Pygame Zero - Space shooter

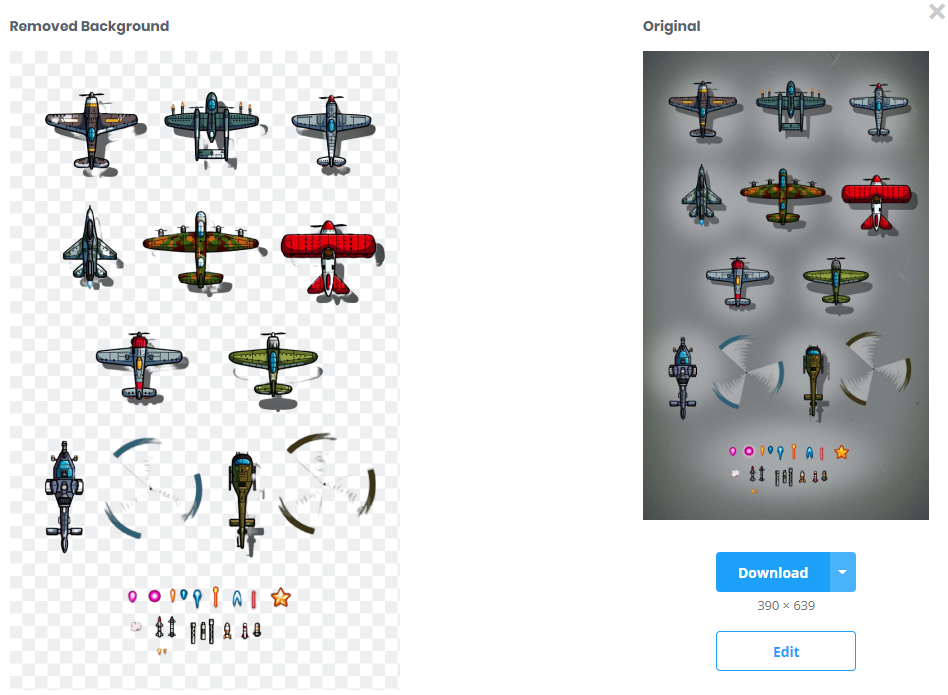
## Design Game

## Prepare Resources

* 1. How to find image ?
     1. Google image “plane game cartoon” and download ( right click to save image )
     2. Game needs png image file ( transparent background ); so use online tool to remove image background: <https://www.remove.bg/zh>

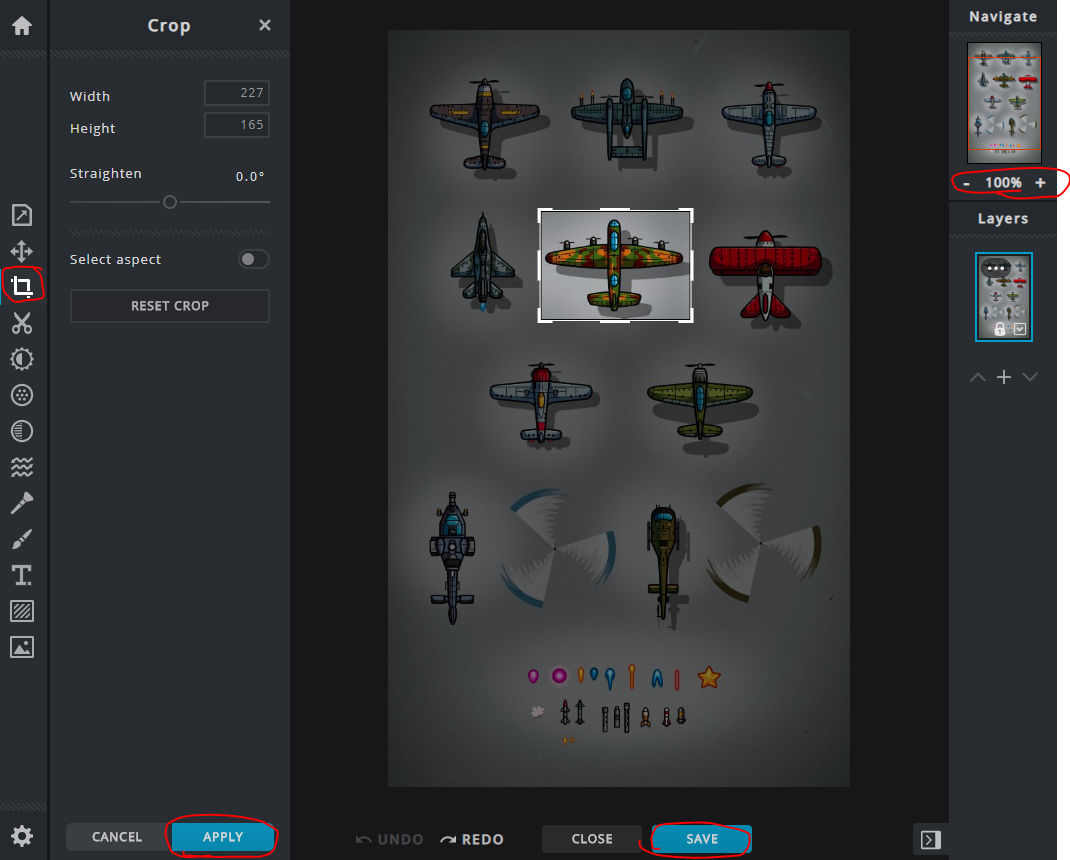
Open/Drag and drop downloaded image into page, it will automatically remove the background, download the updated image in png file

