

String Processing concepts

Boolean Methods(..)

There are several string methods that will return Boolean values:

Method	True if
<code>str.isalnum()</code>	String consists of only alphanumeric characters (no symbols)
<code>str.isalpha()</code>	String consists of only alphabetic characters (no symbols)
<code>str.islower()</code>	String's alphabetic characters are all lower case
<code>str.isnumeric()</code>	String consists of only numeric characters
<code>str.isspace()</code>	String consists of only whitespace characters
<code>str.istitle()</code>	String is in title case
<code>str.isupper()</code>	String's alphabetic characters are all upper case

Use:

```
number = "5"  
letters = "abcdef"  
  
print(number.isnumeric())  
print(letters.isnumeric())
```

Output:

```
True  
False
```

String Processing concepts

String Methods(..)

Method	Description
<code>str.capitalize()</code>	Returns the copy of the string with its first character capitalized and the rest of the letters are in lowercased.
<code>string.casefold()</code>	Returns a lowered case string. It is similar to the <code>lower()</code> method, but the <code>casefold()</code> method converts more characters into lower case.
<code>string.count()</code>	Searches (case-sensitive) the specified substring in the given string and returns an integer indicating occurrences of the substring. Syntax: <code>str.count(substring, start, end)</code> , <code>str.count(substring)</code>
<code>string.endswith()</code>	Returns True if a string ends with the specified suffix (case-sensitive), otherwise returns False. Syntax: <code>str.endswith(suffix, start, end)</code> , <code>str.endswith(suffix)</code>
<code>string.find()</code>	Returns the index of the first occurrence of a substring in the given string (case-sensitive). If the substring is not found it returns -1. Syntax: <code>str.find(substr, start, end)</code> , <code>str.find(substr)</code>
<code>string.index()</code>	Returns the index of the first occurrence of a substring in the given string. Syntax: <code>str.index(substr, start, end)</code> , <code>str.index(substr)</code>
<code>string.join()</code>	Returns a string, which is the concatenation of the string (on which it is called) with the string elements of the specified iterable as an argument. i.e <code>sep = '-->'</code> <code>mystr = 'Hello'</code> <code>print(sep.join(mystr))</code> Output: <code>'H-->e-->l-->l-->o'</code>
<code>string.ljust()</code>	Returns the left justified string with the specified width. If the specified width is more than the string length, then the string's remaining part is filled with the specified fillchar.

String Processing concepts

Method	Description
	<pre>mystr = 'Hi' print(mystr.ljust(4)) Output: 'Hi ' Print(mystr.ljust(4, '-')) Output: 'Hi--' Print(mystr.ljust(2, '-')) Output: 'Hi'</pre>
<code>string.lower()</code>	Returns the copy of the original string wherein all the characters are converted to lowercase.
<code>string.lstrip()</code>	Returns a copy of the string by removing leading characters specified as an argument. <pre>mystr = ' Hello World '</pre> <pre>mystr.lstrip() # removes leading spaces</pre> Output: 'Hello World '
<code>string.partition()</code>	Splits the string at the first occurrence of the specified string separator sep argument and returns a tuple containing three elements, the part before the separator, the separator itself, and the part after the separator. <pre>mystr = 'Hello a World'</pre> <pre>print(mystr.partition(' '))</pre> Output: ('hello', 'a ', 'world')
<code>string.replace()</code>	Returns a copy of the string where all occurrences of a substring are replaced with another substring. Syntax: <code>str.replace(old, new, count)</code> <pre>mystr = 'apples, bananas, apples, apples, cherries'</pre> <pre>print(mystr.replace('apples', 'lemons'))</pre> Output: lemons, bananas, lemons, lemons, cherries
<code>string.rfind()</code>	Returns the highest index of the specified substring (the last occurrence of the substring) in the given string. Syntax: <code>str.replace(old, new, count)</code> <pre>greet = 'Hello World!'</pre> <pre>print('Index of l: ', greet.rfind('l'))</pre> Output: Index of l: 9

String Processing concepts

Method	Description
<code>string.rindex()</code>	Returns the index of the last occurrence of a substring in the given string.
<code>string.rsplit()</code>	Splits a string from the specified separator and returns a list object with string elements. <code>langs = 'C,Python,R,Java,SQL,Hadoop'</code> <code>print(langs.rsplit(','))</code> Output: ['C', 'Python', 'R', 'Java', 'SQL', 'Hadoop']
<code>string.rstrip()</code>	Returns a copy of the string by removing the trailing characters specified as argument.
<code>string.split()</code>	Splits the string from the specified separator and returns a list object with string elements.
<code>string.splitlines()</code>	Splits the string at line boundaries and returns a list of lines in the string.
<code>string.startswith()</code>	Returns True if a string starts with the specified prefix. If not, it returns False.
<code>string.strip()</code>	Returns a copy of the string by removing both the leading and the trailing characters.
<code>string.swapcase()</code>	Returns a copy of the string with uppercase characters converted to lowercase and vice versa. Symbols and letters are ignored.
<code>string.title()</code>	Returns a string where each word starts with an uppercase character, and the remaining characters are lowercase.
<code>string.upper()</code>	Returns a string in the upper case. Symbols and numbers remain unaffected.

