CE151 ASSIGNMENT 1 2017

Set by: Mike Sanderson

Credit: 20% of total module mark

Deadline: 11.59.59, Wednesday 15 November

Submission of this assignment will be via FASer; your program must be demonstrated during your lab in week 8 or 9.

It is expected that marks and feedback will be returned by the beginning of week 11.

You should refer to sections 5 and 7 of the Undergraduate Students' Handbook for details of the University policy regarding late submission and plagiarism; the work handed in must be entirely your own.

Introduction

This assignment involves seven 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 must use Python 3. **Code written in Python 2 will receive no marks.**

You should modify the code at the bottom of the file so that it prints ***your*** name instead of the fictitious name provided.

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 numbers are required the program should check the input and display an appropriate error message if it is not greater than 0**. After outputting the message you may either return immediately from the function (so the user will see the exercise-selection dialogue again) or ask the user to try again; if you choose the latter option any subsequent inputs must also be validated. You must not abort the program.

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. Apart from these additional functions all of your code must be written indented inside the functions `ex1` to `ex7`.

CE151 ASSIGNMENT 1 2017

All the code you produce must be indented within the function bodies so that when run the program starts with the output of your name and the exercise-selection dialogue. The only un-indented lines that you may write are `def` lines for additional functions and `import` statements. Failure to meet this requirement will result in 20% of your mark being deducted.

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 positive values of type `float` representing the width and the length of the hypotenuse of a right-angled triangle and calculate and display the height of the triangle, and also the two interior angles, expressed in degrees. (We already know that one of the angles is 90 degrees.).

If the width specified by the user is not less than the length of the hypotenuse the triangle cannot exist, so you should output an appropriate message instead of trying to calculate the height.

Pythagoras's theorem (rearranged) states that $height^2 = hypotenuse^2 - widtht^2$, so you should use this to calculate the square of the length of the height, and then obtain its square root by calling the function `sqrt` which must be imported from the `math` module.

You must obtain one of the interior angles by calling the function `acos` from the `math` module with an argument of *width*/*hypotenuse*; 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. The second angle can be obtained simply by subtracting the first angle from 90 degrees. You will need to add code to import the `atan` and `degrees` functions

All answers should be displayed with exactly two digits after the decimal point. (Note that you will need to use `format`, not just `round`, to ensure that the two digits are always displayed.)

Exercise 2

The *Fibonacci series* always begins with the values 0,1,1,2,3,5,8,13,21,34. The first two numbers of the series are 0 and 1and each subsequent value is obtained by adding together the two preceding values (e.g. the value 21 was obtained by adding 8 and 13).

In the function `ex2` write code to input a positive integer *n* then calculate and display the first *n* numbers in the Fibonacci series. The numbers must be displayed on a single line separated by single spaces.

Exercise 3

In the function `ex3` write code that will input two positive integers *r* and *c* and display a table with *r* rows and *c* columns, where the value of the *m*th number in the *n*th row is n^m, e.g.

```
1    1    1    1    1
2    4    8   16   32
3    9   27   81  243
4   16   64  256 1024
5   25  125  625 3125
```

The table should have columns of equal width and all numbers should be right-aligned; the required column width should be obtained by calculating the width needed for the largest number in the table (i.e. the one in the bottom right corner) before you perform any output. (To determine how much space is needed for a number `n` you can use `len(str(n))`). The prompts for the inputs should specify which is which, e.g. “How many rows do you want?”.

Exercise 4

In the function `ex4` write code that will input a line of text, split it into words, and display these words one per line, and also 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, but the output must not contain any blank lines.

Exercise 5

In the function `ex5` 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 should print all of them, e.g. if the input was `Andy was here`, the answer should state that both `a` and `e` have two occurrences. If there are no vowels in the line you should display an appropriate message. (Note that, as in the examples, `a` and `A` should be regarded as being instances of the same vowel; also note that in the English language the letter `y` is not regarded as being a vowel.)

Exercise 6

In the function `ex6` write code that will input a line of text from the keyboard into a string and encrypt or decrypt the string using an integer key (also obtained from the keyboard) and display the result. The input string should contain only upper-case letters and/or spaces; if it contains anything else an error message should be displayed and no encryption/decryption should be performed.

You must not write separate code for encryption and decryption – to decrypt we simply supply a negative key (e.g. if we encrypted using key 5 we would decrypt using key -5).

In the ASCII system used to encode characters the upper-case letters are represented by the values 65 to 90 – when encrypting we need to ensure that the encrypted string also comprises only upper-case letters and spaces (so that we can supply it as input for decryption). Hence the encryption algorithm, which should be applied to each non-space character in turn, should be

obtain the ASCII code for the character
subtract 65 (giving a value in the range 0 to 25)
add the key
take the remainder mod 26 (so we've again got a value in the range 0 to 25)
add 65 (to give a value in the range 65 to 90)
convert the ASCII code back to a character

Spaces should not be encrypted – when there is a space in the input string there should be a corresponding space in the output string. The output must be displayed as a single string, not a list of characters

[To obtain the ASCII code for a single-character string `s`, you need to use `ord(s)`; to convert an ASCII code `n` to a character you need to use `chr(n)` – this returns a single-character string.]

Exercise 7

In the function `ex7` write code that will display nested triangles as seen in the diagrams below. 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. In addition to the test for negative input the function should test whether the supplied number is odd and display an appropriate message if it is not.

The full 20 marks for this exercise may be obtained for a fully-working function which displays level 3 triangles; a function which successfully displays level 2 triangles will earn 12 marks.

Level 1

```
*****
 ***
  *
```

Level 2

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

Level 3

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

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

Submission

You should submit your `ass1.py` file to the online FASER submission system before the deadline. In addition you should have your code ready to be tested in your lab in week 8, so if you develop it at home make sure you have a copy on your `M:` drive or on a USB stick.

Marking Scheme

A total of 100 marks is available for the assignment.

Successful completion of each of exercises 1 to 5 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.

20 marks are available for each of exercises 6 and 7.

Up to two marks may be deducted from the mark for any exercise if it fails to perform all of the necessary input validation and up to two marks may be deducted if the output is not in the required style (e.g. spacing, number of decimal places).

The remaining 10 marks are awarded for completion of the two lab exercises set in the week 3 and week 5 labs (5 marks for each).

The only criteria that will be used for assessment are the correct behaviour of the program, the correct use of functions and the use of any specific methods mentioned within the description of the exercises; no marks for this assignment are given for programming style.

If you fail to demonstrate your program in the lab 20% will be deducted from your mark. If you are unable to attend both the week 8 and week 9 labs due to illness or other unforeseen circumstances you must inform me by email (sands@essex.ac.uk) by Wednesday of week 9, so that alternative arrangements for your demonstration can be made.

[Any 20% deduction of marks will not be applied to the marks awarded for the lab exercises.]