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:

```
s = 'abacbdebfgbhhgbabddba'
print('\'b\' separated split -> {}'.format(s.split('b')))
```

Sol: 'b' separated split -> ['a', 'ac', 'de', 'fg', 'hhg', 'a', 'dd', 'a']

2. 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

3. Predict the output:

```
def is_palindrome(s):
  reverse = s[::-1]
  if (s == reverse):
    return True
  return False
s1 = 'racecar'
s2 = 'hippopotamus'
print('\'racecar\' a palindrome ->
{}'.format(is_palindrome(s1)))
print('\'hippopotamus\' a palindrome ->
{}'.format(is_palindrome(s2)))
```

Sol:
```
'racecar' a palindrome -> True
'hippopotamus' a palindrome -> False
```

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:

```
phrase = "This is a regular text"

#look for in 'This is', the rest of the phrase is not included
print(phrase.find('This', 0, 7))

#look for in 'This is a regular'
print(phrase.find('regular', 0, 17))

#look for in 'This is a regul'
print(phrase.find('a', 0, 15))
```

Sol:

```
0
10
8
```

6. Predict the output:

```
String1 = "{} {} {}".format('Bennett', 'For', 'CSE')
print("Print String in default order: ")
print(String1)

# Positional Formatting
String1 = "{1} {0} {2}".format('Bennett', 'For', 'CSE')
print("\nPrint String in Positional order: ")
print(String1)

# Keyword Formatting
String1 = "{l} {f} {g}".format(g='Bennett', f='For', l='CSE')
print("\nPrint String in order of Keywords: ")
print(String1)
```

Sol:

```
Print String in default order:
Bennett For CSE

Print String in Positional order:
For Bennett CSE

Print String in order of Keywords:
CSE For Bennett
```

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 count and display the vowels in a string.

Sample Input:
Hello this is string

Sample Output:
5
['e', 'o', 'i', 'i', 'i']

Sol:

```
def Check_Vow(string, vowels):
    final = [each for each in string if each in vowels]
    print(len(final))
    print(final)

string = input()
vowels = "AaEeIiOoUu"
Check_Vow(string, vowels);
```

9. Write a Python program to remove the nth 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. Write a python program to check the validity of a password, which must have minimum 8 characters, The alphabets must be between [a-z], At least one alphabet should be of

Upper Case [A-Z], At least 1 number or digit between [0-9] and at least 1 character from [_ or @ or $].

Input:
Name@bennett07
Hi@07

Output:
Valid Password
Invalid Password

```
l, u, p, d = 0, 0, 0, 0
s = input()
if (len(s) >= 8):
    for i in s:
        if (i.islower()):
            l+=1

        if (i.isupper()):
            u+=1

        if (i.isdigit()):
            d+=1

        if(i=='@'or i=='$' or i=='_'):
            p+=1
if (l>=1 and u>=1 and p>=1 and d>=1 and l+p+u+d==len(s)):
    print("Valid Password")
else:
    print("Invalid Password")
```