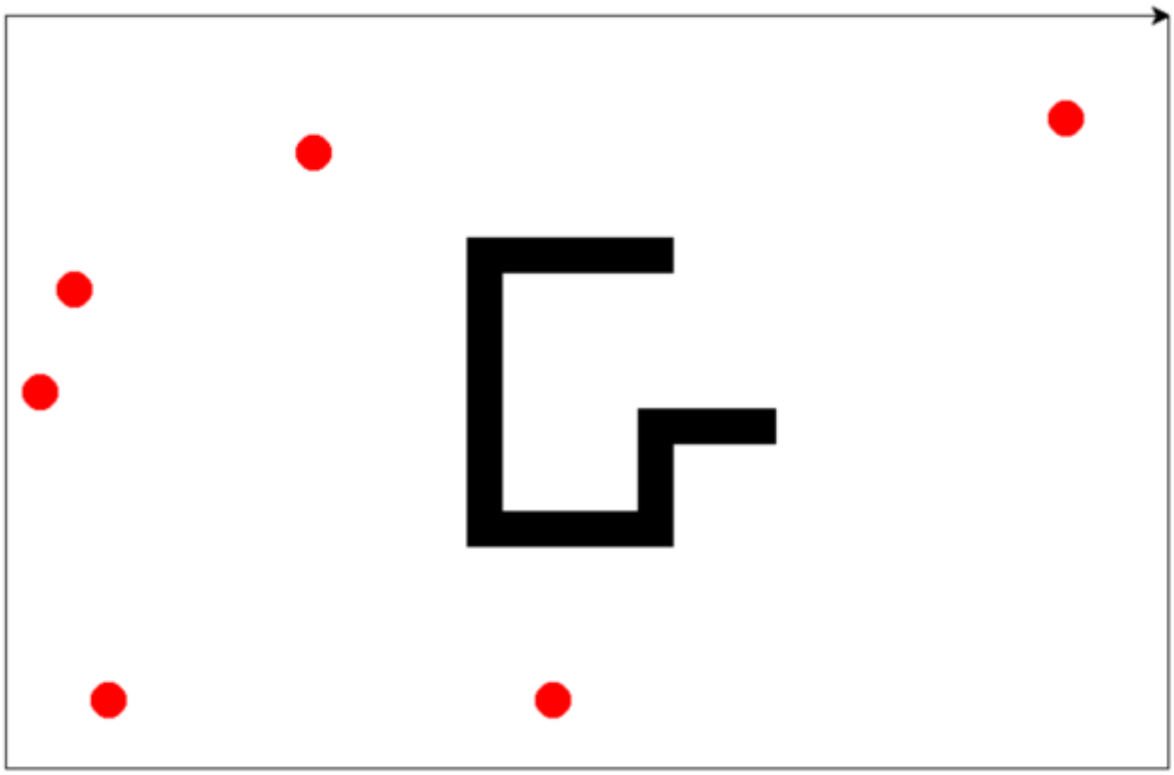
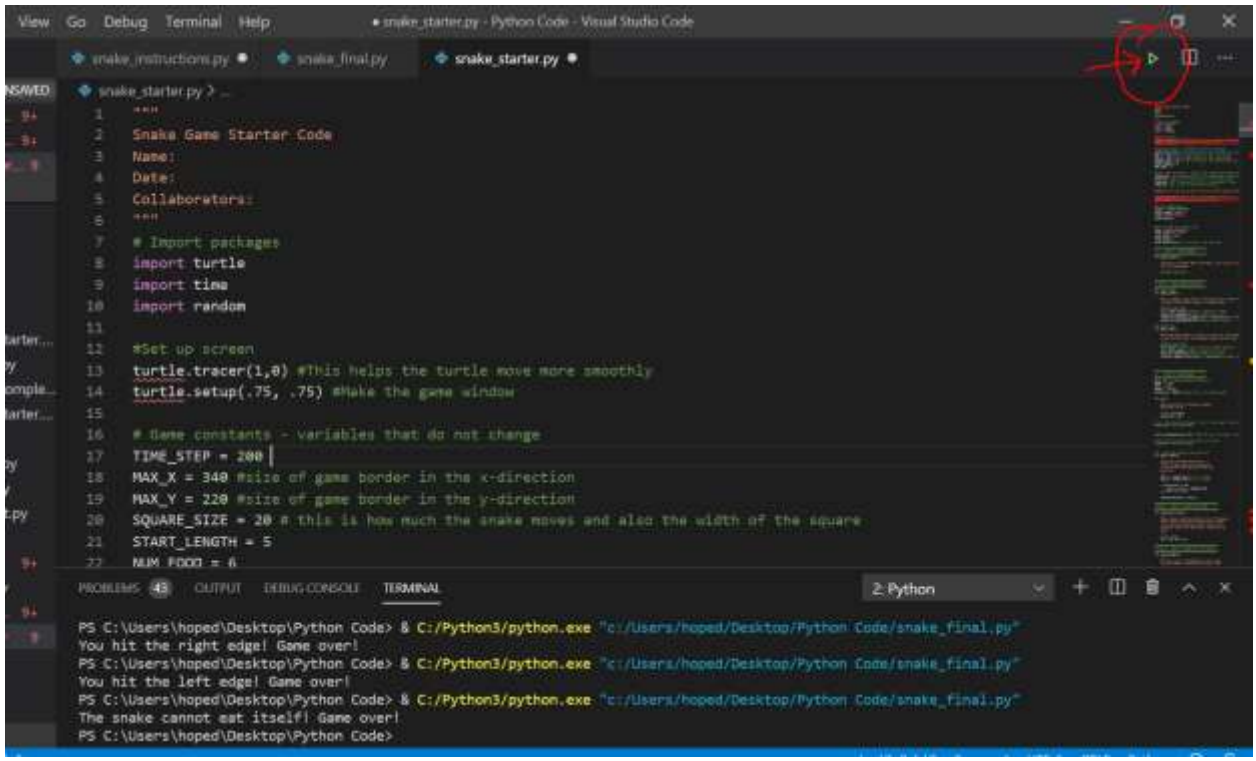