Here’s the example screenshot:

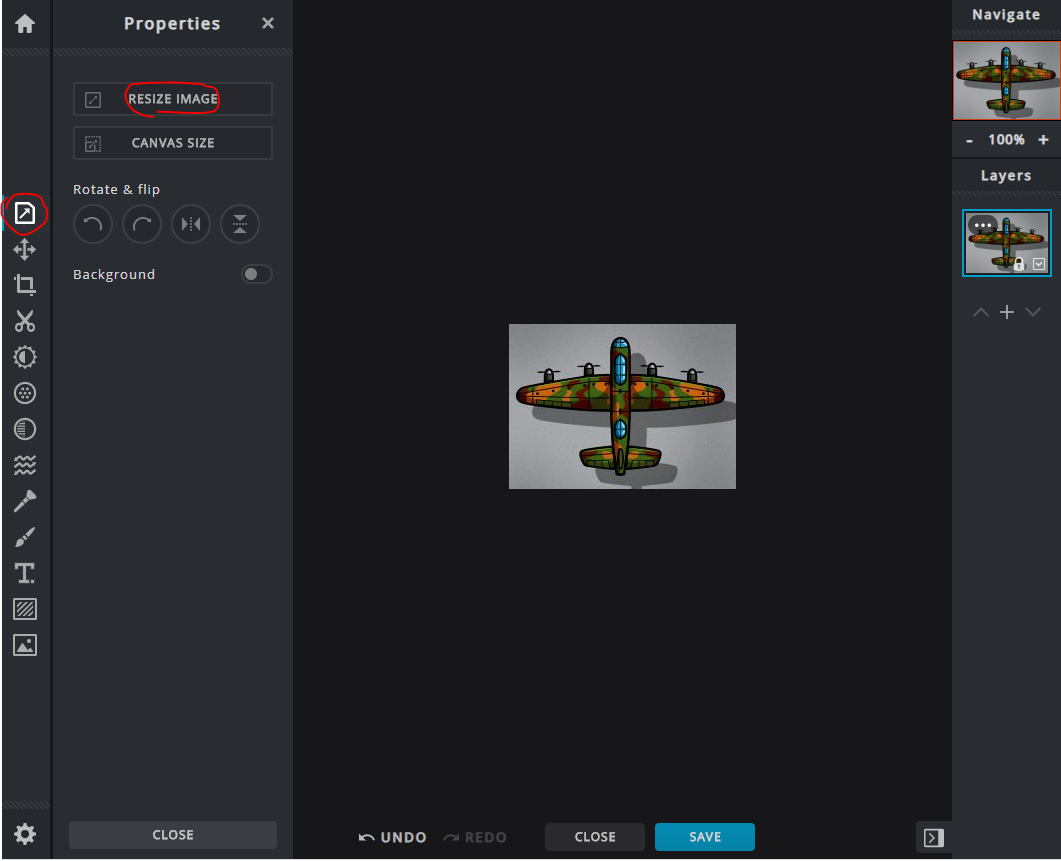


* + 1. If we still need crop and change the image size, we can use Image Online Editor: <https://pixlr.com/x/>

