ECM1408 Programming for Science
Continuous Assessment 1

Date set: Thursday 11th October, 2012
Hand-in date: **12:00 Thursday 25th October, 2012**

This continuous assessment (CA) comprises 15% of the overall module assessment.

This is an **individual** exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

Note that both paper (BART) and electronic submissions are required.

This CA tests your knowledge of the programming in Python that we have covered in the first three weeks of term, particularly lists, loops, conditionals and functions. Note that you will not be able to do all the questions immediately, but functions will be the subject of lectures at the beginning of week 4.

Make sure that you lay your code out so that it is readable and you comment the code appropriately.

Exercises

1. **Syntax errors.** Identify the *syntax* errors in the following Python code, stating why each is an error.

```
1  x = 10
2  y = 4x
3  z = 'Fred's tarantula'
4  w + 12 = x**2
5  mylist = [1, 2, 'three', z+y]
6
7  if x = y
8     for item in L:
9        if item**2 > z:
10          print item
11    else:
12       print z*3
```

Line numbers on the left are merely to help you refer to the lines.

You should submit your answers on paper via BART. **[10 marks]**

2. **Sums of squares.** Write a function `perfect(N)` to compute the sum of the first `N` perfect squares, that is $1^2 + 2^2 + 3^2 + \ldots + N^2$. Use your function to print the sum of the first 30 perfect squares.

   You should submit:

   - A paper copy of the code (via BART).
   - A paper copy of the output of your program showing the sum of the first 30 perfect squares (via BART).
   - An electronic copy of your program in a file named `perfect.py` (electronic submission).

   [10 marks]

3. **Approximating π.** The 14th century Indian mathematician Madhava of Sangaamagrama discovered an infinite series for π. The partial sum of $N+1$ terms of Madhava's series is

$$P_N = \sqrt{12}\sum_{n=0}^{N}\frac{(-3)^{-n}}{2n+1} = \sqrt{12}\left(1 - \frac{1}{3\cdot 3} + \frac{1}{5\cdot 3^2} - \frac{1}{7\cdot 3^3} + \ldots + \frac{(-1)^N}{(2N+1)\cdot 3^N}\right)$$

   As $N \to \infty$ then $P_N \to \pi$.

   Write code to construct a list of the first N partial sums P_N. Thus if $N = 3$, your list should be: `[3.46410161514, 3.07920143568, 3.15618147157]`, where $\sqrt{12} \approx 3.464$, $\sqrt{12}(1-\frac{1}{9}) \approx 3.0792$ and $\sqrt{12}(1 - \frac{1}{9} + \frac{1}{45}) \approx 3.15618$.

   Run your code for $N = 20$.

   Find the accuracy of the approximation as the number of terms increases by printing the difference between each term in your list of partial sums and an accurate value of π. You can get the value of π accurate to about 10^{-16} by importing it from the `math` library:

```
>>> from math import pi
>>> print pi
3.14159265359
```

   Here only the first 11 significant digits are printed but the remainder are accurate. (You may also want to import the `sqrt` function from the `math` library as done in lectures.)

   Roughly how many terms does Madhava need to achieve an accuracy of 1 part in a million?

   You should submit:

   - A paper copy of the code (via BART).
   - A paper copy of the output of your program showing the first 20 partial sums, the accuracy of the approximation and a statement of how many terms are need to achieve an accuracy of 1 part in a million (via BART).
   - An electronic copy of your program in a file named `madhava.py` (electronic submission).

   [20 marks]

4. **Hailstone sequences.** Successive terms x_n in the *hailstone* sequence are generated from an initial positive integer x_1 according to the rule:

$$x_{n+1} = \begin{cases} x_n/2 & \text{if } x_n \text{ is divisible by 2} \\ 3x_n + 1 & \text{otherwise} \end{cases}$$

If $x_n = 1$ the sequence terminates and it is conjectured that every hailstone sequence terminates in a finite number of terms; that is, there are no sequences that go on for ever.

Write a function `hailstone(n)` which returns a list comprising the terms in the hailstone sequence starting with `n`.

Use your function to write a program that prints the hailstone sequences starting with the integers $1, \ldots, 10$.

As you can see, sequences of length 1 and 2 are [1] and $[2, 1]$. Use your function to write a program that finds the longest and shortest hailstone sequences which have initial terms between 3 and 10000, together with the mean length of the sequences.

