Turtle snowflakes

Write code to draw snowflakes with Python Turtle

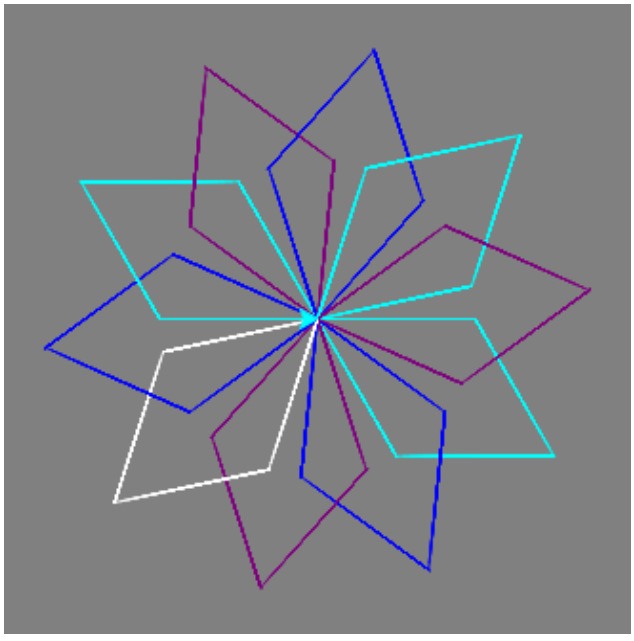
Python

Step 1 Introduction

Create a beautiful landscape of snowflakes using Python Turtle. This is great fun and a great way to start learning how to code with Python.

What you will make

Digital snowflakes out of code, like this one:



What you will learn

By making snowflakes with code you will learn how to:

- Draw lines and make turns with Python Turtle
- Change the pen colour randomly
- Use loops to repeat some instructions and create shapes

- Use more loops to create spiral patterns
- Create a function to draw a snowflake

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (https://www.raspberrypi.org/curriculum/):

- **Combine programming constructs to solve a problem** (https://www.raspberrypi.org/curriculum/programming/builder)

Step 2 What you will need

Hardware

- A computer capable of accessing the **trinket.io** (https://trinket.io) website

or

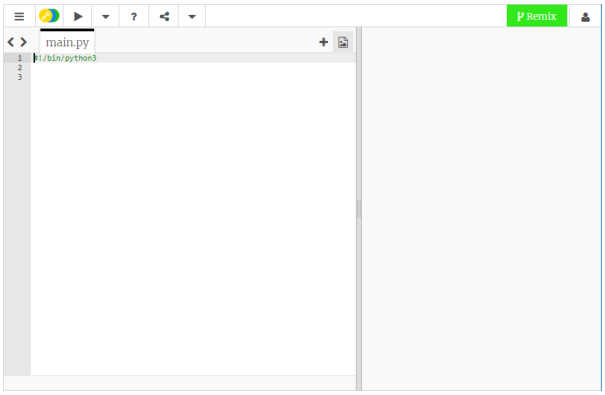
- A computer that has Python 3 installed

Software

This project can be completed in a web browser using **trinket.io** (https://trinket.io).

Step 3 How to draw with Python Turtle

- Open the **blank Python template trinket** (http://jumpto.cc/python-new).
- Type the following into the window that appears:



The line `#!/bin/python3` just tells your computer that we're using Python 3 (the latest version of Python).

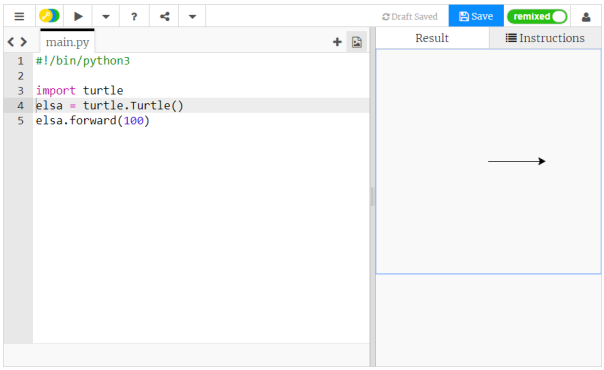
- To begin using Turtle in Python, you need to import the Turtle library. At the top of the text editor window, type `import turtle`.
- Time to give your turtle a name! You can use a variable to do this. I'm naming my turtle `elsa`, but you can name yours whatever you like.

```
elsa = turtle.Turtle()
```

- Now you can tell your turtle what to do, for example, to move forward `100`. Give it a go!

