ciao è Mario

The game has been constructed using five different python files, few images. The latter is stored in its respective folder. That path is defined using the feature import of python use the feature of path from OS. Having all the code in the file is a boring job. The file becomes lengthy and is much more difficult to debug the single code. Importing required files in other files eases our job.

The code contains comments wherever required so that even a noob coder can understand the code ;).

Coming to the files, there are five of them, namely “settings.py”, “main.py”, “sprites.py”, “start.py”, “end.py”. Explanation for them is follows :

Settings :

In this file, all the basic settings of the game like the width and height of the window are defined. This file is imported in the remaining two important files of “main.py” and “sprites.py”. This decreases the effort of the programmer to define those settings again.

This file also has some lists that contain the location of sprites on the screen, that are blit on the screen when the main file called. This is explained further in the ‘main file’ block.

The settings file need is the path module from the os library so that the image directory is defined.

To start with the gameplay, we had to think of a map. Hence we first laid out a map with the positions of the flame, pipes and the pyramid! We inserted lists of bricks, clouds, fire, pipes and coins. The location of clouds and coins are generated randomly. For the bricks and fire, the locations are appended to their respective empty lists which are blit later. These locations are predefined and not random.

So if the settings file is interpreted basically there is no output in the terminal or the window.

Start :

This file has all the classes and functions needed to show the start screen. This file is imported into the main file. This file has some variables defined for the start screen image, the box as a bullet, and a couple of text sprites. Now these are added to a group of sprites which are later drawn on the screen. All this is defined under a

function which is called inside the main file.

Sprites :

The sprites file is very important. This imports all the content from settings. This file mainly consists of the classes of the character(player), the bricks, the pipes, the flag etc etc.. Each class can have their specific methods so that the required thing can occur. This can occur once called in the main file. The classes use lot of variables defined in the settings file which is imported. The inbuilt functions for sprites and pygame are being used to assign images to variables which are made into rectangles to locate them easily.

We used the vector feature of pygame version 1.9.3 which made the tracking of position and velocity of the character easier without using extra variables and complicating the code further.

Main :

The main file is, as the name suggests is the main file. Therefore for the code to run, the main file has to be executed. This has a single class named the game loop. This has methods that are required for the code to run. This also has an exclusive method only for collisions and scrolling. Which also counts the score that is later shown.

All the sprites are added to their respective groups and later blit or drawn on the screen.

As the start and end files are imported, there are exclusive methods only to show the start and end screen.

Now a variable name g is defined as an object the class Game and the required methods are carried out. Thus the game runs from the infinite while loop containing the step by step methods from the class.

We used collision feature of pygame which ensures collision between two classes of sprites. This can be used in many ways. If the character collides with flag, the game must end, or if the character collides with coins, the score must increase and the coin must erase off. It is also used when we checked the condition of the player colliding with the pipe and not moving through it. That was done by some intelligent code.

Now comes the scrolling part. We are not actually scrolling in the game, but moving all the objects away from the screen relatively. This gives us a look that all the player is moving. But actually only the remaining objects are moving and looks like scrolling.

When the game ends, we used an ‘intelligent’ timer to wait for an input for 5 seconds from the user, if there is any, the game respawns else it quits automatically.

End :

This is one of the final file that contains classes related to the end

screen showing the score. This is imported into the main file to get the end screen after the game is done.

The file also has an important function that prints texts on the screen while playing the game that is used to give the instructions to the user and also displays the score.

Here ends the explanation of the code.
So rub your hands and get ready to get into the mario world,
AND
try not to jump into the flames!!

Happy gaming! ;P

References :

Code was done with some help from a youtube channel ‘kidscancode’. The videos subsequently had there code uploaded in github, reading and understanding the code helped us a lot.

Images were taken from ‘Google’. The sprites were some altered in GIMP and others were directly found.

A big thanks to ‘Pokemon FireRed’ to give us the suitable background music for our game, and also the player sprite!

GitHub repository :

CHEERS!