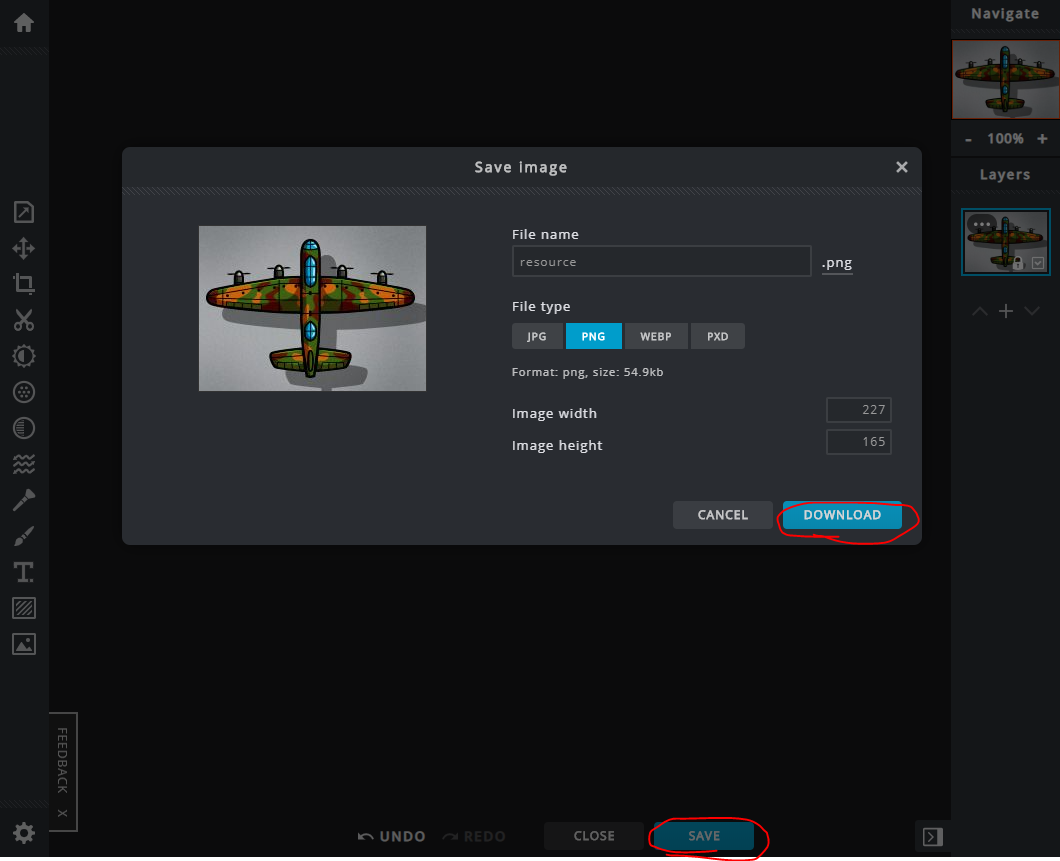
Use CROP to take part of the image



Use Properties to resize/rotate/flip the image

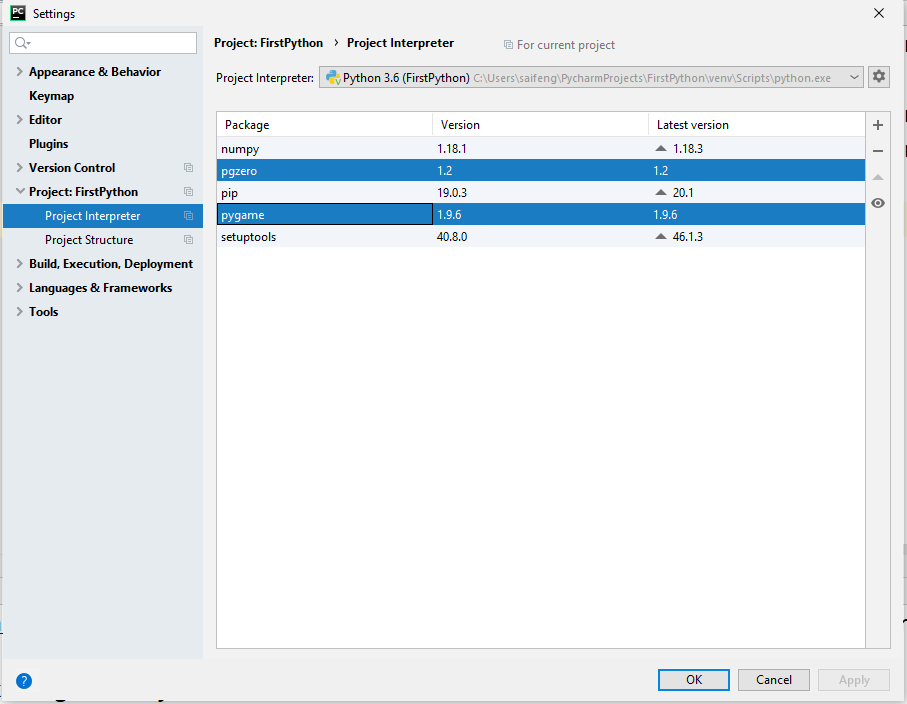


Save and download as png file

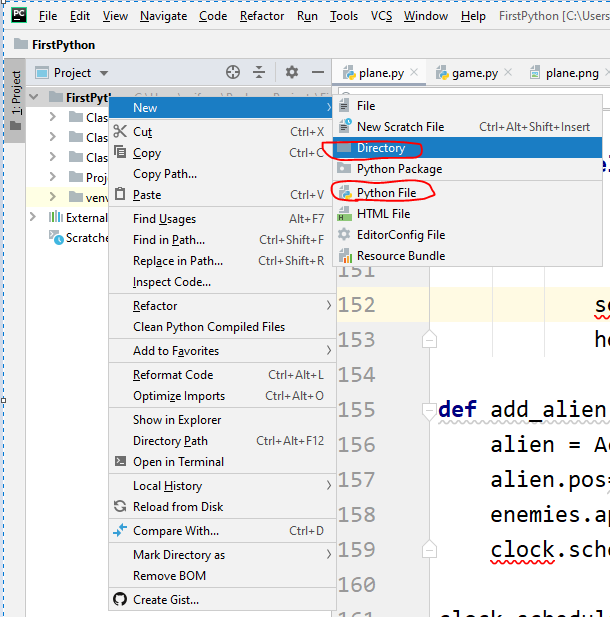


## Create your python game project

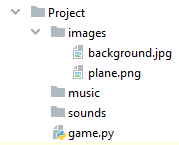
1. Get your development tools ready
   * 1. Python Editor like pycharm: <https://www.jetbrains.com/pycharm/download/>
     2. Install pygame and pgzero



1. Create your project folder by right clicking the destination folder you want to create in



1. Create your project folder structure as below
   * 1. images folder containing all images used in game
     2. music folder containing background music repeat playing
     3. sounds folder containing short sound effects
