ENGG1811 Mid-term Examination
Term 1, 2020

7 April 2020

Instructions

(a) This exam consists of 4 questions. Questions are of equal value. Answer all questions.

(b) Time allowed: 50 minutes.

(c) Each question requires you to submit a separate Python program file for marking.

  (i) You will be provided a submission link for each question.

  (ii) Each question requires a specific filename and the submission system will only accept that particular filename.

  (iii) Ensure that you **save** your file before submission.

  (iv) If the submission system accepts your file, it will also display the contents of your submitted file. This allows you to check.

  (v) You can submit at any time during the exam. There is a grace period at the end of the exam to allow you to make your final submissions.

  (vi) Note that the grace period is only to be used for submission and not for corrections. We will monitor that.

  (vii) We will only mark the last submission that you made.

(d) For each question, an associated test file or testing code has been provided to help you to test your code. The testing code is **rudimentary** and is not thorough. The file or code is provided so that you can write your own tests. It is your responsibility to test your code thoroughly before submission.

(e) You are allowed to consult lecture notes, books and materials on the web.

(f) The exam will be un-supervised. You are expected to do the exam yourselves. No communications are allowed.

Question 1

You have a sensor reading which is stored in a Python variable `x` of float type. Your task is to write a Python code segment which determines a Python variable `output_string`, which is a variable of string type, based on the value of `x`. The table below shows the value of `output_string` for different values of `x`.

Value of `x`	Value of `output_string`
$x > 300$	`good-performance`
$-200 < x \leq 300$	`need-monitoring`
$x \leq -200$	`alert`

Requirements and testing:

- You must write the code segment in the template file `m1.py` provided. The submission system will only accept this filename.
- You must write your code segment to determine `output_string` between the two `#-*-*-*` lines in the file `m1.py`
- You must use the variable names `x` and `output_string`
- The string variable `output_string` can only take one of these three possibilities: `alert`, `need-monitoring` and `good-performance`
- The file `m1.py` contains a rudimentary test. You need to test your code for different values of `x` before submission.
- Make sure that you save your file before submission.

Question 2

Let x and y be two real numbers. Consider the mathematical functions $g(x, y)$ and $h(x, y)$:

$$\begin{aligned} g(x,y) &= \cos(x-y) \\ h(x,y) &= \frac{x^4+1}{y(y+2)} \end{aligned}$$

Your task is to write a Python function `m2_func`, which has two inputs and two outputs, and has the following `def` line:

```
def m2_func(x,y):
```

where you can assume the inputs `x` and `y` are variables of float type. You can associate the Python variables `x` and `y` with, respectively, the real numbers x and y. If you call the Python function with the following statement:

```
fg, fh = m2_func(x,y)
```

then the variables `fg` and `fh` should take on, respectively, the values of $g(x, y)$ and $h(x, y)$.

Requirements and testing:

- You must write the function `m2_func` in a file with the filename `m2.py`. The submission system will only accept this filename. A template file `m2.py` has been provided.
- The file `test_m2.py` contains a rudimentary test. You need to test your function for different values of `x` and `y` before submission. Note that you can assume that `y` will neither be 0 nor -2.
- You do not need to submit `test_m2.py`.
- Make sure that you save your file before submission.

Question 3

Your task is to write a Python code segment that works on a Python list of numbers. Assume that you are given a Python list with float type and we will refer to this list as `list_a`. Your program will use `list_a` to produce another Python list, which we will refer to as `list_b`. These two lists are expected to have the same length, and the entries of these two lists are related by:

$$\begin{aligned} \texttt{list\_b}[0] &= \frac{\texttt{list\_a}[0]-1}{\texttt{list\_a}[0]+1} \\ \texttt{list\_b}[1] &= \frac{\texttt{list\_a}[1]-1}{\texttt{list\_a}[1]+1} \\ \texttt{list\_b}[2] &= \frac{\texttt{list\_a}[2]-1}{\texttt{list\_a}[2]+1} \\ &.... \end{aligned}$$

For example, if `list_a[0]` equals to 1.5, then `list_b[0]` is $\frac{1.5-1}{1.5+1} = \frac{0.5}{2.5} = 0.2$.

Note that your Python code must be able to work with any Python list with the name `list_a`.

You can assume the entries of `list_a` are all of the float type. You can also assume that none of the entries in `list_a` has the value of -1 so there is no division by zero.

Requirements and testing:

- You must write the code segment in the template file `m3.py` provided. The submission system will only accept this filename.
- You must write your code segment to calculate `list_b` from `list_a` between the two `#-*-*-*` lines in the file `m3.py`
- You must use the variable names `list_a` and `list_b`.
- The file `m3.py` prints the value returned by your function for the sample `list_a`, see the comments in the file. You need to manually check whether the returned value is correct or not. You need to test your code for different values of `list_a` before submission.
- Make sure that you save your file before submission.

Question 4

You are asked to write a Python code segment that performs calculations based a given list and a given scalar, which we will refer to as, respectively, `list_x` and `n`. You can assume that `n` is an integer greater than or equal to 1, and `list_x` is a list of numbers with at least `n` elements. Your code segment is required to do the following:

(a) Perform these calculations:

(i) Determine the maximum of the last `n` elements in the list `list_x`

(ii) Determine the maximum of the last $2*$ `n` elements in the list `list_x`

(iii) ...

(iv) Determine the maximum of the last $q*$`n` elements in the list `list_x` where q is the largest integer such that $q*$`n` is smaller than or equal to the number of entries in the given `list_x`.

(b) Create a list which contains the maximums calculated above. The order in which these maximums appear in this list should be in the same order that they are calculated. You MUST assign this list to a variable with the name `list_max`.

As an example. Let us assume `n = 3` and
`x_list = [19, -2, -15, 8, -9, 14, 7, -7, 12, 9, -6, 5, 3]`

The following table illustrates the maximums that will be obtained:

	Elements of the list that are used to compute the maximum	Maximum of the elements in the second column
Last 3 (= `n`) elements	`-6,5,3`	5
Last 6 (= 2*`n`) elements	`-7,12,9,-6,5,3`	12
Last 9 (= 3*`n`) elements	`-9,14,7,-7,12,9,-6,5,3`	14
Last 12 (= 4*`n`) elements	`-2,-15,8,-9,14,7,-7,12,9,-6,5,3`	14

The variable `list_max` should be the list `[5, 12, 14, 14]`. Note that q is 4 in this example because $4 * 3$ is less than or equal to the number of entries in the given `list_x`, but $5 * 3$ is not.

Requirements and testing:

- You must write the code segment in the template file `m4.py` provided. The submission system will only accept this filename.
- You must write your code segment to calculate `list_max` from `list_x` and `n` between the two `#-*-*-*` lines in the file `m4.py`
- You must use the variable names `list_x`, `n` and `list_max`.
- The file `m4.py` contains three test cases which you can select by using the variable `test_num` in Line 14 of the file. The case where `test_num` is 0 corresponds to the example above.
- The file `m4.py` prints your computed `list_max` and the expected output so that you can manually check whether they are correct or not.
- *Hint:* A method is to use the range() function with 3 inputs but there are many other methods.
- Make sure that you save your file before submission.

———— End of exam ————