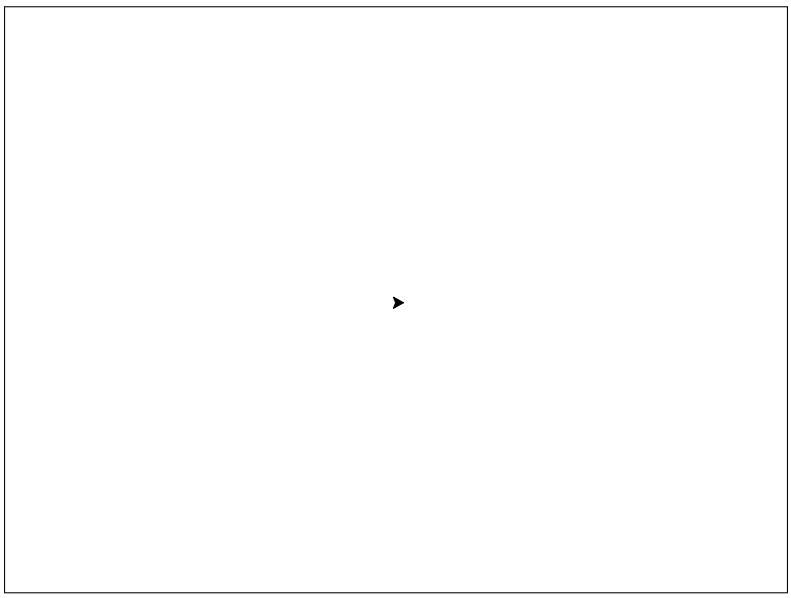
Introduction

Here is a simple turtle programme (`turtle_doing_nothing.py`):

```
1 from turtle import *
2
3 tom=Turtle()
4
5 tom.getscreen()._root.mainloop()
```

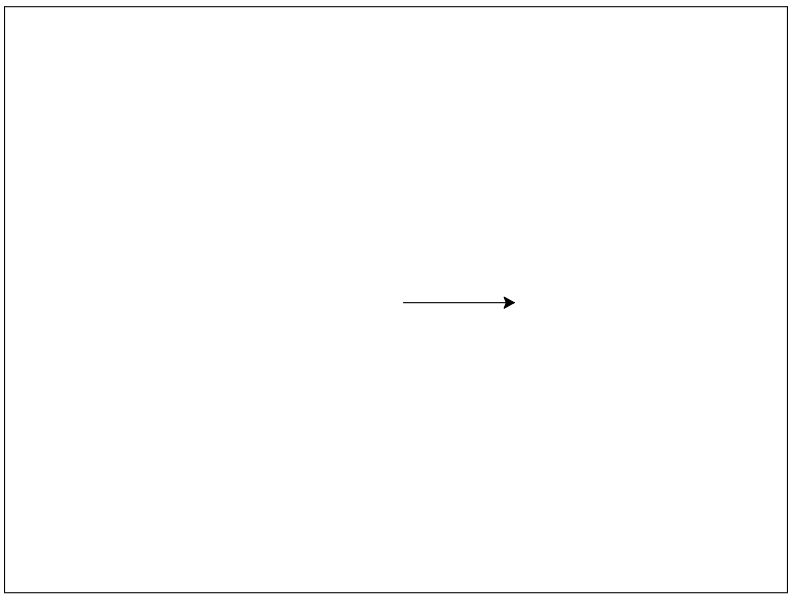
Line 1 and **line 5** aren't worth spending much time on at first, the first line imports the library of commands related to turtle, **line 5** prevents the computer from closing the graphics window when the programme has finished running; we won't include this line again, though it is needed. **Line 3** is important, it tells the computer to make an object, in this case a `Turtle` and call it `tom`, it knows what a `Turtle` is from the library it imported in line 1; in the instructions on what to do when making a `Turtle` the computer is told to open a graphics window and to draw the turtle, a little arrow shape.



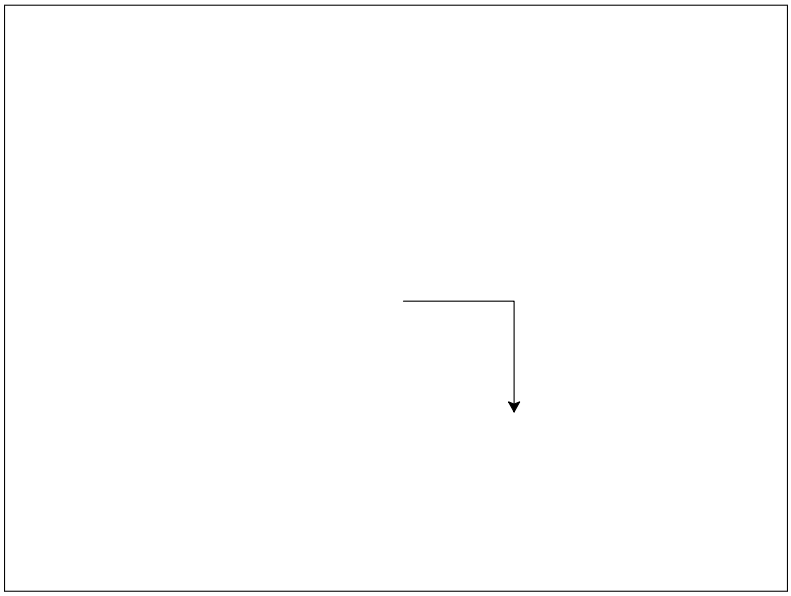
Here the turtle does something (`line.py`):

```
1 from turtle import *
2
3 tom=Turtle()
4
5 tom.forward(100)
```