     4. game.py is the main python file to start the game



## Write your game python code

### Part 1: Create background and main character

**Requirement**: Start a game with background image and character in screen

* + 1. **Setup the size of game area**

Use built-in const variables: WIDTH and HEIGHT to set your game area size (fit the size of background image)

* + 1. **Define the Actor object**

Use Actor object to create image variable mapping to picture

1. The image file has to be placed into images folder
2. The image file name has to be lower case letter, numbers or underscores

plane = Actor(‘image\_file\_name’)

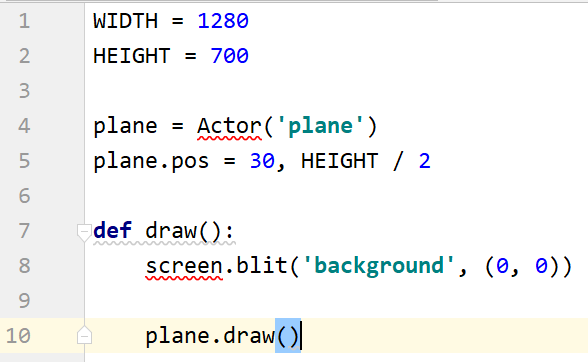
plane.pos = x, y (x, y is the coordinates in screen)

* + 1. **Define the draw function**

draw function is built-in function, which will be invoked by game engine as entry.