SNAKE GAME INSTRUCTIONS

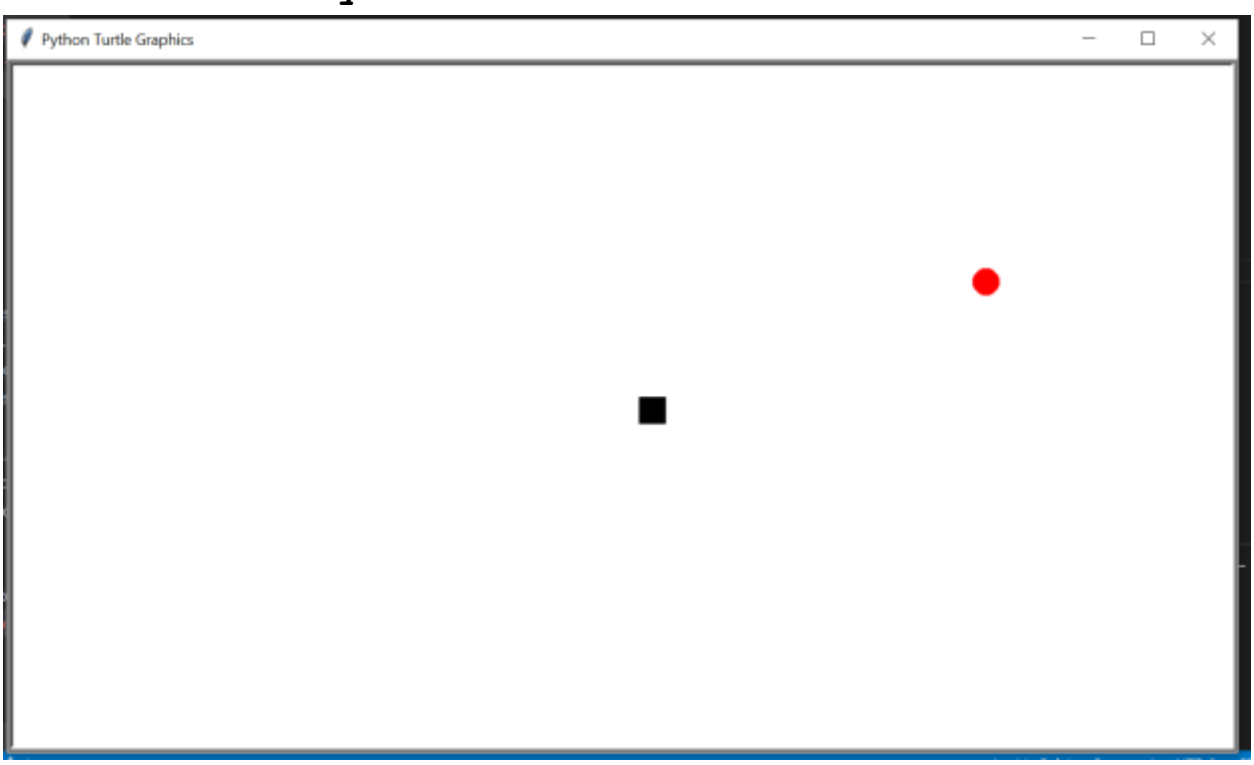
WALKTHROUGH VIDEO

Step 0. Create a new file [0:00 in video]

- On the computer, open up Virtual Studio Code
- Create a new file.
- Save the file as **yourname_snake.py** Make sure to save it in the desktop folder "Python Code".
  PLEASE INCLUDE YOUR NAME IN THE FILE NAME!!!
- Copy and paste the starter code into your file. Run the file to make sure that there are no errors!



