1 Python Week 7 - Exercises

This week, the main goal of the exercises is to get you familiar with using **documentation** for packages; an essential skill for working on real world projects.

1.1 Numpy

Before we begin giving our turtle orders, we're going to have a go at installing a commonly used package for maths, especially linear alegbra (vectors, arrays, metrices etc.).

1. Install the numpy package from the command line or terminal (depending on your operating system) using **pip install numpy**
    - NOTE: If you get an error saying that PIP isn't installed, refer to the pip_help.pdf document in the repository (or ask a TA!)
2. Create a new python script, which imports numpy and creates a numpy array containing ones (this should be two lines of code)
    - See https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.ones.html
3. Search google for *"Create array of random numbers in numpy"* and try and find a way of creating an array containing 10 random numbers in your script
    - print() the array, and run the code a few times to make sure it's working

1.2 Turtle time

Now for the fun stuff. You'll need to make extensive use of the turtle documentation for these exercises: https://docs.python.org/3/library/turtle.html

1.2.1 Try some triangles

Create a basic turtle script (similar to the one in the lectures notes - see the turtle_example.py file in the github repository) which draws a triangle and fills it with a color of your choice.

1. Start with an equilateral triangle
2. Try to make an almost-perfect (no overshooting lines) isoceles triangle

Challenge : Pythagorean Perfection

This is essentially a maths challenge - don't try this during the session!
Create a function which takes three angles and a base length. Then use these angles and the base length (i.e. length of the "bottom" of the triangle) to draw a perfect (no overshooting lines) triangle.

- Ensure that the the triangle you're trying to draw is mathematically valid (i.e. what do angles in a normal triangle add up to?)
- In order to work out the lengths of the triangle's other edges, you'll want to use trigonmetric functions - Numpy can help you here [Make sure you're working in degrees, not radians]

1.2.2 Pretty patterns

1. Using a for loop, create a nice repeated pattern (like a star)
    - Maybe the turtle documentation has some useful examples ;)

Challenge: Symmetry makes me happy

1. Using a while loop, create a closed (i.e. the turtle ends where it started) pattern
    - You'll want to make use of the fact that you can get the turtle's coordinate position using some of the package's functions
2. If your pattern makes use of a repeated number (e.g. turning by a fixed angle many times), wrap it up in the function and allow the user to play around with the value

1.2.3 Solar System

Check the turtle documentation for a way of drawing circles Draw a yellow sun with a green earth some distance away, and show the orbit of the earth by drawing a black circle along its path.

Challenge : Animate

- Try and find a way to make the turtle move so quickly that it draws the entire solar system almost instantly
- Then try and find a way to "clear the screen" and hide the turtle
- Now think of a clever way to repeatedly draw the entire solar system, whilst moving the earth slightly further in its orbit, so that it looks like the earth is actually moving Hint: To make your animation run at a nice rate, you might want to consider using the time package's "time.sleep()" function