For my final project, I have created a snake game. The goal of this game is to increase the score by eating the pixels that randomly show up on the screen. If you maneuver the snake into its body, you lose. Also if you maneuver the snake into the border of the screen, you lose. To maneuver the snake, you need to press the arrow keys (←↑→↓) or use (WASD). Pressing the "up arrow" (↑) or "w" makes the snake go up, the "right arrow" (←) or "a" makes the snake go left, the "left arrow" (→) or "d" makes the snake go left and the "down arrow" (↓) or "s" makes the snake go down.

Creation of a window

The creation of the game required the pygame module as well as the random module. Pygame is a module that helps create games using the python language. In order to create a window using pygame, I assigned a variable with the following code `screen=pygame.display.set_mode([screen_width,screen_height])` the "screen_width" and the "screen_height" are also variables with integer numbers associated with them representing pixels. After running the code, a window will pop up for a second then close, so to fix this, a while loop is needed. All over my code, you will see `while not exitgame:` "exitgame" is just a variable given the boolean value of "False", therefore making it a constant loop. To change the color of the window, pygame has a `.fill()` method which changes the color according to its RGB value in a tuple. This is why there are different variables of color names with their associated RGB values. To get RGB values of colors I used the website, https://www.rapidtables.com/web/color/RGB_Color.html. In addition, `pygame.display.flip()` is a method that is needed in order to update what is displayed in the window, this should also be in the while loop.

In order to rename the window I simply used the set_caption method in pygame the code looks like: `pygame.display.set_caption('Snake Game')`

In order for text to show up on the window, you need to use a method `screen.blit(text, [x,y])` where the text. I followed the code in this website https://www.geeksforgeeks.org/python-display-text-to-pygame-window/. Firstly, we need to get the font and the pygame has a built in font method that can get any font. For this project I used "None" which is just the default font, so I used `font=pygame.font.Font(None,45)`. The second parameter is the size of the font that will display on the screen. Then there is a render method with the font class that will change the wanted message to the desired font and will change the color to a desired color using RGB values. `text = font.render(msg, True, color)`. Lastly, a `screen.blit(text,[x, y])` method is needed to display the text on the desired window. The first parameter is the variable containing the wanted message with the font and the second parameter are coordinates of where the text will be displayed on the screen. I realized this is a lot of repetition throughout the code when I wanted to display multiple text on the screen, so I made a function `def screentext(msg, color, x, y)`.

Adding images to the display I followed the code on this website: https://stackoverflow.com/questions/28005641/how-to-add-a-background-image-into-pygame. It is very similar to displaying the text on the screen, except now we need to get the image from a file, so I used the `image=pygame.image.load('how to.jpg')`. The parameter is a string of the title of the wanted picture with the .jpg. The image variable now contains the image and in order to display it on the window, we need to use the blit method once again: `screen.blit(image,[x, y])`. This works exactly like the way we put text on the screen.

The game (mechanics)

The way I made the game was by using coordinates. The top most left of the screen is (0,0) and the bottom right of the screen is the (screen_width, screen_height). Similar to what we say in assignment 1 of the course. To get inputs from the user, inspiration was taken from this website https://stackoverflow.com/questions/28005641/how-to-add-a-background-image-into-pygame. An iteration through the list of Event objects is needed in order to get specific inputs. `for event in pygame.event.get():`. Then "if" statements will be required to check the events. The different types I used in my project were `event.type == pygame.QUIT` and `event.type == pygame.KEYDOWN`. ".QUIT" is if the user clicks the "x" button of the window and ".KEYDOWN" is when the user presses on a key from the keyboard. I wanted to check the inputs of the Event object which has a .key method, I referred to the pygame documentation of keys https://www.pygame.org/docs/ref/key.html. I set movement as using the "WASD" or arrow keys, in the losing screen press "enter" to play again and the escape key will always exit the game.

Drawing of snake and food

In game variables there are two variables: `snake_x` and `snake_y`. These are the initial coordinates of the snake. There is also an `fps` variable which has an integer value associated with it, this is how many times the display is updated per second. In order to move it, I set a `initspeed` variable which causes the snake to move 4 pixels per tick in the game. So everytime a movement key is pressed, the snake will either move in 2 directions which makes the movement easy, it will either move in the x-coordinate or the y-coordinate. Thus, if the user presses the up arrow for example, the snake moves up so with this coordinate system in the negative y-direction, and now we can set `speedx` = 0 and `speedy = -initspeed`. To move the snake, we add to the initial coordinates of the snake the speedx and speedy values to their corresponding coordinates.

Food also has two variables: `foodx` and `foody`, but I wanted them randomized so I used the random module to generate a number between 0 and the maximum height for the `foody` and 0 and the maximum width for the `foodx`.

Pygame has a draw method which I used rectangles for both the food and the snake. `pygame.draw.rect(screen, GREEN, [foodx, foody, 20, 20])`. The first parameter is the display, color, a list in which the first 2 indices are the coordinates of where the rectangle will be drawn and the last 2 are the sizes of the rectangle with x and y values.

Increasing score

Since everything is revolved around coordinates, to check if the snake interacts with the food, I just used math, in which the subtraction of the snake_x and the foodx as well as the subtraction of the snake_y and the foody will be 0 if they are directly on eachother. I realized that the hitbox didn’t really register correctly as the size of the food particle is [20,20], so I implemented a less than comparison which will give a more reasonable hit box. Everytime this if statement is true it adds to the counter. And a new `foodx` and `foody` coordinate are generated. Also adds to the snake length.

Length of snake

I had no idea how to increase the length of the snake while making the whole snake move at the same time. I was struggling with the fact that my snake would increase in size, but it would be a long line rectangle that would move and the tail would keep trailing. Meaning that eventually my whole screen would turn black and cover everything. I consulted this website https://www.edureka.co/blog/snake-game-with-pygame/, and they created a new list containing the head coordinates of the snake and appended the list to another list: body. Then if the length of the body list is greater than the snake length, it would delete the first index in the body list, which represents the tail. They had also made a function to draw the snake on the screen, since the snake is updating every tick, it needs to draw a new pixel every updated coordinate, this iterates through every coordinate of the body, therefore making the snake move as a whole without leaving a trail. This successfully grows the snake and makes the snake move as a whole rectangle.

Losing

Need to check that the snake doesn’t run into itself or else you lose, so if the list of the head is in the list of the body, then the game is lost. Also if the snake in the x is greater than the screen_width the game is lost and so on. This just represents the borders of the game.

Multiple screens

Took inspiration to make a welcome screen, and the game screen. If user presses enter then it does game.

https://stackoverflow.com/questions/13801828/how-to-create-a-play-again-option-with-pygame

High score

Opens a hs.txt file and writes the high score if it is bigger than the original. Everytime game is running, it will display the highest score.