CompSci 101 - Assignment 01

Due: 4:30pm, 21st August 2020.

Worth: This assignment is marked out of 30 and is worth 3% of your final mark.

Topics covered:

- Arithmetic operators, printing output, manipulating string objects, string slicing, generating random numbers, getting user input.

The work done on this assignment **MUST** be your own work. Think carefully about any problems you come across, and try to solve them yourself before you ask anyone else for help. Under no circumstances should you use code written by another person in your assignment solution. Under no circumstances should you share the code written by you in in this assignment with another person.

VERY IMPORTANT:

This assignment has two sections.

Section A is marked using Coderunner3 (20 marks). For this section you need to develop four programs. Each program is described in Section A of this document. Each of the Section A programs are to be done in IDLE and then submitted using CodeRunner3.

Section B

Section B of this assignment (10 marks). For this section you need to develop one program. The program is described in Section B of this document.

Submission

Section A - Questions 1, 2, 3 and 4 are submitted using CodeRunner3.

Section B – Question 5. Submit your completed Assignment 1 Question 5 Python program using the Assignment Dropbox:

https://adb.auckland.ac.nz/Home/

NOTES:

- This assignment is marked out of 30 and is worth 3% of your final mark. Five marks out of 30 are assigned for the style of your program (program docstring, variable names, etc.).
- Only use the features taught in CompSci 101. When solving these questions you must only use content covered in Lectures 1 to 6.

Assignment 1 Section A

Develop the Questions 1, 2, 3 and 4 programs on your computer using IDLE. Once you are happy that your program code executes correctly, submit the program code to CodeRunner3:

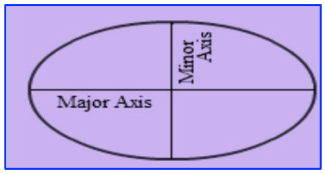
https://coderunner3.auckland.ac.nz/moodle/

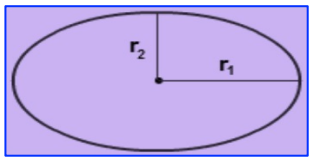
When you press the `Check` button in CodeRunner3, you will receive immediate feedback telling you if you have passed all the tests for the program. You need to validate one program at a time.

When you have successfully completed all four questions and your code passes all the tests click the "`Finish Attempt`" button followed by the "`Submit all and finish`" button.

Question 1. The Cost of Fencing and laying grass in an Elliptical area

An ellipse has two axes – a major axis and a minor axis. The longest chord of the ellipse is the major axis (2 * `major_radius`). The chord, perpendicular to the major axis, which bisects the major axis at the centre is the minor axis (2 * `minor_radius`).





The following are the two formulae for calculating the area and the perimeter (distance around the edge of the ellipse) of an ellipse:

$$Area\ of\ the\ Ellipse = \pi r_1 r_2 \qquad Perimeter\ of\ the\ Ellipse = 2\pi\sqrt{\frac{r_1^2 + r_2^2}{2}}$$

Write a program which calculates the total cost of fencing and laying grass over a field which is in the shape of an ellipse. The four variables below give the information needed to calculate the total cost. The two prices are in dollars and the major and minor axes are in metres. The area and the perimeter should both be rounded to the nearest whole number before the cost of fencing and laying the grass is calculated. Copy the following statements into the start of your program (you will change the values assigned to these four variables when you test your program):

```
fencing_per_metre = 75
grass_per_square_metre = 20
major_radius = 10
minor_radius = 5
```

Below are two example outputs from the completed program (using different values for the initial four variables). Your program must give the correct output in the same format as the outputs in the two examples below:

```
**************************************************
Cost of laying grass (157 square metres): $3140
Cost of fencing (50 metres): $3750
Total cost: $6890
**************************************************
```

```
**************************************************
Cost of laying grass (75 square metres): $1875
Cost of fencing (38 metres): $3420
Total cost: $5295
**************************************************
```

The first and last lines of the output are 50 stars.

The second output uses the following initial values:

```
fencing_per_metre = 90
grass_per_square_metre = 25
major_radius = 8
minor_radius = 3
```

Once you are happy that your program code executes correctly, copy all the program code into CodeRunner3 **except for the initial four statements which initialise the four variables** and press the 'Check' button.

<----------------------->

Question 2. Display the digital root of a four digit number

Write a program which, given a four digit whole number, calculates the digital root of the number by first calculating the sum of the four digits making up the number and then totalling the digits in the result of that calculation (a single digit whole number - 1 to 9 inclusive). For example, given the four digit number 3498:

- The beginning number is 3498
- The sum its digits is 3 + 4 + 9 + 8 = 24
- The number is now 24
- The sum of its digits is 2 + 4 = 6
- The resulting digital root of 3498 is 6.

The program prints a single line of output: the string `'The number: '` followed by the given four digit number, followed by the string `' The digital root: '` and finally, the single digit total. The first line of your program initialises the `number` variable in the following way (you will need to change the value assigned to this variable when you test your program):

```
number = 3498
```

Note: you will only need to add the digits of the number twice because you are always starting with a two digit number and the maximum sum of four single digits is 36 (for the initial four digit number 9999).

Hint: when doing the second sum of the digits you need to make sure that the number always has two digits: convert the number into a string and concatenate a `"0"` to the string.

Your program must give the correct output in the same format as the outputs in the four examples below (the initial variable assignment is shown on the right hand side and is not part of the output):

`The number: 3498 The digital root: 6`	`number = 3498`
`The number: 9999 The digital root: 9`	`number = 9999`
`The number: 1000 The digital root: 1`	`number = 1000`
`The number: 1234 The digital root: 1`	`number = 1234`

Once you are happy that your program code executes correctly, submit all the program code other than the initial statement which initialises the `number` variable to CodeRunner3.

