

Project 2: Scary Spot the Difference

Python can handle pictures and sounds as well as turtle-based graphics, and those can be used to great effect as a prank on your friends — a spot-the-difference game with a scary secret at its heart, perfect for Halloween!

This project needs two images — your spot-the-difference image plus a ‘scary’ surprise image — and a sound file. Click on the Raspberry Pi icon to load the Raspberry Pi menu, choose the **Internet** category, and click on **Chromium Web Browser**. Once it’s open, type **rptl.io/spot-pic** into the address bar followed by the **ENTER** key. Right-click on the picture and click on **Save image as**, choose the **Home** folder from the list on the left-hand side, then click **Save**. Click back on Chromium’s address bar, then type **rptl.io/scary-pic** followed by the **ENTER** key. As before, right-click the picture, click **Save image as**, choose the **Home** folder, then click **Save**.

To get the sound file, click back into the address bar and type **rptl.io/scream** followed by the **ENTER** key. This file — the sound of a scream to give your player a real surprise — will download automatically. It needs to be moved to the **Home** folder before you can use it. Open a new File Manager window, go to your **Downloads** folder, and find the **en_images_....wav** file you just downloaded. Right-click the file and click **Rename**, then change its name to **scream.wav**. Right-click **scream.wav**, then click **Cut**. Finally, click on **Home Folder** at the top-left of the file manager, right-click in any empty space in the large file viewing window on the right, and click **Paste**. You can now close the Chromium and File Manager windows.

Click the **New** icon  in the Thonny toolbar to begin a new project. As before, you’re going use a library to extend Python’s capabilities. This time it’s the **pygame** library, which, as the name suggests, was created with games in mind. Type the following:

```
import pygame
```

You’ll need some parts from other libraries and a subsection of the Pygame library too. Import these by typing the following:

```
from pygame.locals import *
from time import sleep
from random import randrange
```

The **from** instruction works differently to the **import** instruction by allowing you to import only the parts of a library you need rather than the whole library. Next, you need to set up Pygame; this is called *initialisation*. Pygame

needs to know the width and height of the player's monitor or TV, known as its *resolution*. Type the following:

```
pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
```

The final step in setting up Pygame is to create its main window, which Pygame calls a screen. Type the following:

```
screen = pygame.display.set_mode((width, height))
pygame.display.update()
# Your code goes here
pygame.quit()
```

Note the commented line in the middle: this is where your program will go. For now, though, click the **Save** icon , save your program as **Spot the Difference.py**, then click the **Run** icon  and watch. Pygame will create a window, fill it with a black background, and then almost immediately close the window as it reaches the instruction to quit. Aside from a short message in the shell (**Figure 5-14**), the program hasn't achieved much so far.

