Boolean Methods(..)

There are several string methods that will return Boolean values:

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

Use:

```
number = "5"
letters = "abcdef"

print(number.isnumeric())
print(letters.isnumeric())
```

```
Output:
True
False
```

String Methods(..)

Method	Description
str.capitalize()	Returns the copy of the string with its first character capitalized and the rest of the letters are in lowercased.
string.casefold()	Returns a lowered case string. It is similar to the lower() method, but the casefold() method converts more characters into lower case.
string.count()	Searches (case-sensitive) the specified substring in the given string and returns an integer indicating occurrences of the substring. Syntex: str.count(substring, start, end), str.count(substring)
string.endswith()	Returns True if a string ends with the specified suffix (case-sensitive), otherwise returns False. Syntex: str. endswith (suffix, start, end), str.endswith (suffix)
string.find()	Returns the index of the first occurence of a substring in the given string (case-sensitive). If the substring is not found it returns -1. Syntex: str.find(substr, start, end), str.find(substr)
string.index()	Returns the index of the first occurence of a substring in the given string. Syntex: str.index(substr, start, end), str.index(substr)
string.join()	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 sep = '-->' mystr = 'Hello' print(sep.join(mystr)) Output: `'H-->e-->l-->l-->o'`
string.ljust()	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.

Method	Description
	mystr = 'Hi' print(mystr.ljust(4)) Output: 'Hi ' Print(mystr.ljust(4, '-')) Output: 'Hi--' Print(mystr.ljust(2, '-')) Output: 'Hi'
string.lower()	Returns the copy of the original string wherein all the characters are converted to lowercase.
string.lstrip()	Returns a copy of the string by removing leading characters specified as an argument. mystr = ' Hello World ' mystr.lstrip() # removes leading spaces Output: 'Hello World '
string.partition()	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. mystr = 'Hello a World' print(mystr.partition(' ')) Output: ('hello', 'a ', 'world')
string.replace()	Returns a copy of the string where all occurrences of a substring are replaced with another substring. Syntax: str.replace(old, new, count) mystr = 'apples, bananas, apples, apples, cherries' print(mystr.replace('apples','lemons')) Output: lemons, bananas, lemons, lemons, cherries
string.rfind()	Returns the highest index of the specified substring (the last occurrence of the substring) in the given string. Syntax: str.replace(old, new, count) greet = 'Hello World!' print('Index of l: ', greet.rfind('l')) Output: Index of l: 9

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

Q1. Predict the output

```
word = "Python Programming"
print (word[0])
print (word[0:1])
print (word[0:3])
print (word[:3])
print (word[-3:])
print (word[3:])
print (word[:-3])
```

Sol.

```
P   #get one char of the word
P   #get one char of the word
Pyt #get the first three char
Pyt #get the first three char
Ing #get the last three char
hon Programming #get all but the three first char
Python Programm #get all but the three last character
```

Q2. Predict the output:

```
def string_length(str1):
    count = 0
    for char in str1:
        count += 1
    return count
print(string_length('Python Programming'))
```

Sol: 18

Q3. Predict the output

```
word = " Python Programming"

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

Sol.

```
a.['', 'Python', 'Programming']
b.False
c.True
d. ********** # prints ten *
```

Q4. Predict the output:

```
def fun_char(str1):
  char = str1[0]
  str1 = str1.replace(char, '$')
  str1 = char + str1[1:]
  return str1
print(fun_char('restart'))
```

Sol: resta$t

Q5. Predict the output

```
a = input("enter colour name= ")
if(a==red):
 print ("this colour indicates to stop your vehicle")
elif(a==yellow):
 print ("this colour indicates be ready to go")
elif(a==green):
 print ("this colour indicates your vehicle is ready to
go")
else:
 print ("colour name is not valid... thank you!!")
```

If you will get error then fix it.

Result.

```
 enter colour name= red
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-70-c60760ade985> in <module>
      1 a = input("enter colour name= ")
----> 2 if(a== red):
      3  print ("this colour indicates to stop your vehicle")
      4 elif(a == yellow):
      5  print ("this colour indicates be ready to go")

NameError: name 'red' is not defined
```

Sol.
```
if(a=='red'):
elif(a=='yellow'):
elif(a=='green'):
```

Q6. Predict the output:

```
def chars_mix_up(a, b):
  new_a = b[:2] + a[2:]
  new_b = a[:2] + b[2:]

  return new_a + ' ' + new_b
print(chars_mix('abc', '123'))
```

Sol. 12c ab3

Q7. Predict the output:

```
sub_string1 = 'ice'
sub_string2 = 'glue'
string1 = 'ice cream'
if sub_string1 in string1:
    print("There is " + sub_string1 + " in " + string1)
if sub_string2 not in string1:
    print("Phew! No " + sub_string2 + " in " + string1)
```

Sol.

```
There is ice in ice cream
Phew! No glue in ice cream
```

Q8. A user will provide a string, and your task is to count the frequency of characters in that string. Write a python program for this

E.g.,

Sample String: `Bennett.edu.in`
Result: {'B': 1, 'e': 3, 'n': 3, 't': 2, '.': 2, 'd': 1, 'u': 1, 'i': 1}

Sol.

```
def char_frequency(str1):
    dict = {}
    for n in str1:
        keys = dict.keys()
        if n in keys:
            dict[n] += 1
        else:
            dict[n] = 1
    return dict
print(char_frequency('Bennett.edu.in'))
```

Q9. A user will provide a string, and he asked you to perform the following task:

For the given string where all occurrences of its first char have been changed to '#', except the first char itself.

E.g.,

String1: `Python Programming`
Result: `Python #rogramming`

String2: `Python Python`
Result: `Python #ython`

Sol.

```
def change_char(str1):
  char = str1[0]
  str1 = str1.replace(char, '#')
  str1 = char + str1[1:]

  return str1

print(change_char('Python Programming'))
print(change_char('Python Python'))
print(change_char('restart result'))
```

Q10. Your course instructor asked you to write a program to remove duplicate characters of a given string.
E.g.,

String1: python exercises practice solution
Result: python exrcisalu

String2: BU4resource
Result: BU4resouc

```
Sol.

from collections import OrderedDict
def remove_duplicate(str1):
  return "".join(OrderedDict.fromkeys(str1))

print(remove_duplicate("python exercises practice
solution"))
print(remove_duplicate("BU4resource"))
```

Q11. Write a Python program that accepts a comma separated sequence of words as input and prints the unique words in sorted form (alphanumerically).

E.g.,

String1: red, white, black, red, green, black
Result: black, green, red, white, red

Sol.

```
items = input("Input comma separated sequence of words")
words = [word for word in items.split(",")]
print(",".join(sorted(list(set(words)))))
```

Q12. Write a Python program to count 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.'))
```