Project1

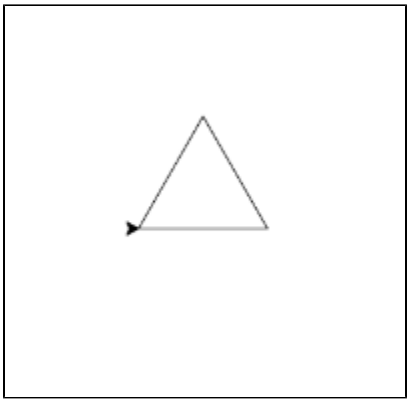
In this project I developed algorithms, or sets of specific instructions, to perform tasks like creating or drawing a simple images. To do this, I used a program known as Python turtle which is developed to read code containing directional instructions and draw the corresponding image. My task was to write an algorithm for a number of different shapes and allow turtle commands to draw the shape.

Task 1 was basic project setup. I mounted my directory to the Personal server where I could safely save my project work on the Colby server. Next, I opened Terminal which is a computer interface that can execute text-based commands. In Terminal, I navigated to a folder named Project1 in my Personal server to make sure the work I was using/creating was from my Project1 folder. The last stage of the setup was opening Text Wrangler which is the program where we use Python code to write our programs. The first thing thing I did at the beginning of every program was tell Text Wrangler I want to use Python turtle. To do this, I wrote the line: from turtle import * and I used simple directional commands like forward() with a pixel distance and left() with an angle command for the turtle drawing cursor to move in every single task.

For **task 2** I chose to create a triangle. After writing the first line of "from turtle import *" I then wrote the Python code that corresponds to turtle drawing a triangle. I saved this code in my Project1 folder as shapeA.py. The code was a set of directional instructions telling the turtle cursor where, what, and how much to draw. The code looked like:

```
forward(100)
left(120)
forward(100)
left(120)
forward(100)
left(120)
```

This is ***ShapeA:***(required image1)

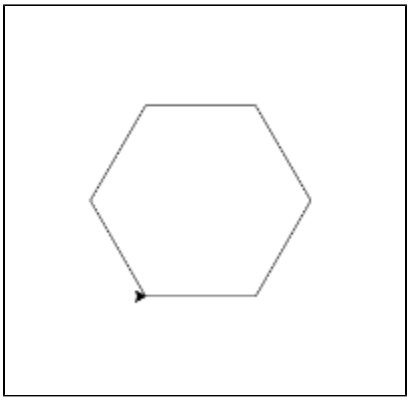


Task 3 was to repeat the same process but with a different shape. I decided to draw a hexagon. In a new file titled shapeB.py I wrote the correspond code to instruct Python turtle to draw a hexagon. The first two lines of code were as follows:

```
forward(100)
left(60)
```

I repeated these two lines six times to complete the code commands to draw a hexagon that looked like this:

This is ***ShapeB:***(required image2)



Task 4 was to create a new python file called shapes.py and execute a number of instructions in the file. The first step in **task 4** was to make shapeA and shapeB from the previous two files into functions. To do this, I defined the function shapeA as a list of python commands; the same list of python commands used to draw shapeA in the file called shapeA.py. To create a function for shapeA the code looked like this:

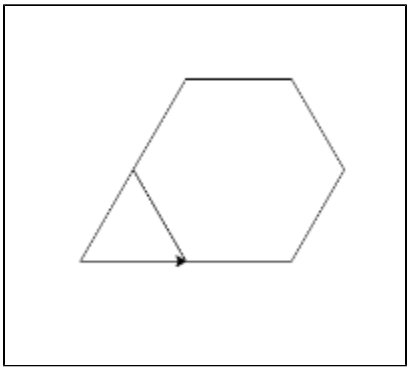
```
def shapeA():
    forward(100)
    left(120)
    forward(100)
    left(120)
    forward(100)
    left(120)
```

Notice that below the first line of code all the python commands are tabbed over. This tells Text Wrangler that this code corresponds to the function shapeA(): Next, these same steps were carried out to define a function of shapeB():

