# Python Lesson 6

This will be our final lesson of the term. You asked to learn some more about turtle graphics, so here they are, more turtles!

*A well-known scientist once gave a public lecture on astronomy. He described how the earth orbits around the sun and how the sun, in turn, orbits around the centre of a vast collection of stars called our galaxy.*

*At the end of the lecture, a little old lady at the back of the room got up and said: "What you have told us is rubbish. The world is really a flat plate supported on the back of a giant tortoise."*

*The scientist gave a superior smile before replying, "What is the tortoise standing on?"*

*"You're very clever, young man, very clever," said the old lady. "**But it's turtles all the way down!**"*

- from "A Brief History of Time" by Stephen Hawking

## Filling Shapes

To create a filled shape, first turn on filling with your preferred colour:

```
turtle.fillcolor("red")
turtle.begin_fill()
```

Then draw the shape that you want (a square with sides of 200 pixels – "picture elements" or dots):

```
for side in range(0, 4):
    turtle.forward(200)
    turtle.left(90)
```

Finally, tell Python that you have finished drawing the shape and that it should now be filled in:

```
turtle.end_fill()
```

## Choosing Colours

We've seen how you can give turtle graphics an English name for the colour to draw. Python knows a lot of colour names (more than 750), which are listed here!

https://www.tcl.tk/man/tcl8.4/TkCmd/colors.htm

What happens if you want to use a colour name that Python doesn't recognise, like "amber"? You'll get an error.

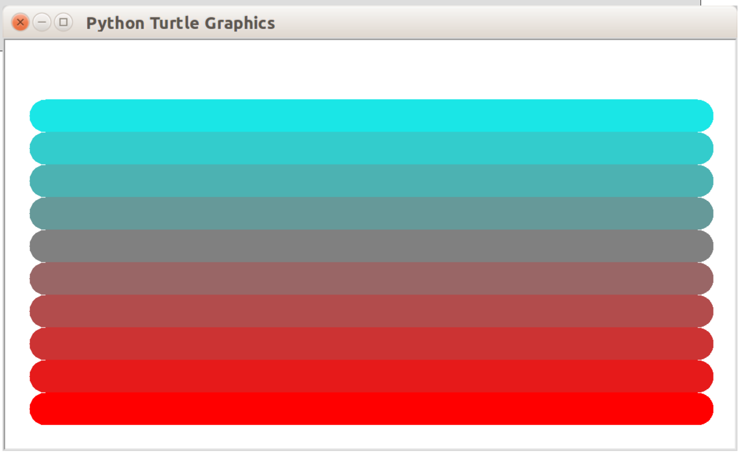
```
turtle.fillcolor("amber") # Raises an error
```

Fortunately, this doesn't mean that it's impossible to use this colour! What you need to do instead is specify how much red, green and blue (RGB) to mix together to make the colour that you want. Each value is between zero and 1. Red, green and blue are the primary colours of light, you can make any colour that your screen can display. In fact, computer screens are made out of tiny red, green and blue lights (LEDs).

```
# "amber" is lots of red (100%), quite a lot of green (75%) but no blue (0%)
turtle.fillcolor( (1.0, 0.75, 0.0) )
```

Using RGB numbers instead of names has an advantage that you can calculate colours with maths, or use the `random.random()` function to pick random amounts of each colour component!

```
import turtle

turtle.reset()
turtle.width(50)

for line in range(0, 10):
    turtle.penup()
    turtle.goto(-500, line*50)
    turtle.pendown()
    turtle.color(((10-line)/10, line/10, line/10))
    turtle.forward(1000)
```



## Many Turtles

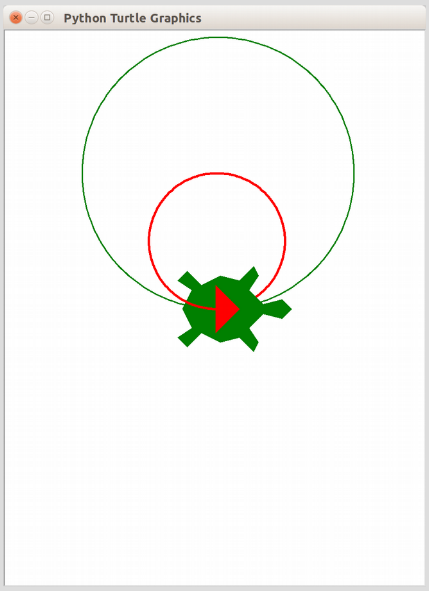
So far we've only used a single turtle in each picture.

You can create your own turtles and store them in a variable using

```
variable_name=turtle.Turtle(shape="shapename")
```

Here we create two turtles, one with a turtle shape and the other with an arrow shape. We store them in two variables, green_turtle and red_arrow.

