Geometric Patterns using Python Turtle

GETTING STARTED

What will you do?

In this tutorial, you will learn how to write code in **Python** and create simple geometric patterns using the **Python Turtle** tool.

What do you need?

You will need a computer with the Python programming language installed on it. You can either use a Raspberry Pi or a computer running Windows or Mac OS.

The easiest way to code in Python is to use an Integrated Development Environment (IDE). You can either use **IDLE** (the standard Python Integrated Development Environment) or **Thonny Python IDE**. Both IDLE and Thonny use **Python 3**.

If you are using a Raspberry Pi, you can either open **IDLE** or **Thonny Python IDE** by following these steps:

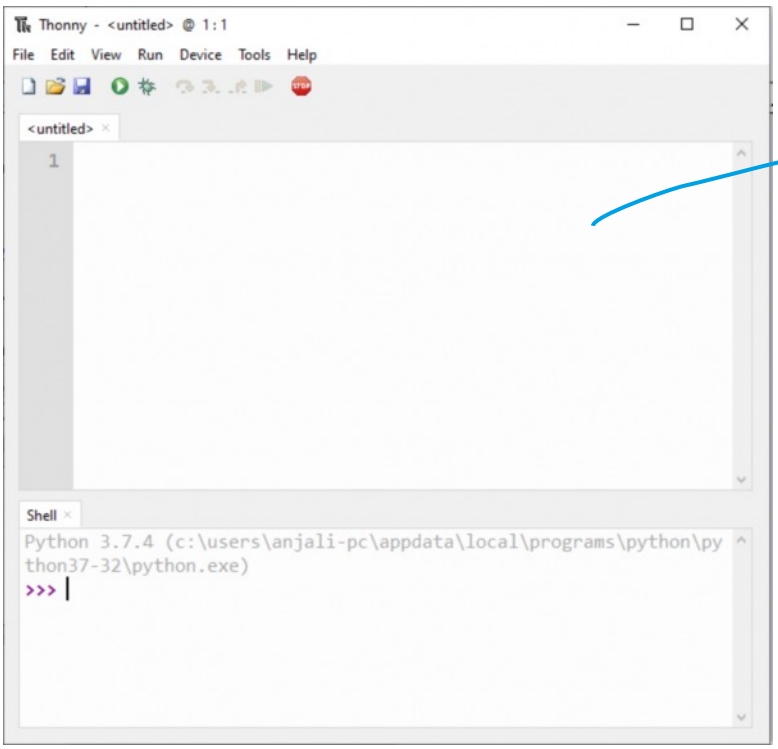
- Open the **applications menu** (**MENU** button) and select **Programming**;
- Click on **Python 3 (IDLE) or Thonny Python IDE**

Otherwise, if you are using a computer running Mac OS or Windows, you can download **IDLE** from https://www.python.org/downloads/ and install the latest version on your machine.

GETTING STARTED

IDE

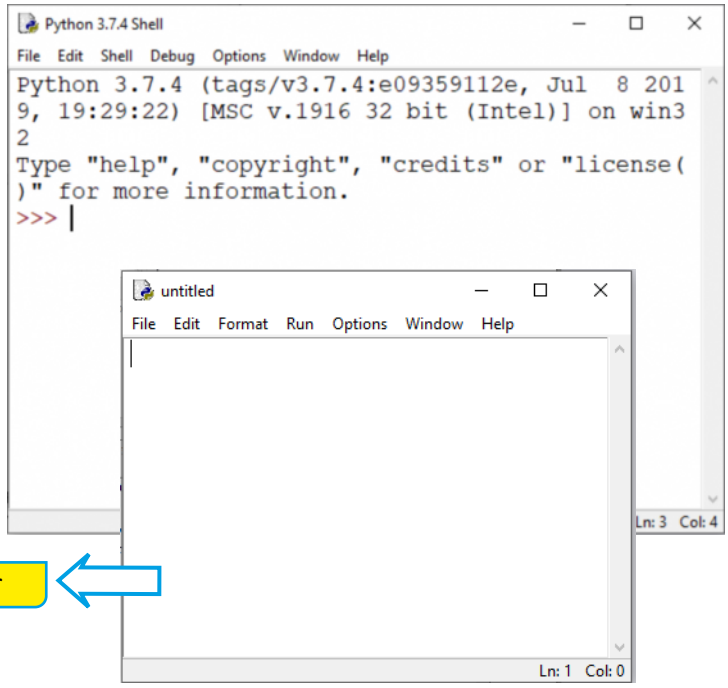
Depending on which IDE you are using, you will see one of the two interfaces shown below:



Start typing your code here

Thonny IDE

IDLE



Python Shell

Text Editor

Once the Python Shell is open in IDLE, click on **FILE** and then **NEW FILE**. This will open another window, the **TEXT EDITOR**, where you can start to write your Python program ...

TURTLE LIBRARY

In the text editor window, **CLICK ON THE BLANK SPACE** to start typing.

