Assignement A2
ITI1120 Section E

Read the instructions below carefully. The instructions must be followed. This assignment is worth 5% of your grade. The assignment is due on Monday 17th of October Midnight. No late assignments will be accepted.

The goal of this assignment is to learn and practice (via programming and quizzes) the concepts that we have learned in the last two weeks: function design, function calls, branching (that is, if statements), strings, for-loops, range function and lists.
Some of these concepts (range function and lists) will be covered more on Monday/Wednesday class (3/5 Oct). But you can certainly already work on many of the questions.

This assignment has two parts. Each part explains what needs to be submitted. Put all those required documents into a folder called a2_xxxxxx, zip it and submit it as explained in lab 1. (In particular, the folder should have the following files:
a2_part1_xxxxxx.py,
a2_part2_xxxxxx.py and a2_part2_xxxxxx.txt

For each function that you design for this assignment you have to include docstrings that specify:
- type contract
- description about what the function does (while mentioning parameter names)
- preconditions, if any
If you do not know what this means revisit the lectures/labs from the past, or read section 3.6 from the "Practical Programming" textbook or alternatively view the following short video:
https://www.youtube.com/watch?v=Q5GR138-Mmk&index=12&list=PLkHsKoi6eZnwpn7P5G8gEBebAY8Jbky4N

****

PART 1 (20 points)

****

Suppose you are asked to design a software tool that helps an elementary school student learn arithmetic operations. The software allows the student to select the arithmetic operation she or he wishes to study. The student chooses from a menu one of two arithmetic operations: Addition and Multiplication. Then the student is asked how many questions would he/she like to be tested on. That number is stored in variable called n. Based on the student's choice and answer, the software tests the student with exactly n questions. If n is zero no test should be performed). For each of the n questions, two random positive one-digit integers are generated; then the student is asked to enter the answer for the arithmetic operation applied to the two numbers.

At the end, the software displays a message “Well done! Congratulations.” if at least 80% of the questions are answered correctly; if at least 60% but less than 80% of the questions is answered correctly, the program should display "Not too bad but please study and practice some more.", otherwise, the program should display "Please study more and ask your teacher for help.".

a) Implement a Python function, called, perform_test, that will execute all the arithmetic tests for a student for multiplication or addition operations. The function has two input parameters:
   i. A or a for addition;
   ii. M or m for multiplication.
b) The second one is a positive integer n representing the number of questions in the test. Then it gets the student to answer n questions as follows:
   i. Randomly generates two positive one-digit integers.
   ii. Ask the student to enter the answer for the arithmetic operation of the two numbers.
   iii. Checks if the result is correct. If the answer is incorrect, it provides the correct answer.

As questions are answered, the correct answers are counted. The number of correct answers is returned by the function.

(Outside of the function) implement the main part of the program to interact with the student to obtain the choice for either multiplication or addition and the number of questions, then call the function developed in part (a) to test the student (recall that the function returns the number of correct answers). Then print one of three possible messages to the student according to the criteria mentioned above:

a) “Well done! Congratulations.” or
b) "Not too bad but please study and practice some more." or
c) "Please study more and ask your teacher for help.",

Store your program in file called a2_part1_xxxxxx.py
Test it by pressing Run Module.

View the sequence of running the code (attachment : run_code.docx)   as demo for running your code.

Note that to generate a random number first import module called random and then use the following function random.randint

Here is what help(random.randint) gives:
"randint(a, b) method of random. Random instance
    Return random integer in range [a, b], including both end points."

****

lis

This part resembles assignment 1. Each question asks you to design one function. Put all these functions (for all the questions in this part) in one file only, called a2_part2_xxxxxx.py (where xxxxxx is replaced with your student number). Within this file, a2_part2_xxxxxx.py, separate your answers (i.e. code) to each question with a comment that looks like this:

```
#####################################################################
  # Question X
#####################################################################
```

Similarly to assignment 1, in addition to a2_part2_xxxxxx.py you have to submit a separate file called a2_part2_xxxxxx.txt with your function tests copy pasted there. If you do not know what this means, read the Assignment 1 description again.

Your program must run without syntax errors.   The questions that have syntax errors will receive zero points. More specifically, for each of the functions below, I have provided one or more tests to test your functions with. To obtain a partial mark your function may not necessarily give the correct answer on these tests. But if your function gives any kind of py thon error when run on the tests provided bellow, that question will be marked with zero points.

