School of Information Systems
Academic Year 2019/20
Term I

IS111: Introduction to Programming

Lab Exercise 04: Functions

Optional submission deadline: **Sun 15 Sep 2019 @ 2359 hrs**

INSTRUCTIONS

You are expected to read through the accompanying lesson 04 notes before attempting lab exercise 04. You should attempt the questions using **Visual Studio Code**. Each question should be submitted on a different file.

Do not submit Level 1 questions; use the discussion forum for that.

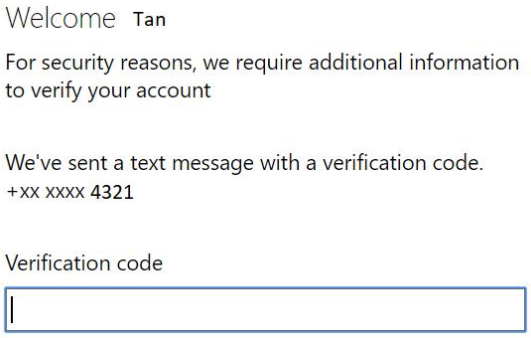
To submit, you should:

1. Name your 3 files as **week4-qn2-1.py**, **week4-qn2-2.py** and **week4-qn3-1.py**.
2. Submit the 3 .py files to **eLearn → Assignments** by the stipulated deadline.

Level 1 Questions (no submission required; use the discussion forum)

1.1 Mobile Number Masking

Mobile numbers may sometimes be masked, to preserve privacy and minimize the possibility of spam. For instance, the first four digits of a mobile number 87654321 may be masked into xxxx4321.



Write a Python function `mask_mobile` that takes in an integer parameter `mobile_number` containing an 8-digit mobile number, and then returns the masked mobile number.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(mask_mobile(87654321) == '****4321')
print(mask_mobile(91234567) == '****4567')
```

1.2 Email Masking

An email address is typically of the format local-part@domain, where the local part may be up to 64 characters long and the domain may have a maximum of 255 characters. Email masking hides all or part of the local part of the email address. An example of a masked email address abcdef@gmail.com is given by ******@gmail.com.

Write a Python function `mask_email` that takes in a string parameter `email_address` containing a *valid* email address, and then returns the masked email address.

You can use the following code snippet to test your code:

```
print(mask_email('abcdef@gmail.com') == '******@gmail.com')
print(mask_email('is111@gmail.com') == '*****@gmail.com')
```

1.3 Quadratic Equation

A quadratic equation is given by the form $ax^2 + bx + c = 0$, where $a \neq 0$. It can be solved using the formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. The quadratic equation may have:

- 2 solutions - which happens when $b^2 - 4ac > 0$;
- 1 solution - which happens when $b^2 - 4ac = 0$; or
- 0 (or no) solution - which happens when $b^2 - 4ac < 0$.

Write a Python function `get_num_solutions` that takes in 3 integer parameters *a*, *b* and *c*, and returns the number of solutions for the quadratic equation $ax^2 + bx + c = 0$.

You can use the following code snippet to test your code:

```
print(get_num_solutions(4,5,6) == 0)
print(get_num_solutions(3,6,3) == 1)
print(get_num_solutions(2,4,1) == 2)
```

Level 2 Questions

2.1 IPPT Passing Criteria

In Singapore, IPPT is a mandatory annual (within a 12-month period, i.e. your IPPT Window) requirement for all eligible NSmen with PES A, B/B1, and B2/C1 status. According to the passing criteria, the IPPT comprises 3 stations:

- push ups;
- sit ups; and
- 2.4 km run.

The minimum and maximum scores for each of the three stations are as follows:

Station	Minimum Score	Maximum Score
push ups	0	25
sit ups	0	25
2.4 km run	0	50

In order to pass, NSmen will need to:

- score a minimum of 1 point in each station; and
- achieve a total score of at least 51 out of 100 points.

Write a Python function `pass_ippt` that takes in 3 integer parameters `pushup_score`, `situp_score` and `run_score`. The function `pass_ippt` should return `True` if the NSman has passed his IPPT (based on the two criteria mentioned above), and `False` if the NSman has failed his IPPT.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(pass_ippt(10,20,20) == False)
print(pass_ippt(20,0,40) == False)
print(pass_ippt(20,15,33) == True)
```

2.2 Christmas Tree Lights

You are given the task of decorating the Christmas tree with lights, using a budget `$x`. Each light costs `$y`. The lights have to be decorated on the tree, in the following way:

- the first row contains only 1 light,
- the second row contains 2 lights,
- the third row contains 3 lights,
- and so on.

