**TURTLE SNOWFLAKES**

Draw beautiful and intricate patterns with Python Turtle, while learning how to code with Python. In this exercise, you may name your turtle anything you wish. You might consider one of the Teenage Mutant Ninja Turtles such as Leonardo.

HOW TO DRAW WITH PYTHON TURTLE

To begin, you will need to open the Python programming environment IDLE. You can do this by selecting it from the Main Menu with your mouse.

Then click on File and New Window to open a blank text editor window to write your code in.

Save your empty file by clicking on File, Save As and naming it snowflake.py.

To begin using Turtle in Python you need to import the Turtle library. At the top of the text editor window type import turtle.

Next, a command is needed to create the window which will display your turtle drawings:turtle.Screen(). To save having to type it out every time you need the command, you can store it as a variable:

window=turtle.Screen()

*Note: There is a capital letter used here for Screen. It is important to get uppercase and lowercase syntax correct in Python in order for your code to work.*

Time to give your turtle a name; you can use a variable to do this. I'm naming my turtle Elsa but you can name yours whatever you like.

elsa=turtle.Turtle()

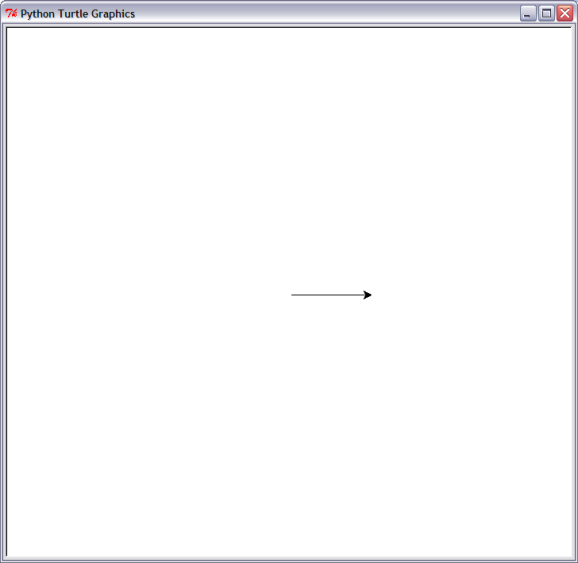
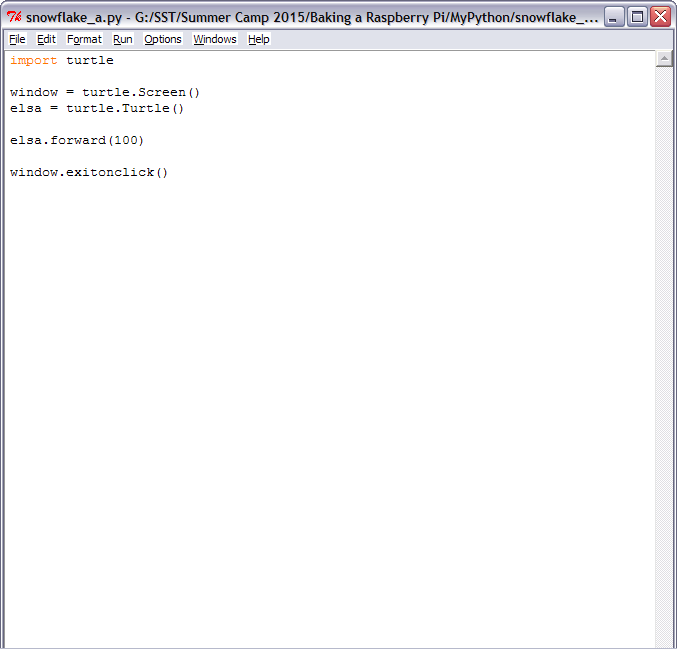
Now you can tell your turtle what to do; for example, to move forward 100. Try it out.

elsa.forward(100)

To complete your first turtle program, add window.exitonclick(); this means that the turtle window will stay open so you can see your drawing for as long as you like. It will only close when you click on the x in the top right hand corner of the window with your mouse.

window.exitonclick()

Click on File and Save, and then click on Run and Run Module to run your first turtle program. What happens?



HOW TO TURN WITH PYTHON TURTLE

Your turtle is pointing in one direction, which is good news if you want to draw a line, but it is time to think about drawing a shape like a square!

Underneath the line elsa.forward(100) in your Python code, add:

elsa.right(90)

*Note: Turtle uses angles in degrees. There are 360 degrees in a circle. How many are in the right angle of a square? That's right: 90. The value 90 inside the brackets in elsa.right(90) is in degrees. So this line is telling your turtle to turn right by 90 degrees.*

Add another instruction underneath to move your turtle forward by 100:

elsa.forward(100)

Save and run your code to see what happens.

You are on your way to creating a square. What do you need to add to your code in order to complete the square?

USING LOOPS TO CREATE SHAPES

To create a square, you have repeated some lines of code. It is not the most efficent way of doing it. Instead of typing out many lines of code, it makes more sense to use a loop.

Instead of code to create a square like this:

elsa.forward(100)

elsa.right(90)

elsa.forward(100)

elsa.right(90)

elsa.forward(100)

elsa.right(90)

elsa.forward(100)

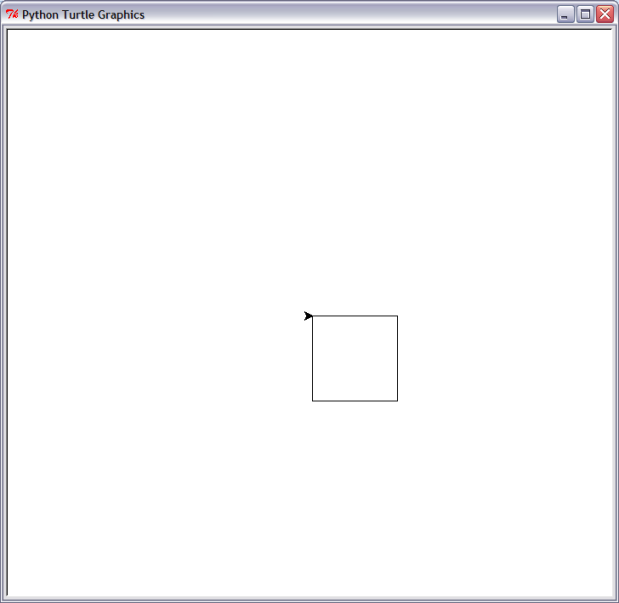
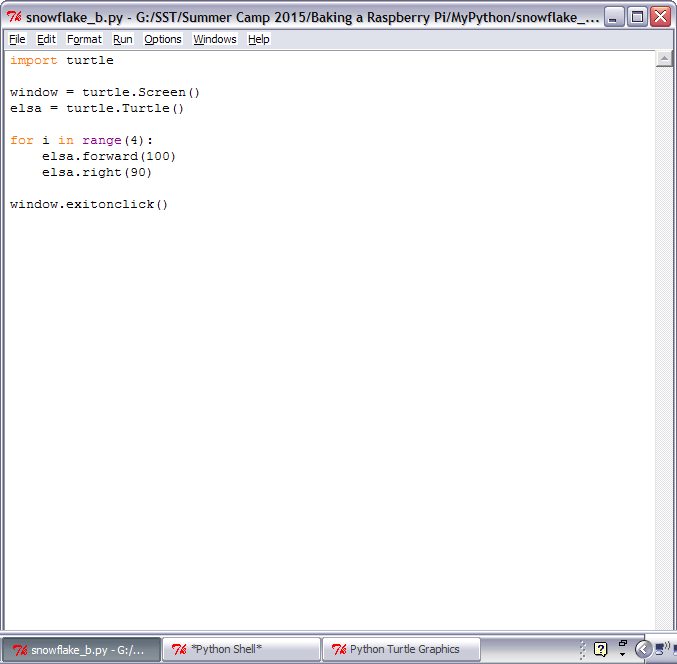
You can type:

for i in range(4):

elsa.forward(100)

elsa.right(90)

Try it yourself and see what happens when you save and run your code.



