Now**,Creating The Head Of Our Snake ,**For that first we will make three game specific variables:First two are x and y co-ordinates of our snake’s head in gameWindow and “snake\_size” is length and breadth of our snake’s head.Than, we need to make rectangle with this data! For that we will use “rect()” function. It takes surface, color, rect as argument. In rect we gave a list in which we told where the rectangle should be placed(co-ordinates) and, length and breadth of the rectangle.

**Moving Our Snake And Setting Game FPS,for that clock,** **W**e will make a clock. It is a class in “time.py" which has a “tick” function. “tick” takes framerate as an argument. It updates the frame according to the frames we want per second. When we click on a key, we want our snake to move continuously with a defined speed, clock does that.

Similarly, we will make “if” blocks for every direction! We will have total 4 “if” statements(for each direction)

For every key there is an identifier kind of thing which verifies this key is pressed. We simply checked which key is pressed and added/subtracted in x/y co-ordinates according to that!

Now,Giving Speed To Our Snake In X and Y directions Speed means it is covering distance. In our game distance is in co-ordinates(x and y) which are snake\_x and snake\_y. So it just means that we have to keep increasing values of co-ordinates like it’s in a loop. We already are in a “while” loop so we just have to increment values, but we also have to see how much and in which axis to increment? For this we will make two variables:velocity\_x = 4velocity\_y = 4Now we just have to increment

Next issue is that it is moving straight and diagonally, we want only straight. There is a simple solution for that, diagonal movement happens when there is more than one force applied. Here are two, “velocity\_x” and “velocity\_y” so just null the other velocity when one is in play!

Creating Food For Snake To Eat Getting random number:

To get a random number we will use random module’s “randint” function. “rand” stands for random and “int” means integer. It takes a range(start and end) as argument and returns a random number between that range!

For food we are just making a red rectangle like we made black rectangle(snake’s head).

**Adding Score, Replotting Food & Changing Size**, We simply saw the difference between snake’s co-ordinates and food’s co-ordinates, if they are lower than 6 then it means that they are kinda overlapping and it will add and replot food. In “if” block we will add score and replot food as both things should be done when overlapping happens! “abs” function returns absolute value which in simple term means it will return positive value. We just have to keep doing it whenever it overlaps which means, a loop! For that we just have to bring that code(to plot food) in game loop. Also, it should replot only when it overlaps so it will go in “if” block!

Increasing snake length:

We will make two variables “snk\_list” and “snk\_length”. “snk\_list” will be a list of list. It will have co-ordinates of the snake’s rectangles. “snk\_length” will have an integer and we will increment it’s value everytime our snake eats food.

**Continue…..**

Game over:

There are only two reasons for game over: 1)Hit wall 2) Collision

Hit wall:

There are 4 walls so 4 possibilities/4 ways to hit. We will make an “if” statement for this and “if” statement will have 4 conditions(one for each wall):

We are saying, if snake’s co-ordinates(x and y) are less than 0 then game is over and if they are greater than screen width then also game is over. Our game screen is between 0 to screen\_width so if snake is on any other co-ordinate(less than 0 or greater than screen\_width) then it’s out of screen. Hence, game over!

Right now we just changed a variable from “False” to “True”, we haven’t done anything to make it stop the game. For that we will write one more “if” statement which says if game\_over is “True” then fill the screen with white and write “Game Over! Press Enter To Continue”! In previous blog we made “text\_screen” function to display score and we made “red” color/variable in blog 9. It’s not over yet tho. Rest of the code is below this “if” statement so that is playing as well so we will put all that in “else” block but then we will see one more problem. Events are in “else” block so we won’t be able to call “QUIT” event. For that we will add necessary events in both blocks!,Here necessary events are “QUIT” and “K\_RETURN”. “K\_RETURN” is for enter, so that we can restart the game and you can see we used “gameloop()” function for restarting the game as I mentioned in the start.

Collision:

For collision we will do a very simple thing. We will write an “if” statement which will check if variable “head”(it has real time co-ordinates of snake’s head) is in “snk\_list”(co-ordinates of all rectangles in snake).

What is collision? In this game it just means that snake’s head hit any part(rectangles) of snake! So we are simply seeing if snake’s head and snake’s any other body part is on the same co-ordinate. All co-ordinates are in “snk\_list” and snake’s head’s co-ordinates are in “head” variable, they are also in “snk\_list” so we are just seeing if except “head” there is any other rectangle on the same co-ordinate! With list slicing we are checking all co-ordinates except head’s co-ordinates!

Highscore

To display it first we have to store it somewhere and it’s obvious we can’t store it in the same python file as it will reset everytime we run the file. So we have to make a new file and store it there. There will be three tasks:

Reading high score from file

Displaying high score in gameWindow

Writing/storing new high score when game is over

We gave two arguments to “open” function, one is file name and second is mode. “r” means read mode, then we stored file data in “f”

So now high score will keep changing when score is greater than high score. Now we need to display it! The way we displayed “Score”, we used “text\_screen” function, in the same function, in the same string, we will give high score

Here we are opening in “w”(write) mode. It overwrites previous data so “30” in file would be erased and new high score will be written.

Adding Music and Background Image In Pygame

In this blog we will learn to add music and background image in our game! But first let’s fix a bug. Bug is that if “hiscore.txt” is not in folder then it crashes. So we have to code it to make “hiscore.txt” if it’s not there!

Making file automatically:For this first we will import “os” module, to check if it’s already there in folder or not. But let’s first import “os” module:

Now, we will use an “if” statement to check if the file exists and create if it doesn’t:

If file doesn’t exist “os.path.exists” will return “False” but if it doesn’t exist then we want to create file and to do that we have to go in “if” block so we used “not”! To make “False” into “True”, then we opened file in “w” mode because in write mode, it automatically makes a new file!

Adding music:

First we have to initialize mixer:

then we have to load sound:

now we play it:

There is really nothing more to it. It is that easy! Just put this line in your code, wherever you want to play it, like this:

This is just before starting the game!

Background image:

Like there was a “load” function for loading music, there is a “load” function for loading image too:

Now, we will transform image to screen height and width:

“scale” takes 2 arguments, one is image and second is a tuple, tuple will have 2 values, new height and width of image(transformed).

It still won’t show the image because we haven’t written anything to display it. It’s like we have made variable and saying why isn’t it printing? So to display it, we will use “blit”. Last time we talked about “blit” was in blog 17. Code to blit:

It takes 2 arguments. One is image and other is co-ordinates, co-ordinates of where to place it!

And it’s done!