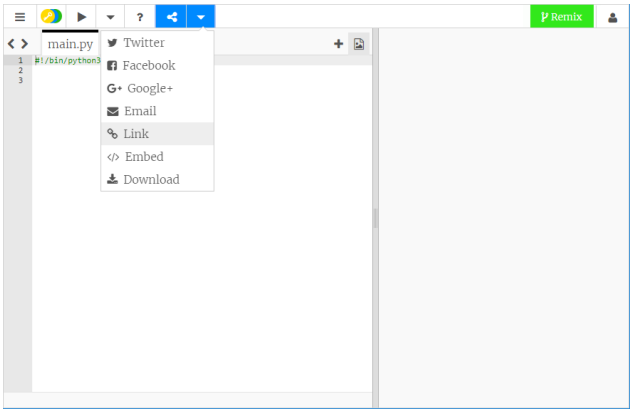
```
elsa.forward(100)
```

- Click on **Run** to run your first Turtle program. What happens?



You don't need a Trinket account to save your projects!

If you don't have a Trinket account, click the down arrow and then click **Link**. This will give you a link that you can save and come back to later. Just remember that you'll need to do this every time you make changes to your code, as the link will change!



If you have a Trinket account, you can click **Remix** to save your own copy of the trinket.

Step 4 How to turn with Python Turtle

Your turtle is only moving in one direction so far. This is good news if you want to draw a straight line, but to draw a shape such as a square, your turtle is going to have to turn.

- Below 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 degrees does the right angle of a square have? 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 below 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?

Try adding the following code and running you program:

```
elsa.right(90)
elsa.forward(100)
```

What was the result? How can you finish the shape to draw a square?

Step 5 Using loops to create shapes

To create a square, you have repeated some lines of code. This is not the most efficent way of doing it. Instead of typing out many lines of code, it's easier to use a loop.

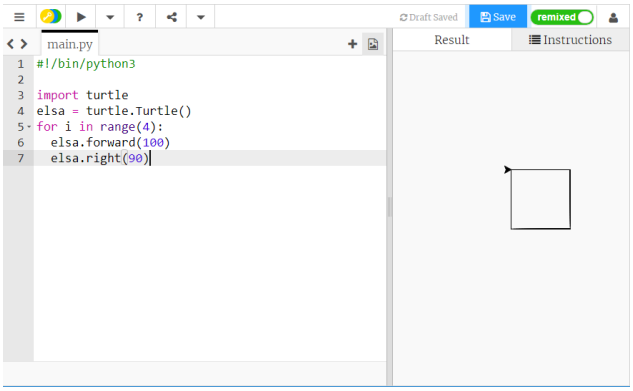
Instead of code to create a square like this:

```
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.



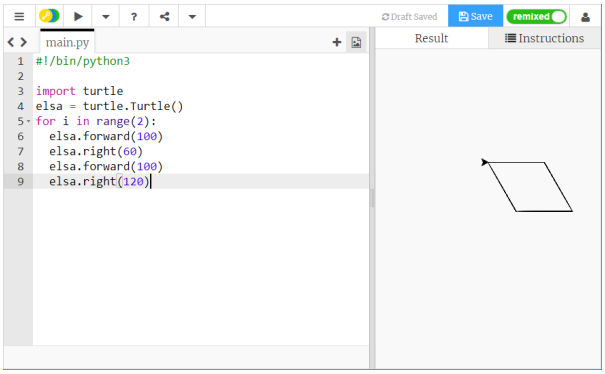
Step 6 Creating spiral patterns

Enough squares! 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)
```

This 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 do this to easily make a drawing that looks like a snowflake.