Your functions will be tested both with provided examples and with some other examples.

=====================
Question 1: (5 points)
=====================

Write a function, size_format(b), that takes an integer, b, that represents a number of bytes and returns a readable string representation of this number of bytes.
The representation uses the metric units of bytes (B), kilobytes (KB), megabytes (MB), gigabytes (GB), and terrabytes (TB) and includes 1 significant decimal place.
If b is a negative number, the function should return a string 'buy a new hard disk'.

Testing your function:

```
>>> size_format(623)
'623B'
>>> size_format(1500)
'1.5KB'
>>> size_format(2732888)
'2.7MB'
>>> size_format(2762888)
'2.8MB'
>>> size_format(-11230)
'buy a new hard disk'
```

=====================

Question 2: (5 points)

=====================

(Suppose the following is true for this question, even if there are some exceptions in real life.)

French country names are feminine when they end with the letter e, masculine otherwise, except for the following countries which are masculine even though they end with e: le Belize, le Cambodge, le Mexique, le Mozambique, le Zaire, le Zimbabwe. Write a function call add_article that takes as input a string s containing a name of a country and returns a string with article added: le for masculine and la for feminine, such as le Canada and la Belgique. However, if the country name starts with a vowel, use l'; for example, l'Italie. For the following plural countries, use les: les Etats-Unis and les Pay-Bas. You may assume that the strings your functions will be tested with contain proper country names with first letter capitalized and that the only plural countries are Etats-Unis and Pay-Bas.

Testing your function:

```
>>> add_article("Canada")
'le Canada'
>>> add_article("Cambodge")
'le Cambodge'
>>> add_article("Belgique")
'la Belgique'
>>> add_article("Italie")
"l'Italie"
>>> add_article("Pay-Bas")
'les Pay-Bas'
```

=====================

Question 3: (5 points)

=====================

Write a function called factorial that takes as input one number, n,

and returns the value n*(n-1)*(n-2)*...*2*1. You may not use the factorial(x) function from the math module. Roll
your own implementation. You may assume that n is a non-negative
integer

Testing your function:

```
>>> factorial(0)
1
>>> factorial(1)
1
>>> factorial(2) 2
>>> factorial(3) 6
>>> factorial(4) 24
>>> factorial(5) 120
>>> factorial(500)
1220136825991110068701238785423046926253574342803192842
19241358838584537315388199760
5496447502203281863013616477148203584163378722078177200
48078520515932928547790757193
9330603772960859086270429174547882424912726344305670173
27076946106280231045264421887
8789465754777149863494367781037644274033827365397471386
47787849543848959553753799042
3241061271326984327745715546309977202781014561081188373
70953101635632443298702956389
6628911658974769572087926928871281780070265174507768410
71962439039432253642260523494
5850129918571501248706961568141625359056693423813008856
24924689156412677565448188650
6593847951775360894005745238940335798476363944905313062
32374906644504882466507594673
5862074637925184200459369692981022263971952597190945217
82333175693458150855233282076
2820023402626907898342451712006207714640979456116127629
14595123722991334016955236385
0942885592018727433795173014586357570828355780158735432
76888868012039988238470215146
7605445407663535984174430480128938313896881639487469658
81750450692636533817505547812864000000000000000000000
0000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000
```

=====================

Question 4: (5 points)

=====================

Call an integer special if it is non-negative and divisible by four, except that any non-negative integer divisible by 100 is not special unless it is also divisible by 400.   Write a function, special_count, that takes as input a non-empty list l of integers and returns the number of special integers it contains.

Testing your function:

```
>>> special_count([2020, 600, 800, 22])
2
>>> special_count([44,-1200,100, 0])
2
```

=====================
Question 5: (5 points)
=====================
If there is a vote at a meeting, there are several possible outcomes based on the number of yes and no votes (abstains are not counted). If all the votes are yes, then the proposal passes "unanimously", if at least 2/3 of the votes are yes, then the proposal passes with "super majority", if at least 1/2 of the votes are yes, then the proposal passes by "simple majority", and otherwise it fails. Write a function called vote that asks a user to enter all yes-s and no-s and abstained-s and then press enter.   The function then prints the outcome of the vote. You may assume that the user will enter at least one yes or no vote. You may assume that the user will enter yes or y for yes votes, no or n for no votes, and abstained for abstained votes. Finally, you may assume that the only words that the user will enter are yes, y, no, n and abstained but in any combination of lower-case/upper-case letters as can be seen in below tests:

Testing your function: (in the tests below the parts that the user enters are in brown)

```
>>> vote()
Enter the yes, no, abstained votes one by one and then press enter:
  yes Yes yes yes Y abstained abstained y yes yes yes
proposal passes unanimously
>>> vote()
```

Enter the yes, no, abstained votes one by one and then press enter:
 yes,yes, no,   YES, no, yes, abstained, yEs,   Y,no
proposal passes with super majority
>>> vote()
Enter the yes, no, abstained votes one by one and then press enter:
 abstained no abstained yes no yes no yes yes yes no
proposal passes with simple majority
>>> vote()
Enter the yes, no, abstain votes one by one and then press enter:
 no yes no no no, yes yes yes no
proposal fails

=====================
Question 6: (5+5=10 points)
=====================

Write a function that takes as input a positive integer n, produces n random integers in the range [-100, 100], and prints all those integers, the minimum of all those integers and the average of all those integers. You should write two versions of this function stats_v1 and stats_v2. In stats_v1, you must use a list to store all those random numbers and only then compute the rest. In stats_v2, neither lists nor tuples nor any other way to store all of those random numbers is allowed. In stats_v2 use a for loop and compute the minimum and the average on the "fly".

```
>>> stats_v1(1)
The minimum and the average of the following numbers:
23
is 23 and 23.0
>>>
>>> stats_v1(3)
The minimum and the average of the following numbers:
88 -10 84
is -10 and 54.0
>>>
>>> stats_v2(4)
The minimum and the average of the following numbers:
77 69 21 83
```

```
is 21 and 62.5
>>>
>>> stats_v2(4)
The minimum and the average of the following numbers:
31 -52 -58 34
is -58 and -11.25
```

Note that by default the print function prints the given arguments and then goes to a new line. You can change this default behaviour of print function by using the keyword argument `end=<string>`. For example, if you want print function to print an empty space instead of a new line when it finishes use `end=' '`.
For example:

```
print(1, end=' ')
print(2, end=' ')
print(3)
```

prints

```
1 2 3
```

=====================
Question 7: (5 points)
=====================
Write a function emphasize that takes as an input a string s and returns a string with a blank space inserted between every pair of consecutive characters in s.

Testing your function:

```
>>> emphasize('v')
'v'
>>> emphasize('   song ?   tr a ')
'       s o n g     ?       t r     a   '
>>> emphasize('')
''
>>> emphasize('very important')
'v e r y     i m p o r t a n t'
>>> emphasize(' really?')
'   r e a l l y ?'
```

=====================
Question 8: (5 points)
=====================
Write a function crypto that takes as an input a string s and returns an encrypted string where encryption proceeds as follows: in the new string the character of s should appear in the following order in the new string: last, first, second_to_last, second, third_from_the_back, third …. See examples below.
\
Testing your function:

```
>>> crypto('Good Day')
'yGaoDo d'
>>> crypto('Good Days')
'sGyoaoDd '
>>> crypto(',4?tr')
'r,t4?'
```

- `>>> crypto('ab')`
- `'ba'`
- `>>> crypto('a')`
- `'a'`
- `>>> crypto('')`
- `''`

=====================
Question 9: (5 points)
=====================
Write a function stranger_things that takes as an input two non-empty lists. The elements of that list can be numbers, strings or boolean values. The function should return True if

- if both l1 and l2 have the same number of elements **and**
- if the 1st element of l1 is **the same** as the 1st element of l2 **and**
- if the 2nd element of l1 is **not the same** as the 2nd element of l2 **and**
- if the 3rd element of l1 is **the same** as the 3rd element of l2 **and**
- if the 4th element of l1 is **not the same** as the 4th element of l2 **and**
- ….

Otherwise, it returns False.

(In other, True is returned if and only if the lists have the same number of elements and every pair of elements of l1 and l2 that are at the same even index have to be the same and every pair of elements of l1 and l2 that are at the same odd index have to be distinct). See few examples below.

Testing your function:

```
>>> stranger_things([1],[2,"aha"])
False
>>> stranger_things([1,2,True, '7'],[1,False,True,5])
True
>>> stranger_things([1,2,3,4],[1,2,3,4])
False
>>> stranger_things([1,2,4],[1,'2',3])
False
>>> stranger_things([1,2,4],[1,'2',4])
True
>>> stranger_things([],[])
```