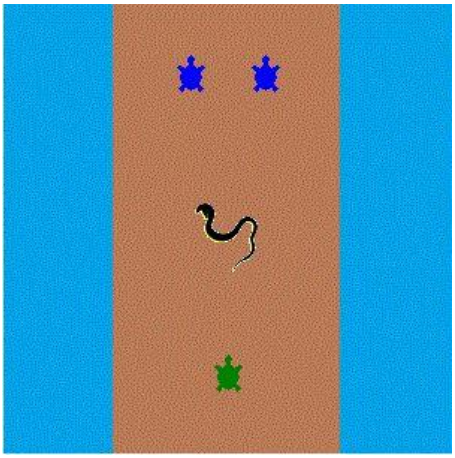
Lesson 10: Turtle Obstacle Game

In this lesson, we'll learn how to make a game using our turtle skills and taking input from the user.

The point of the game is to move a green turtle around a snake to its family, and not fall in the water.



PART 1: Setting up the Turtle and Scene

Open the file called “turtle_obstacle_student.py” and you will see many lines of code already written for you broken into four parts.

Part one should have code that looks familiar from earlier lessons. It will start by importing the turtle module. Then it will print the game information and instructions to play. The player turtle (myT) is created and made green. We set the background of the screen to our image called “street2.gif” that shows a road with the snake in the center and water on both sides.

The final section of code in part one makes two more turtles, which will be blue representing the finish goal. We call them momT and dadT.

Notice that we call penup() after we make our turtles. This is because we don’t want our turtles to draw when we move them.

Since our turtles are being setup to their starting point, we'll set the speed to 0 (fastest).

After we move myT to its starting position, we then change myT’s speed to 1 (slowest).

PART 2: Using a Variable to see Player Position

Part two of the given code should be empty except for a comment.

In this part we will be learning how to store a turtle's position (x and y coordinate) into a variable to be used later.

Try adding code below the comment as shown here:

```
# Set a variable "location" representing myT's position

location = myT.position()
```

In the location variable, we are storing the (x,y) coordinates of the myT turtle, which refers to the location of the center of our player turtle.

The function position() is from the turtle module that returns a 2DVector type which is basically a **list/array** containing two **floats** (the x coordinate and the y coordinate).

Remember that the coordinates at the center of the screen are (0, 0), and we moved our player turtle to (0, -100). That means right now location is equal to (0.0, -100.0)

If we need to see the x coordinate of myT, we can now use location[0] to show us the float value of where on the x axis our turtle is. To see the y coordinate of myT we would use location[1].

This line of code is very important for the game to work the way we want it to. It will let us see if the player turtle is in the water, on the snake, or in between the two blue turtles

We will need to update location every time the turtle is moved. We can simply do that by adding this line of code after the turtle is moved. We will cover this in part four.

PART 3: Moving our Turtle

Here, we'll take input from the user, and use the input as commands to move and rotate the turtle.

In this part we will not be giving you the completed code, but it should be easy to figure out what is needed based on your earlier lessons.

The code that is needed is the commands to turn and move the player when the input asks for it. We assign our input to the variable command. Then we check if the command is a valid option; if so, we do it. The code below shows where code is needed with the bold and underlined comments.

Put in the missing lines of code where it is bold and underlined:

```
. . .
# Repeat the following until the player reaches the goal
while(True): # So long as the player isn't at the goal
    command = input("Direction: ")
    # Do the command the player puts in
    if(command == "rt"):
        #add code to turn myT to the right 90 degrees
    elif(command == "lt"):
        #add code to turn myT to the left 90 degrees
    elif(command == "fd"):
        #add code to move myT 50 pixels forward
    elif(command == "q"):
        print("Quitting")
        break        #break from while loop to end game
    else:
        # Tell the player if they put in an invalid command
        print("Not a valid command")
```

Now, save your file with a new name such as turtle_obstacle_yourrname.

