Introduction to Computer Science (ICS 3U)
String Manipulation and Predefined Function Assignment

Instructions:
Create a program for each of the following problems using the concepts covered on the Strings and Predefined Functions Worksheet and the Math and Data Conversion Worksheet. Note that some of the programs or parts of programs are Level 4 programs and should only be done after all of the other programs are complete. Save your work in a folder called **Functions**.

1. Create a program that will ask the user to input a word. The program will then tell the user if there is an "s" in the word. Save your program as **Functions1.py**

2. a) Create a program that will ask the user to input their full name (first, middle and last), one name at a time. The computer should then output the user's initials separated by a period(.). Save your program as **Functions2.py**

   b) Have the computer output the initials of **any** name (ie. multiple 1st, middle or last names, no middle name, etc.) without asking the user how many names they have. Re-save your program as **Functions2.py** *(Level 4)*
   Hint: See this example program to get you started on the Level 4 section

3. Create a program that will ask the user to input a sentence. The computer should determine if the last letter is punctuation (i.e. a period, an exclamation mark or a question mark) and output an appropriate message. If the punctuation is not correct, output an appropriate message. Save your program as **Functions3.py**

4. Create a program that will allow the user to input a word and will output the reverse of the word (i.e. the reverse of RUN is NUR). Save your program as **Functions4.py**

   Hint 1: See this example program to get you started with outputting the string one character at a time.

   Hint 2: See this example program to get you started with keeping the cursor on the same line when outputting more than one string in a loop.

5. Create a program which will output the total number of characters (including blanks) in a series of words inputted by the user. It should then output the average number of characters in each word. Save your program as **Functions5.py**

   Hint 1: Count the number of spaces to calculate the number of words.

   Hint 2: Remember that you cannot include the spaces when calculating the average length of the words.

6. Create a program that will allow the user to input a string of any size. It should then allow the user to choose to do one of the following:
    1) Output the entire string in lower case letters
    2) Output the entire string in capitals
    3) Output the entire string with only the first letter capitalized
    4) Output the entire string with only the 1st letter of each word in the string capitalized. *(Level 4)*
        Hint: See this example program to get you started on the Level 4 section

    Save your program as **Functions6.py**

7. a) Create a program which will allow the user to input a string. The program should then output the same string in a "secret code" which will be unreadable. Save your program as **Functions7.py**

    Hint: See this example program to get you started with working with the string one character at a time.

    b) Add a section to your program which will decode any message created by your program in (a). Re-save your program as **Functions7.py**

    c) Improve your program so that the user is asked whether they wish to code or decode the word after they input it. Re-save your program as **Functions7.py**

8. One of the problems with creating a program that requires integer input is that it crashes if the user inputs anything other than a legitimate number. Remember that a good programmer will attempt to reduce the number of program crashes to an absolute minimum. Later we will learn how to do this with exceptions but in the meantime, some of the data conversion predefined functions will be used to achieve this.

    a) Create a program which will prompt the user to input a single positive integer value. If proper positive integer data is inputted, the program should calculate and output the following:
        - The square root of the value
        - The sine of the value
        - The area and circumference of the circle when assuming that the inputted value is the circle's radius

    Save your program as **Functions8.py**

    b) Update your program so if incorrect data is inputted (ie. a string, a float or a negative integer), the user should be prompted to input the correct data. Only when correct data is inputted should the program continue. Re-save your program as **Functions8.py**

9. Remember that we talked about Unicode when you did the ASCII assignment. Python uses Unicode to represent characters in strings. Unicode can represent characters from 0000 to FFFF (65,535 characters) which represents many of the characters in many different languages. See a complete Unicode table at http://www.tamasoft.co.jp/en/general-info/unicode.html

To see some of the characters available in Unicode, type the following into your computer to see the results. Note that \u indicates that it is a Unicode character. Remember, Unicode characters are in hexadecimal (base 16) values.

print ("\u023B")
print ("\u0050")
print ("\u20AC")
print ("\u0038")

print ("\uCB00")
print ("\uD403")
print ("\u01E4")
print ("\u0787")

Create a program that will allow the user to input a starting unicode value and an ending unicode value and have the range of Unicode characters outputted. Save your program as **Functions9.py**

Hint 1: Unfortunately, you cannot simply concatenate the "/u" onto a number to make a Unicode value. This will result in an error. The **chr ()** function, however, will covert integers to Unicode.

Hint 2: Unicode values are hexadecimal (base 16) values. In order to use a for loop, you will need to convert the hexadecimal values into integer values. This is done using the **int ()** function that you are familiar with and adding a 2nd parameter which is the base. For example:

print (int ("AAA", 16)) will output 2730 which is the decimal equivalent of the hexadecimal number 'AAA'

10. Remember that a valid Python variable name can have letters, numbers and an underscore (_) but cannot have most other special characters and cannot start with a number. It should also begin with a lower case letter but can contain capitals. Create a program which will allow the user to input a potential variable name and output whether it is a proper variable name or not. If it is not, output what the problem is. Save your program as **Functions10.py** *(Level 4)*

11. a) Create a program which will have the user input string of three or more words. The program should then calculate and output the length of the longest and shortest word in the string. Save your program as **Functions11.py** *(Level 4)*

Hint: See this example program to get you started.

b) Improve your program so that the longest and shortest words are outputted as well. Re-save your program as **Functions11.py** *(Level 4)*