Girls' Programming Network

Flappy Bird!

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Amanda Hogan
Isabella Hogan
Renee Noble

A massive thanks to our sponsors for supporting us!

ATLASSIAN

Intro to Python

Part 0: Setting up

Task 0.1: Start Programming

Follow the instructions from the lecture to set up your flappy bird python file!

Hint

Don't forget to ask the tutors for help if you're not sure what to do next

Task 0.2: You've got a blank space, so write your name!

At the top of the file edit the comment to write your name!
Any line starting with # is a comment.

```
# This is a comment
```

Task 0.3: Some scaffolding

Now read through the other comments to see the different steps we'll be doing.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

- ☐ You should have a file called flappy_bird.py
- ☐ Your file has your name at the top in a comment
- ☐ Your file has a bunch of scaffolding comments
- ☐ Run your file with F5 key and it does nothing!!

Part 1: Making a game

Intro to Pygame

Task 1.1: Print a welcome and the rules

Welcome the player and print the rules!

Use some print statements to make it happen when you run your code:

Put this under the print welcome comment

```
The game is about to start!
Click the mouse to "flap" upwards
Dodge the pipes and the floor
Good luck and have fun!
```

Task 1.2: Get Pygame Zero

PygameZero will give us the tools we need to write our game code. Let's import it!

1. At the top of your file, under the start modules comment **import Pygame Zero**

2. At the bottom of your file, under the runs everything comments,**make Pygame Zero GO!**

Hint

To import and start Pygame Zero we can use this code:

```
import pgzrun
...
pgzrun.go()
```

Task 1.3: Make a screen

Before we make our game we need to set up the screen to display it on!

1. **Make a screen** that is 800 pixels wide and 600 pixels tall. This should be under the create constants comment

2. **Run** your code! Do you see a big dark box?

Hint

To make a screen that is 100 pixels wide and 200 pixels tall you could use this code

```
WIDTH = 100
HEIGHT = 200
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

- ☐ Print a welcome
- ☐ Print the rules
- ☐ Try running your code!
- ☐ Shows a dark screen that is the same size as the background image

Pygame Images

Part 2: Images!

Task 2.1: Save all the images!

We need to load all the images that we need for our game!

1. **Download** the images onto our computer. Go to https://www.girlsprogramming.network/flappy-bird-images and download all the images there.

2. **Double click** on the zip folder to *unzip* the the folder to get to the images

3. **Move the images folder** into the "flappy_bird" folder

Task 2.2: Backgrounds are better!!

Let's start our game by adding the background to the screen! We just downloaded the pic! We'll be using a **function** to **draw** things to our screen.

1. Under the comment "make background" make a new Actor() called `background` with an image of "bg".

2. Directly under that make it so that the actor's x value is 400 and y value is 300

3. **Create a function** call `draw()` under the draw everything to screen comment

4. Inside your function, ***draw background to the screen*** This should be under the set background image comment.

5. Make sure the code you wrote in Step 4 is ***indented inside your function***.

Hint: How to function!

How do I indent things inside my function?
Use the Tab key to make the parts you want inside your function go inside!

```
def myFunction():
    # this line is indented, it's part of the function!
# this line is not indented, it isn't part of the function.
```

Hint: Make an actor!

To make an actor with an x value of 10 and a y value of 25, it should look like this;

```
myCharacter = Actor("characterImage")
myCharacter.x = 10
myCharacter.y = 25
```

Hint: Draw an actor!

We can use draw, since we already programmed the x, y position for our background Actor.

Here's an example:

```
myCharacter.draw()
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

- ☐ Saved all the images to the same place as your code
- ☐ Made the background appear on the game screen
- ☐ Run your code!

★ BONUS 2.3: Funky Backgrounds!!

Waiting for the next lecture? Try adding this bonus feature!!

We can use any image as a background!

1. Go on the **internet** to find a cool image that you want to be the background of the game. Must be a png
2. Save the image into your **images folder**
3. Where you've set the background image replace **"bg"** with the **name of your file**

Try and make sure the new image is a similar size to the original and if it's too small try to make it bigger

Events & If Statements

Part 3: The Bird is the Word!

Task 3.1: Where's that bird?

This game is meant to be Flappy Bird but there's no bird on our screen! Let's fix that.