CREATING SPIRAL PATTERNS

Don't be a square! Let's create some different shapes and repeat them to make a snowflake-like spiral.

Replace the code for your square with the following:

for i in range(2):

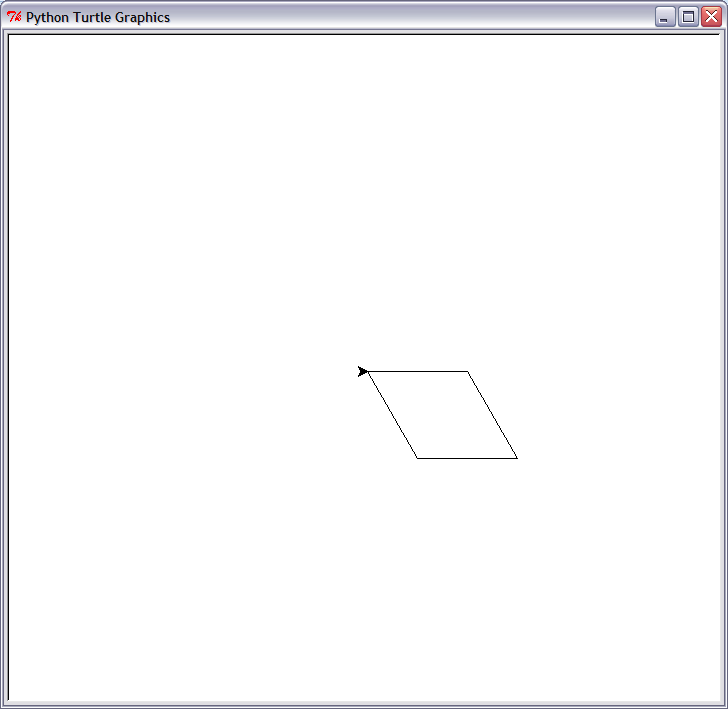
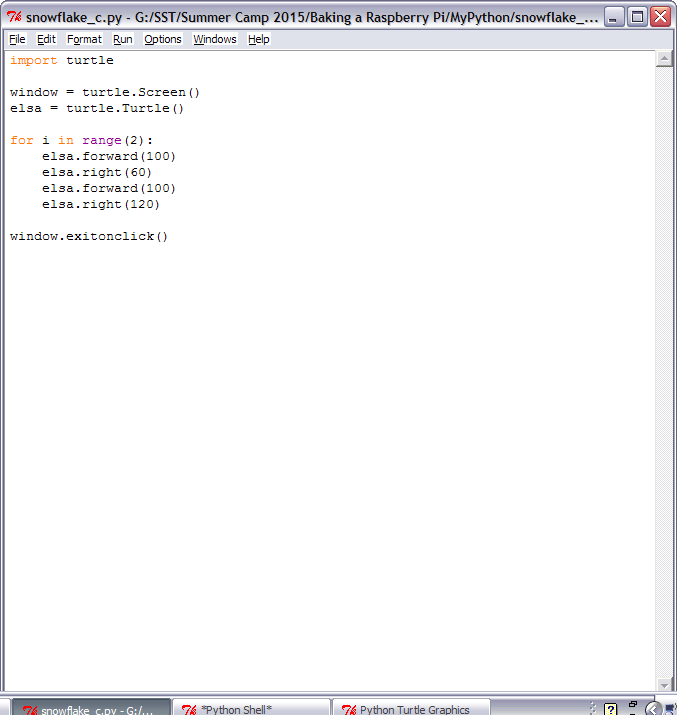
elsa.forward(100)

elsa.right(60)

elsa.forward(100)

elsa.right(120)

It will draw a shape called a parallelogram. You can see what it looks like by saving and running your code.