Test your program, it should show your green turtle in the bottom, the snake in the center, and the two blue turtles at the top. Click on the python shell where the text with game info and instruction should have appeared. Then type the command *fd* next to the text "Direction: " and press Enter to see if the turtle on the screen moves forward. Try typing more than one command. You should notice that the turtle can move over the snake and walk on the ground and water. Next, we will add a feature that will assign actions based on where the turtle is.

PART 4: Checking Location to see if we Win or Lose

In this part, we will be using our player location variable to see if we win or lose. We need to update the location then check if our player has collided with the snake, or is back with its family.

First, we use `location = myT.position()` to update the location variable to the current position of the turtle.

Second, we use an if statement to check if the turtle is exactly between the two blue turtles. The x coordinate of myT would be 0 and if the y coordinate would be 100.

Finally, we use an if statement to check if the turtle is exactly in the center of the screen which means he's on the snake. This x coordinate of myT would be 0 and if the y coordinate would be 0.

Add the following code under the PART FOUR comment. <u>Be sure to indent the code to be in the while loop and if statements.</u> See the → below.

```
    ###PART FOUR###

    """

→   location = myT.position()

→   if(int(location[0]) == 0 and int(location[1]) == 100):
    →   print("Nice job! You got to the turtle back to its family!")
    →   break
→   if(int(location[0]) == 0 and int(location[1]) == 0):
    →   print("You ran into the snake!")
    →   break     #break from while loop to end game
```

Once you finish adding the code, test your code by running the module. Now, try moving the turtle forward into the snake. The screen should close once that happens, and a text will show up in the shell “You ran into the snake!” and “Game Over”

Now run your module again, but this time move around the snake, and move the turtle between the two blue turtles. The screen should close again, and text should pop up saying “Nice job! You got the turtle back to its family!” and “Game Over”

The game is almost complete! All we need is some code to check if the turtle is on the water, which is in the next section.

PART 5: BONUS: Adding Water Condition

For this bonus section, we will not be giving you the code.

You will need to add the code to check if the player is on the water. If so, you need to print a message like “You fell in the water!” and then break out of the while loop.

It will be very similar to the code we wrote in part four where we checked if the player ran into the snake or moved between his parents.

Add this code right under the similar code from part four. Make sure you indent it to be in the while loop.

A good hint is that the dirt road is about 130 pixels wide and the player turtle starts in the middle of the road, with his x coordinate equal to 0. Also remember the player moves 50 pixels forward each time the command “fd” is entered. Try using the x coordinate to see if the turtle is too far to the left or right of the dirt path.

PART 6: BONUS: Keeping the Player on the Screen

Although our game is complete, there is still one problem you may find. The turtle can go off the screen if they go down or up the path far enough. This is fairly easy to fix.

The pseudocode is:

```
If the user inputs the command “fd”,
        If the player is at the top of screen and facing up
                Print “not a valid command”
        Else if the player is at the bottom of screen and facing down
                Print “not a valid command”
        Else
                Move the player forward 50 pixels
```

The screen height is 300 pixels. So the player is at the bottom of the screen if the y coordinate is less than or equal to -150. And the player is at the top of the screen if the y coordinate is greater than or equal to 150.

In order to see if the player is facing up or down, we need to find the turn angle by using the function **heading()** from the turtle module. Remember the angle starts at 0 degrees when the turtle is facing right. The angle increases counter-clockwise, so the angle facing up is 90 degrees. The angle facing left is 180 degrees, and the angle facing down is 270 degrees.

Using heading() will return the direction angle as a **float**. So that means, just like our x and y coordinates in position(), we need to convert it to an int if we want to compare it to another int.

Add the following code under the forward command if statement:

```
. . .
elif(command == "fd"):
    if(int(location[1]) >= 150 and int(myT.heading()) == 90):
        print("Not a valid command")
    elif(create code here that checks if the turtle is below the screen):
        print("Not a valid command")
    else:
        myT.forward(50)
```