1. **Check** that you have the **bird image file called bird.png** where you saved your code. This should be in your images folder.

2. Create an ***Actor*** using the bird image.
   Store it in a variable **named bird.** Do this after the make bird comment

3. After you make the bird set the **x value** to be **160** and the **y value** to be **300**

Task 3.2: Show me the bird!

Now we're ready to add the bird to the screen!

1. Go to your **draw function**

2. **After** the draw actors comment, ***draw*** your bird.
   Make sure this is indented inside your function.

Task 3.3: Make the bird fly

We need to make the bird move downwards on the screen

1. Make a new **function** called `update()` after the "updates everything" comment

2. You need to **increase** the bird's y value **by 1** by adding one to its value under the updating bird comment

Pygame zero does all the work for looping and frames so don't worry about that.

Hint

We need to save a new y value to bird. We can do this the same way we originally set it.

e.g.

```
myCharacter.y = myCharacter.y + 5
```

Task 3.4: Falling off the screen

When our bird touches the bottom of the screen it should end the game.

1. **Import** a new module called `sys` directly under where you `import pgzrun`
2. In the *update* function under the "bird hits bottom of screen" comment you need to test if the bird's y value is **more** than the **height** of the screen. Use an if statement!
3. If it is, print `"Game Over!"` then you need to put the line `sys.exit()` to immediately **stop the game.**

Task 3.5: Moving the bird

Our bird should "flap" upwards when the mouse is clicked.

1. Make a **new function** called `on_mouse_down()` after the moving comment
2. **Decrease** the bird's y value by 50

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

- ☐ There is a background still on the screen
- ☐ There is a bird on the screen and its falling
- ☐ When you click the bird moves back up
- ☐ Run your code!

★ BONUS 3.6: Funky Bird!

Waiting for the next lecture? Try adding this bonus feature!!
We can use any image as our bird!

1. Go on the **internet** to find a cool image that you want to be the **character** of the game. Must be a jpg,png, or gif
2. Save the image into your **images folder**
3. Where you've set the bird's image replace **"bird"** with the **name of your file**

Try and make sure the new image is a similar size to the original as otherwise the game may be too hard.

Part 4: Pipes!

We'll be making some pipes to make the game playable

Task 4.1: Make a pipe

Let's make a pipe!

1. Create a new actor called `topPipe1` with an image of `"top"`. This should be under the make pipes comment
2. Set it's **x value** to be 266
3. Set it's **y value** to be -10

Hint

Look at your code for your bird actor if you've forgotten how to make one

Task 4.2: Updating it!

Now we need to make it so it shows up and so it moves to the left when the game is run.

1. In the draw function under where you're drawing the bird **draw** the topPipe1 actor
2. Go to the **update function** under the updating pipes comment
3. **Decrease** topPipe1's x value by 1

Task 4.3: Now what?

Now that we have our new pipe we need to make some more. We're going to start with 5 more (three groups of a top and bottom pipe).

1. The **first bottom pipe** should have an **x value** of 266 and a **y value** of 800
2. The **second top pipe** should have an **x value** of 532 and a **y value** of -200
3. The **second bottom pipe** should have an **x value** of 532 and a **y value** of 560
4. The **third top pipe** should have an **x value** of 798 and a **y value** of -120
5. The **third bottom pipe** should have an **x value** of 798 and a **y value** of 710

The bottom pipes should be called something like `bottomPipe1` and should have an image of `"bottom"`.

Task 4.4: Now what? pt.2

Now that we have our new pipes we need to make a list to store all of them.

1. Above where you made all your pipes but under the making pipes comment, make a **list called pipes**

2. After you've created each pipe **add** it to the list.

Hint

To create and add something to a list it looks like this:

```
myList = []
myList.append(myCharacter)
```

Task 4.5: Move those pipes!

You now need to update and draw all the pipes.

1. In the draw function, where you drew topPipe1, replace that line with a **loop** that **loops through every item within pipes** and draws them

2. In the update function, where you update topPipe1's x value, replace that line with a **loop** that **updates the x value** of each item in pipes

Hint

To loop through a list you need to do something along the lines of:

```
for thing in myList:
   thing  = thing + 5
```

Task 4.6: Colliding!

Run your code and see what happens.
If the bird hits the pipes nothing happens right? Let's make that work.