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

Write a Python function `calculate_rows_of_lights` that takes in two parameters:

- `budget`, which is the total amount of money that you are allowed to spend; and
- `cost_per_light`, which is the cost of each light.

Your function should return the maximum number of rows of lights that can be decorated using the `budget` that you are given, and taking into consideration the `cost_per_light`.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(calculate_rows_of_lights(50,2) == 6)
print(calculate_rows_of_lights(100,3) == 7)
print(calculate_rows_of_lights(10,12) == 0)
```

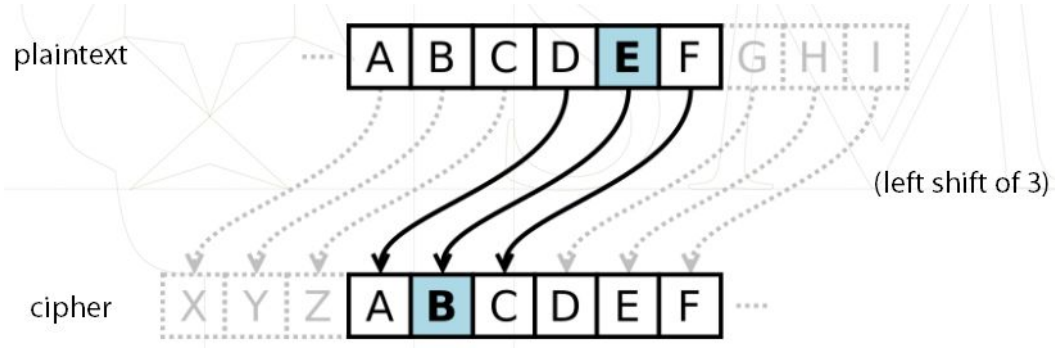
💬 **You do not have to print the Christmas tree.**

💬 **This question was part of the lab test last year.**

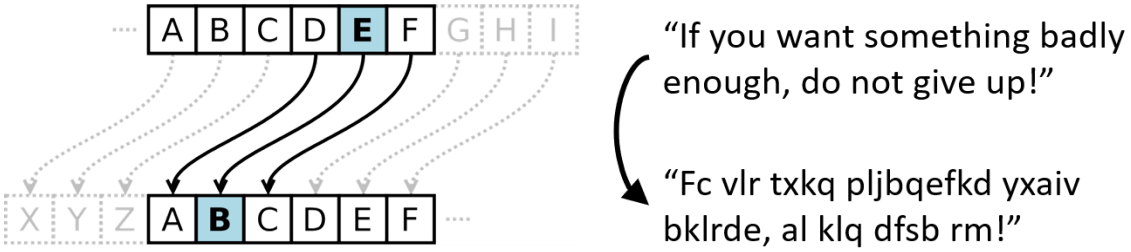
3 Brownie Points 🍪

3.1 Caesar Cipher

The Caesar Cipher is one of the simplest and most well-known encryption technique. It is named after Julius Caesar, who used it in his private correspondences. In Caesar Cipher, each letter in the plaintext is replaced by a letter that is some fixed number of positions up/down the alphabet, to form the ciphertext.



Here is an example of the plaintext and the corresponding ciphertext, after a left shift of 3.



Write a Python function encode_caesar that takes in two parameters `plaintext` and `key`, encrypts the `plaintext` and then returns the ciphertext (which has been encoded using the key). The `plaintext` is the message to be encrypted to ciphertext, and `key` is the shift (which can be right or left). A key value of 3 indicates that each character in the plaintext should be shifted right by 3, and key value of -3 indicates that each character in the plaintext should be shifted left by 3.

You can use the following code snippet to test your code:

```
# the following statements should print True
print(encode_caesar("If you want something badly enough, do not
give up!", -3) == "Fc vlr txkq pljbqefkd yxaiv bklrde, al klq
dfsb rm!")
print(encode_caesar("Programming is SO FUN!", 12) ==
"Bdasdmyyuzs ue EA RGZ!")
```

💾 05 Sep 2019