<----------->

Question 3. Display a date in two different formats

Write a program which, given a day number, a month number and a year, displays the date in two different formats: Standard format and New Zealand format. The first three lines of your program initialise the variables in the following way (you will need to change the values assigned to these three variables when you test your program):

```
day = 6
month = 7
year = 1976
```

The Standard format is the format `"yyyy-mm-dd"`.

The New Zealand format is the format `"dd/mm/yyyy"`.

Note: any single digit day or month number should **ALWAYS** has a `"0"` in front of it.

Hint: to make sure a number always has exactly two digits: convert the number into a string, concatenate a `"0"` in front of the number (string) and slice the last two characters of the string.

Your program must give the correct output in the same format as the outputs in the four examples below (the initial variable assignment is shown on the right hand side and is not part of the output):

```
Standard format: 1976-07-06
New Zealand format: 06/07/1976
```

```
day = 6
month = 7
year = 1976
```

```
Standard format: 1999-05-23
New Zealand format: 23/05/1999
```

```
day = 23
month = 5
year = 1999
```

```
Standard format: 2019-10-01
New Zealand format: 01/10/2019
```

```
day = 1
month = 10
year = 2019
```

```
Standard format: 2020-12-16
New Zealand format: 16/12/2020
```

```
day = 16
month = 12
year = 2020
```

Once you are happy that your program code executes correctly, submit all the program code other than the initial lines which initialise the `day`, `month` and `year` to CodeRunner3.

<----------------------------->

Question 4. Six Letter Word Generator

Write a program which generates a six letter random word. All the letters of the word at even indexes (0, 2, 4) should be random consonants (no repetitions allowed) and all the letters at odd indexes (1, 3, 5) should be the vowels at the same position in the `vowels` string as the consonant before it, e.g. tubaha, cevaki, rijezi, soturi, tupeso. This means that you will only be generating three random indexes. Below are the consonants and vowels which are allowed for the random word. Note that both strings have the same length. Paste these two lines of code into your program.

```
consonants = "bcdfghjklmnprstvwz"
vowels =     "aeiouaeiouaeiouaei"
```

The six letter word which is generated is displayed within two rows of 16 stars and the random word is centred. As well, there is a blank line before the first line of stars and a blank line after the last line of stars. Below are three different example outputs obtained by executing the completed program three times.

```
****************
****************
     zisoba
****************
****************
```

```
****************
****************
     lokiri
****************
****************
```

```
****************
****************
     guzilo
****************
****************
```

Once you are happy that your program code executes correctly, copy all the program code into CodeRunner3 except for the initial two statements which initialise the variables and press the 'Check' button.

<---------------------------->

Assignment 1 Section B (10 marks)

For Question 5, submit your completed Python program using the Assignment Dropbox:

https://adb.auckland.ac.nz/Home/

IMPORTANT: Your program MUST include a docstring at the top of the file (containing your name, your username and a correct description of the program) and your program MUST be named correctly (i.e. as stated in the question).

Question 5. A dice game

Write a program which implements a dice game. The aim of the game is to reach a score as close as possible to 20 in three rounds. Name the program `'YourUsernameA1Q5.py'`, e.g., `afer023A1Q5.py`.

Each round consists of rolling five random dice, then, when prompted, the player enters:

- either `"+"` or `"–"` to indicate if the player wishes to add or subtract the number to their current total,
- the positions of two of the five dice values where the two dice values chosen form a two digit score e.g. if the user first chooses the position of a dice with the value 3 and then chooses the position of a dice with the value 5, then 35 is added to (or subtracted from) the user's current total (the first dice chosen is the tens digit and the second dice chosen is the units digit).

To choose whether to add, to subtract and the number itself the user enters the following information on **a single line**:

> either `"+ "` or `"– "` followed by the position number (1, 2, 3, 4 or 5) of the tens digit, followed by a space and finally the position number (1, 2, 3, 4 or 5) of the units digit.

The prompt requesting the user input is made up of 2 spaces followed by `"Add/Subtract tens units: "`.

The random dice are displayed with one space between each dice, e.g.

```
Your dice: 3 5 3 4 1
```

This process is repeated three times. Copy and paste the following statement which initialises the user's current total, into your program:

```
current_total = 0
```

On the next page are two example outputs using the completed program (the user input is shown in blue). Your program must give the output in the same format as the outputs in the two examples below.

Note: the top string of "*" symbols has a length of 42 and the bottom string of "*" symbols has a length of 16.

```
******************************************
REACH 20 IN THREE ROUNDS! CURRENT TOTAL: 0
******************************************

Round 1
The dice: 6 6 1 4 6
  Add/Subtract tens units: + 3 1
The number: 16

Current total: 16


Round 2
The dice: 4 6 6 3 5
  Add/Subtract tens units: + 4 2
The number: 36

Current total: 52


Round 3
The dice: 6 5 2 2 3
  Add/Subtract tens units: - 5 3
The number: -32


****************
Final total: 20
Out by: 0
****************
```

```
******************************************
REACH 20 IN THREE ROUNDS! CURRENT TOTAL: 0
******************************************

Round 1
The dice: 4 5 2 4 6
  Add/Subtract tens units: + 3 5
The number: 26

Current total: 26


Round 2
The dice: 6 4 1 3 3
  Add/Subtract tens units: - 3 1
The number: -16

Current total: 10


Round 3
The dice: 6 5 4 3 5
  Add/Subtract tens units: + 4 3
The number: 34


****************
Final total: 44
Out by: 24
****************
```

<----------------------->