Every time, the objects are changed, the draw function will be called to redraw them in screen.

1. Screen is built-in variable to control screen. blit function is to put background image, which is file from images folder, with (0 ,0) is the coordinate of left-up corner of the image
2. plane.draw(); call draw function of Actor object to draw things in screen.
   * 1. **Sample codes for Part 1**



### Part 2: Control your object by using keyboard

**Requirement**: Use keys to control your object to move up/down/left/right

* + 1. **Define the update function**

Update is a built-in function, which will be called 60 times per second. In where, you can change the properties of objects, like color, position or even image file.

* + 1. **Keyboard event**

Keyboard is a build-in object to listen to any events from key press.

Use if condition to check which key is pressed and change the x or y of object, which will be updated in screen in draw function.

Move bigger the value is changed, how fast of the object will be moving in screen.

* + 1. **Sample codes for Part 1**

## 

### Part 3: Control music & sounds in your game

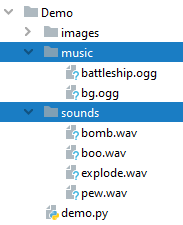
**Requirement**: Add background music in your game and sound effects when events happen such as shooting bullets

* + 1. **Prepare your game music & sounds resources**

All resources should be placed in the correct folder and with lower case file name

Sample website for free music: <https://patrickdearteaga.com/royalty-free-music/>

Sample website for free sounds: <https://www.freesoundeffects.com/>



* + 1. **Coding your music and sounds**

Music: play longer track, such as background music

**music**.play(“filename”)

**music**.pause()

**music**.stop()

Sound: play shorter track, such as sound effect

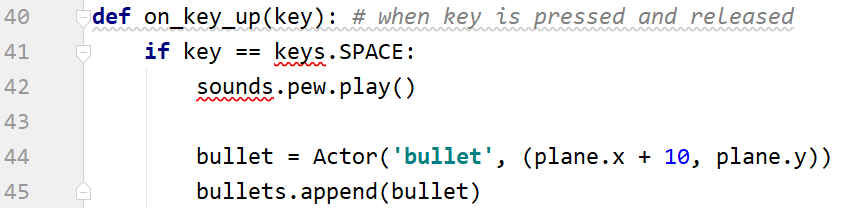
**sounds**.filename.play()

### Part 4: Add bullet when key is pressed

**Requirement**: Shoot bullets when key is pressed

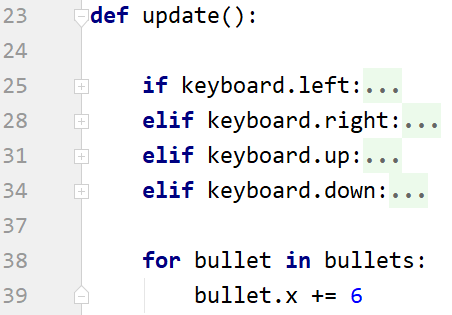
* + 1. **on\_key\_up listen to key release event**

Check which key is released: key == keys.SPACE



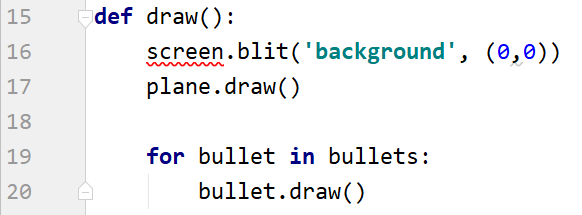
* + 1. **Create a bullet object and bullets list**

1. Define global list variable bullets = [], containing all created bullets
2. Create Actor with image name and position (x, y)
   * 1. **Change the position of bullets**
3. In update function, which is invoked by engine 60 times per second
4. Change bullet position to move bullet to right



* + 1. **Draw bullets**

1. Draw bullet in game area



### Part 5: Add aliens to your game

**Requirements**: Randomly display aliens in your screen as your enemies

* + 1. **Add clock schedule**

Often when writing a game, you will want to schedule some game event to occur at a later time. For example, we want to add a new alien into screen every 5 seconds. So we can use the clock object to schedule a function to happen in the future.

