Extension: Scoring!

Storing a high score in a file so that it remains when the program close

Task 1.0: Setting up

Let's get a new Highscore.txt file!

1. Inside your Flappy_Bird folder, make a new file called "Highcore.txt"

Task 1.1: Setting up pt.2

Let's make a scoring function!

1. After your on_mouse_click() function make a new one called something like scoring()
2. Make a variable up in the create constants section that stores a string that you can display on the screen as it will change based on whether the person beat the high score or not. Call it outText
3. Go to where you are changing game over to be true in your update function
4. Call the function. You don't need to put any parameters in or store a return as there will be none
5. In your draw function where you are displaying text after the game is over, make sure the text this function will generate is displayed

Task 1.2: Reading the file

Let's use our function now

1. In your function make sure that both score and outText are global as we will be using them
2. Open the highscore file in read mode and read the first (and only line).
3. Check if the line is empty. If it is, set a new variable called highscore to 0
4. Otherwise set the variable to whatever is in the line as an integer

Hint

Reading a file line looks like this:

```
with open("myText.txt","r") as file:
    line = file.readline()
```

Task 1.2: Win or Lose

Now we need to work out if the player beat the high score

1. Under the last section, open the file again but this time in writing mode.

2. If the player's score is higher than the stored highscore write the player's score into the file and change outText to something like:
   ```
   f"Congratulations you beat the highscore! Your score was: {score}"
   ```

3. Otherwise, write the previously stored highscore back into the file and change outText to something like:
   ```
   f"Your score was: {score} The current highscore is: {highscore}"
   ```

4. Close the file. This step is really important because without closing the file, it won't write anything into it.

Hint

Writing to a file looks like this:

```
with open("myText.txt","w") as file:
    file.write("This is a line added to the file\n")
file.close()
```

Remember to close your file or it will not actually write to it!

☑ CHECKPOINT ☑

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

- ☐ You are tracking a score
- ☐ At the end of the game the highest score will be added to a file

Extension: Scoring pt.2!

We're storing some scores but let's take a name to store the score under so we can compare personal high scores

Task 2.1: Taking a name

We need a name from the player!

1. In the create constants section, make a variable that will store the player's name as a string.

2. At the top of your scoring function where you're setting score and outText to global, set the name variable to global as well

3. Directly under that use the name variable to take an input asking for the player's name

4. Then make a boolean to store whether the player has won or not and set it to False

Task 2.2: Finding the personal high score

Now that we're going to be storing a name as well as a score we need to change some things.

1. Manually open your highscores.txt file and delete what's already in it as it will cause an error

2. Right before you open the line, create a dictionary called something like scores
3. The key will be the name and the value will be the score so don't worry about repeated keys as we will only ever have one of each name

4. Where you are currently opening the file and only read the first line you'll need to open the file and loop through it line by line

5. Then you will need to check if the line isn't empty and if it isn't, you'll need to strip the "\n" so it doesn't have a newline. Then you'll need to split the line into two sections. The first half (before the ": ") will be the name and the second half (after the ": ") will be the score.

6. Next, before you open the file again for writing, you need to check if the player's name is in the dictionary. If it is you need to check if the player's current score is

higher than the stored one. If it is, change the won flag to be true and change the value in the dictionary to be the player's current score

7. If the player's name is not in the dictionary, add it to the dictionary with their current score and set won to true

8. Then you need to loop through scores and write every line to the file and close the file

9. The last thing you need to do is to change outText based on whether the player has won or not

Hint

One way to read a file line by line, putting each line into a list looks like;

```
data = {}
with open("MyFile.txt","r") as file:
   nextLine = file.readline()
   while nextLine:
       nextLine = nextLine.strip("\n")
       nextLine = nextLine.strip()
       data[nextLine[0]] = nextLine[1]
       nextLine = file.readline()
```

Task 2.3: Displaying the name while typing

We need to use the screen to ask the user for their name

1. Make a flag in the create constants section called something like entered that will be true one the player has entered their name

2. Make a new function called get_name. This function is going to be run a lot and it's going to update the partial name that's being run on the screen.

3. In this function make outText global and then write an if statement to check is entered is False. If it is change outText to be f"What is your name?\n{name}". This is so that on your game over screen the text will ask the users name and update every time the user types a new letter in their name

Task 2.4: Checking if a key has been pressed

Pygame zero makes it very difficult to take input so we're going to do some wacky stuff

1. In your create constants section make a new flag called keyUp and set it to True. This flag is just to make sure we don't accidentally track a held key as multiple key presses.

2. Make two new functions just above your runs everything section. One needs to be `on_key_down()` and the other needs to be `on_key_up()`

3. The `on_key_up()` function is pretty simple. It should make the keyUp flag global, then it should make sure the game is over. If it is, then it should change keyUp to True

Task 2.5: Checking the key

Now let's work on the `on_key_down()` function

1. First you need to make keyUp, entered and name global

2. Next you need to check if the game is over and the key is up and the name hasn't been entered yet.

3. If all of that is true, change keyUp to False.

4. Then you need an if statement to check if the key that's been pressed is the return or enter key. If it is, switch entered to True and call our scoring() function

5. Otherwise, make another if statement to check what key it is. For any of the letters it should add that letter to the end of the name. For a space key it should add a space. For a backspace it should remove the last letter of name. For anything else it shouldn't do anything.

Hint

To test the key pressed you need to write this;

```
if keyboard[keys.A]:
   #returns true when the "a" key is pressed
if keyboard[keys.B]:
   #returns true when the "b" key is pressed
if keyboard[keys.RETURN]:
   #returns true when the enter key is pressed
```

Task 2.5: Cleaning up

We are now calling our scoring function in places we shouldn't be and we're not calling our get_name function at all so let's fix that.

1. In the update function where you are calling scoring(), delete it. The only place that should call scoring() is when we make entered true in the on_key_down function

2. We need to call the get name function in our draw function, only in the section for when the game is over

Run your game! It should take a name now

★ BONUS 9.6: Extra special characters!

Now that you are testing for the basic lower case letters and space and backspace, you can add as many characters as you want. Here are some of the most common ones;

BACKQUOTE	`
MINUS	-
EQUALS	=
BACKSLASH	\
QUOTE	“
COMMA	,
PERIOD	.
SLASH	/
SEMICOLON	;
0 … 9	K_0 … K_9

You can also test for the left or right shift (keyboard[keys.LSHIFT] or keyboard[keys.RSHIFT]) at the same time as a key to recognise them as capitals.