CE151 ASSIGNMENT 1 2014

Set by: Mike Sanderson

Credit: 20% of total module mark

Deadline: 11.59.59, Thursday 13 November

Submission of this assignment will be via the online submission system; your program must be demonstrated during your lab in week 8 (18 November or 20 November).

It is expected that marks and feedback will be returned on or before Wednesday of week 10.

You should refer to pages 24 and 40 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 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 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**.

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 `ex8`.

All the code you produce must be indented within the function bodies so that when run the program starts with 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).

CE151 ASSIGNMENT 1 2014

Exercise 1

In the function `ex1` 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 $hypotenuse^2 = width^2 + height^2$, so you should use this to calculate the square of the hypotenuse length, 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 exactly one digit after the decimal point.

Exercise 2

The *Fibonacci series* begins with the values 1,1,2,3,5,8,13,21,34. The first two numbers of the series are both 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 commas, with no spaces.

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 the *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
```

(Make sure that your function gives the correct answer when small numbers are input: 2 and 3 are prime, but 1 is not.)

Exercise 4

The binomial coefficient of two positive integers *x* and *y*, when *y*<*x* can be calculated using the formula *x* * (*x*-1) * … * (*x*-*y*+1) / (*y* * (*y*-1) * … * 1). (The value of this expression will always be an integer.) When *y* is equal to 1 the value of this expression simplifies to *x*, and when *y* is equal to *x* the value simplifies to 1, so no calculation needs to be performed in these two cases. When *y*>*x* the value is 0.

In the function `ex4` write code that will input two positive integers *x* and *y* and calculate and display (as an integer) their binomial coefficient. In the cases where the answer is neither 1 nor 0 you should first calculate the two products (using loops) and then use the integer division operator (`//`) to perform the division.

Exercise 5

In the function `ex5` 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 6

In the function `ex6` write code that will input a line of text and calculate and print the least-frequently-occurring vowel in the line, along with the number of times it occurs, e.g. if the line was `Are you about`, the least frequent vowel would be `e` with one occurrence. (You should ignore vowels that do not occur, so the output for this example above should ***not*** be `i` with no 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.)

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 can earn 16 marks)

Leap Years

A year is a leap year (in which February has 29 days) if it is a multiple of 4, with the exception that a year which is a multiple of 100 is a leap year only if it is a multiple of 400 (so for example 2000 was a leap year but 2100 will not be).

Exercise 8

In the function `ex8` write code that will input a date in the format *dd*/*mm*/*yyyy*, check that it is in the future, and calculate and display the number of days between today and the date that has been input (including the input date, but not including today). For example if tomorrow's date is supplied the result should be 1.

You may assume that the user will supply input in the correct format but must check that the numbers represent a valid date, i.e. the month should be between 1 and 12 and the day should be between 1 and the number of days in the month.

If the date is not valid or is not in the future an appropriate message should be output.

[Refer to **`hello1.py`** from the lab 2 solutions for details of how to input a date and get today's date. You should use a list similar to that seen on slide 4 from part 4 of the lecture slides to store the number of days in each month. If the date that has been input is in the current month the calculation is simple; otherwise it is suggested that you work out how many days are left in the current month, add to this the numbers of days in all complete months between the current month and the month of the input date, and finally add the days between the first day of that month and the input date. If one or more of the months is February you will need to check whether it has 28 or 29 days – the best way to do this is to write a function which takes a year as an argument and returns a boolean value indicating whether the year is a leap year.]

Submission

You should submit your `ass1.py` file to the 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.

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.

20 marks are available for each of exercises 7 and 8.

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. 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 8 lab due to illness or other unforeseen circumstances you must inform me by email (sands@essex.ac.uk) no later than Monday of week 9, so that alternative arrangements for your demonstration can be made.