CompSci 101 - Assignment 01

Due: 4:30pm, 15th January 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.

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. Once you have all 4 programs correct in CodeRunner, click the `"Finish Attempt"` button followed by the `"Submit all and finish"` button.

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. Four 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

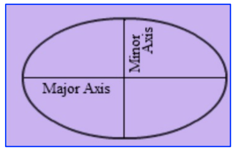
Develop the Questions 1, 2, 3 and 4 programs on your computer using IDLE. Once you are happy that your program code executes correctly, submit the program code to CodeRunner:

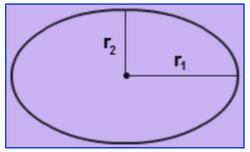
https://coderunner3.auckland.ac.nz/moodle/

When you press the `Check` button in CodeRunner, 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. The chord perpendicular to the major axis is the minor axis which bisects the major axis at the centre.





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

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

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 are both rounded to the nearest whole number and then 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
```

Write your program in a file named `'YourUsernameA1Q1.py'`, e.g., `afer023A1Q1.py`. 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 CodeRunner **except for the initial four statements which initialise the four variables** and press the 'Check' button.

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

Question 2. Time Difference using a 24 hour clock

Write a program which calculates the difference in hours and minutes between two times. The two times are given using a 24 hour clock, e.g., 21:34 is 34 minutes past 9pm. Copy the following statements which initialise the two times into the start of your program (you will change the values assigned to these four variables when you test your program).

```
time1_hour = 5
time1_minute = 23
time2_hour = 23
time2_minute = 14
```

You can assume that the second time is always later than the first time. Write your program in a file named `'YourUsernameA1Q2.py'`, e.g., `afer023A1Q2.py`. Below are three 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 three examples below:

```
Hours and minutes from 5:23 to 23:14 is:
   17 hours and 51 minutes.
```

```
Hours and minutes from 11:25 to 15:6 is:
   3 hours and 41 minutes.
```

```
Hours and minutes from 13:32 to 13:36 is:
   0 hours and 4 minutes.
```

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

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

Question 3. Print a parallelogram of the letters of a five character word

Write a program which prints a 5 letter word (I have used 5 letter names in my examples) in the form of a parallelogram as in the examples below. Nine lines of the output print sections of the word, the first line prints the first letter of the word, the next line prints the next two letters of the word, etc. , the fifth line prints the whole word and the next three lines contract the word until the last line which prints just the last letter of the word. Each of these 9 lines is indented by four spaces and the whole nine lines are enclosed inside four rows of stars. Each row of stars is indented by two spaces. Copy the following statement into the start of your program.

```
word = "MARIA"
```

You can assume that the word always has five letters. Write your program in a file named `'YourUsernameA1Q3.py'`, e.g., `afer023A1Q3.py`. Below are two example outputs from the completed program (using a different value for the initial variable). Your program must give the correct output in the same format as the outputs in the two examples below:

```
  *********
  *********
        M
       AR
      ARI
     ARIA
    MARIA
    ARIA
    RIA
    IA
    A
  *********
  *********
```

```
  *********
  *********
        J
       AM
      AME
     AMES
    JAMES
    AMES
    MES
    ES
    S
  *********
  *********
```

Once you are happy that your program code executes correctly, copy all the program code into CodeRunner **except for the initial statement which initialises the variable** and press the 'Check' button.

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

Question 4. Encryption/decryption program

Write a program which encrypts (or decrypts) a message. The message is **always** 25 letters in length. Lines 2 and 3 of the code below ensure that the message is always 25 letters in length.

```
message = "Pagg rm rfomieugisanrn t!"

message = message * 25
message = message[0: 25] #Make sure the message length is exactly 25
```

Copy the above statements which initialise the `message` variable into your program. The program uses the following encryption method. Firstly the 25 letter message is broken up into five 5 letter words, i.e., a five by five square of letters. Then the message is recreated by going down the five by five square of letters from left to right. For example, the message `"abcdefghijklmnopqrstuvwxy"` is reshaped into five rows:

```
abcde
fghij
klmno
pqrst
uvwxy
```

and then the message is reconstructed going down each column starting from the first column to the last column: `"afkpubglqvchmrwdinsxejoty"`

Write your program in a file named `'YourUsernameA1Q4.py'`, e.g., `afer023A1Q4.py`. Below are three example outputs from the completed program (using different values for the `message` variable). Your program must give the correct output in the same format as the outputs in the three examples below:

```
Original message: E  l ntbidjhraaoiinyyslt!
Encrypted message: Enjoy this brilliant day!
```

```
Original message: Meet at 11 in the library
Encrypted message: Ma hbetiere n at1 lr 1tiy
```

```
Original message: Ma hbetiere n at1 lr 1tiy
Encrypted message: Meet at 11 in the library
```

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

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

Assignment 1 Section B (10 marks)

For Question 5, submit your completed Python programs 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. Name the program `'YourUsernameA1Q5.py'`, e.g., `afer023A1Q5.py`. The aim of the game is to reach a score as close as possible to 100 (but not over 100) in three rounds. Each round consists of throwing five random dice, the user then chooses two of the dice values where the two dice values chosen form a two digit score which is added to the user's current total, e.g., if the user first chooses a dice with the value 3 and then a dice with the value 5, 35 is added to the user's total (the first dice chosen is the tens digit and the second dice chosen is the units digit). The random dice are displayed with one space between each dice, e.g.,

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

and to choose the dice, the user enters a number 1, 2, 3, 4 or 5 indicating which of the five dice they wish to choose, i.e., **the position of the dice** (NOT the value of the dice). This process is repeated three times. Below is the statement which initialises the user's current score:

```
current_total = 0
```

Copy this statement into your program. Below are two example outputs using the completed program (the user input is shown in a pink font). Your program must give the output in the same format as the outputs in the two examples below. Note that the top string of `"*"` symbols has a length of 35 and the bottom string of `"*"` symbols has a length of 16.

```
***********************************
REACH 100 IN THREE ROUNDS!
***********************************

Round 1
Your dice: 2 2 1 5 2
  Tens? 5
  Units? 1
Dice value: 22

Your current total: 22


Round 2
Your dice: 2 2 4 3 2
  Tens? 4
  Units? 3
Dice value: 34

Your current total: 56


Round 3
Your dice: 4 4 3 4 1
  Tens? 2
  Units? 1
Dice value: 44

****************
Final score: 100
****************
```

```
***********************************
REACH 100 IN THREE ROUNDS!
***********************************

Round 1
Your dice: 3 3 1 5 1
  Tens? 2
  Units? 4
Dice value: 35

Your current total: 35


Round 2
Your dice: 1 4 4 3 2
  Tens? 3
  Units? 4
Dice value: 43

Your current total: 78


Round 3
Your dice: 3 4 6 6 3
  Tens? 1
  Units? 5
Dice value: 33

****************
Final score: 111
****************
```

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