When you are done with step 0 - the screen should look like this when you run the code:



REMEMBER TO PUT YOUR NAME AT THE TOP OF THE FILE

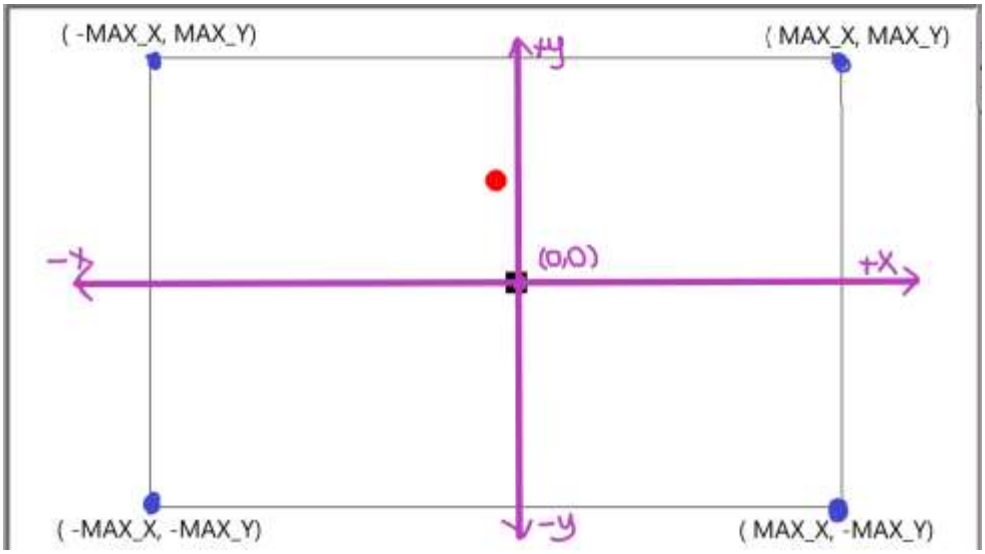
```
"""
Snake Game Starter Code
Name: YOUR NAME
Date: THE DATE
Collaborators: WHO ARE YOU WORKING WITH? (CAN BE NO ONE)
"""
```

STEP 1 - MAKE BORDER [2:39 in video]

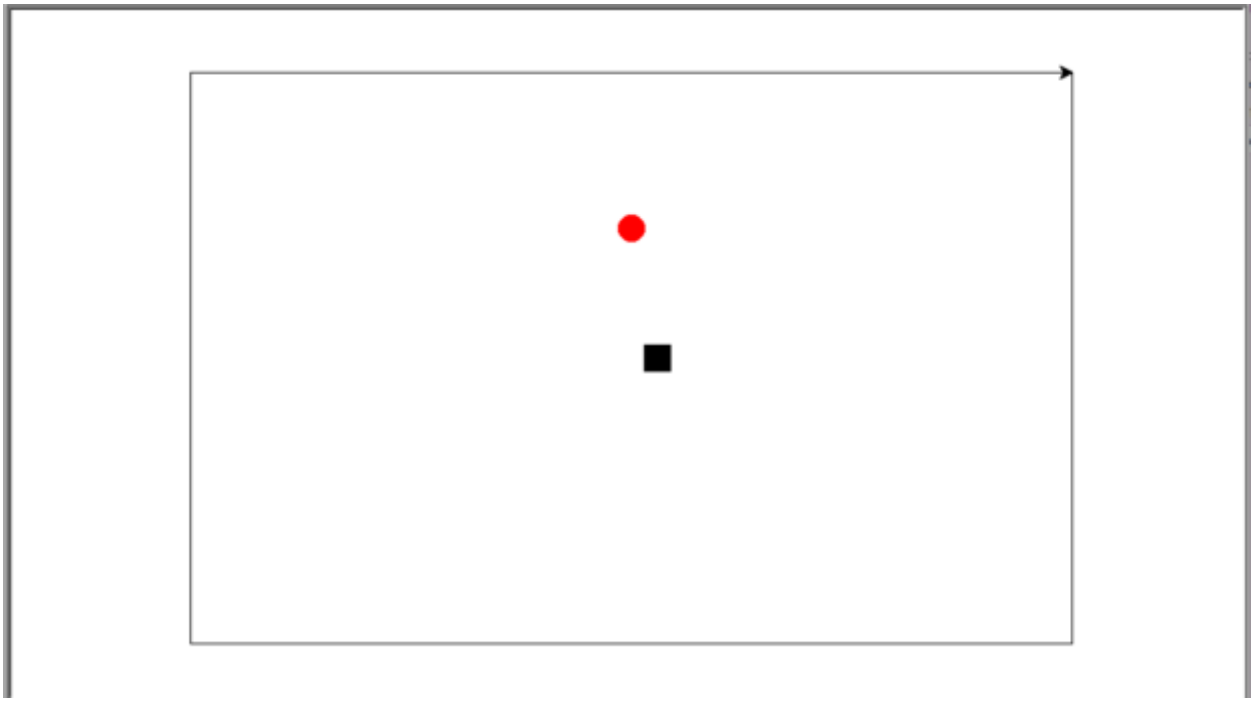
- Complete the border of the game by using turtle, MAX_X and MAX_Y.

```
##################################
#      STEP 1 - MAKE BORDER      #
##################################
def make_border():
    '''
    Using the variables MAX_X and MAX_Y, use turtle to draw a border
    for the snake game.
    '''
    ###YOUR CODE HERE
```

- Hint: turtle.penup(), turtle.pendown() and turtle.goto(MAX_X, MAX_Y) should come in handy
- Hint: here is a diagram of the different points of the walls. You just need to write code to go to these points!