**clock.schedule\_interval(callback, interval)**

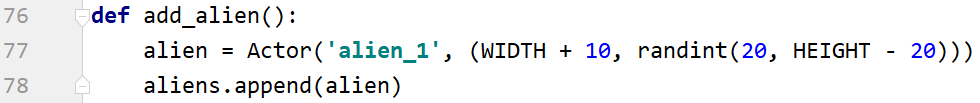
Schedule callback to be called repeatedly.

Parameters: callback - A callback function that takes no arguments

Interval - The interval in seconds between calls to callback



* + 1. **Create a new function to add alien**



\* make sure you have alien picture ready in your images folder. If you can’t make your own, you can download from my shared driver: <https://drive.google.com/drive/u/0/folders/1XneVFsUj8TrzYt8pftxNVoZlUm75ES4i>

\* the coordinate value **(WIDTH + 10, randint(20, HEIGHT - 20))**, is make sure the alien is appear beyond the right side of game area and position randomly between top and bottom at the beginning.

\* Don’t forget to import randint at the beginning of your codes

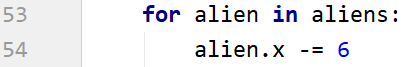


\* Create a global list variable aliens above draw function, which is hold all your alien in screen.

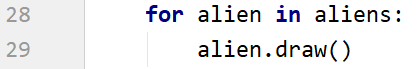


* + 1. **Update the position for each alien in update function.**

Please refer to the same logic for bullet.



* + 1. **Don’t forget draw alien in your draw function**



### Part 6: Check bullet collide with alien

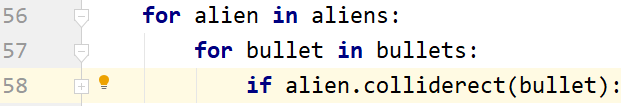
**Requirements**: When bullet hits alien, it should be removed from screen after explosion

* + 1. **Use colliderect to check if 2 objects hit**

To test if any 2 actor objects collide with each other, we use colliderect() function. For example

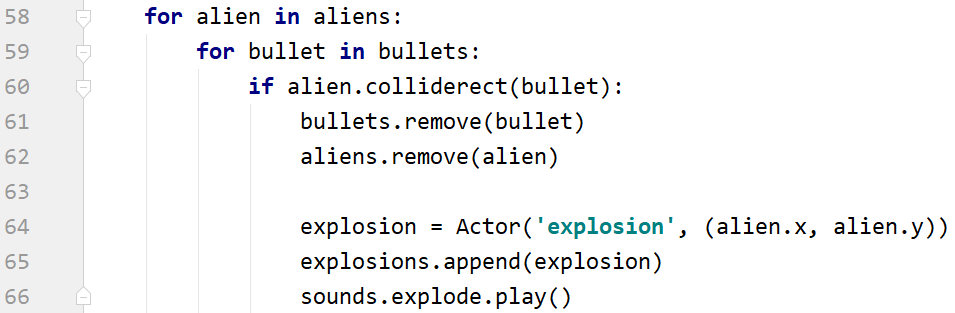
object1.colliderect(object2)

In our game, we need check if any bullet hit any alien, so we need have nested loops ( loop in loop ) to check every possible collision between object in 2 lists



* + 1. **Remove the alien after explosion**

Once a bullet hit one alien, both bullet and alien should disappear from screen. For which, we can just remove them from each list; Meanwhile, we need display explosion effect, which is just display explosion image and play sound



\* We should put the check in **update** function, since the check is happening all the time

\* remove hitting bullet and alien from lists, so they won’t be drawn in screen anymore

\* create explosion image and add into explosions list; The position of explosion should be exactly the same as last position of alien.

\* Make sure you have explosion image and global variable explosions created

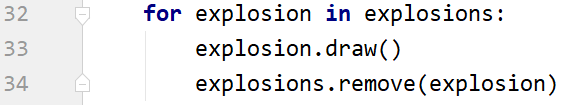


\* sound explode play. Make sure you have explode sound file sounds folder

\* You can find explosion image and explode sound by yourself online. If not, you can download from my google driver: <https://drive.google.com/drive/u/0/folders/1XneVFsUj8TrzYt8pftxNVoZlUm75ES4i>

* + 1. **Remove explosion after displaying**

To display the explosion in screen, don’t forget the draw it in draw function. But after first draw, you should remove it immediately since it’s one time displaying.