Start by importing the Python turtle library in to your program. Type the following line:

```
import turtle
```

You can give your turtle a name, we're going to call ours 'ellie':

```
ellie = turtle.Turtle()
```

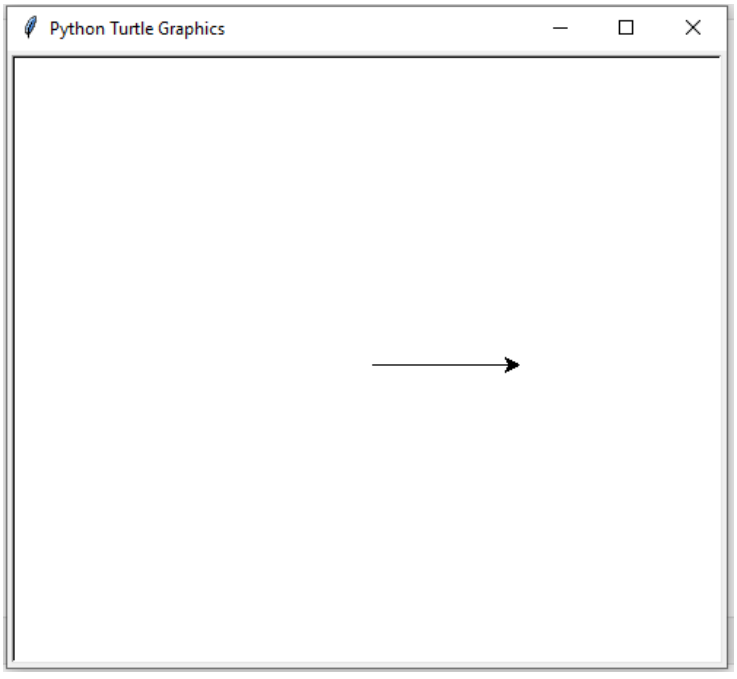
Finally, make the turtle move around the screen. Let's start with a simple forward movement…

```
ellie.forward(100)
```

This should make the 'turtle' move forward 100 pixels. To see if this works, you need to run the program. To do this, **PRESS F5** on the keyboard.

You should be asked to **SAVE** your program first. Click **OK** and **GIVE IT A NAME** (you can call it "turtle").

When the code runs, the Python Turtle Graphics windows should come into focus and display the turtle moving forward 100 'steps'!



MAKE A SHAPE

Let's draw a shape next, like a square.

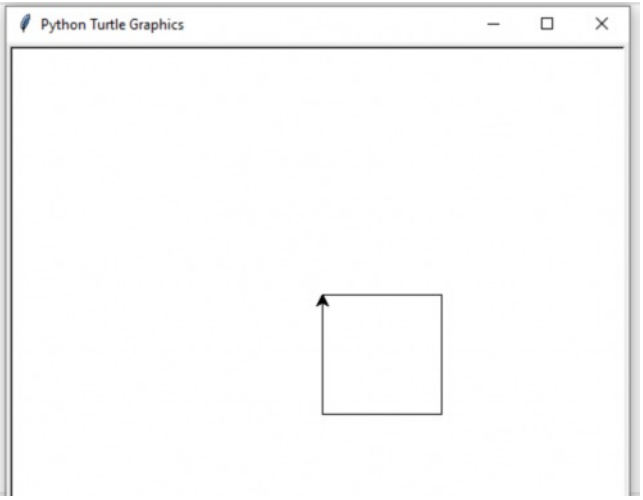
From the previous code, your turtle should have moved forward. Next we need to make it turn right, then go forward again and so on.

The code looks something like this (**type these lines under the previous code**):

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

REMEMBER:
- PRESS F5 ON THE KEYBOARD TO RUN
- SAVE WHEN ASKED TO

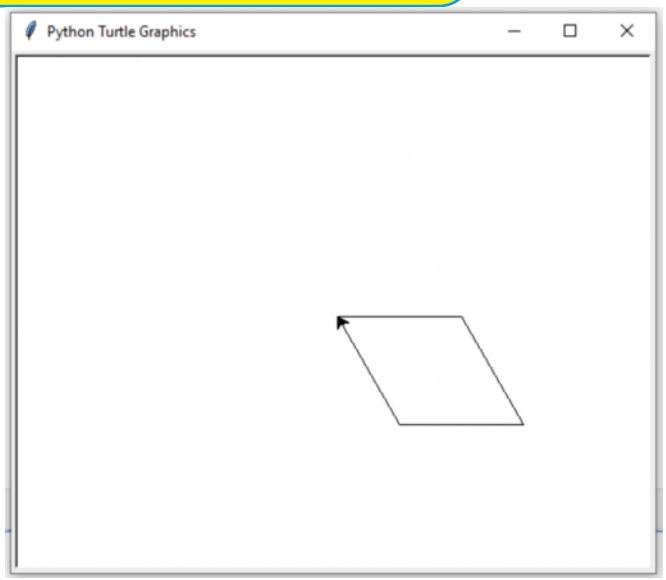
```
1  import turtle
2
3  ellie =  turtle.Turtle()
4
5  ellie.forward(100)
6  ellie.right(90)
7  ellie.forward(100)
8  ellie.right(90)
9  ellie.forward(100)
10 ellie.right(90)
11 ellie.forward(100)
```



You can try and draw a parallelogram, instead…

```
ellie.right(60)
ellie.forward(100)
ellie.right(120)
ellie.forward(100)
ellie.right(60)
ellie.forward(100)
```

REMEMBER:
- Type these lines under `ellie.forward(100)` instead of the lines above



LOOPS

The square that you just made had the 'forward' and 'right' movements being repeated. **Repetition** or **Iteration** in code can be put in a **loop**.