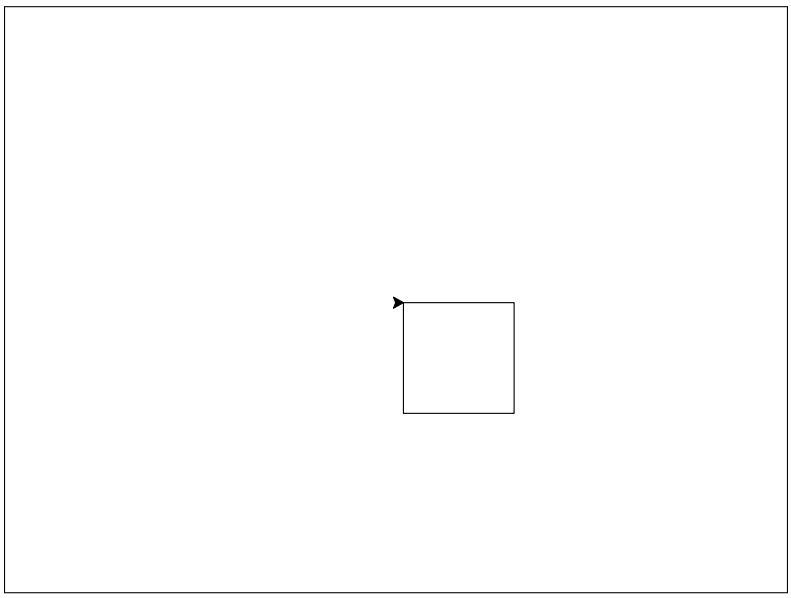
The extra line, **line 5**, tells the turtle to move forward by 100 units, this is an important piece of Python syntax, to tell an object to do something you use a dot followed by the command, here it tells the `Turtle` called `tom` to perform the command `forward`. Of course, the command has to make sense for whatever type of object it is dotted onto, but here it does, `forward` is one of the defined commands for a `Turtle` object.



`Turtle` objects have another command `right(90)` which turns the turtle by 90°. QUESTION: Can you write a programme to draw this:



QUESTION: How about a square?



Perhaps you programme to draw a square looked like this

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

There are two problems with this, most obviously it is boring writing in the same two lines again and again; secondly, the programme is inflexible and hard to read, we'll deal with the inflexible bit later, but as for the hard to read, to know that it draws four lines and has four corners you need to count the lines; it would be better if the 'fourness' was more apparent, as it is in this programme (`square_loop`):

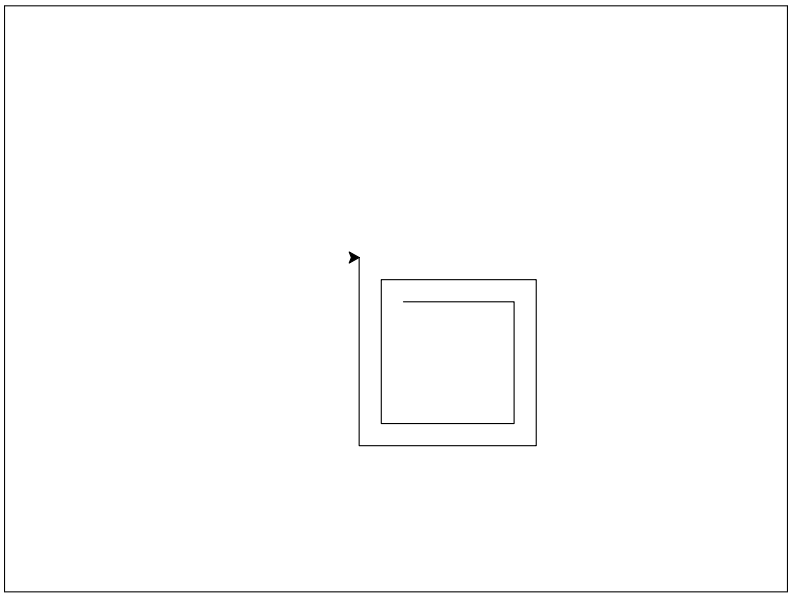
```
1  from turtle import *
2
3  tom=Turtle()
4
5  for i in range(0,4):
6      tom.forward(100)
7      tom.right(90)
```

The business part of this program is **line 5**; `range(0,4)` is the list of numbers `[0,1,2,3]`, so starting at zero and ending one before four; the full command says that `i` takes each value in this list in turn and then does all the stuff belonging to the command. Here the command

is a `for`, a command that says 'do everything that belongs to the for once for every value the variable, in this case i, is instructed to take'. In Python stuff belonging to a command is indented, so that means it does **line 6** and **line 7** once for each item in the list, that is four times. Of course, in this programme `i` isn't used for anything except counting but it could be (`spiral1.py`):

```
1 from turtle import *
2
3 tom=Turtle()
4
5 for i in range(0,8):
6     tom.forward(100+10*i)
7     tom.right(90)
```

giving



`i` is variable, it stores some data, in this case a number which changes each time the programme goes around the `for` loop. Once slightly confusing thing is 'scoping', which is where the variable is defined; the `i` is only defined inside the `for` loop, that is, it only exists while the programme is executing **line 5** to **line 7**, but that's fine, that's where we use it. QUESTION: Can you draw a triangular spiral like this: