ECMM171 Programming for Engineers
Assignment 1: Basic Python

Date set: Tuesday 17th October
Hand-in date: **12:00 (noon) Thursday 2nd November 2017**

- This assignment will count for **30%** of your total grade in the course.
- Remember that all work you submit should be **your own work**. This assignment is not meant to be a team exercise. The University treats plagiarism very seriously.
- Ensure that your program runs **without any error** on the Harrison computers. Additionally, ensure that you pay careful attention to the instructions and marking scheme below, particularly in regards to defining specific functions, commenting your code appropriately and ensuring its readability.
- You should submit your coursework using the Harrison online coursework submission system, which you can find at `http://empslocal.ex.ac.uk/submit/`, by the deadline of **12:00 (noon) Thursday 2nd November 2017**. You should submit all source code and any other requested materials in a single compressed `zip`, `rar` or `tar` file.

  You will be sent an email by the submission system asking you to confirm your submission by following a link. Your submission is not confirmed until you do this. **Failure to follow the link will result in your file not being saved and your submission not being marked.**
- If you have more general or administrative problems please e-mail me. Always include the course number (ECMM171) in the subject of your e-mail.

This assignment consists of three questions that should be completed throughout as the material is presented in lectures. As a guide for how the material maps to questions:

- **Question 1:** requires use of `if` and basic input which is covered in day 1;
- **Question 2:** requires functions (covered in day 3), testing and file input (covered in day 4);
- **Question 3:** requires functions (covered in day 3), string formatting (covered in day 4) and `NumPy` and plotting with `matplotlib` (covered in day 5).

The questions follow below: read them carefully, and consider the range of things that could go wrong with respect to user input.

1 Questions

1. The roots of a quadratic equation $ax^2 + bx + c$ can be found by using the quadratic equation

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

   Write a program called `quadratic.py`, which reads in values of a, b and c from the user and solves the equation. Your program should:

   - read a, b and c (you may assume these will be supplied as integers);

- tell the user whether there are two real roots, two complex roots, or one repeated root;
- print out the value of the roots.

Here is an example of how your program should run in the terminal:

```
Enter the value of a: 1
Enter the value of b: 5
Enter the value of c: 6
The equation has two real roots:
    x1 =  -3.0
    x2 =  -2.0
```

2. A *palindrome* is a word that remains the same when all of its letters are reversed: for example *"radar"* and *"tenet"* are both palindromes. Write a program called `palindrome.py` that:

   - defines a function `is_palindrome(word)`, which should determine if an individual `word` is a palindrome, and return `True` or `False` accordingly. For example:

     ```
     >>> is_palindrome('tenet')
     True
     >>> is_palindrome('cheese')
     False
     ```

     Your function should be appropriately documented with a docstring and tested using doctesting, and be able to deal with unexpected input sensibly.

   - Defines a `main` block that will read words from a file `words.txt` (which can be found the ELE page) and outputs all palindromes into a new file called `palindromes.txt`, one per line.

3. Consider the problem of launching a projectile at a given speed v at some initial height y_0 in a vacuum (so that there are no effects due to drag). Your task is to write a program called `projectile.py` that will:

   - print the distance travelled, maximum height attained and travel time for a range of launch angles;
   - produce a plot of their trajectories using `matplotlib`. An example of an ideally-formatted plot is shown in figure 1.

Here is an example of how your program should run in the terminal:

```
Enter height [m]: 10
Enter initial speed [m/s]: 30

Angle [deg]   Distance [m]   Max height [m]   Travel time [s]
------------------------------------------------------------
10            60.6965       11               2.0544
20            79.3819       15               2.8159
30            94.0794       21               3.6211
40            101.0092      29               4.3953
50            98.0790       37               5.0861
60            84.8576       44               5.6572
70            62.4105       51               6.0825
80            33.0520       54               6.3446
```

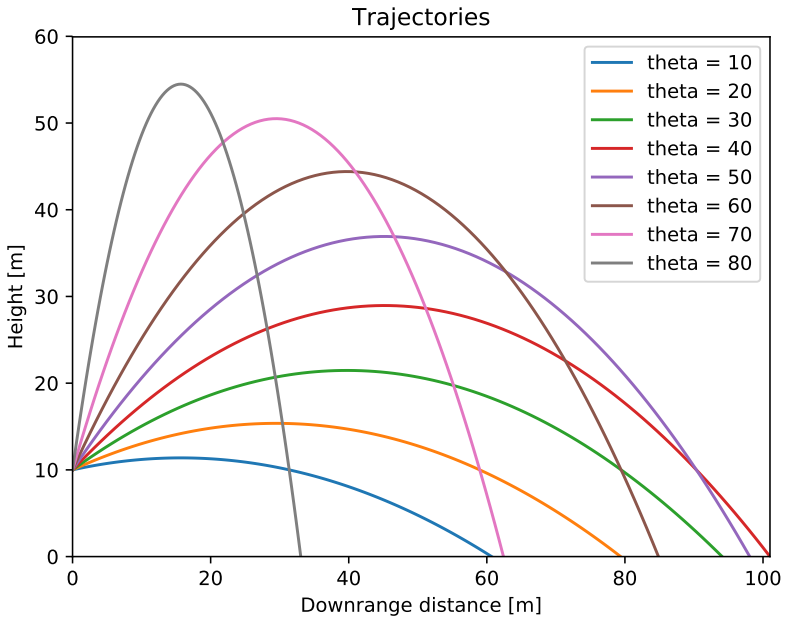

Figure 1: Example plot showing the trajectories for a projectile fired at initial speed $v = 30\,\mathrm{ms}^{-1}$ and height $y_0 = 10\,\mathrm{m}$ at various launching angles.

Your program should:

- Define a function called `fire_projectile`, which:
  - takes as arguments the initial velocity, launch height and launch angle;
  - returns the distance travelled, maximum height attained and travel time;
  - calls the `plot` command from `matplotlib` to plot the projectile's trajectory profile;
  - be documented using a docstring. There is no need for a doctest in this function.
- Define a `main` block, which:
  - asks the user for a (valid) initial height and speed;
  - calls `fire_projectile` for the angles of $\theta = 10°, 20°, \ldots, 80°$ and prints out the table shown above (with correct formatting);
  - calls the relevant commands from `matplotlib` to produce a correctly labelled plot with the correct limits on x and y axes (which should both start from 0). This plot should be saved as a PDF to a file called `trajectories.pdf`.

For your calculations, you can use the formulae:

$$\text{Distance travelled}: \quad d = \frac{v\cos\theta}{g}\left(v\sin\theta + \sqrt{(v\sin\theta)^2 + 2gy_0}\right)$$

$$\text{Height at position } x: \quad y = y_0 + x\tan\theta - \frac{gx^2}{2(v\cos\theta)^2}$$

$$\text{Travel time}: \quad t = \frac{d}{v\cos\theta}$$

where v is the initial velocity, y_0 is the launch height, θ is the launch angle and $g = 9.81\,\mathrm{ms}^{-2}$ is acceleration due to gravity.

2 Assessment criteria

Your submission will be assessed on the following criteria:

- Fully working implementation for question 1. **[20%]**
- Question 2:
  - Fully working implementation of `is_palindrome`. **[10%]**
  - An appropriate docstring/doctest for `is_palindrome`. **[5%]**
  - Fully working implementation of `main` block. **[20%]**
- Question 3:
  - Fully working implementation of `fire_projectile`. **[10%]**
  - Correctly formatted table of output. **[10%]**
  - Appropriate calls to `matplotlib` and correctly labelled figure. **[10%]**
- General discretionary marks for good programming technique, structure and commenting. **[15%]**

Important notes:

- 'Fully working' means that these routines run without any errors in the installed version of Python, they include comments and they behave as expected for a range of test cases.
- You should consider what invalid inputs to your functions might be supplied and act accordingly. For example if an invalid type is given as an argument you should check this using `assert`, and you should catch exceptions where appropriate.
- Routines that will not run due to an error will receive zero marks.
- Routines that run but are incomplete are still eligible to receive partial credit depending on their level of completeness, so make sure you submit a working program even if you have not fully managed to finish all the tasks.
- If there are no comments in the routine, 20% will be deducted from your mark. If the comments are not enough to sufficiently explain the code, then 10% will be deducted.

  Think carefully about what comments you should use: functions should use proper docstrings, and key loops and sections of the code should be documented properly. On the other hand, programs that have huge amounts comments for every line of code, are not appropriate!
- Figures should have appropriate axis labels.

3 Submission

You should package your `quadratic.py` for question 1, `palindrome.py` for question 2 and `projectile.py` for question 3 into a single `zip`, `rar` or `tar` file, and submit using the Harrison online coursework submission system as described at the top of this document.