CE151 ASSIGNMENT 1 2012

Set by: Mike Sanderson

Credit: 20% of total module mark

Deadline: 11.59.59, Wednesday 14 November

Submission of this assignment will be via the online submission system; your program will be tested during your lab in week 7 (on 15 November or 16 November).

It is expected that marks and feedback will be returned by the beginning of week 10.

You should refer to pages 24-25 and 62-64 of the Undergraduate Students' Handbook for details of the departmental policy regarding late submission and university regulations regarding plagiarism; the work handed in must be entirely your own.

Introduction

This assignment involves eight individual exercises, each of which should be written in a function within a single file. You must use the template file, `ass1.py`, supplied in the CE151 course materials area, which contains code to facilitate testing of the exercises. You should modify the code at the bottom of the file so that it prints ***your*** name instead of the fictitious name provided; apart from this all of your code must be written inside the functions `ex1` to `ex8`.

For each exercise that you attempted you should delete the words "not attempted" from the `print` statement at the beginning of the function. Your code to perform the exercise should be placed within the function body beneath the `print` statement. Each exercise involves the writing of a function that will input data from the keyboard, perform some processing and output results. The functions should prompt the user for the required data. When the user is expected to type an integer or a real number you may assume that he or she will do so; however, where positive or odd numbers are required the program should check the input and display an appropriate error message if it is invalid.

You are encouraged to define and use additional functions where appropriate. If you do so, each function definition should appear immediately above the (first) exercise that uses it – these functions should be documented using the style seen in labs 3 and 4.

You may use features of the Python language and library that have not yet been covered in CE151, but must not use any third-party modules (i.e. modules that are not provided as part of the Python download).

Exercise 1

In the function `ex1` write code to input two integers *m* and *n* and calculate and display the sum of all the integers from *m* to *n* e.g. if 4 and 8 were entered (in either order) the sum to be calculated would be 4+5+6+7+8.

The sum should be calculated in two ways: (i) using a loop to repeatedly add numbers to a total; and (ii) using the formula $(n+m)\times(n-m+1)/2$ (where *n* is the larger of the two numbers). Both answers (which should of course be the same) should be output as integers (along with messages indicating which is which).

Exercise 2

In the function `ex2` write code to input two positive values of type `float` representing the width and height of a right-angled triangle and calculate and display the length of the hypotenuse, and also the three interior angles, expressed in degrees.

Pythagoras's theorem states that $\textit{hypotenuse}^{2} = \textit{width}^{2} + \textit{height}^{2}$, so you should use this to calculate the square of the hypotenuse, and then obtain its square root by calling the function `sqrt` from the `math` module.

You can obtain one of the interior angles by calling the function `atan` from the `math` module with an argument of *height*/*width*; this will return a result in radians; to convert this to degrees the result must be passed as an argument to another function called `degrees` (also from the `math` module) which will return the required result. Since it is known that one of the angles is 90 degrees and the sum of the other two is 90 degrees we can now easily find all three angles.

All answers should be displayed with two digits after the decimal point.

Exercise 3

In the function `ex3` write code that will input a positive integer *n* and check whether it is a prime number. This should be done by checking the remainder obtained when dividing the number *n* by 2 and then by all odd numbers between 3 and *n*/2; the number will be prime if and only if all of the remainders are non-zero. The function should output a message of the form

```
The number is prime
```

or

```
The number is not prime
```

(The number 1 is ***not*** a prime number – make sure that your code gives the correct answer when 1 is input.)

Exercise 4

In the function `ex4` write code that will input two positive integers *r* and *c* and display a multiplication table with *r* rows and *c* columns, e.g. if the inputs were 5 and 6 the output should look like

```
   1   2   3   4   5   6
   2   4   6   8  10  12
   3   6   9  12  15  18
   4   8  12  16  20  24
   5  10  15  20  25  30
```

The table should be neatly aligned; you may assume that all the values to be displayed are less than 999. The prompts for input should indicate which is which (e.g. “How many rows do you want?”).

Exercise 5

In the function `ex5` write code that will input a line of text, split it into words, and calculate and print the length of the longest and shortest words. You should regard any sequence of consecutive non-space characters as a single word; there may be more than one space between adjacent words in the input line.