You can put loops inside of other loops. This is good news for us as we can make a drawing that looks like a snowflake. Above for i in range(2): type:

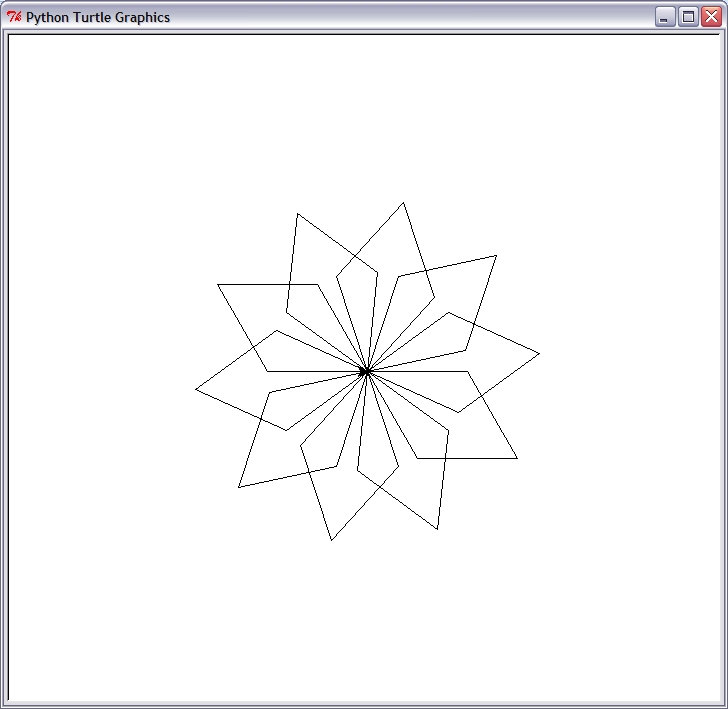
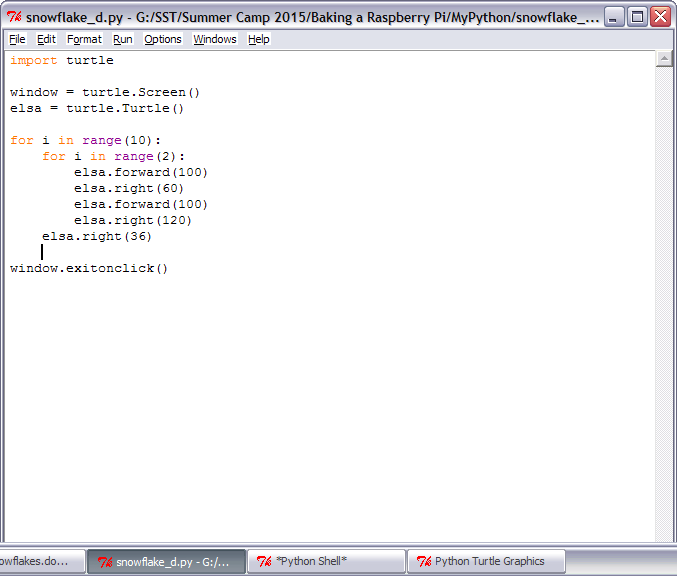
for i in range(10):

How many times will this loop go round?

Move to the bottom of your sequence of code and press the space bar four times to indent your code. Indentation in Python is very important to ensure that your code works as you expect. Then type:

elsa.right(36)

Save and run your code to see what happens. You should see a drawing like this:



CHANGING THE PEN COLOR RANDOMLY

So far the turtle has been drawing black lines on a white background. Time for some color!

To set the color of the turtle, use the following instruction underneath the naming of your turtle and before the loops:

elsa.color("cyan")

*Note: The spelling of color is different in other countries. In the US it is spelled 'color' and in Python it has to be spelled in the American way for it to work.*

I have chosen to use the color cyan but you can use any from this list:

"blue"