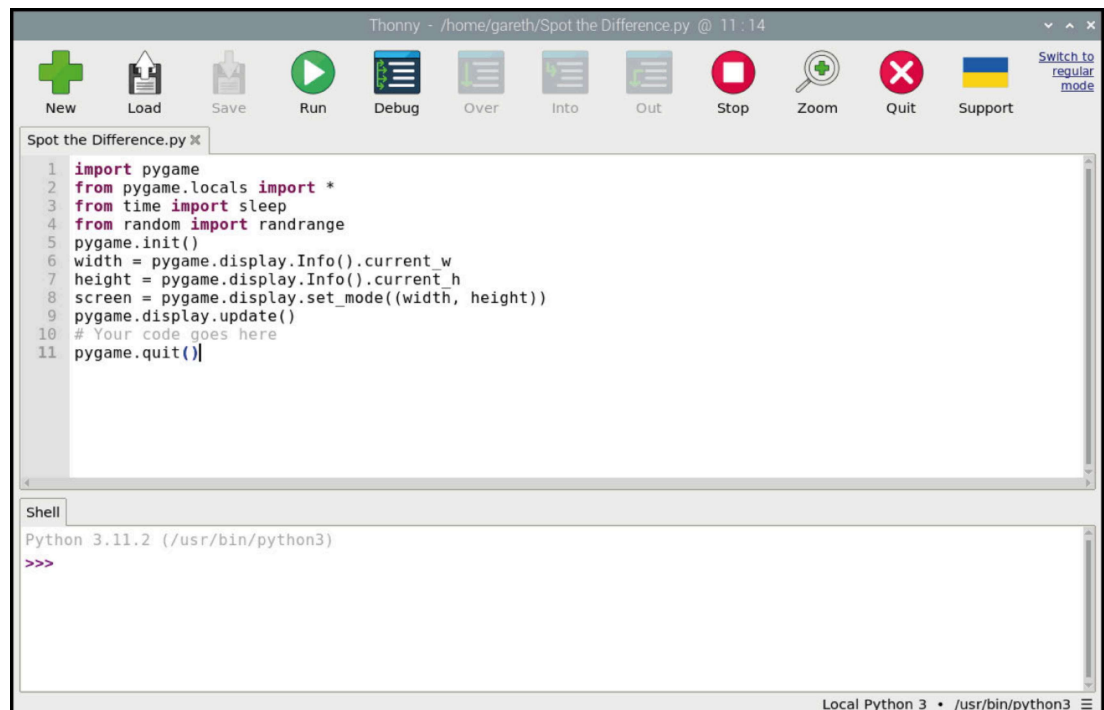


Figure 5-14 Your program is functional, but doesn't do much yet

To display your spot-the-difference image, delete the comment above **pygame.quit()** and type the following in the space:

```
difference = pygame.image.load('spot_the_diff.png')
```

To make sure the image fills the screen, you need to scale it to match your player's monitor or TV's resolution. Type the following:

```
difference = pygame.transform.scale(difference, (width, height))
```

Now that the image is in memory, you need to tell Pygame to actually display it on the screen — a process known as *blitting*, or a *bit block transfer*. Type the following:

```
screen.blit(difference, (0, 0))  
pygame.display.update()
```

The first of these lines copies the image onto the screen, starting at the top-left corner; the second tells Pygame to redraw the screen. Without this second line, the image will be in the correct place in memory, but you'll never see it!

Click the **Run** icon , and the image, shown in **Figure 5-15**, will briefly appear on screen.



Figure 5-15 Your spot-the-difference image

To have the image on screen for a longer period of time, add the following line just above `pygame.quit()`:

```
sleep(3)
```

Click the **Run** icon  again, and the image will stay on the screen longer. Add your surprise image by typing the following just below the line `pygame.display.update()`:

```
zombie = pygame.image.load('scary_face.png')
zombie = pygame.transform.scale(zombie, (width, height))
```

Add a delay, so the zombie image doesn't appear right away:

```
sleep(3)
```

Then blit the image to the screen and update to show it to the player:

```
screen.blit(zombie, (0,0))
pygame.display.update()
```

Click the **Run** icon  and watch what happens: Pygame will load your spot-the-difference image, but after three seconds it will be replaced with the scary zombie (**Figure 5-16**)!



Figure 5-16 It'll give someone a scary surprise

Having the delay set at three seconds makes things a bit too predictable, though. Change the line `sleep(3)` above `screen.blit(zombie, (0,0))` to:

```
sleep(randrange(5, 15))
```

This picks a random number between 5 and 15 and delays the program for that long. Next, add the following line just above your `sleep` instruction to load the scream sound file:

```
scream = pygame.mixer.Sound('scream.wav')
```

Type the following on a new line after your sleep instruction to start playing the sound. It should kick in just ahead of the scary image actually being shown to the player:

```
scream.play()
```

Finally, tell Pygame to stop playing the sound by typing the following line just above `pygame.quit()`:

```
scream.stop()
```

Click the **Run** icon  and admire your handiwork: after a few seconds of innocent spot-the-difference fun, your scary zombie will appear alongside a blood-curdling shriek — sure to give your friends a fright! If you find that the zombie picture appears before the sound starts playing, you can compensate by adding a small delay just after your `scream.play()` instruction and before your `screen.blit` instruction:

```
sleep(0.4)
```

Your finished program should look like this:

```
import pygame
from pygame.locals import *
from time import sleep
from random import randrange

pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
screen = pygame.display.set_mode((width, height))
pygame.display.update()

difference = pygame.image.load('spot_the_diff.png')
difference = pygame.transform.scale(difference, (width, height))
screen.blit(difference, (0, 0))
pygame.display.update()

zombie = pygame.image.load('scary_face.png')
```