Exercise 6

In the function `ex6` write code that will input a line of text and calculate and print the most-frequently-occurring vowel in the line, along with the number of times it occurs, e.g. if the line was `Are you bart`, the most frequent vowel would be `a` (or `A`) with two occurrences. If there is more than one vowel with the same number of occurrences you may print any of them, e.g. if the input was `Bart was here`, the answer may be either `a` or `e` with two occurrences. If there are no vowels in the line you should display an appropriate message. (Note that, for example, `a` and `A` should be regarded as being instances of the same vowel.)

The Bubble Sort Algorithm

The bubble sort algorithm is one of many algorithms that can be used to sort a list into ascending order.
We can describe it informally as follows

```
Compare the first two items; if the first is greater than the second swap them
Compare the second and third items and swap them if necessary
Continue these pair-wise comparisons until the end of the list has been reached
At this stage the last item in the list is guaranteed to have the largest value, but the other items
  might not be in the correct place
Repeat the whole process but stop before reaching the last item
After this the last two items are guaranteed to be in the correct place
Continue to repeat the process, each time looking at one fewer item than the previous time
Eventually there will only be one pair of items to compare (the first and second items in the list); after
  this pair has been compared (and swapped if necessary) the sorting is complete
```

Exercise 7

In the function `ex7` write code that will input a series of non-negative integers one at a time, add each one to a list, then use the bubble sort algorithm to sort the list and finally print the sorted list. A negative number should be used to indicate that input is complete; this should not be stored in the list.

You must write your own code for bubble sort; you are not allowed to use the Python `sort` method in this exercise.

The function can be made more efficient; it the smallest items in the list are not near the end it is likely that the list will be sorted some time before the algorithm terminates. We can detect this by checking whether any swaps were made in a single iteration through the list. To gain the full marks for this exercise you must add code to check for this. (A program that works correctly but does not terminate the looping as soon as it can do so will earn about 12 of the 15 marks available for this exercise).

Exercise 8

In the function `ex8` write code that will display nested triangles as seen in the diagrams on the last page of this document. At each level the complete triangle for the previous level is placed into an extra outer triangle. The user should be asked to input the two characters to be used and the width of the innermost triangle, which must be odd. The function should test whether the supplied number is odd and display an appropriate message if it is not.

The full 25 marks for this exercise may be obtained for a fully-working function which can display level 4 triangles; a function which successfully displays level 3 triangles will earn 16 marks and one which successfully displays level 2 triangles will earn 8 marks. Attempts that do not work correctly will not earn more than 7 marks, so a working level 2 function is better than a level 3 or level 4 version that does give the correct output.

Submission

You should submit your `ass1.py` file to the online submission system before the deadline (if you do not know how to use the system, go to the CSEE student intranet and follow the Electronic Submission Guide link). In addition you should have your code ready to be tested in your lab in week 7, so if you develop it at home make sure you have a copy on your `M:` drive.

Marking Scheme

A total of 100 marks is available for the assignment.

Successful completion of each of exercises 1 to 6 will earn 10 marks; functions which work correctly with some inputs but fail with other inputs or meet some but not all of the requirements may earn between 5 and 9 marks.

15 marks are available for exercise 7 and 25 marks are available for exercise 8.

Two marks will be deducted from the mark for any exercise that fails to perform all of the necessary input validation and two marks will be deducted if the output is not in the required style (e.g. number of decimal places).

If you fail to demonstrate your program in the lab 20% will be deducted from your mark. If you are unable to attend the week 7 lab due to illness or other unforeseen circumstances you must inform me by email (sands@essex.ac.uk) by the end of week 7, so that alternative arrangements for your demonstration can be made.

Diagrams for Exercise 8

```
Level 1  Level 2          Level 3                          Level 4

*****       .       ***********.***********                        .
 ***       ...       *********...*********                        ...
  *       .....       *******.....*******                        .....
         .*****.       *****.*****.*****                        .......
        ...***...       ***...***...***                        .........
       .....*.....       *.....*.....*                        ...........
                          ***********                        .............
                           *********                        ...............
                            *******                        .................
                             *****                        ...................
                              ***                        .....................
                               *                        .......................
                                                       .***********.***********.
                                                      ...*********...*********...
                                                     .....*******.....*******.....
                                                    .......*****.*****.*****.......
                                                   .........***...***...***.........
                                                  ...........*.....*.....*...........
                                                 .............***********.............
                                                ...............*********...............
                                               .................*******.................
                                              ...................*****...................
                                             .....................***.....................
                                            .......................*.......................
```

[These triangles all have a width of 5 for the innermost triangle.]