- When you are finished, your game should have a border like this:



Before moving on to step 2, go to the **Snake Game Student Submission** document and answer the Design Reflection Questions for step 1!!!

STEP 2- MAKE SNAKE AND FOOD [5:08 in video]

- **2A** Currently, the make_snake() function only makes one extra piece of the snake. Write code so the snake has **START_LENGTH** stamps (pieces).

```
def make_snake():
    '''
    Draw a snake at the start of the game with a while loop
    to make START_LENGTH number of snake pieces.
    '''
    ###YOUR CODE HERE
    id = snake.stamp() #makes a stamp of a snake
    snake_ids.append(id) #adds stamp id to list
    snake_pos.append( [snake.xcor(), snake.ycor()] ) #adds location of stamp
    snake.forward(SQUARE_SIZE) #moves snake forward
```

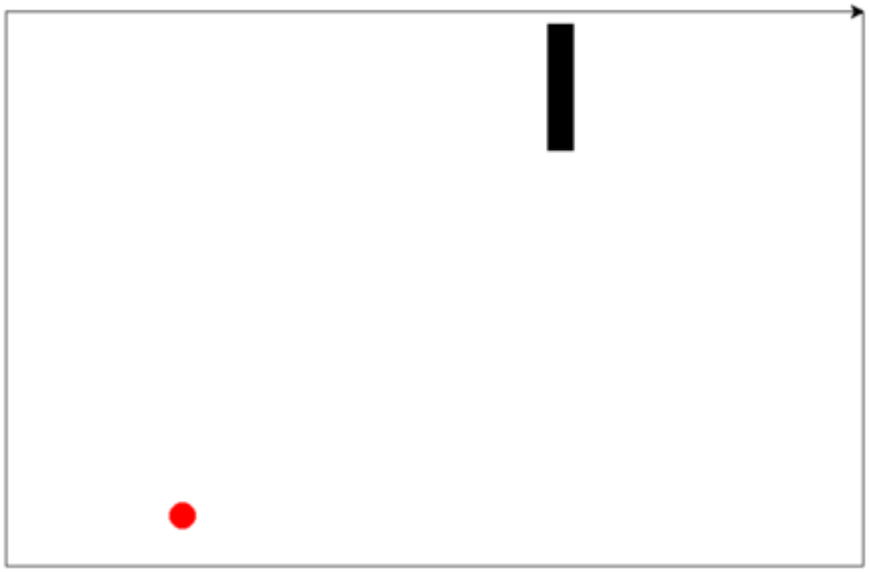
- Add a while loop to the make_snake() function that will make the snake longer. Solution below:

```
def make_snake():
    '''
    Draw a snake at the start of the game with a while loop
    to make START_LENGTH number of snake pieces.
    '''
    i = 0
    while i < START_LENGTH :
        id = snake.stamp() #makes a stamp of a snake
        snake_ids.append(id) #adds stamp id to list
        snake_pos.append( [snake.xcor(), snake.ycor()] ) #adds location of stamp
        snake.forward(SQUARE_SIZE) #moves snake forward
        i = i + 1
```

- What does snake.stamp() do? [This website]() provides a nice tutorial. Basically instead of making a lot of turtles (which slows the game down), the code just stamps (leave a copy of) the turtle so that whenever the snake moves the copy will stay behind. When the turtle leaves a stamp, it creates an id (a unique number) that is used to remove the stamp later using the clearstamp(id) function. Whenever the snake moves forward, it creates a new stamp in the new location and deletes (clears) the oldest

one. This is what the move_body() function at the bottom of the code does for you.

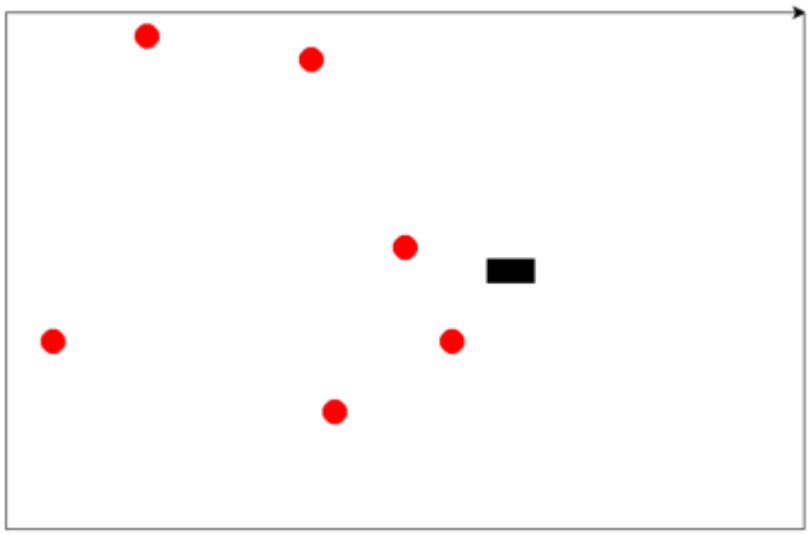
- *When you are done with the make_snake() function, the snake should look like this when you press the UP BUTTON*