There are different kinds of loops. We will be using the 'for' loop here:

```
for i in range(4):
    ellie.forward(100)
    ellie.right(90)
```

Note where the 'for' loop gets slotted in

```
1  import turtle
2
3  ellie =  turtle.Turtle()
4
5  for i in range(4):
6      ellie.forward(100)
7      ellie.right(90)
8
```

How will you put the parallelogram code in a loop?

GEOMETRIC PATTERNS

Now, you know how Python turtle works, let's make some patterns!

Open a new text editor window [**NOTE**: In Thonny, you get this by clicking on **File** → **New**]. In the new text editor window, **CLICK ON THE BLANK SPACE** to start typing.

Type in the following code:

```
import turtle

ellie = turtle.Turtle()

def pattern():
    for i in range(4):
        ellie.right(90)
        ellie.forward(100)

for num in range(8):
    ellie.pensize(2)
    pattern()
    ellie.left(45)
```

Our earlier code to draw a square but now put in a Python function called pattern() which can be 'called' from anywhere in the program.

'Calling' the function pattern() 8 times.

TRY THESE:

- Can you change the size of the pattern? [HINT: Play with the numbers in the 'forward' and 'backward' functions]
- Can you change the thickness of the lines being drawn? [HINT: pensize()]
- Is the drawing speed slow for you? Add this line after you've created your turtle:

  ```
  ellie.speed(0)
  ```

  Here are other 'speeds' you can play with: 1 (slowest), 3 (slow), 6 (normal), and 10 (fast).

ADD SOME COLOUR

Well done on getting this far!

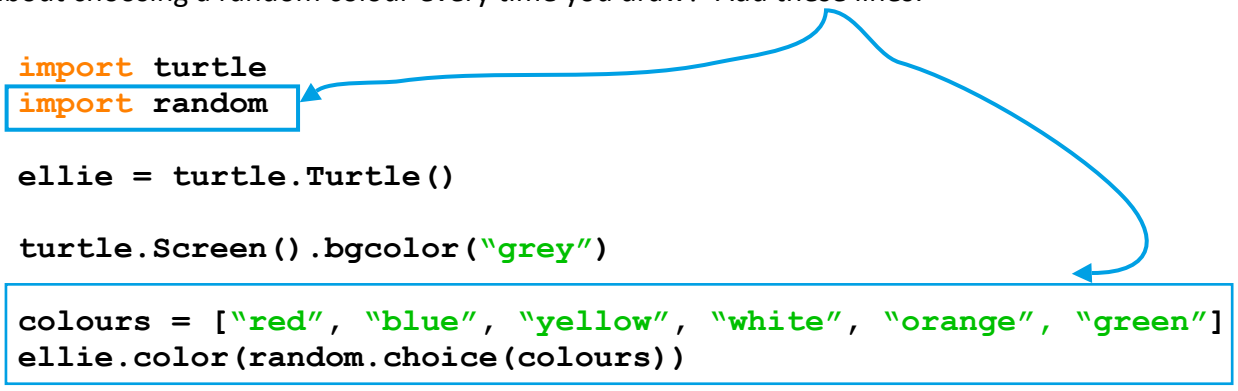
Let's add a little bit of colour to your creation…

Add these lines after you've created your turtle:

```
turtle.Screen().bgcolor("grey")
ellie.color("white")
```

You don't have to stick with grey and white, you can use other colours. Experiment!

How about choosing a random colour every time you draw? Add these lines:



```
import turtle
import random

ellie = turtle.Turtle()

turtle.Screen().bgcolor("grey")

colours = ["red", "blue", "yellow", "white", "orange", "green"]
ellie.color(random.choice(colours))

def pattern():
    for i in range(4):
      ellie.right(90)
      ellie.forward(100)

for num in range(8):
    ellie.pensize(2)
    ellie.color(random.choice(colours))
    pattern()
    ellie.left(45)
```

TRY THESE:

- Change the angle the turtle turns after it has drawn one square and see what that does to your pattern. [HINT: Play with the number in `ellie.left(45)`]
- Remember the parallelogram pattern we did earlier? Can you change the square being drawn above into a parallelogram and draw a new pattern?
- What other geometric figures can you draw? [HINT: Try changing the angle (in `ellie.right()`) and length (in `ellie.forward()`).]

Fantastic work! Well done!