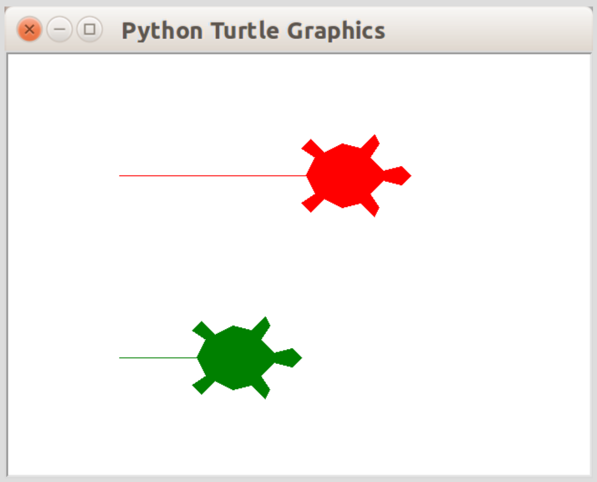
```
import turtle

def draw_circle(the_turtle, size):
    for side in range(0, 180):
        the_turtle.forward(size)
        the_turtle.left(2)

green_turtle=turtle.Turtle(shape="turtle")
green_turtle.color("green")
green_turtle.shapesize(10)
green_turtle.pensize(3)

red_arrow=turtle.Turtle(shape="arrow")
red_arrow.color("red")
red_arrow.shapesize(5)
red_arrow.pensize(5)

draw_circle(green_turtle, 10)
draw_circle(red_arrow, 5)
```



## Clicking on Turtles

You can detect when someone clicks on a turtle with a mouse pointer, and trigger an action. Call `turtle.onclick(`*`your_function`*`)`, then function called *`your_function`* will get called when the turtle is clicked. In this example, we create two turtles and move them forward towards a finish line when they are clicked – a turtle race!

```
import turtle

turtle.title("Turtle Race") # Sets the window title

turtle1=turtle.Turtle(shape="turtle")
turtle2=turtle.Turtle(shape="turtle")

def win(the_turtle):
    """Spin the turtle when it wins"""
    for i in range(0, 180):
        the_turtle.left(4)

def turtle1_clicked(x, y):
    turtle1.forward(10)
    if turtle1.xcor() >= 200:
        win(turtle1)
        reset_turtles()

def turtle2_clicked(x, y):
    turtle2.forward(10)
    if turtle2.xcor() >= 200:
        win(turtle2)
        reset_turtles()

def reset_turtles():
    turtle1.reset()
    turtle1.color("red")
    turtle1.shapesize(5)
    turtle1.penup()
    turtle1.goto(-200, 100)
    turtle1.pendown()
    turtle1.onclick(turtle1_clicked)
    turtle2.reset()
    turtle2.color("green")
    turtle2.shapesize(5)
    turtle2.penup()
    turtle2.goto(-200, -100)
    turtle2.pendown()
    turtle2.onclick(turtle2_clicked)

reset_turtles()
```

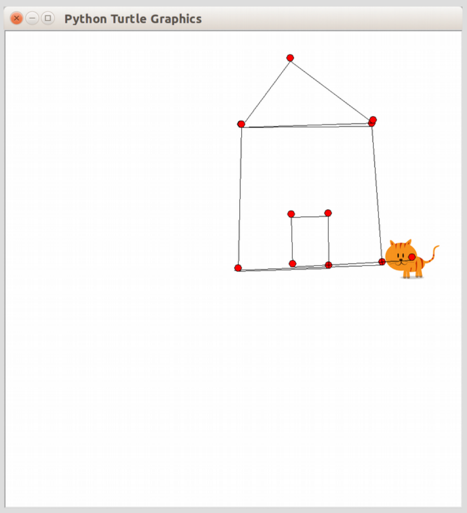


The function that you provide to "onclick" takes two parameters, x and y, which are the coordinates of where the mouse pointer was clicked. You can also register a function "onrelease" so that you can detect the coordinates of where the mouse pointer was released (unclicked). There's an example of this below.

## Making your own Turtle Shapes

You can save images as "gif" files and import these to create your own turtle shapes. This program lets us draw with a cat-turtle! You can click on the cat, move the pointer, then release the click to move the cat to that location. It will draw a line between the two points.

```
import turtle

turtle.register_shape("cat.gif")
cat=turtle.Turtle("cat.gif")
cat.speed("fastest")

def cat_released(x, y):
    cat.goto(x, y)
    cat.fillcolor("red")
    cat.begin_fill()
    for i in range(0, 45):
        cat.left(8)
        cat.forward(1)
    cat.end_fill()

cat.onrelease(cat_released)
```



Now that you can create many turtles, detect when they have been clicked on and dropped, and you can create your own pictures for each turtle, you might have enough information to draw a picture of a chess board with all the different types of pieces, and let the player move them around! You can be creative.

# What's next? Learning more about Python

*I hope you've enjoyed this series of Python lessons. The lessons and other materials can be downloaded from https://github.com/chrisglencross/python-lessons/.*

If you want to learn more about Python, for example over the summer holidays, here are some places you can look. There's much more to learn about Python; the language has many features and modules that we haven't covered this term.

http://www.python.org

This is where you can get Python and read the documentation!

http://interactivepython.org/runestone/static/thinkcspy/index.html

A tutorial for learning Python, with lots of videos.

http://www.pygame.org/

Pygame is a module which is designed for writing arcade games in Python, with sound, music and graphics. We haven't had time this term to try it (and it isn't installed on the computers at school!) but if you want to learn about writing games in Python, without limiting yourselves to just turtle graphics or text adventures, you can try it at home.

http://programarcadegames.com/

A tutorial for getting started with writing games with Python and Pygame.

There are also lots of books available. You might be able to find one in the library. If you're going to get a book, I would suggest finding one that focuses on Python version 3! Python 2 is a little bit different to what we've done here, but the changes are fairly small.