### Part 7: Add score

**Requirement**: We need display score in screen to showing how many aliens got killed.

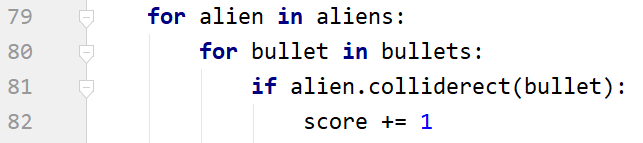
* + 1. **Introduce score variable**

We need create a global variable to hold score value, could be initialized with value 0; Global variable is defined outside any functions and accessible in any functions



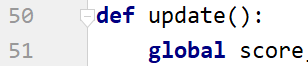
* + 1. **Update the score value**

Increase the score value every time kill the alien ( bullet hits the alien )



\* In the update function, where you already check the collision, add score += 1 in the same condition.

\* Don’t forget to global score, since you’re changing the value of a global variable in the function



* + 1. **Display score value in screen**

Display the score in the screen in draw function



\* The text function contains 3 parameters

1. String displayed in screen, make sure you type casting score to string variable
2. Attribute color = “white”
3. Attribute position: topleft=(10, 10); which is related to the game area

### Part 8: Add game\_over

**Requirement**: When plane hit the alien, game should be over. All characters should be removed and final score is displayed

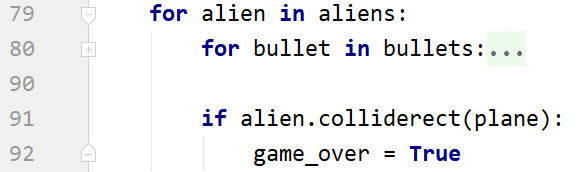
* + 1. **Introduce game\_over variable**

We need have a variable to hold the current game status, which has to be global variable since it need be used in multiple functions



* + 1. **Change game\_over in some cases**

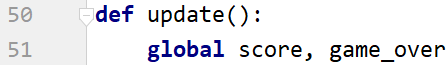
When alien hits plane, game should be over. So game\_over value should be changed to True.



\* The collision check could be placed in the update function, where we’re checking collision between bullet and alien.

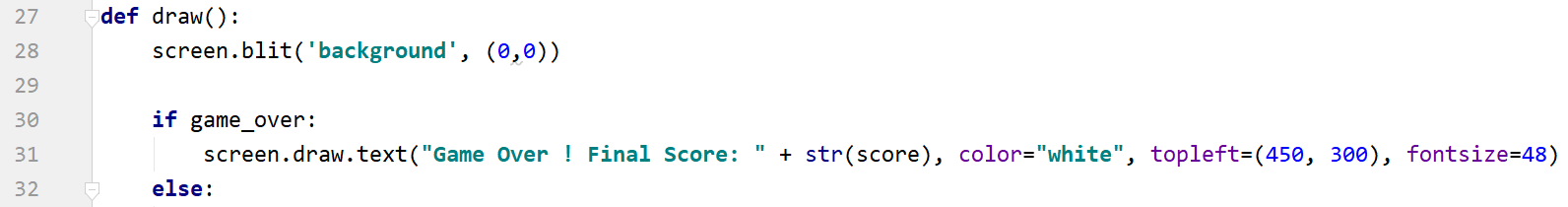
\* alien.colliderect(plane) check should be included in the loop of aliens, make sure the indentation is correct.

\* We’re changing the value of a global variable game\_over, so don’t forget the global reference it at the beginning of the update function.



* + 1. **End game and display final score**

In the draw function, when engine redraw objects in screen, we should firstly check if game is over; If yes, then no need to draw anything but “Game Over” and final score text in screen. If game\_over is False, keep redrawing all objects in screen.



\* Only draw objects if game\_over is False

\* screen.draw.text to have 4 parameters

1. Text to be displayed
2. Color attribute
3. Position attribute, place center
4. Font size attribute, make the final message bigger

## Test your game

1. In your terminal, use command “pgzrun game.py”
   * 1. Make sure you’re in the correct directory to execute the command. Use cd to change the directory; For example “cd ..” is to back parent directory, “cd Project” is to get into the Project folder
     2. Or you can select the python file, right click to select “Open in Terminal”, which will open a new terminal with the correct directory
     3. Press enter to execute the game