- Above the line `for i in range(2):` for your parallelogram, type:

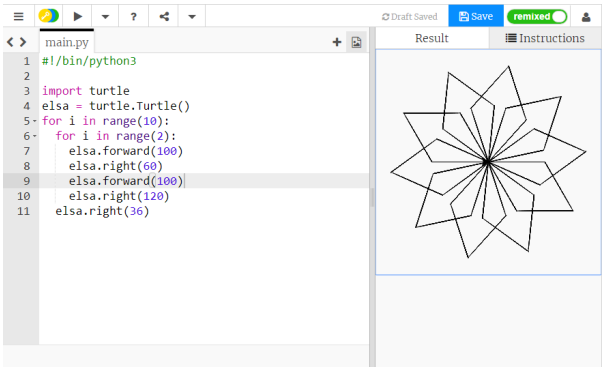
```
for i in range(10):
```

How many times will this loop go round?

- Move your cursor to the line below your sequence of code, and press the space bar four times to **indent** the code you're about to write. Indentation in Python is very important to ensure that your code works as you expect! Now type:

```
elsa.right(36)
```

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



Step 7 Changing the pen colour randomly

So far the turtle has been drawing black lines on a white background. Now it's time to add colour!

- To set the colour of the turtle, move you're cursor below where you named your turtle and before your loops, and type in the following:

```
elsa.color("cyan")
```

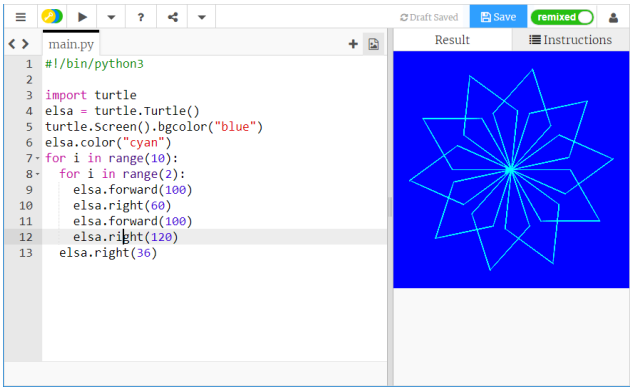
Note: The spelling of 'colour' is different in other countries. In the US, it is spelled 'color', and in Python it has to be spelled the American way to work.

I have chosen to use the colour **cyan**, but you can use any from this list:

- "blue"
- "magenta"
- "grey"
- "purple"

You can also change the colour of the background window. To set the colour of the background, use this instruction below the code you've just written:

```
turtle.Screen().bgcolor("blue")
```



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

- First you will need to import the `random` library: below `import turtle`, type `import random`.
- Next, change the background colour from `"blue"` to `"grey"`.
- Below that line, create a variable called `colours` to store a list of the colours to select from, like this:

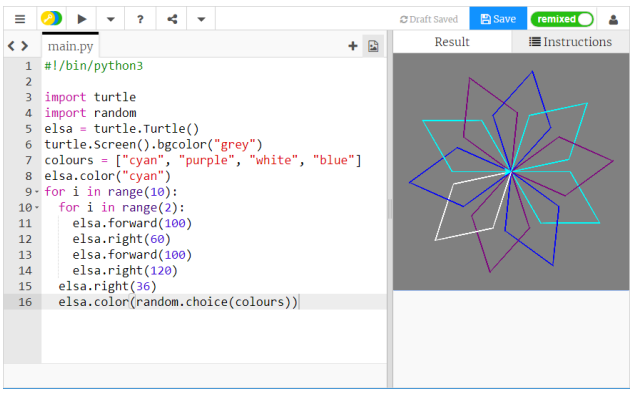
```
colours = ["cyan", "purple", "white", "blue"]
```

- At the end of the spiral loop, below `elsa.right(36)`, type:

```
elsa.color(random.choice(colours))
```

Note: make sure this line is also indented, so that your program knows it's within the loop.

