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.

1. Predict the output:

```
def string_length(str1):
    count = 0
    for char in str1:
        count += 1
    return count
print(string_length('bennettuniversity.edu.in'))
```

Sol: 24

2. Predict the output:

```
def string_both(str):
  if len(str) < 2:
    return ''
  return str[0:2] + str[-2:]
print(string_both('bennettresource'))
print(string_both('bennett'))
print(string_both('b'))
```

Sol:
```
bece
bett
```

3. Predict the output:

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

Sol: benn$ett

4. Predict the output:

```
def chars_mix(a, b):
  new_a = b[:2] + a[2:]
  new_b = a[:2] + b[2:]
  return new_a + ' ' + new_b
print(chars_mix('abc', 'pqr'))
```

Sol: pqc abr

5. Predict the output:

```
def add_string(str1):
  length = len(str1)
  if length > 2:
    if str1[-3:] == 'ing':
      str1 += 'ly'
    else:
      str1 += 'ing'
  return str1
print(add_string('ab'))
print(add_string('abc'))
print(add_string('string'))
```

Sol:

```
ab
abcing
stringly
```

6. Write a Python program 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.

Sample String :
'The lyrics is not that poor!'
'The lyrics is poor!'

Expected 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!'))
```

7. Predict the output:

```
def find_long(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_long(["PHP", "Exercises", "Backend"])
print(result[1])
print(result[0])
```

Sol:

```
Exercises
9
```

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

Sample Input:
abcd
12345
Sample Output:
dbca
52341

Sol:

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

9. Write a Python program to remove the n^{th} index character from a nonempty string.

Sample Input:
Python, 2
Python, 3

Sample output:
Pyhon
Pyton

Sol:

```
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))
```

10. Predict the output:

```
items = 'red, black, pink, green, black, green, pink, red'
words = [word for word in items.split(",")]
print(",".join(sorted(list(set(words)))))
```

Sol: black, green, pink, red,

11. 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("Python Exercises"))
print(case_str("Java"))
print(case_str("NumPy"))
```

Sol:

```
pYTHON eXERCISES
jAVA
nUMpY
```

12. Write a Python program to delete all occurrences of a specified character in a given string.

Sample Input:
all occurrences of a specified character in a given string

Sample Output:
ll occurrences of specified chrcter in given string

Sol:

```
def delete(str1, ch):
     result = str1.replace(ch, "")
     return(result)
str_text = "Delete all occurrences of a specified character in a
given string"
ch='a'
print(delete(str_text, ch))
```

13. Predict the output of the following code.

```
def inter(str1, str2):
    result = ""
    for ch in str1:
        if ch in str2 and not ch in result:
            result += ch
    return result

str1 = 'Python3'
str2 = 'Python2.7'

print(inter(str1, str2))
```

Sol: Python