from datetime import datetime
4
5   # create our pygame window in the upper left corner of the screen
6   os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
7
8   # build up a list of all of the pictures we took TODAY
9   filePattern = '/home/pi/Pictures/{}*.jpg'.format(datetime.now().strftime("%Y-%b-%d"))
10  lstFileNames = glob.glob(filePattern)  # this python module creates a list of filenames
11  lstFileNames.sort()  # sort the list of picture filenames in the order the pictures were taken
12
13  if len(lstFileNames) == 0:
14      print("ERROR: there aren't any files with this pattern: {}".format(filePattern))
15
16  pygame.init()
17  screen = pygame.display.set_mode((640, 480))  # set the window to the same size as the pictures
18  pygame.display.set_caption('Photobooth Pics')
```

Referring to the following picture, starting from line 20, we loop over the list of our picture filenames, or in other words, the pictures that we took today using the photobooth program. We print the filename out (will display within the bottom window of the Thonny IDE) and we set a timer which we use to determine how long each picture is shown. Specifically, in line 24, we are setting the value of the variable *timer* to 3, meaning each picture will be displayed for 3 seconds before changing to the next picture taken today.


Thonny - /home/pi/Camera/slideshowBH.py @ 1 : 67
File Edit View Run Tools Help
slideshowBH.py
Shell

```
18  pygame.display.set_caption('Photobooth Pics')
19
20  for filename in lstFileNames:
21      print(filename)
22      clock = pygame.time.Clock()
23      dt = 0  # capture the change in clock ticks
24      timer = 3  # seconds of delay between pictures
25
26      image = pygame.image.load(filename)  # load a picture from the list
27      screen.fill((255,255,255))
28      screen.blit(image, (0,0))
29
30      while True:
31          if timer <= 0:
32              break
33
34          for e in pygame.event.get():
35              if e in pygame.event.get():
36                  if e.type == pygame.QUIT:
37                      pygame.quit()
38
39
40          pygame.display.update()
41          dt = clock.tick(30) / 1000  # record 1 second since the last clock tick
42          timer -= dt
43
44  pygame.quit()
```

```
Python 3.5.3 (/usr/bin/python3)
>>> %Run photobooth.py
>>>
```

In lines 27 - 28, we set up our pygame window by first loading in the current loop iteration's filename, setting the pygame window's background to white, and then initialize the screen (screen.blit) to include the image file we've loaded.

We then enter the endless pygame loop (line 30), which we break out of when our timer value counts down to 0. Just as within our photobooth file, we start a pygame window event loop, which pygame requires. Unlike our photobooth, we really aren't listening for a specific event.

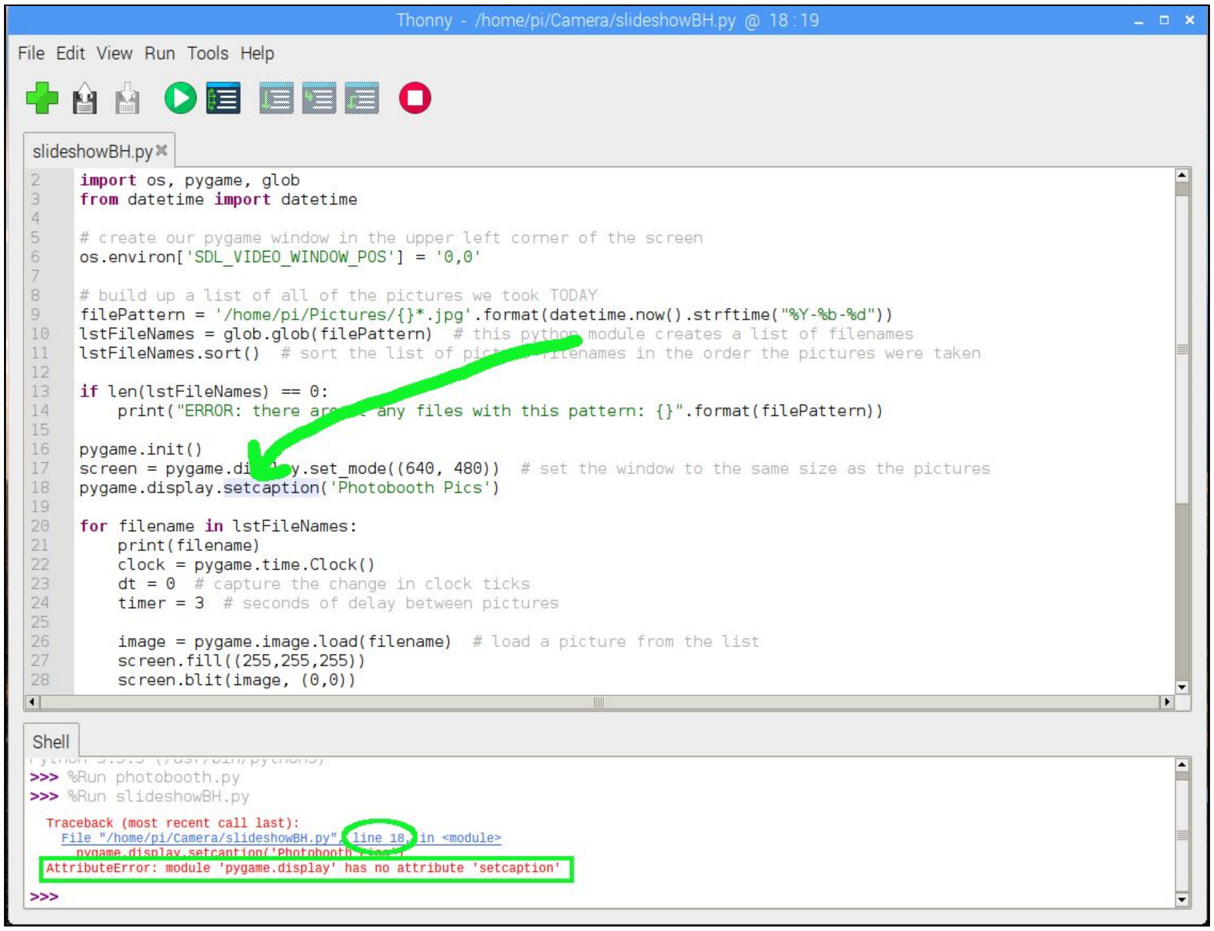
After checking the pygame event queue, we update our pygame window display (line 40) which actually displays our image on the screen. Line 41 returns the number of clock ticks in one second, and subtracts that amount (one second) from our timer. Once the timer counts down to 0 (3 .. 2 .. 1 .. 0) we break out of our endless "while True" loop on line 32 and we move onto the next picture within the lstFileNames list.

Finally, on line 44, we shutdown the pygame system before we end our program.

<u>STEP #2 RUN (and debug) your new slideshow program!</u>

As we mentioned in the photobooth program project, it's nearly IMPOSSIBLE to copy 40+ lines of unfamiliar code without making an error! Don't be surprised (in fact you can EXPECT) that your code won't run the first time you try it. Don't give up - learning to program requires patience and persistence!

With that being said, let’s go ahead and click on the Thonny IDE’s RUN button (green button with white arrow) and try to run our code. One of three things will happen: (1) a window will appear in the upper left corner of our screen, containing the first of the photobooth pictures taken today. (2) The error message ”ERROR: there aren't any files with this pattern: YYYY-mmm-dd” will appear (YYYY-mmm-dd will be an actual date, like 2020-Jun-3) within the bottom window of the Thonny IDE. Try to use the photobooth program to take some pictures and then run the slideshow program again. (3) an error message will appear resulting from an error we made copying a line of code.


Thonny - /home/pi/Camera/slideshowBH.py @ 18 : 19
File Edit View Run Tools Help
slideshowBH.py
Shell