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



More colours

There are a lot more colours you can choose from! Have a look at **this website** (https://wiki.tcl.tk/37701) for a complete list.

Step 8 Using a function to draw a snowflake

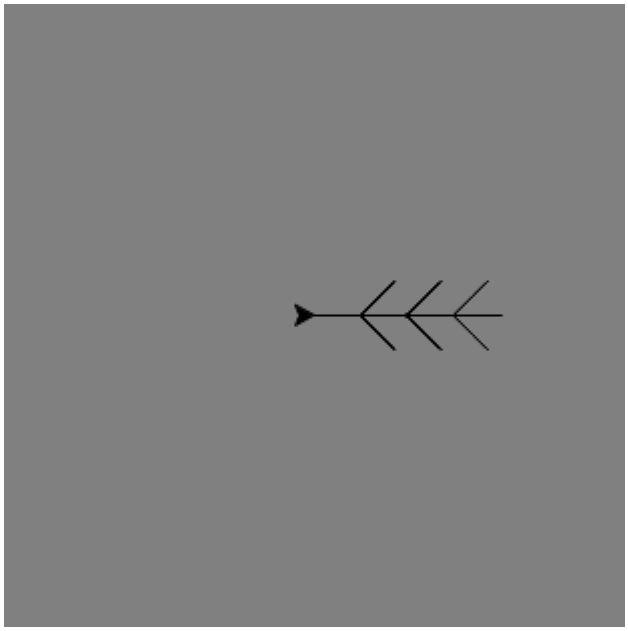
Your parallelogram snowflake is cool, but it does not look as snowflake-like as it could. Let's fix that!

For this drawing, we need to move the turtle from the centre of the window. The `penup()` and `pendown()` instructions let us do this without drawing a line, just like picking up a real pen from the paper and moving it somewhere else to start writing.

- Type the following instructions below the `colours` list:

```
elsa.penup()
elsa.forward(90)
elsa.left(45)
elsa.pendown()
```

Let's write the code to draw one branch of a snowflake, and store it inside a **function**. Then you can simply repeat it over and over to create a complete snowflake.



- Define a function called `branch` by typing:

```
def branch():
```

- Remove the code for the parallelogram snowflake loops. Add the following code indented inside the `branch` function:

```
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 to check that all your indentation is correct, otherwise your code won't work!

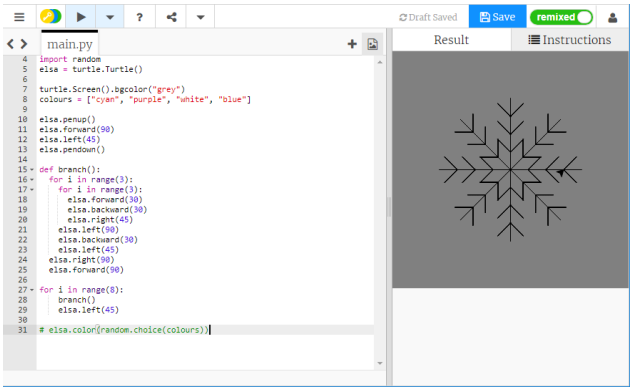
- Write a final section of code to **call** the `branch` function (which means to run it) eight times. You can use a loop again as for your last snowflake:

```
for i in range(8):
    branch()
```

```
    elsa.left(45)
```

- Put a `#` at the start of the `elsa.color(random.choice(colours))` 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!



Step 9 Challenge: every snowflake is different

- Can you uncomment the random colour instruction and place it in the function, so that each branch has a different colour?
- Can you create a snowflake function and then repeat it all over the window to create a snowflake landscape?
- Can you draw snowflakes of different colours and different sizes, like those in Carrie Anne's video **Make snowflakes with code** [(https://www.youtube.com/watch?v=DHmeX7YTHBY)](https://www.youtube.com/watch?v=DHmeX7YTHBY)?