Hint:

- The `%` operator finds the remainder when its left operand is divided by the right operand. For example:

```
>>> 6 % 5
1
>>> 4 % 5
4
>>> 10 % 5
0
>>>
```

You should submit:

- The output from your program showing the hailstone sequences which start with $1, \ldots, 10$, the longest and the shortest hailstone sequences you have found, together with the mean length of the sequences (paper via BART).
- Hardcopy of your program `hailstone.py` (paper via BART).
- Your program in a file `hailstone.py` (electronic submission).

[30 marks]

5. **Words with vowels.** The aim of this exercise is to find words in a file of words that contain all the vowels, 'a', 'e', 'i', 'o' and 'u'. The file of words is the dictionary file `lowercasewords.txt` on the module resources website `http://empslocal.ex.ac.uk/studyres/ECM1408`. This file contains 235886 words, one word per line and all the letters are in lower case.

Write a function `has_all_vowels(word)` which takes the string `word` and returns the Boolean `True` if `word` contains all five vowels, but `False` if it does not.

Use your function write a program that constructs a list of all the words in `lowercasewords.txt` which contain all the vowels. Your program should find the number of these words and print out all the words that have a length equal to the shortest one.

Hints:

- You can make a list, here called `lines`, whose elements are the lines of the file `myfile`:

  ```
  lines = open('myfile', 'r').readlines()
  ```

  (Note that the file must be in the same directory as the directory in which you invoke the Python interpreter.)

  You can then strip the newlines from each line in this list to get just the words with:

  ```
  words = []
  for line in lines:
      words.append(line.strip())
  ```

- Remember that you can test whether one string is contained in another with the `in` operator.

You should submit:

- The output from your program showing the number of words containing all the vowels and the shortest words containing all the vowels (paper via BART).
- Hardcopies of your program `vowels.py` (paper via BART).
- Your program in a file `vowels.py` (electronic submission).

[30 marks]

[Total 100 marks]

Submitting your work

The CA requires both paper and electronic submissions.

Paper You should submit the answer to question 1 and paper copies of the code and any output for **all** the other questions to the Harrison Student Services Office by the deadline of **12:00 Thursday 25th October, 2012**. Markers will not be able to give feedback if you do not submit hardcopies of your code and marks will be deducted if you fail to do so.

Paper submissions should have the BART cover sheet securely attached to the front and should be anonymous (that is, the marker should not be able to tell you are from the submission). If this is the first time you have used BART, please make sure that you understand the procedure beforehand and leave plenty of time as there are often queues close to the deadline.

Where you are asked for paper copies of the output of your code, please cut and paste the output from the terminal rather than taking a screenshot, because the screenshot is often illegible after printing.

Electronic You should submit the files containing the code for each question via the electronic submission system at `http://empslocal.ex.ac.uk/submit/`. Make sure that your code is in files with the names specified in the questions. Then use `zip` or `rar` or `tar` to compress these into a single file, and upload this file using the submit system. You must do this by the deadline.

You will be sent an email by the submit system asking you to confirm your submission by following a link. Your submission is not confirmed until you do this. It is best to do it straightaway, but there is a few hours leeway after the deadline has passed. It is possible to unsubmit and resubmit electronic coursework — follow the instructions on the submission website.

Marking criteria

Work will be marked against the following criteria. Although it varies a bit from question to question they all have approximately equal weight.

- **Does your algorithm correctly solve the problem?**
  In most of these exercises the algorithm has been described in the question, but not always in complete detail and some decisions are left to you.
- **Does the code correctly implement the algorithm?**
  Have you written correct code?
- **Is the code syntactically correct?**
  Is your program a legal Python program regardless of whether it implements the algorithm?
- **Is the code beautiful or ugly?**
  Is the implementation clear and efficient or is it unclear and inefficient? Is the code well structured? Have you made good use of functions?
- **Is the code well laid out and commented?**
  Is there a comment describing what the code does? Are the comments describing the major portions of the code or particularly tricky bits? Do functions have a docstring? Although Python insists that you use indentation to show the structure of your code, have you used space to make the code clear to human readers?

There are 10% penalties for:

- Not submitting hardcopies of your programs.
- Not naming files as instructed in the questions.