Module 2:1 Challenge

*Connect the dots...with Turtles!*

***Learning Topics Covered:***

* Looping (For and While)
* Functions
* Turtles
* File Reading

***Introduction***

For your first Challenge, you will be asked to connect the dots, but in python! This Challenge will test your understanding of what we have covered so far: basic programming, loops and functions.

***How to Play***

For our special connect the dots game, I will give you several a special text file. Instead of dots on a piece of paper, you will get a list of (x,y) points in the order you need to connect them.

You will create a program for part 1 which does several things:

1. Set up a turtle screen.
2. Read the file.
3. For each point in the file:
	1. Draw a line to that point

For part 2 of the Challenge, you will add in color to your “connect the dots” game.

***Text Files***

You can find the text files available for testing at the box link provided below:

<https://iastate.box.com/s/v5gxzbv23n97m6zb3l6huat0hhd911dt>

The box link has several files:

* 1\_FILENAME.txt
	+ These files are used for your Part 1 program. Use these to test and view how your part 1 code works.
* 2\_FILENAME.txt
	+ These files are used for your Part 2 program. You CANNOT use these files with your part 1 code.
* Curve\_file\_generator.py
	+ This file is explained at the end of the challenge

***Part 1: Connect the Dots***

***Part 1 Goals (In a Suggested Order):***

* Set up a turtle screen with 1 turtle
* Open up a file and read from it
* Draw a picture based on the text in the file

***Part A: Set up a Turtle Screen.***

For this part, just set up a single turtle screen (with whatever size you want), and put one turtle on it. If you don’t remember the code to get this done, here is a link to a helpful guide:

<http://www.openbookproject.net/thinkcs/archive/python/thinkcspy3e_abandoned/ch03.html>

***Part B: Open a file and read from it.***

While we have never explicitly covered file reading, the code is very short. Below is an example of python code used to open a file, and how to *iterate* through it.



Set this up in your project. I would recommend printing out every line in your file to make sure you are reading it correctly. Once you know it is working, you don’t have to print out every line anymore.

***Part C: Connect the Dots***

This is the most difficult part of this project. I will break this task up into pseudocode for you:

**

* If you are stumped about how to turn each line into a coordinate point, look at what [string.split()](https://www.w3schools.com/python/ref_string_split.asp) does.
* If you are stumped about how to draw lines between the points, look at how we can move the turtle from point to point. Mess around with *(your turtle variable).goto(x,y)* and figure out a solution.
* If your turtle is moving, but not drawing lines, make sure you call *pendown()* on your turtle!

***If you have made it this far, congratulations! You should now be able to test your program with any of the 1\_FILENAME.text files. If you still have time, try out the next challenge!***

***Part 2: Use Color***

***Part 2 Goal:***

Modify your existing program to use color!

This part of the project is just an extension of what you worked on in project 1. Now, instead of just having coordinate points, your file will have special *color* lines that look like the following:



Each color line will start with a “!” character, followed by an RGB color. When your program encounters this line, it should change the turtle’s color to whatever RGB color is requested (after the !).

The most important function for this section is *turtlename.color(R, G, B).*

Here are some questions to help you get started:

* How can my program figure out when it is on a color line instead of a coordinate point line?
* How do I change the color of the turtle when I find one of these lines?

***IMPORTANT:*** If you are getting the error “bad color sequence” when using the color function, you need to call *(whatever your turtle screen is called)*.colormode(255).

***When you finish this section, you have accomplished a lot! Not only can you read a file and draw a picture, but now you can distinguish between two different “commands”. If you still have time, keep going! What else can you do to make your program more interesting? Take a look at the next section to start extending your image library.***

***Part 3 (For Fun): Custom Image Files***

If you have made it this far, congratulations! You have finished the programming work of the challenge. Now you can enjoy the extension - making your own image files! So far we have created a system which reads in points and colors to draw a single-line image.

Download *curve\_file\_generator.py* and *curve\_gen\_example.py*. Open up the *curve\_gen\_example.py*. This file is a demo on how to create your own, custom images! We can generate our custom images in the following way:

1. Create a custom function to produce x, y points
2. Feed the function into the *curve\_file\_generator* to produce a text file.
3. Feed the new text file into your program

The file *curve\_gen\_example.py* outputs a text file called “filename.txt”. This file, when fed through your program, produces this image:



So how does the example generate a text file? Let’s walk through the code:



*Lines 1 and 2:*

We need to import any modules we need. The *minimum* you need to include is line 1. Make sure you include it in the exact way shown above.

*Lines 5 - 15*

This is our “custom function”. The input is a single number which we can call “time”. Your function needs to produce a coordinate point (x, y) from a single value: time.

Let’s imagine time is measured in seconds. For our custom function, our curve will look like the following:

 # Seconds is Negative: (-1,-1), (-2, -2), (-3, -3), etc.

 # Seconds is Greater than 20: (400, 20), (441, 20), etc

 # of Seconds is > 0 and < 20: We plot a special math function, sine.

For your custom function, you can do just about anything. Mess around with how x and y change with the time value. If you are having trouble thinking of ideas, try modifying the example code with your own ideas!

*Line 18:*

This is the function we need to call to generate the text file. This is a **big** line, so lets break it up.

 “Filename.txt”: This is the file we are outputting. Name it whatever you want.

 Example\_curve: This is the name of the function we are using to make points.

 gen.float\_list(-50, 50, 1): See below.

gen.float\_list:

This is a special function that generates our “time” for us. You should always use this function, but feel free to change the numbers inside. The first and second values are *start* and *end* times, and the third value is the *interval*. For example:

 gen.float\_list(0, 5, 1) ⇒ [0, 1, 2, 3, 4, 5]

 gen.float\_list(0, 5, 0.5) ⇒ [0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]

**Using these tools, try to make your own curves. Feel free to take a screenshot of your work and post it on Discord. Also feel free to check out the other cool stuff others have come up with!**