Strings

May 11, 2020

1 Strings Exercises

1.0.1 Introduction

Python Strings are Arrays. They are arrays of bytes representing unicode characters. We can assign a string literal enclosed in a single quotation (`'Udacity'`) or double quotation (`"Udacity"`) marks to a variable.

```
str1 = "Udacity"
```

Common String Methods Let us see some common methods that we use on string variables. Refer here for more

```
In [62]: str1 = "Udacity"


         # LENGTH
         print(len(str1))                # 7


         # CHANGE CASE
         # The `lower()` and `upper` method returns the string in lower case and upper case resp
         print(str1.lower())               # udacity
         print(str1.upper())               # UDACITY


         # SLICING
         # string_var[lower_index : upper_index]
         # Note that the upper_index is not inclusive.
         print(str1[1:6])                # dacit
         print(str1[:6])                      # Udacit. A blank index means "all from that end
         print(str1[1:])                      # dacity
         print(str1[::-1])                      # yticadU

         # A negative index means start slicing from the end-of-string
         print(str1[-6:-1])               # dacit
```

```
# STRIP
# `strip()` removes any whitespace from the beginning or the end
str2 = "    Udacity    "
print(str2.strip())               # Udacity


# REPLACE/SUBSTITUTE A CHARACTER IN THE STRING
# The replace() method replaces all occurances a character in a string with another cha
print(str1.replace('y', "B")) #UdacitB


# SPLIT INTO SUB-STRINGS
# The split() method splits a string into substrings based on the separator that we spe
str3 = "Welcome, Constance!"
print(str3.split(",")) # ['Welcome', ' Constance!']


# CONCATENATION
print(str3 + " " + str1) # Welcome, Constance! Udacity
marks = 100
# print(str3 + " You have scored a perfect " + marks) # TypeError: can only concatenate
print(str3 + " You have scored a perfect " + format(marks)) # format() method converts


# SORT A STRING
# We can use sorted() method that sort any instance of *iterable*. The characters are c
print(sorted(str3)) # [' ', '!', ',', 'C', 'W', 'a', 'c', 'c', 'e', 'e', 'e', 'l', 'm',
```

```
7
udacity
UDACITY
dacit
Udacit
dacity
yticadU
dacit
Udacity
UdacitB
['Welcome', ' Constance!']
Welcome, Constance! Udacity
Welcome, Constance! You have scored a perfect 100
[' ', '!', ',', 'C', 'W', 'a', 'c', 'c', 'e', 'e', 'e', 'l', 'm', 'n', 'n', 'o', 'o', 's', 't']
```

Let us do some exercises to practice our work with string manipulation.

Exercise 1. Reverse Strings In this first exercise, the goal is to write a function that takes a string as input and then returns the reversed string.

For example, if the input is the string `"water"`, then the output should be `"retaw"`.

While you're working on the function and trying to figure out how to manipulate the string, it may help to use the `print` statement so you can see the effects of whatever you're trying.

Note - You can use built-in method `len()` on the string.

```
In [34]: # Code

         def string_reverser(our_string):

             """
             Reverse the input string

             Args:
                our_string(string): String to be reversed
             Returns:
                string: The reversed string
             """

             # TODO: Write your solution here
             length = len(our_string)
             output = ""
             for item in range(length):
                 output += our_string[length - item - 1]
             return output
             pass
```

```
In [35]: # Test Cases

         print ("Pass" if ('retaw' == string_reverser('water')) else "Fail")
         print ("Pass" if ('!noitalupinam gnirts gnicitcarP' == string_reverser('Practicing stri
         print ("Pass" if ('3432 :si edoc esuoh ehT' == string_reverser('The house code is: 2343
```

```
Pass
Pass
Pass
```

Hide Solution

```
In [ ]: # Solution

        def string_reverser(our_string):

            """
            Reverse the input string

            Args:
```

```
        our_string(string): String to be reversed
    Returns:
        string: The reversed string
    """

    # New empty string for us to build on
    new_string = ""

    # Iterate over old string
    for i in range(len(our_string)):
        # Grab the charecter from the back of the string and add them to the new string
        new_string += our_string[(len(our_string)-1)-i]

    # Return our solution
    return new_string


# Test Cases

print ("Pass" if ('retaw' == string_reverser('water')) else "Fail")
print ("Pass" if ('!noitalupinam gnirts gnicitcarP' == string_reverser('Practicing strin
print ("Pass" if ('3432 :si edoc esuoh ehT' == string_reverser('The house code is: 2343'
```

Exercise 2. Anagrams The goal of this exercise is to write some code to determine if two strings are anagrams of each other.

An anagram is a word (or phrase) that is formed by rearranging the letters of another word (or phrase).

For example: - "rat" is an anagram of "art" - "alert" is an anagram of "alter" - "Slot machines" is an anagram of "Cash lost in me"

Your function should take two strings as input and return `True` if the two words are anagrams and `False` if they are not.

