# ASSIGNMENT 2

Read the instructions below carefully. The instructions must be followed. This assignment is worth 5% of your grade. The assignment is due on Monday 19th of October 8AM. 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 ...
Some of these concepts will be covered more on Monday/Tuesday class (5/6 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 view the coursera 6-min video
"Function Design Recipe" in Week 2. (or revise what we did in class and labs)

**********
PART 1 (15 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; the first one is an integer, 0 or 1, that represents the required operation (1 for multiplication and 0 for addition), 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:

1. Randomly generates two positive one-digit integers.
2. Ask the student to enter the answer for the arithmetic operation of the two numbers.
3. 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.

b) (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 ("Well done! Congratulations." or "Not too bad but please study and practice some more." or "Please study more and ask your teacher for help.", as determined by the criteria listed above).

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

View the video I made called a2_part1_example_run.mp4 to see how your program should behave when you run it.

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."

**********
PART 2
**********

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 python 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 called in_or_out_square that takes as input five numbers (i.e. the function has 5 input parameters). The first two input parameters are numbers that represent the x and y coordinates of the bottom left corner of
a square. The third number represents the length of the side of the square. The fourth and fifth number represent the x and y coordinates of some query point. (Notice that the first three numbers completely define a square and its position in the plane). (From this description you should conclude that your function has 5 input parameters)

You may assume that all 5 parameters are numbers. However if the side length is a negative number, the function should return the string "invalid side length".

Otherwise, if the side length is positive, your function should test if the given query point is inside of the given square. A point on the boundary of a square is considered to be inside the square. If the query point is inside the given square, the function should return one string containing nicely formatted sentence stating the results. See the example function calls below

Testing your function:

```
>>> in_or_out_square(0, 0, -2.5, 0.5, 1.5)
'invalid side length'
>>> in_or_out_square(0, 0, 2.5, 0.5, 1.5)
'The given query point (0.5, 1.5) is inside of the square.'
>>> in_or_out_square(2.5, 1, 1, -1, 1.5)
'The given query point (-1.0, 1.5) is outside of the square.'
```

==========
Question 2: (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. (Thus your function has one input parameter)
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)
1220136825991110068701238785423046926253574342803192842192413588385845373
5496447502203281863013616477148203584163378722078177200480785205159329285
9330603772960859086270429174547882424912726344305670173270769461062802310
8789465754777149863494367781037644274033827365397471386477878495438489595
3241061271326984327745715546309977202781014561081188373709531016356324432
6628911658974769572087926928871281780070265174507768410719624390394322536
5850129918571501248706961568141625359056693423813008856249246891564126775
6593847951775360894005745238940335798476363944905313062323749066445048824
5862074637925184200459369692981022263971952597190945217823331756934581508
2820023402626907898342451712006207714640979456116127629145951237229913401
0942885592018727433795173014586357570828355780158735432768888680120399882
7605445407663535984174430480128938313896881639487469658817504506926365338
```

==========
Question 3: (5 points)
==========

Write a function, strange_count, that takes as input a string s and prints the number of characters in s that are lower case letters of English alphabet between 'b' and 's' (including 'b' and 's') or digits between '3' and '7' (including '3' and '7'). (Thus function strange_count has one input parameter, a string s)

Testing your function:

```
>>> strange_count('2aAb3?eE')
3
>>> strange_count('16ABCDEFz')
1
```

==========
Question 4: (5 points)
==========

Write a function called vote_percentage that takes as input a string, results. Thus the function has one input parameter, called results. Your function should count the number of substrings 'yes' in the string results and the number of substrings 'no' in the string results, and it should return the percentage of 'yes' (among all 'yes' and 'no'). (You may assume that string results has at least one yes or no and that the only words present are yes, no and/or abstained)

Testing your function:

```
>>> vote_percentage('yes yes yes yes yes abstained abstained yes yes yes yes')
1.0
>>> vote_percentage('yes,yes, no,  yes, no, yes, abstained, yes,  yes,no')
0.6666666666666666
>>> vote_percentage('abstained no abstained yes no yes no yes yes yes no')
0.5555555555555556
```

```
>>> vote_percentage('no yes no no no, yes yes yes no')
0.4444444444444444
```

==========
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 solution must involve making a call to function vote_percentage

Testing your function:

```
>>> vote()
Enter the yes, no, abstained votes one by one and then press enter:
yes yes yes yes yes abstained abstained yes 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,  yes,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
```

**********

CLARIFICATION ABOUT Question 6 and 7

The next two questions ask you to solve the same problems in two different ways. In particular:

In neither Question 6 nor Question 7, you can call your own functions (eg. I do not want that your Q6 calls Q7 or that your Q7 calls your Q6).

In your Question 6 solution, you can use any python's functions/methods but no if statements and no loops.

In your Question 7 solution, exactly the opposite. You cannot use python methods (those called with dot operator) but rather once you get input from the user, find a solution using only loops, variables and if statements.

**********

==========
Question 6: (5 points)
==========
Question 6 and 7 ask you to solve the exact same problem, except that in Q6 you are allowed to use any string methods and Q7 is more restrictive.

Roman numerals use symbols M, D, C, X, V, and I whose decimal values are M = 1000, D = 500, C = 100, X = 10, V = 5, I = 1. For example, the Roman numeral MDCXVII corresponds to 1000+500+100+10+5+1+1=1617. There are more complicated rules, e.g. IV usually is 4, but we'll use a simple version of Roman numerals where we just accumulate the values of all symbols. E.g. MIIIMMDCM we'll evaluate as 4*1000 + 3*1 + 1*500 + 1*100 = 4603. Write a function called roman that asks a user to enter a roman number using capital letters M, D, C, X and I and returns a decimal numeral computed according to the above simplified rules.

Testing your function:

```
>>> roman()
Enter a roman number using capital letters M, D, C, X and I: MIIIMMDCM
4603
>>> roman()
Enter a roman number using capital letters M, D, C, X and I: IV
6
>>> roman()
Enter a roman number using capital letters M, D, C, X and I: MDCXVII
1617
>>>
>>> roman()
Enter a roman number using capital letters M, D, C, X and I: IIIVIII
11
```

==========
Question 7: (5 points)
==========
Repeat the previous question except that this time you are not allowed to use any string methods that need a dot to be called. Basically use only for loop over a string. Call the function roman_v2.

Tests are the same as above. Example

```
>>> roman_v2()
Enter a roman number using capital letters M, D, C, X and I: MIIIMMDCM
4603
```

==========
Question 8: (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.

more examples

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 ?'
```

corrected

==========

Question 9: (5 points)

==========

Write a function crypto that takes as an input a string s and returns an encrypted string where encryption proceeds as follows: split the text up into blocks of two letters each and swap each pair of letters (where spaces/punctuation, etc. is treated like letters). If the input string

Testing your function:

```
>>> crypto('Secret Message')
'eSrcteM seaseg'
>>> crypto('Secret Messages')
'eSrcteM seasegs'
>>> crypto(',4?tr')
'4,t?r'
```