- It's ok if the snake grows and then shrinks at first - it is because the move_snake() function isn't fixed yet. This will be fixed in step 3.
- **2B** [8:06 in video] Currently, there is one piece of food for the turtle to eat. Add a while loop to make_food() to make NUM_FOOD pieces of food for the turtle to eat

```
#2B MAKE FOOD
def make_food():
    '''
    Make NUM_FOOD number of pieces of food for the snake
    to eat at the start of the game using a while loop.
    '''
    ###YOUR CODE HERE
    move_food() #moves food turtle to random location
    id = food.stamp() #makes a stamp of the food
    food_ids.append(id) #adds stamp id to list
    food_pos.append([food.xcor(), food.ycor()]) #adds location of stamp
```

- Hint: this is VERY similar to the while loop from make_snake(). Instead of using START_LENGTH, use NUM_FOOD.
- When you are done the game should look something like this when it starts:



Before moving on to step 3, go to the **Snake Game Student Submission** document and answer the Design Reflection Questions for step 2!!!

STEP 3 - MOVE SNAKE [8:23 in video]

- **3A** Write the down(), left() and right() functions to change the value of the variable direction.

```
##################################
#      STEP 3 - MOVE SNAKE       #
##################################
UP = "Up" # snake Movement directions
DOWN = "Down"
LEFT = "Left"
RIGHT = "Right"
direction = RIGHT #snake starts off moving right


def up():
    '''
    When up button is pressed, change
    direction to UP
    '''
    global direction
    direction = UP

# 3A - Make functions down(), left(), and right() that change direction
#####WRITE YOUR CODE HERE!!
```

- Hint: Follow the pattern given in the up() function.
- You must include the line **global direction** inside these functions. This is what allows us to change the value of direction without causing errors!
- **3B** [8:44 in video] To get direction to change when the buttons are pressed, add turtle.onkeypress commands for down, left and right

```
turtle.onkeypress(up, UP) # When UP key is pressed, call up() function.

# 3B - Do the same for the other arrow keys
#####WRITE YOUR CODE HERE!!
```

  - Hint: follow the pattern for the up function. Note that the first input is the function name, the second is a variable in all capital letters.

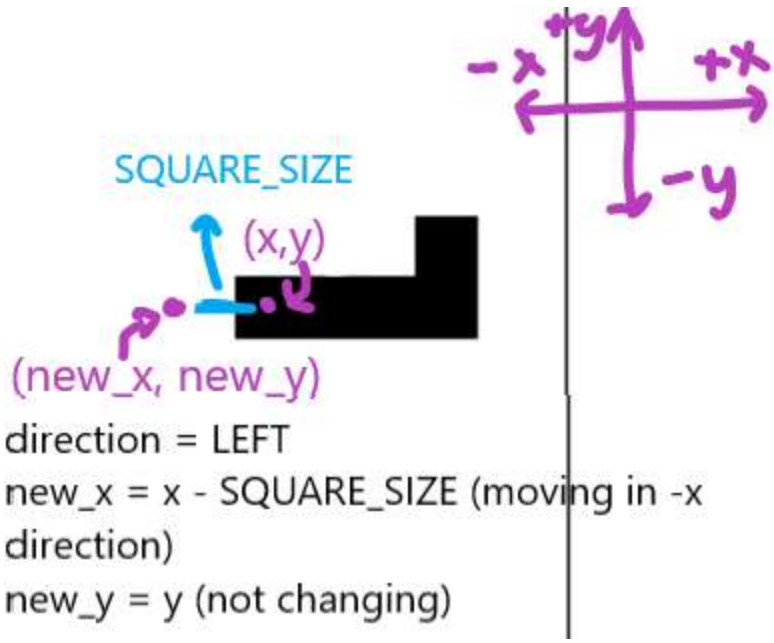
- **3C** [9:14 in video] Currently when you run the code, only pressing the up button will cause the snake to move up. Write if statements for DOWN, LEFT and RIGHT so that the snake can move in all four directions!

```
#3C - Move the snake to new position
def move_snake():
    '''
    Using the variable direction,
    find the new_x and new_y position
    of the snake and then goto that new
    position.
    '''
    new_x = snake.xcor()
    new_y = snake.ycor()

    if direction == UP:
        new_y = new_y + SQUARE_SIZE
    ### YOUR CODE HERE

    snake.goto(new_x, new_y)
```

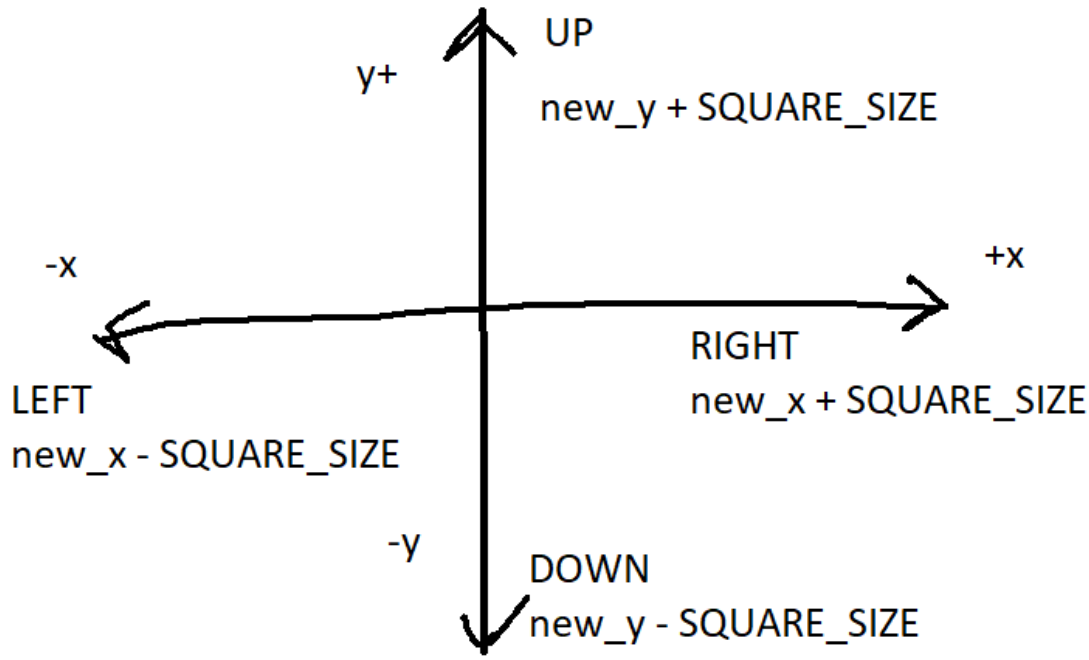
- Hint: the following diagrams might be useful



direction = LEFT
new_x = x - SQUARE_SIZE (moving in -x direction)
new_y = y (not changing)

- Hint: this is the code for the UP and LEFT directions. How would you write code for DOWN and RIGHT ? (look at the following diagram for hints)

```
if direction == UP:
    new_y = new_y + SQUARE_SIZE
### YOUR CODE HERE
if direction == LEFT:
    new_x = new_x - SQUARE_SIZE
```



- When you are done with the move_snake function, the snake should move continuously and be able to turn in all four directions.

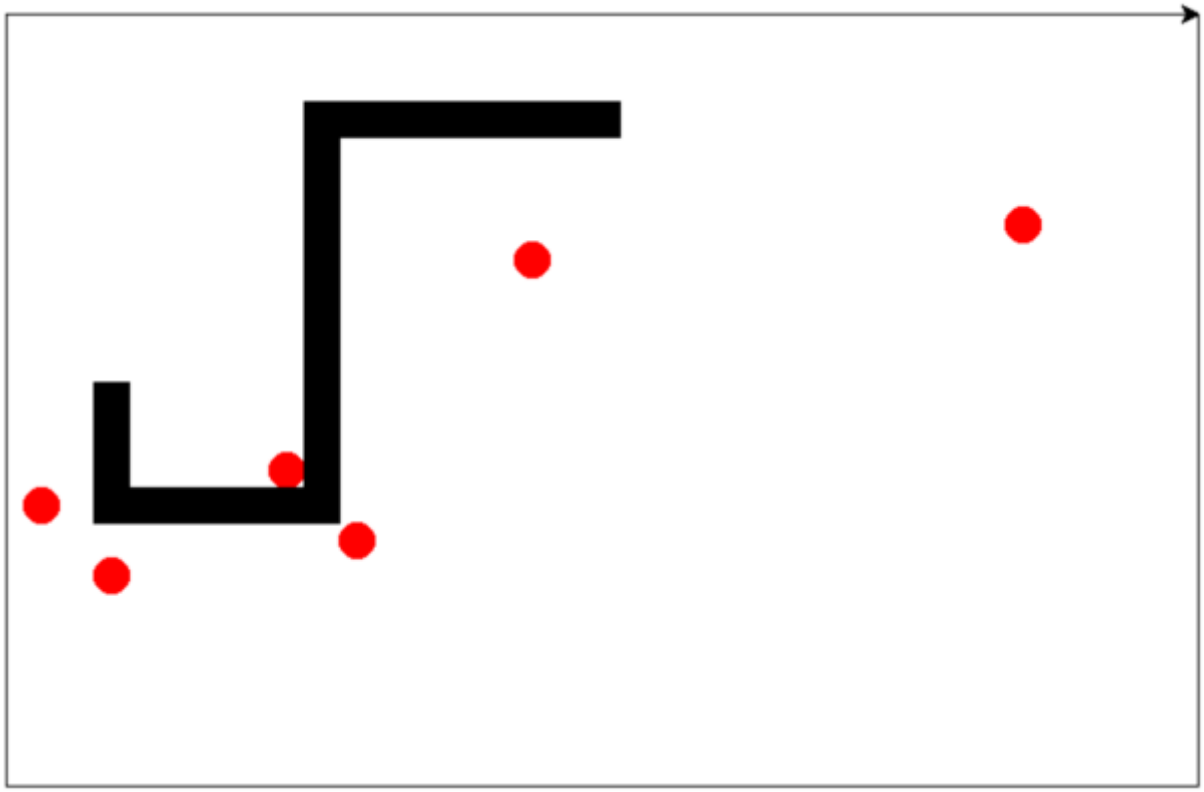
Before moving on to step 4, go to the **Snake Game Student Submission** document and answer the Design Reflection Questions for step 3!!!

STEP 4 - CHECK IF SNAKE EATS FOOD [10:26 in video]

- Now it's time to check if the snake is eating food! This code is a bit tricky, so here is the solution. Write this code in the check_eat_food() function!

```python
# STEP 4 - CHECK IF SNAKE EATS FOOD      #
##########################################
def check_eat_food():
    '''
    Using snake location and the list of food_pos, make a while loop that checks if the snake
    is in the same location as the food. If so, increase score by 1 and make a new piece of food
    '''
    global score
    x = snake.xcor()
    y = snake.ycor()
    i = 0
    while i < len(food_pos):
        food_loc = food_pos[i] #food location [x,y]
        food_x = food_loc[0] # get x position of food
        food_y = food_loc[1] # get y position of food
        if x == food_x and y == food_y : #snake has eaten food
            score += 1 #increase score
            food.clearstamp(food_ids[i]) # delete stamp
            move_food() #move to a new location
            food_ids[i] = food.stamp() #make a new stamp
            food_pos[i] = [food.xcor(), food.ycor()] # remember new location
        i = i + 1
```

- **Note - you don't have to include the comments (#green code that starts with #)**
- Note: score+= 1 is the same as writing score = score + 1 !
- Want to know how this code works? Look at the video for an explanation!
- When you are done, the snake should be able to eat food and grow!



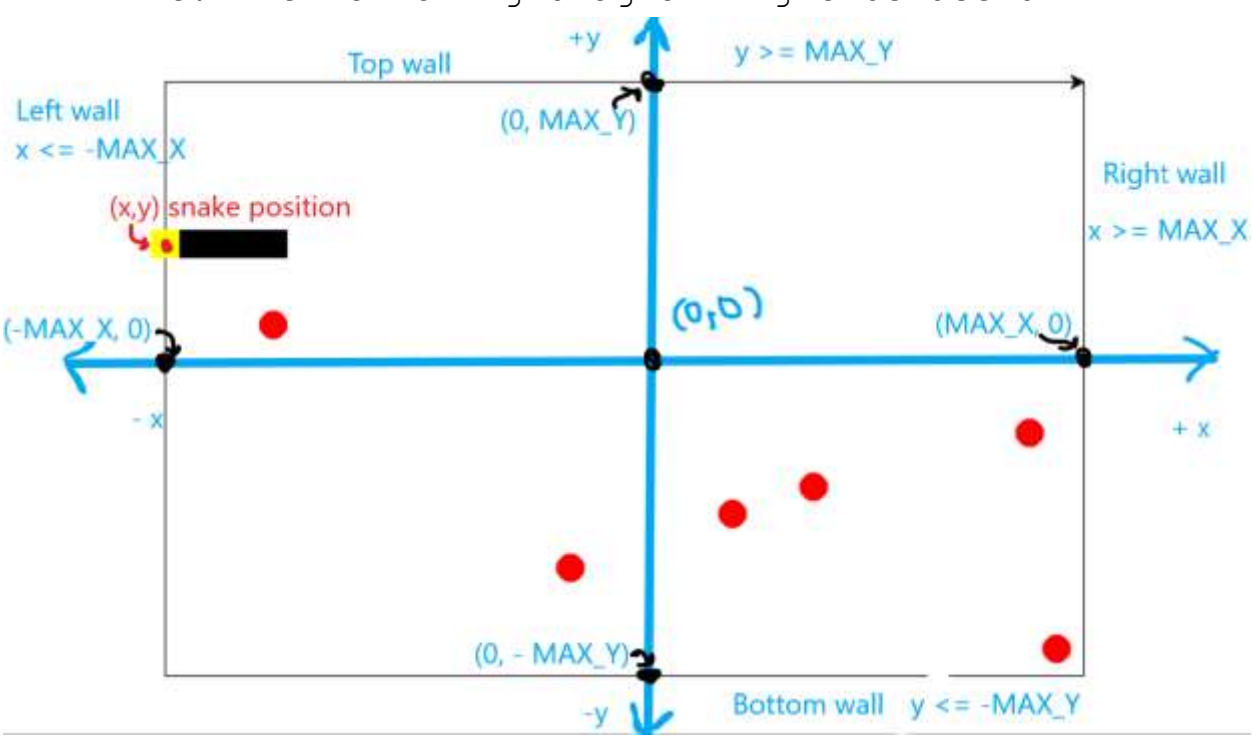
Before moving on to step 5, go to the **Snake Game Student Submission** document and answer the Design Reflection Questions for step 4!!!