The last step in **task 4** was to create a third a final function called shapeC(): The purpose of shapeC was to draw both shapeA and shapeB 100pixels apart. To do this I wrote code with the function definitions I already defined earlier in the shapes.py file. The code for shapeC looked like this:

```
def shapeC():
    shapeA()
    forward(100)
    shapeB()
```

The resulting image was ***ShapeC:***



Task 5 was to expand on the file shapes.py with new instructions to create a flexible shape-drawing function with variables that enable the caller to decide the specific distance(number of pixels) to travel forward when drawing. First, I created a shapeD function that defined the tasks to draw a pentagon with a variable distances. To this I created the variable "distance" and wrote that instead of a numerical distance allowing the user to choose a distance to assign to that variable. The code looked like:

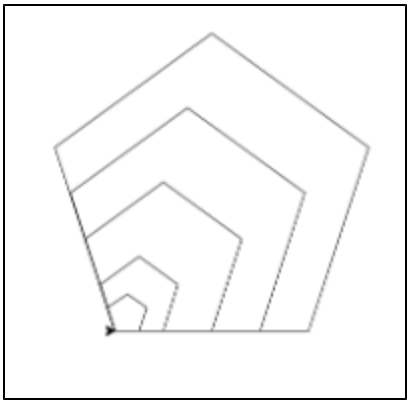
```
def shapeD( distance ):
    forward( distance )
    left(72)
```

I repeated those first two lines five more times to complete the code instructions for turtle to draw a pentagon.

Then, I created a shapeE function that calls the shapeD function multiple times but draws it with a different distance variable each time. The code for shapeE looked like this:

```
def shapeE():
    shapeD(25)
    shapeD(50)
    shapeD(100)
    shapeD(150)
    shapeD(200)
```

This is ***ShapeE:***(required image3)



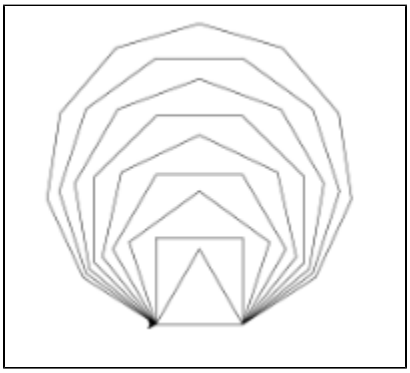
After completing the five required tasks for Project1, I worked on a few of the optional **extensions** for the project. My first extension, **extension1**, was one of the suggested extensions to create a function that will draw a N-Gon. To do this, I defined a function called shape() that took two variables, both side length("distance") and the number of sides("nsides"). Then, I created a for loop in range(0 to "nsides") that would loop my instructions starting at 0 up to the variable I input for "nsides". Within the loop, I move forward and adjust my angle depending on the number of sides by moving left by 360.0/nsides. The code for the shape() function and the for loop looked like this:

```
def shape(distance , nsides):
    for i in range(0,nsides):
        forward(distance)
        left(360.0/nsides)
```

Then, I called to function shape() with a number of different variables:

```
shape(100 , 3)
shape(100 , 4)
shape(100 , 5)
```
I repeated this all the way up to shape(100 , 11) making python turtle draw everything from a triangle to a hendecagon(11-sided polygon)

The resulting image was ***Extension1 Image:***(ngon)



Extension2 was an extension of my own creation. I wanted to create something random so I decided to code the drawing of a flower. To create a more complex image like a flower, I had to use some new commands besides just forward() and left(). I used up() and down() to tell the cursor to pickup off the screen and not draw for a certain distance and then go back down when I wanted. My code to draw a flower contained four different functions: stem(), leaves(), petals(), and ultimately a combination of those parts titled flower(). The code for stem() was as follows:

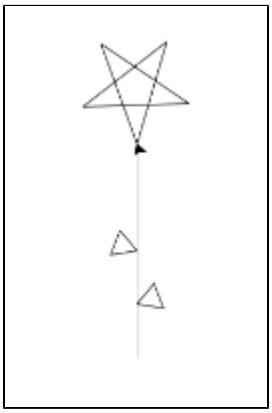
```
def stem(distance):
    left(90)
    forward(distance)
```

The function code definitions for leaves() and petals() were similar but much longer. And finally, the most important function that combined them all was flower() and its code looked like this:

```
def flower():
    stem(200)
    right(180)
    forward(200)
    left(90)
```

```
    leaves()
    left(20)
    petals(100)
```

This is ***Extension2 Image:***(flower)



My third extension, **extension3**, was based off of one of the suggested extensions. **Extension3** is based off the hierarchy of functions idea but with a twist of my own. Basically, I defined two separate shape functions called diamond() and trapezoid() that contained the algorithms to draw their respective shapes. The diamond drawing function had two variables, "start" and "distance" that were allowed the user to input the start angle of the cursor and the distance of pixels drawn. The diamond function was as follows:

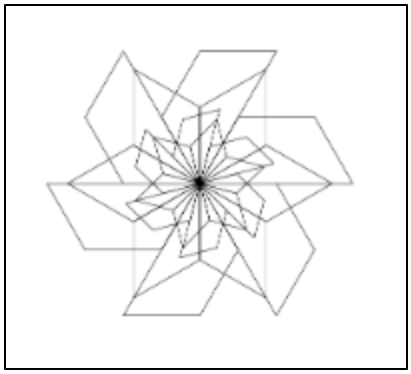
```
def diamond(start , distance):
    left(start)
    forward(distance)
    left(60)
    forward(distance)
    left(120)
    forward(distance)
    left(60)
    forward(distance)
```

The trapezoid drawing function for **extension3** had three variables "start1", "distance1" and "distance2" becuase a trapezoid has different bottom and top lengths. The trapezoid function looked like this:

```
left(start1)
    forward(distance1)
    left(120)
    forward(distance2)
    left(60)
    forward(distance2)
    left(60)
    forward(distance2)
```

Finally, for **extension3**, I created a function named star() that drew trapezoids and diamonds of different sizes in the shape of a star. This was the image star() created:

Extension3 Image:(star)

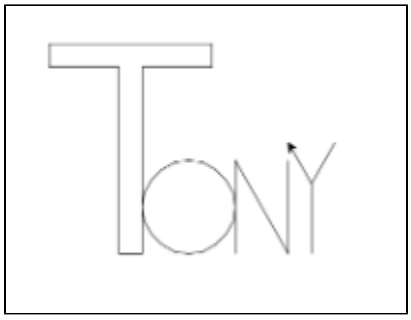


My fourth and final extension was an extension piece that I came up with just messing around with TextWrangler. **Extension4** simply contains the code directions to make python turtle draw my name. I define each letter, T(), O(), N(), Y(), as a function and combined them all in the function Tony() which looked like this:

```
def Tony():
    T()
    left(90)
    up()
    forward(75)
    down()
    O()
```

....all the way to the function for letter Y()! The resulting looked like this:

Extension4 Image:(name)



In the end, through this **Project1** I learned some basics of simple programming. I learned how to run applications like Terminal and TextWrangler and how to mount my directory on Personal server in the Colby server to save my work. Through completing a number of tasks and extensions I gained a basic knowledge of Python coding as well as Python turtle commands. I learned where and how to correctly save my work and how to write some basic code in Python and turtle.

Who helped me:

A number of people helped me during this project: Classmate Stefan Kohli helped me write the code from the star that acts as the petals for the flower on my **extension2.** TA's Vlad and Erin helped me create and understand the basics of a for loop in Python for my **extension1.** During lab write-up, I worked along side classmates and fellow coder Stefan Kohli and Julia Saul. TA Mike helped me figure out how to properly label my Wiki Writeup. Finally, Professor Taylor and Professor Maxwell answered a number of small questions for me throughout the project.