Computer Science 1 (../index.html) CPSC 1301K (../index.html)

String Exercises

These excercises provide the opportunity for you to practice string fundmentals that persist throughout your studies in cs.

Submission Instructions

For each of the following practice assignments, save your solution in a `.py` file, with the name being `strings` and the number of the assignment. For example, for Strings01: Names, save your solution in a file named `strings01.py`. Then, submit that file the respective assignment on codePost.io (https://codePost.io).
To register for a free account, go to https://codepost.io/signup/join?code=I5039NQNWJ (https://codepost.io/signup/join?code=I5039NQNWJ). Register with your CSU email address. Sometimes it takes more than one try, so please work on this well before the deadline. Additionally, some students have had better luck with using the “Forgot password” link.

Strings01: Iterating Over a String.py

Ask the user to input a string. Use a for loop to display each letter of a string.
Example:

```
Please enter a string: Geeks
G
e
e
k
s
```

Strings02: Reverse the String.py

Write a Python script that requests a string from the user and displays the string in reverse order.
Example:

```
Please enter a string: hello
olleh
```

Strings03: Uppercase Letters.py

Complete the countUpper() function to return the number of uppercase letters in string.
Examples:

```
countUpper( 'Jake went to Publix in Columbus Georgia.') returns 4
countUpper( 'python') returns 0
countUpper( 'CPSC 1301K') returns 5
```

Provided code

```
def countUpper( string ):
    print( countUpper( 'Jake went to Publix in Columbus Georgia.') )
    print( countUpper( 'python') )
    print( countUpper( 'CPSC 1301K') )
```

Strings04: Changing Hills.py

Complete the swapCaps() function to change all lowercase letters in string to uppercase letters and all uppercase letters to lowercase letters. Anything else remains the same.
Examples:

```
swapCaps( 'Hope you are all enjoying October' ) returns 'hOPE YOU ARE ALL ENJOYING oCTOBER'
swapCaps( 'i hope my caps lock does not get stuck on' ) returns 'I HOPE MY CAPS LOCK DOES NOT G
ET STUCK ON'
```

Provided code:

```
def swapCaps( string ):
```

Strings05: Password.py

Complete the goodPassword() function to return True if password is a good password. Otherwise, have it return False. Criteria for a good password are:

- At least 8 characters
- At least 1 uppercase letter
- At least 1 lowercase letter

- At least 1 number
- At least 1 symbol

Examples:

```
goodPassword( 'P@ssword1' ) returns True
goodPassword( 'password' ) returns False
```

Provided code:

```
def goodPassword( password ):
```

Strings06: Scrambled.py

Complete the `isScrambled()` function to return `True` if `stringA` can be reordered to make `stringB`. Otherwise, return `False`. Ignore spaces and capitalization. Note, you can not use a `list` for this assignment.

Examples:
`isScrambled( 'Easy', 'Yase' )` returns `True`
`isScrambled( 'Easy', 'EasyEasy' )` returns `False`
`isScrambled( 'eye', 'eyes' )` returns `False`
`isScrambled( 'abcdefghijklmnopqrstuvwxyz', 'zyxwvutsrqponmlkjihgfedcba' )` returns `True`
`isScrambled( 'Game Test', 'tamegest' )` returns `True`

Hint:
One solution is to remove all spaces and make a lowercase version of each string. Then, compare the lengths to make sure that they're equal. Then, for each character in the first string, replace the first occurrence of it in the second string with an empty string. You can use the `str` method `replace()` to do this. Then, test that the second string is now an empty string.

Provided code:

```
def isScrambled( stringA, stringB ):
```

Complete the `firstMiddleLast()` function to return a new string with the first, middle, and last character from `string`. If `string` has an even number of characters, grab the middle two characters. It is safe to assume that no string passed in will be less than 3 characters.

Examples:
`firstMiddleLast( 'hello' )` returns `'hlo'`
`firstMiddleLast( 'Grounds' )` returns `'Gus'`
`firstMiddleLast( 'plural' )` returns `'purl'`
`firstMiddleLast( 'password' )` returns `'pswd'`
Provided code:

```
def firstMiddleLast( string ):

```

Strings08: Split and Swap

Complete the `splitAndSwap()` function to return a `str` with the first and last halves swapped. For example, if the parameter was "`moon`", then it would return "`onmo`" by splitting it down the middle: "`mo` | `on`" and then swapping the first and last halves. If the word has an odd number of characters, ignore the middle character.

Examples:
`splitAndSwap( 'moon' )` returns `'onmo'`
`splitAndSwap( 'orange' )` returns `'ngeora'`
`splitAndSwap( 'oranges' )` returns `'gesora'` (Notice that the middle letter was ignored and not part of the return value)
`splitAndSwap( 'hello' )` returns `'lohe'`

Provided code:

```
def splitAndSwap( string ):

```

Strings09: Palindromes

A palindrome is a word spelled the same forwards and backwards. This can also apply to any phrases that might be the same forwards and backwards (if you ignore the spaces). Complete the `isPalindrome()` function to return `True` if the string is a palindrome or `False` otherwise. For this assignment, ignore capitalization.

Examples:
`isPalindrome( 'Eevee' )` returns `True`
`isPalindrome( 'nurses run' )` returns `True`
`isPalindrome( 'Hello There' )` returns `False`
Provided code:

```
def isPalindrome( string ):

```

Strings10: Letter Grade

Write a Python program that:

1. requests a grade between 0 and 100 from the user
2. displays "`Your letter grade is`" and then the letter grade according to the following scale:

- 90 or above: `'A'`
- 80-89: `'B'`
- 70-79: `'C'`
- 60-69: `'D'`
- <60: `'F'`

Use a `str` method to determine if the user typed something other than an integer value. Additionally, verify that the value is between 0 and 100. If the user did not enter a number or entered an invalid number, repeatedly

1. display "`Sorry, this program only accepts values between 0 and 100`"
2. prompt the user,

until valid input is given.

Example 1:

```
Please enter a grade (between 0 and 100): perfect
```

```
Sorry, this program only accepts values between 0 and 100.
```

```
Please enter a grade (between 0 and 100): ABCDF
```

```
Sorry, this program only accepts values between 0 and 100.
```

```
Please enter a grade (between 0 and 100): 85
```

```
Your letter grade is B
```

Example 2:

```
Please enter a grade (between 0 and 100): -4

Sorry, this program only accepts values between 0 and 100.

Please enter a grade (between 0 and 100): -99

Sorry, this program only accepts values between 0 and 100.

Please enter a grade (between 0 and 100): A+

Sorry, this program only accepts values between 0 and 100.

Please enter a grade (between 0 and 100): Ok, just A

Sorry, this program only accepts values between 0 and 100.

Please enter a grade (between 0 and 100): A

Sorry, this program only accepts values between 0 and 100.

Please enter a grade (between 0 and 100): 91

Your letter grade is A
```

Feel free to add more functions to this program to help separate the work. For example, you could have a validation function that returns `True` or `False`, depending on if the user input is a valid grade.

Provided code:

```
def main( ):



main()
```