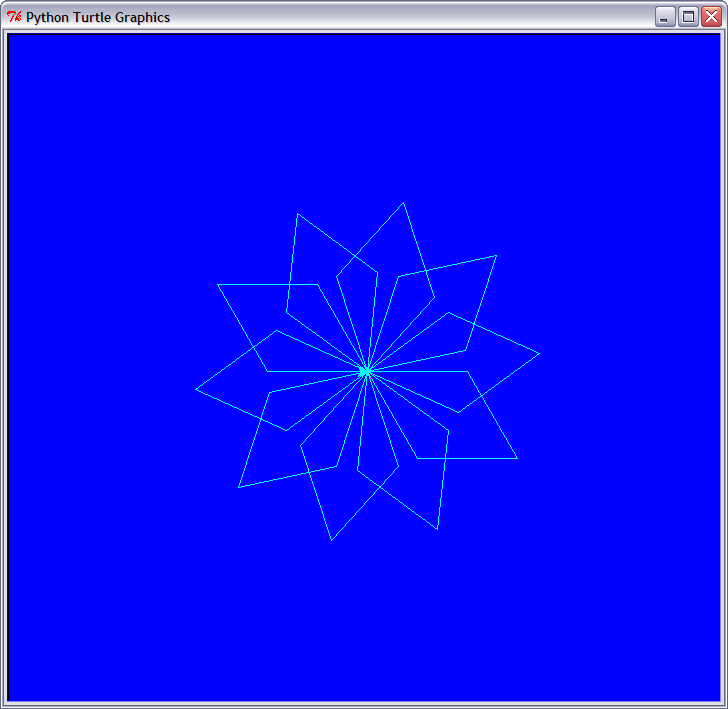
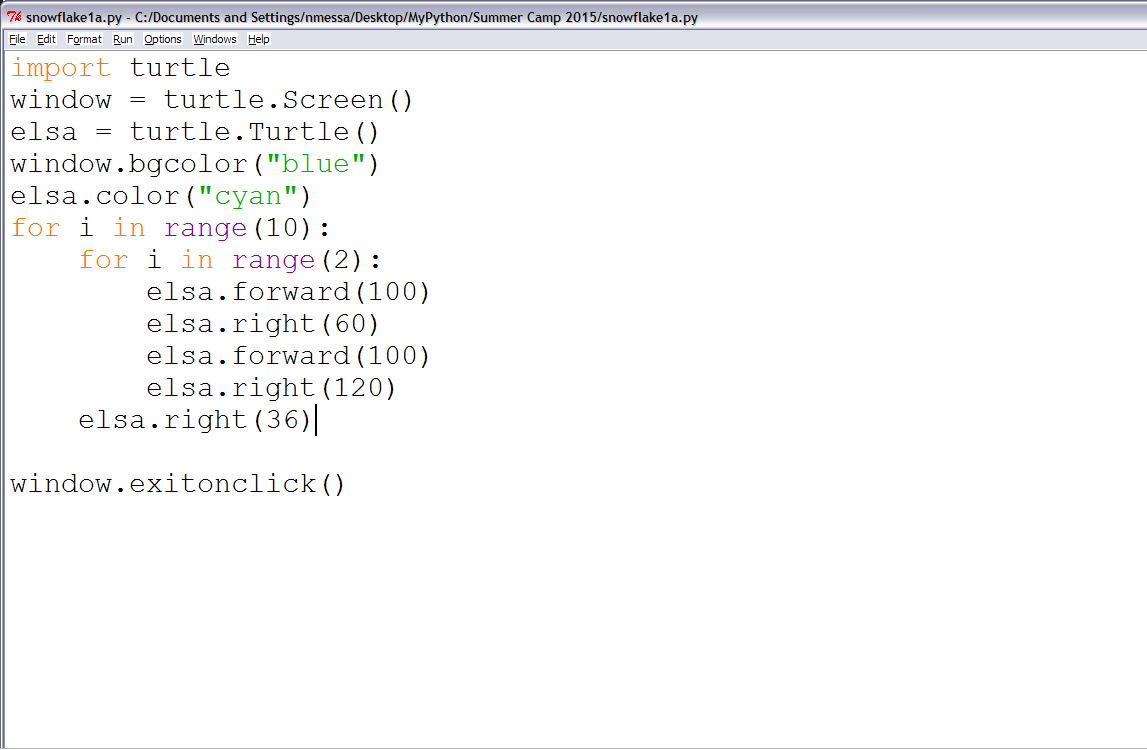
"magenta"

"grey"

"purple"

You can also change the color of the background window. To set the colour of the background use this instruction:

wn.bgcolor("blue")



For fun you can add a random color for your turtle, so that every time you run your code you will get a slightly different result.