String Processing concepts

Q1. Predict the output:

```
def add_string(str1):  
    length = len(str1)  
    if length > 2:  
        if str1[-3:] == 'ian':  
            str1 += 'ly'  
        else:  
            str1 += 'ian'  
    return str1  
print(add_string('BU'))  
print(add_string('Bennett'))  
print(add_string('IIT'))
```

Sol:

```
BU  
Bennettian  
IITian
```

Q2. Predict the output:

```
def string_both(str):  
    if len(str) < 2:  
        return ''  
    return str[0:2] + str[-2:]  
print(string_both('BU University'))  
print(string_both('Bennett'))  
print(string_both('B'))
```

Sol: BUty
 Bett

String Processing concepts

Q3. Predict the output:

```
word = "Bennett University"
print (word[0])
print (word[0:1])
print (word[0:3])
print (word[:3])
print (word[-3:])
print (word[3:])
print (word[:-3])
```

Sol.

```
B    #get one char of the word
B    #get one char of the word
Ben  #get the first three char
Ben  #get the first three char
ity  #get the last three char
nett University #get all but the three first char
Bennett Univers #get all but the three last character
```

Q4. Predict the output

```
word = " Bennett University "
```



```
a.word.split(' ')
b.word.startswith("p")
c.word.endswith("y")
d.print('$'* 10)
```

Sol.

```
a. ['', 'Bennett', 'University', '']
b. False
c. False
d. $$$$$$$$$$ # prints $ ten times
```

String Processing concepts

Q5. Predict the output

```
def remove_char(str, n):  
    first_part = str[:n]  
    last_part = str[n+1:]  
    return first_part + last_part  
print(remove_char('Python', 2))  
print(remove_char('Python', 3))
```

Sol.

```
Pyhon  
Pyton
```

Q6. Predict the output:

```
def case_str(str1):  
    result_str = ""  
    for item in str1:  
        if item.isupper():  
            result_str += item.lower()  
        else:  
            result_str += item.upper()  
    return result_str  
print(case_str("Bennett University"))  
print(case_str("Python"))  
print(case_str("Programming"))
```

Sol:

```
bENNETT uNIVERSITY  
pYTHON  
pROGRAMMING
```

String Processing concepts

Q7. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return empty string

E.g.,

String1: 'BU4resources'

Result: 'BUes'

String2: 'U4'

Result: 'U4U4'

String3: 'B'

Result: Empty String

Sol.

```
def string_both_ends(str):  
    if len(str) < 2:  
        return ''  
  
    return str[0:2] + str[-2:]  
  
print(string_both_ends('BU4resources'))  
print(string_both_ends('U4'))  
print(string_both_ends('B'))
```


String Processing concepts

Q8. Your course instructor asked you to write programming which takes the input as a string from the user. Now your task is to find the first appearance of the substring 'not' and 'poor' from a given string, if 'not' follows the 'poor', replace the whole 'not...'poor' substring with 'good'. Return the resulting string.

E.g.,

String1: 'The lyrics is not that poor!'

'The lyrics is poor!'

Result: 'The lyrics is good!'

'The lyrics is poor!'

Sol.

```
def not_poor(str1):
    snot = str1.find('not')
    spoor = str1.find('poor')

    if spoor > snot and snot>0 and spoor>0:
        str1 = str1.replace(str1[snot:(spoor+4)], 'good')
        return str1
    else:
        return str1

print(not_poor('The lyrics is not that poor!'))
print(not_poor('The lyrics is poor!'))
```

String Processing concepts

Q9. Your course instructor asked you to write a python program that operates taking the input from the user as a string and counting the occurrences of each word in a given sentence.

E.g.,

String: the quick brown fox jumps over the lazy dog

Result: 'the': 2, 'quick': 1, 'brown': 1, 'fox': 1, 'jumps': 1, 'over': 1, 'lazy': 1, 'dog.': 1

Sol.

```
def word_count(str):  
    counts = dict()  
    words = str.split()  
  
    for word in words:  
        if word in counts:  
            counts[word] += 1  
        else:  
            counts[word] = 1  
  
    return counts  
  
print(word_count('the quick brown fox jumps over the lazy  
dog.'))
```

String Processing concepts

Q10. Write a Python program to change a given string to a new string where the first and last chars have been exchanged.

Input:

abcd
12345

Output:

dbca
52341

Sol:

```
def change_sring(str1):  
    return str1[-1:] + str1[1:-1] + str1[:1]  
print(change_sring('abcd'))  
print(change_sring('12345'))
```

String Processing concepts

Q11. Write a Python function that takes a list of words and return the longest word and the length of the longest one.

E.g.,

Input: List = ["Python", "Exercises", "Programming"]

Output: Longest word: Programming
Length of the longest word: 11

Sol.

```
def find_longest_word(words_list):
    word_len = []
    for n in words_list:
        word_len.append((len(n), n))
    word_len.sort()
    return word_len[-1][0], word_len[-1][1]
result = find_longest_word(["Python", "Exercises",
"Programming"])
print("\nLongest word: ",result[1])
print("Length of the longest word: ",result[0])
```

String Processing concepts

Q12. Write a Python program to remove the characters which have odd index values of a given string.

E.g.,

String1: abcdef

Result: ace

String1: python

Result: pto

Sol.

```
def odd_values_string(str):  
    result = ""  
    for i in range(len(str)):  
        if i % 2 == 0:  
            result = result + str[i]  
    return result  
  
print(odd_values_string('abcdef'))  
print(odd_values_string('python'))
```