1. Go to the update function, under the **"bird hits pipes"** comment

2. Loop through the pipes to test if the bird has hit them. If it has, print `"Game Over!"` and `exit` the program

Hint

To test if one actor has hit another the code should look like:

```
if myCharacter.colliderect(myWall):
    print("Hit!")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:

- ☐ The game has some moving pipes
- ☐ The game stops when the bird hits the ground
- ☐ The game stops when the bird hits the pipes

Part 5: A better game!

Task 5.1: Wrapping around!

Our pipes move from right to left but drop off the end of the screen let's make them move to the right side when that happens

1. In the loop where you update the pipes
2. Write an **if statement** to see if the pipe's **x value** is less than -44
3. If it is, set it's x value to be the **width** of the screen (800)

Task 5.2 Scoring

We're going to make it so the player can earn some points to make the game more fun.

1. Make a variable called **score** that starts at zero in your create constants section
2. At the top of the **update function** above the updating bird comment make the variable **"global"**

Hint

To make a variable global you need to write

```
global myVariable
```

Task 5.3 Scoring pt2.

Then we want to add a point every time a set of two pipes rolls around to the other side of the screen. To do this we're going to modify our for loop.

1. Go to the **for loop** under the updating pipes comment
2. Inside the if statement to wrap around the pipe and under where you reset the pipe's x value to the width of the screen create a new if statement that tests whether the pipe's image is `"top"`
3. If it is, add increase the score by one

Hint: making an index loop

To make a loop where we loop through index values it looks something like this:

```
for index in range(len(myList)):
    print(myList[index])
```

Hint: finding out what image an actor is using

You can check what image an actor is displaying in a similar way to how you can check its x or y value. You will need to use code similar to:

```
if myActor.image == "myImage":
    # do a thing
```

Task 5.4: Now what?

Now that we are detecting when the game is over and we're tracking a score, we should print the final score when the game is over.

1. In the **if statement** that checks whether the bird hits the bottom of the screen, before the sys.exit(), **print** out the score

2. In the **if statements** that check if a bird has hit a pipe, before the sys.exit(), **print** our the score

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

- ☐ Your game ends after the bird collides with the wall or pipes
- ☐ You are printing a score when the game is over

★ BONUS 5.8: On screen score!

Waiting for the next lecture? Try adding this bonus feature!!

We are currently printing the score once the game has finished, but we should try to display the score by drawing it onto the screen in the draw function using code like:

```
screen.draw.text(f"My Number is:
{myNum}",topleft=(0,0),color=(0,0,0), fontsize = 30)
```

Play around with the formatting given here. If you want some more formatting options you can go to this link: https://pygame-zero.readthedocs.io/en/stable/ptext.html

Remember you need to make score global at the top of your draw function for this

Part 6: Random

We're going to make a screen that tells you the game is over

Task 6.1: Random!

The next step of our game is to make the y values of our pipes random each time they're created.

1. Under where you import pygame zero and sys you need to import random

Hint

The syntax to import random is a little different from what you're used to. It should say;

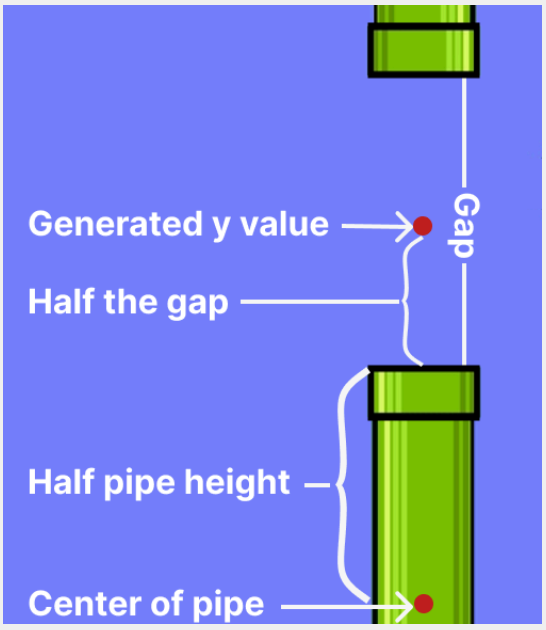
```
from random import *
```

Task 6.2: Y coordinates

We'll need a new way to find the y coordinates of the pipes. Here's how we'll do it

1. First we'll generate a y **value** between two points.
2. Then to find the y coordinate of the **top pipe** we'll use:
   The generated y value - (half the height of the pipe + half the gap between the pipes)
3. Then to find the y coordinate of the **bottom pipe** we'll use:
   The generated y value + half the height of the pipe + half the gap between the pipes

This can be seen in this image.



Task 6.3: Y coordinates pt.2

Let's change our y values to match

1. First you need to make **a gap variable** under where you've made all of your constant variables. It should be **210**

2. Above each set of a top and bottom pipe make a variable that is equal to a **random number between 155 and 445**

3. Where you assign your **top pipes' y values** replace the current number with: the generated y value - (300 + half of the gap)

4. Where you assign your **bottom pipes' y values** replace the current number with: the generated y vale + 300 + half of the gap

5. Where your pipes wrap around,on the line above your loop, pick a **new random number** between 155 and 445.

6. Then inside where you are testing if the pipe is a top pipe (for the scoring) update the **pipe's y value** with the formula for the top pipe

7. Create an else and update the **pipe's y value** with the formula for a bottom pipe

Hint

To find a random number between two values you need to use the code:

```
number = randint(min,max)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 7:

☐ Your pipes are randomly generated

Part 7: Game Over!

We're going to make a screen that tells you the game is over

Task 7.1: Making a game over variable!

We need to store whether it is game over or not

1. In the create constants section under the other variables, create a boolean variable called **gameOver**

2. Initially set gameOver to be **False**

Hint

To make a flag variable it should look like:

```
myFlag = True
```

Task 7.2: Using the flag

We need to store when the game is actually over.

1. At the top of the **update function** above the updating bird comment, make the **gameOver** variable **"global".**

2. Go through the **update()** function

3. Every line that says sys.exit() should be replaced with a line that sets **gameOver** to be **True**.

Task 7.3: Making a game over screen!

If the game is over, it should display a different screen.

1. In your **draw()** function, before the draw background comment, write an **if statement** that tests if gameOver is **True**.

2. If it is, **fill the background** with a solid colour and **display text** that says something like; “Game Over! Your score was:” then their score

3. Make your code to **set the screen** and **draw the actors** be inside the **else** section of this if.

4. Go to the **update()** function and make an if statement so that everything that’s currently in the update function only happens if the game isn’t over.

Hint

To display a single colour as a background the code looks something like:

```
screen.fill((0,0,0))
```

To write text to the screen the code looks something like:

```
screen.draw.text(f"Some Text",topleft=(10,10),color=(0,0,0),
fontsize = 30)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 8:

☐ Your game shows a basic game over screen when it should

★ BONUS 7.4: Funky Screens!

Waiting for the next lecture? Try adding this bonus feature!!

Now that you have your basic game over screen you can change some things and add images or whatever you want to make the coolest game over screen.

For some more text formatting go to this link:
https://pygame-zero.readthedocs.io/en/stable/ptext.html

For some other screen design code go to this link:
https://pygame-zero.readthedocs.io/en/stable/builtins.html#screen

8. Extension: Changing Gaps!

Your game currently only has one possible distance between the pipes, let's use random to change that!

Task 8.1: Making a gap variable

Let's make a temporary variable that generates a random gap size:

1. Remove the **gap variable** from your constants.

2. Directly above where you are generating a center y value, make a new variable to generate a number between 160 and 260. This will act as the size of the gap between those two pipes

3. Where you generate a **y value** you need to change the **minimum** and **max** numbers so they're dependent on it's respective generated gap.
    - **The minimum value** should now be; half the height of the pipes(300) + the generated gap - 250
    - **The maximum value** should now be; 850 - (half the height of the pipes + half the generated gap)

4. Change the top and bottom pipes' y values to use their **generated gap**

Task 8.2: Resetting the gaps

You need to change where you wrap the pipes around the screen.

1. Go to the for loop where you are **resetting the x values** of top and bottom pipes

2. Above where you are generating a new y value for the loop, generate a new gap value. Change the center generation and top and bottom pipes' y values as before

3. **Update** the y coordinate generation and the top and bottom pipes' y values like you did in the last task.

☑ CHECKPOINT ☑

If you can tick all of these off you can work on some other extensions and bonuses.

☐ Your pipes have different sized gaps between them