First you will need to import the random library. Underneath import turtle type import random.

Next, change the background colour from "blue" to "grey".

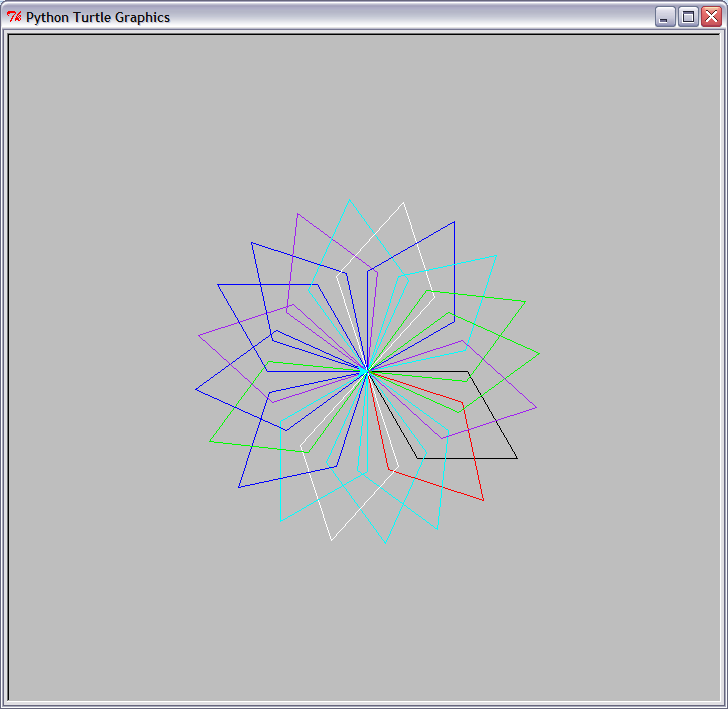
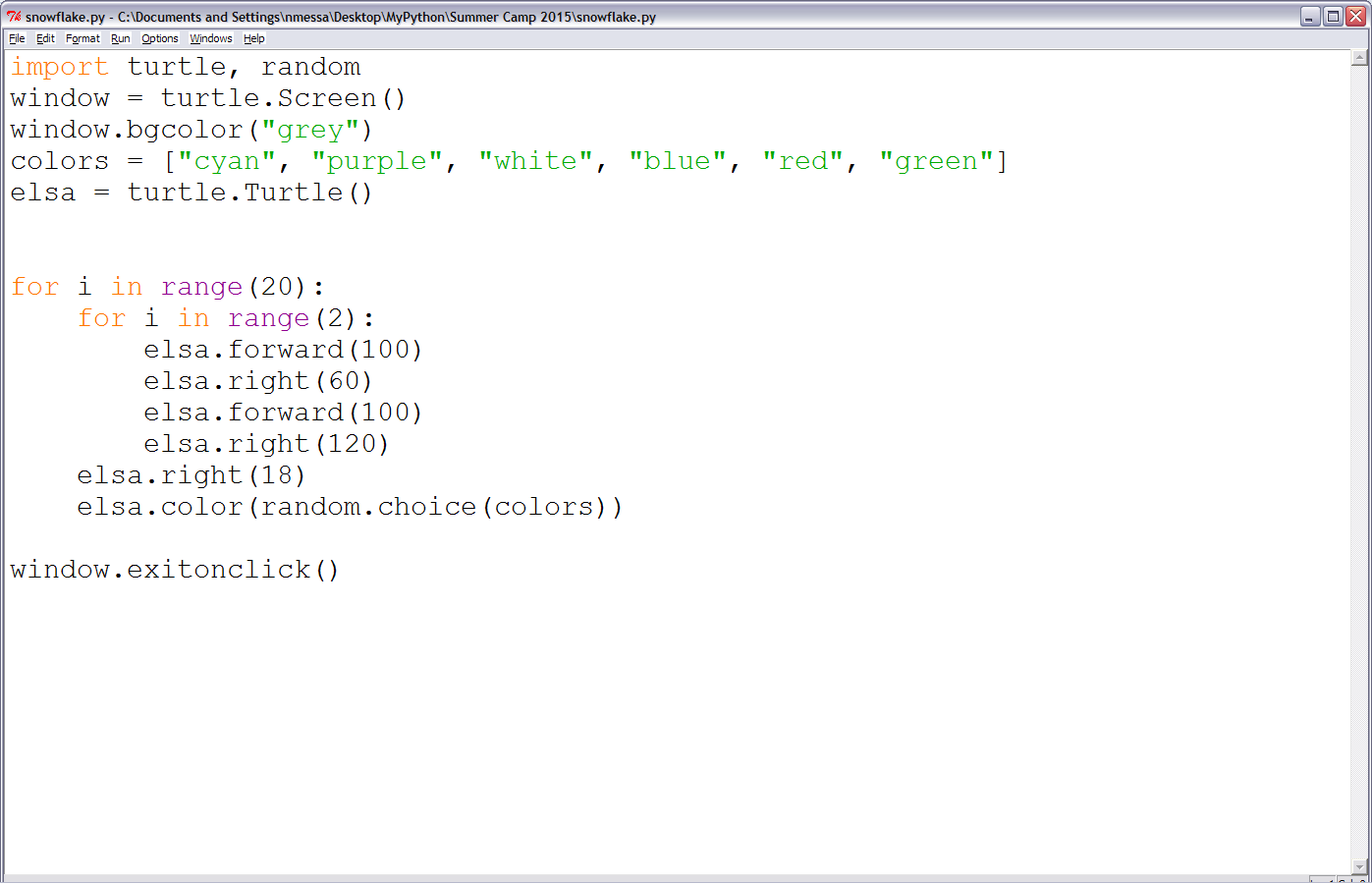
Below, create a variable called colors to store a list of the colors to select from, like this:

colors=["cyan","purple","white","blue"]

At the end of the spiral loop, underneath elsa.right(36), type:

elsa.color(random.choice(colors))

Save and run your code for a multi-colored snowflake!



USING A FUNCTION TO DRAW A SNOWFLAKE

Your parallelogram snowflake is cool, but it does not look as snowflake-like as it could. Let's fix that!

For this drawing, we need to move the turtle from starting in the centre of the window. Thepenup() and pendown() instructions allow this to happen without drawing a line. It's like picking up a real pen from the paper and moving it somewhere else to start to write. Type the following instructions underneath the colors list:

elsa.penup()

elsa.forward(90)

elsa.left(45)

elsa.pendown()

Next, define a function by typing:

def branch():

*Note: The code to draw one branch of the snowflake will be stored inside this function, so that you can repeat it over and over to create an entire snowflake.*

Remove the code for the parallelogram snowflake loops and replace it with:

for i in range(3):

for i in range(3):

elsa.forward(30)

elsa.backward(30)

elsa.right(45)

elsa.left(90)

elsa.backward(30)

elsa.left(45)

elsa.right(90)

elsa.forward(90)

*Note: Remember that indentation is important. Make sure that you check your indentation is correct or your code might not work!*

The last section of this code will call the snowflake function and repeat it eight times. You can use a loop in the same way as with the last snowflake:

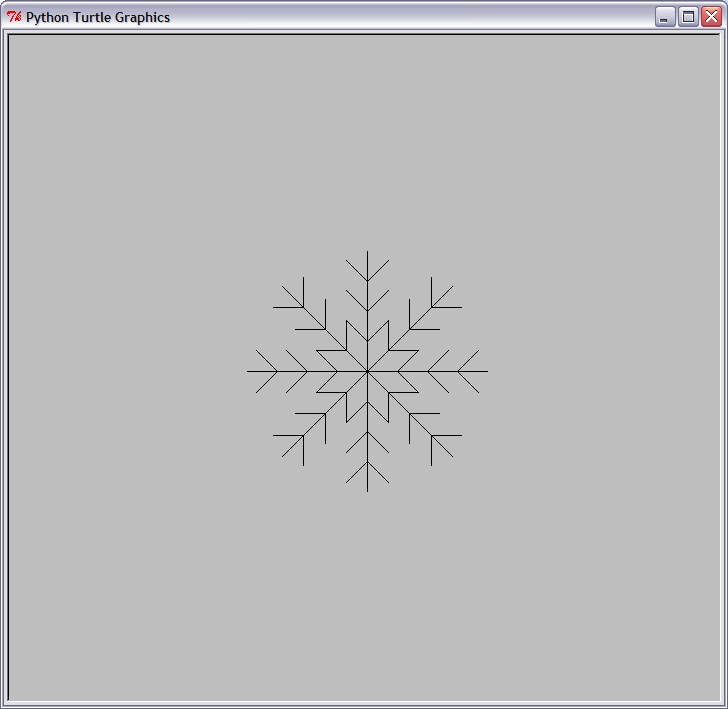
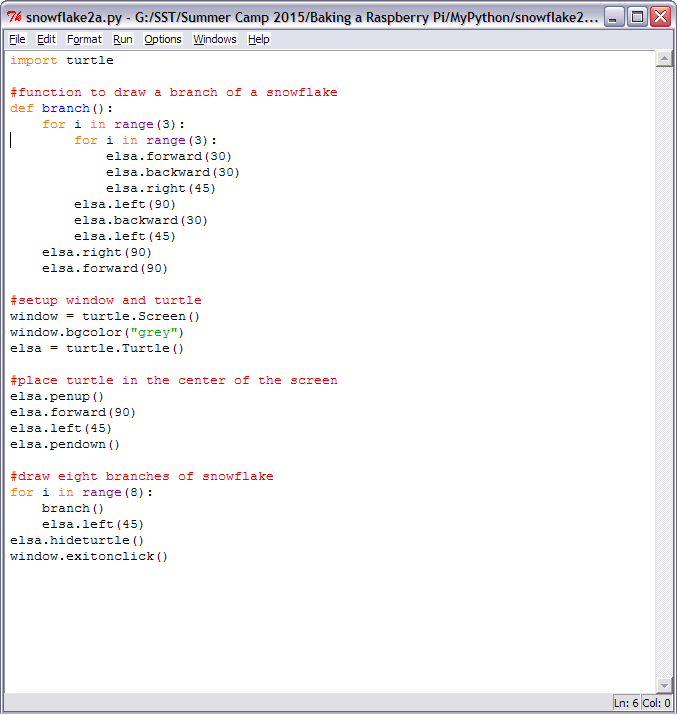
for i in range(8):

branch()

elsa.left(45)

Use a # at the start of the elsa.color(random.choice(colors)) instruction to turn it into a comment. This means that the computer will skip that line of code. You could delete the line, but you might want to use it to add colour to your snowflake later on.

Save and run your code, and a snowflake should appear before your eyes!



**Finally we will create 4 snowflakes at random locations**

import turtle, random

colors = ["white", "red", "cyan", "yellow"]

#function to draw a branch of a snowflake

def branch():

for i in range(3):

for i in range(3):

elsa.forward(30)

elsa.backward(30)

elsa.right(45)

elsa.left(90)

elsa.backward(30)

elsa.left(45)

elsa.right(90)

elsa.forward(90)

def place():

elsa.penup()

x = random.randint(-200, 200)

y = random.randint(-200, 200)

elsa.goto(x, y)

elsa.pendown()

#setup window and turtle

window = turtle.Screen()

window.bgcolor("grey")

elsa = turtle.Turtle()

elsa.color("white")

#place turtle in the center of the screen

elsa.penup()

elsa.forward(90)

elsa.left(45)

elsa.pendown()

#draw four snowflakes at random locations

for i in range(4):

place()

elsa.color(colors[i])

#draw eight branches of snowflake

for i in range(8):

branch()

elsa.left(45)

elsa.hideturtle()

window.exitonclick()