STEP 5 - CHECK IF SNAKE HITS THE WALLS [12:08 in video]

- Currently, if the snake hits the top wall the game ends, but the snake can move outside the other walls!
- Add if statements to check_edges() that check if the snake is outside the left, bottom and right walls and if it is end the game by calling the game_over() function.

```python
##########################################
# STEP 5 - CHECK IF SNAKE HITS THE WALLS #
##########################################
def check_edges():
    '''
    Get the x and y coordinates of the snake
    If the snake is outside of the top, left
    right, or bottom walls, call game_over() function
    '''
    x = snake.xcor()
    y = snake.ycor()
    if y >= MAX_Y:
        print("You hit the top wall! Game over!")
        game_over()
    ### YOUR CODE HERE
```

- Hint: The following diagram might be useful



- When you are done, the game should end if the snake hits any of the four walls.

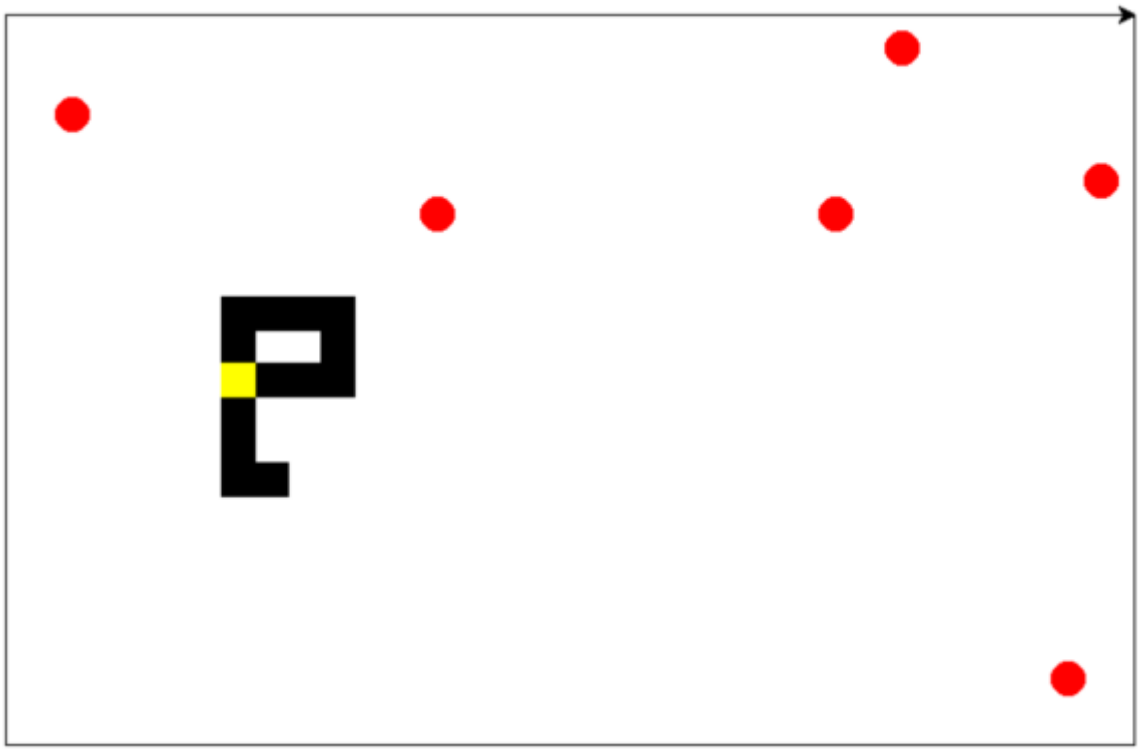
Before moving on to step 6, go to the **Snake Game Student Submission** document and answer the Design Reflection Questions for step 5!!!

STEP 6 - CHECK IF SNAKE EATS ITSELF [13:23 in video]

- Complete the check_eat_self() function

```
##########################################
# STEP 6 - CHECK IF SNAKE EATS ITSELF    #
##########################################
def check_eat_self():
    '''
    Check if the snake's head is in the same location
    as the rest of the body of the snake. If it
    is, call the game_over() function
    '''
    ### YOUR CODE HERE
```

- Hint: Write a while loop that goes through the snake_pos list and check to see if the snake's current x and y position is the same.
- Hint: This is VERY similar to check_eat_food() except use snake_pos instead of food_pos!
- When you are finished, the snake should die if it eats itself:



Now you are done with all the required features of the snake game!!

Go to the **Snake Game Student Submission** document and make sure you have answered all the questions and copied your code before you submit!

When you are finished:

WALKTHROUGH VIDEO FOR EXTRA FEATURES

- Add new features to your game!
  - Some ideas:
    - Add a pause button to the game **[0:00]**
    - Show the score **[3:11]**
    - Add different levels of difficulty
    - Make the game harder (faster) as you increase the score **[5:21]**
    - Make a game over screen when the snake dies **[8:06]**

- Make sure you have submitted all assignments on Repl.it/student
- Help other students
- Work on test corrections