You can assume the following about the input strings: - No punctuation - No numbers - No special characters

Note - You can use built-in methods `len()`, `lower()` and `sort()` on strings.

```
In [42]: # Code

    def anagram_checker(str1, str2):

        """
        Check if the input strings are anagrams of each other

        Args:
            str1(string),str2(string): Strings to be checked
        Returns:
            bool: Indicates whether strings are anagrams
        """
```

```
        # TODO: Write your solution here
        str1 = str1.replace(" ", "").lower()
        str2 = str2.replace(" ", "").lower()
        if len(str1) == len(str2):
            if sorted(str1) == sorted(str2):
                return True
        return False
        pass
```

In [43]:
```
# Test Cases

print ("Pass" if not (anagram_checker('water','waiter')) else "Fail")
print ("Pass" if anagram_checker('Dormitory','Dirty room') else "Fail")
print ("Pass" if anagram_checker('Slot machines', 'Cash lost in me') else "Fail")
print ("Pass" if not (anagram_checker('A gentleman','Elegant men')) else "Fail")
print ("Pass" if anagram_checker('Time and tide wait for no man','Notified madman into
```

```
Pass
Pass
Pass
Pass
Pass
```

Hide Solution

In [2]:
```
# Solution

def anagram_checker(str1, str2):

    """
    Check if the input strings are anagrams

    Args:
       str1(string),str2(string): Strings to be checked if they are anagrams
    Returns:
       bool: If strings are anagrams or not
    """

    # Clean strings and convert to lower case
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Compare the length of both strings
    if len(str1) == len(str2):
        # Sort each string and compare
        if sorted(str1) == sorted(str2):
            return True
```

```
            return False


        print ("Pass" if not (anagram_checker('water','waiter')) else "Fail")
        print ("Pass" if anagram_checker('Dormitory','Dirty room') else "Fail")
        print ("Pass" if anagram_checker('Slot machines', 'Cash lost in me') else "Fail")
        print ("Pass" if not (anagram_checker('A gentleman','Elegant men')) else "Fail")
        print ("Pass" if anagram_checker('Time and tide wait for no man','Notified madman into w
        print ("Pass" if (anagram_checker('rat','art')) else "Fail")
```

```
Pass
Pass
Pass
Pass
Pass
Pass
```

Exercise 3. Reverse the words in sentence Given a sentence, reverse each word in the sentence while keeping the order the same!

In [60]:
```
# Code

def word_flipper(our_string):

    """
    Flip the individual words in a sentence

    Args:
       our_string(string): String with words to flip
    Returns:
       string: String with words flipped
    """

    # TODO: Write your solution here
    reversed
    flipped = ""
    our_array = our_string.split(" ")
    for item in our_array:
        for idx, char in enumerate(item):
            flipped += item[len(item) - 1 - idx]
        flipped += " "
    return flipped.strip()
    pass
```

In [61]:
```
# Test Cases
```

```
print ("Pass" if ('retaw' == word_flipper('water')) else "Fail")
print ("Pass" if ('sihT si na elpmaxe' == word_flipper('This is an example')) else "Fai
print ("Pass" if ('sihT si eno llams pets rof ...' == word_flipper('This is one small s
```

```
Pass
Pass
Pass
```

Show Solution

Exercise 4. Hamming Distance

```
In [ ]: # Solution

        def word_flipper(our_string):

            """
            Flip the individual words in a sentence

            Args:
               our_string(string): Strings to have individual words flip
            Returns:
               string: String with words flipped
            """

            word_list = our_string.split(" ")

            for idx in range(len(word_list)):
                word_list[idx] = word_list[idx][::-1]

            return " ".join(word_list)

        print ("Pass" if ('retaw' == word_flipper('water')) else "Fail")
        print ("Pass" if ('sihT si na elpmaxe' == word_flipper('This is an example')) else "Fail
        print ("Pass" if ('sihT si eno llams pets rof ...' == word_flipper('This is one small st
```

In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. Calculate the Hamming distace for the following test cases.

```
In [71]: # Code

         def hamming_distance(str1, str2):

             """
             Calculate the hamming distance of the two strings

             Args:
```

```
            str1(string),str2(string): Strings to be used for finding the hamming distance
        Returns:
            int: Hamming Distance
        """

        # TODO: Write your solution here
        diff = 0
        if len(str1) == len(str2):
            for idx in range(len(str1)):
                if str1[idx] != str2[idx]:
                    diff += 1
        return diff if diff != 0 else None

        pass
```

In [72]:
```
# Test Cases

print ("Pass" if (10 == hamming_distance('ACTTGACCGGG','GATCCGGTACA')) else "Fail")
print ("Pass" if  (1 == hamming_distance('shove','stove')) else "Fail")
print ("Pass" if  (None == hamming_distance('Slot machines', 'Cash lost in me')) else "
print ("Pass" if  (9 == hamming_distance('A gentleman','Elegant men')) else "Fail")
print ("Pass" if  (2 == hamming_distance('0101010100011101','0101010100010001')) else "
```

```
Pass
Pass
Pass
Pass
Pass
```

Hide Solution

In [ ]:
```
# Solution

def hamming_distance(str1, str2):

    """
    Calculate the hamming distance of the two strings

    Args:
       str1(string),str2(string): Strings to be used for finding the hamming distance
    Returns:
       int: Hamming Distance
    """

    if len(str1) == len(str2):
        count = 0

        for char in range(len(str1)):
```

```
            if str1[char] != str2[char]:
                count+=1

        return count

    return None



print ("Pass" if (10 == hamming_distance('ACTTGACCGGG','GATCCGGTACA')) else "Fail")
print ("Pass" if  (1 == hamming_distance('shove','stove')) else "Fail")
print ("Pass" if  (None == hamming_distance('Slot machines', 'Cash lost in me')) else "F
print ("Pass" if  (9 == hamming_distance('A gentleman','Elegant men')) else "Fail")
print ("Pass" if  (2 == hamming_distance('0101010100011101','0101010100010001')) else "F
```