```
2   import os, pygame, glob
3   from datetime import datetime
4
5   # create our pygame window in the upper left corner of the screen
6   os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'
7
8   # build up a list of all of the pictures we took TODAY
9   filePattern = '/home/pi/Pictures/{}*.jpg'.format(datetime.now().strftime("%Y-%b-%d"))
10  lstFileNames = glob.glob(filePattern)  # this python module creates a list of filenames
11  lstFileNames.sort()  # sort the list of picture filenames in the order the pictures were taken
12
13  if len(lstFileNames) == 0:
14      print("ERROR: there aren't any files with this pattern: {}".format(filePattern))
15
16  pygame.init()
17  screen = pygame.display.set_mode((640, 480))  # set the window to the same size as the pictures
18  pygame.display.setcaption('Photobooth Pics')
19
20  for filename in lstFileNames:
21      print(filename)
22      clock = pygame.time.Clock()
23      dt = 0  # capture the change in clock ticks
24      timer = 3  # seconds of delay between pictures
25
26      image = pygame.image.load(filename)  # load a picture from the list
27      screen.fill((255,255,255))
28      screen.blit(image, (0,0))
```

```
>>> %Run photobooth.py
>>> %Run slideshowBH.py

  Traceback (most recent call last):
    File "/home/pi/Camera/slideshowBH.py", line 18, in <module>
      pygame.display.setcaption('Photobooth Pics')
  AttributeError: module 'pygame.display' has no attribute 'setcaption'

>>>
```

As an example, in line 18 we should have copied pygame.display.set_caption, but we inadvertently copied pygame.display.setcaption instead, which raised the error above.

Click on the Thonny IDE’s STOP button (red button with white square) to stop your code from running. Correct your code and save it, then click on the RUN button once again to try again!

Once you have corrected any errors, continue to press the STOP button (which will change that button to RUN) and then RUN to play your slideshow! Don’t forget you can go back and take a few more pictures with your photobooth program which will be automatically added to your slideshow.

The entire Slideshow coding listing follows for your convenience:

```
# SLIDESHOW PROGRAM (see: http://github/GSOCSTEMPatrol/Photobooth)
import os, pygame, glob
from datetime import datetime

# create our pygame window in the upper left corner of the screen
os.environ['SDL_VIDEO_WINDOW_POS'] = '0,0'

# build up a list of all of the pictures we took TODAY
filePattern = '/home/pi/Pictures/{}*.jpg'.format(datetime.now().strftime("%Y-%b-%d"))
lstFileNames = glob.glob(filePattern)  # this python module creates a list of filenames
lstFileNames.sort()  # sort the list of picture filenames in the order the pictures were
taken

if len(lstFileNames) == 0:
    print("ERROR: there aren't any files with this pattern: {}".format(filePattern))

pygame.init()
screen = pygame.display.set_mode((640, 480))  # set the window to the same size as the
pictures
pygame.display.set_caption('Photobooth Pics')

for filename in lstFileNames:
    print(filename)
    clock = pygame.time.Clock()
    dt = 0  # capture the change in clock ticks
    timer = 3  # seconds of delay between pictures

    image = pygame.image.load(filename)  # load a picture from the list
    screen.fill((255,255,255))
    screen.blit(image, (0,0))

    while True:
        if timer <= 0:
            break

        for e in pygame.event.get():
            if e in pygame.event.get():
                if e.type == pygame.QUIT:
                    pygame.quit()

        pygame.display.update()
        dt = clock.tick(30) / 1000  # record 1 second since the last clock tick
        timer -= dt

pygame.quit()
```