Classwork 2

Assignment 1:

Define a function **myList2Text** which receives a list of strings and numbers and returns a string made from concatenation of all the members in the list. For example: if your list is **[1, ' cow, ', 2, ' geese ' 3, ' chickens ', 5.5, ' Sandwiches ', 3, 2.0, ' Liters ', 'of ', 'hell', '.']** , **myList2Text(myList)** should return : **'1 cow, 2 geese 3 chickens 5.5 Sandwiches 32.0 Liters of hell.'**

Assignment 2:

Define a function **myListMean** which receives a list and calculates the mean value of all the **numbers (numbers only, not strings!)** in the list, prints it and prints the index of a number closest to the mean result by its value. List contains both strings and numbers. Call example **MyListMean(myList)**. (You may use any library or function.)

Example:
If given list is:
List: [1.0, 'abc', 12, 'aaa', '1000', 'hello', 'aaaaa', 2, '1223']

Your print should be:	Because:
Mean Value: 5	**<= (1.0 + 12 + 2)/3 = 5**
Index: 7	**<= the index of number 2 is 7**

Assignment 3:

Define a function **list_files** which receives a directory path **dir** and returns a list of names of all the files in the specified directory. If the function receives nothing, it returns a list of all the files in the current work directory. Call examples: **list_files()**, **list_files(dir)**.

Assignment 4:

Define a function **create_3darray** which receives a tuple **(n,m,k)** and returns 3-dimensional ndarray (**n**-by-**m**-by-**k**) of zeros. Call example **create_3darray((n,m,k))**.

Assignment 5:

In cryptography, a Caesar cipher is a very simple encryption techniques in which each letter in the plain text is replaced by another letter from the alphabet. The new letter is chosen by adding some fixed number to the position of current one. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals. ROT-13 ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. In Python, the key for ROT-13 may be represented by means of the following dictionary:

key = {'a':'n', 'b':'o', 'c':'p', 'd':'q', 'e':'r', 'f':'s', 'g':'t', 'h':'u',
'i':'v', 'j':'w', 'k':'x', 'l':'y', 'm':'z', 'n':'a', 'o':'b', 'p':'c',
'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k',
'y':'l', 'z':'m', 'A':'N', 'B':'O', 'C':'P', 'D':'Q', 'E':'R', 'F':'S',
'G':'T', 'H':'U', 'I':'V', 'J':'W', 'K':'X', 'L':'Y', 'M':'Z', 'N':'A',
'O':'B', 'P':'C', 'Q':'D', 'R':'E', 'S':'F', 'T':'G', 'U':'H', 'V':'I',
'W':'J', 'X':'K', 'Y':'L', 'Z':'M'}

Your task in this exercise is to implement an encoder/decoder of ROT-13.

Note that since English has 26 characters, your ROT-13 program will be able to both encode and decode texts written in English.

Define a function named **myRot13**. It receives a message **text.** If a message is encoded the function returns a decoded message, if a message is not encoded the function returns an encoded message. Call example **myRot13(text)**.
Both received and returned texts must be in English and their type should be string.
NOTE: If a character which you want to convert is not a letter of English alphabet, leave it as it is. If you encounter a backslash ('\'), skip it and the next symbol which is located right after it. For instance if you get a **('!Nn\nz\ttT \\n?')**, your function should return a **('!Aa\nm\tgG \\a?')**. (Note! return value and print representation are not the same!). Once you're done, you will be able to read the following secret message: **'Pnrfne pvcure?\n V zhpu cersre\t Pnrfne fnynq!'**

Assignment 6:

Define a function **countMyWords** which counts the occurrences of each word in a given string and prints the result.
For example: **CountMyWords('Hello Hello We Do Hi no No no we Donot Hey HE he heY')** should print:

hello :	2
we :	2
do :	1
hi :	1
no :	3
donot :	1
hey :	2
he :	2

Assignment 7:

A robot moves in a plane starting from the original point (0,0). The robot can move toward UP, DOWN, LEFT and RIGHT with a given steps. Write a function **robot_dist(UP, DOWN, LEFT, RIGHT**) which receives 4 integers assigning the numbers of steps in the specified directions. The function will return the Euclidian distance between the current position after a sequence of movements and an original point, and the phase angle (positive directions are up and right).
Example:
If the following integers are provided as an input to the program:
UP = 15
DOWN = 12
LEFT = 7
RIGHT = 3
then the output of the program should be:

Distance :	5.0
Angle :	2.498