String Methods and Built-in Functions

1. len()

Returns the length of the given string passed as the argument.

```
>>> str="this is my book"
>>>len(str)
15
```

2. capitalize()

Return a string with its first character capitalized and the rest lower cased.

```
>>> str="this Is stRing example .... wow!!!" -
 >>> str.capitalize()
 'This is string example....wow!!!'
```

3. title ()

Return a title cased version of the string, where words start with an uppercase character and the remaining characters are lowercase. It will return unexpected result in cases where words have apostrophe etc.

```
>>> str="this is string example .... wow!!!"
 >>> str.title()
 'This Is String Example....Wow!!!'
 >>> str="this isn't a float example .... wow!!!"
 >>> str.title ()
 "This Isn'T A Float Example .... Wow!!!"
```

4. **upper ()**

Return a string with all the cased characters converted to uppercase.

```
>>> str="this is string example "
 >>> str.upper()
 'THIS IS STRING EXAMPLE....WOW!!!'
```

5. **lower()**

Return a string with all the cased characters converted to lowercase.

```
>>> str="THIS IS STRING EXAMPLE .... WOW!!!"
 >>> str.lower()
 'this is string example....wow!!!'
```

6. **count(sub[,start[,end]])**

Return the number of non-overlapping occurrences of sub-string sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

```
>>> str="this is string example .... wow!!!"
 >>> sub="i"
 >>> str.count(sub,4,40)
 2
 >>> sub="wow"
 >>> str.count(sub)
 1
```

7. **find(sub[,start[,end]])**

Return the lowest index in the string where sub-string sub is found, such that sub is contained in the slice s [start: end]. Optional arguments start and end are interpreted as in slice notation. Return -1, if sub is not found.

```
>>> str1 = "this is string example .... wow!!!"
>>> str2="exam"
>>> str1.find(str2)
15
>>> str1.find(str2,2,10)
15
>>> strl.find(str2,40)
-1
```

8. **index(sub[,start[,end]])**

Like find (), but raise ValueError when the sub-string is not found.

```
>>> str1="this is string example....wow!!!"
>>> str2="exam"
>>> str1.index(str2)
15
>>> str1.index(str2,10) .
15
>>> str1.index(str2,40)
Traceback (most recent call last):
File "<pyshell#38>", line 1, in str1.index(str2, 40)
ValueError: substring not found
```

9. isalnum()

Return True, if all characters in the string are alphanumeric, otherwise False is returned.

```
>>> str="this2009"
 >>> str.isalnum()
 True
 >>> str="this is string example....wow!!!"
 >>> str.isalnum()
 False
```

10. **isalpha()**

Return True, if all characters in the string are alphabetic, otherwise False is returned.

```
>>> str="this"
 >>> str.isalpha()
 True
 >>> str="this is string example .... wow!!!"
 >>> str.isalpha() ,
 False
```

11. isdigit()

Return True, if all characters in the string are digits, otherwise False is returned.

```
>>> str="this2009”
 >>> str.isdigit()
 False
```

```
>>> str="2009"
>>> str.isdigit()
True
```

12. isspace()

Return True, if there are only whitespace characters in the string, otherwise False is returned.

```
>>> str=" "
>>> str.isspace ()
True
>>> str="This is string example .... wow!!!"
>>> str.isspace ()
False
```

13. **islower()**

Return True, if all cased characters in the string are in lowercase and there is at least one cased character, False otherwise.

```
>>> str="THIS is string example....wow!!!"
>>> str.islower()
False
>>> str="this is string example .... wow!!!"
>>> str.islower()
True
>>> str="this2009"
>>> str.islower()
```

```
True
>>> str="2009"
>>> str.islower()
False
```

14. **isupper ()**

Return True, if all cased characters in the string are uppercase and there is at least one cased character, otherwise False is returned.

```
>>> str="THIS IS STRING EXAMPLE .... WOW!!!"
>>> str.isupper()
True
>>> str="THIS is string example .... wow!!!"
>>> str.isupper ()
False
```

15. **istitle ()**

Return True, if the string is title cased, otherwise False is returned.

```
>>> str="This Is String Example ... Wow!!!"
>>> str.istitle ()
True
>>> str="This is string example .... wow!!!"
>>> str.istitle()
False
```

16. strip(chars)

Return a string with the leading and trailing characters removed. The chars argument is a string specifying the set of characters to be removed. If omitted or None, the chars argument defaults to removing whitespaces.

```
>>> str="OOOOOOOthis is string example... .wow! !!0000000"
 >>> str.strip ('0')
 'this is string example....wow!!!'
```

17. **partition(sep)**

Split the string at the first occurrence of sep, and return a tuple containing the part before the separator, the separator itself, and the part after the separator.

```
>>> str=" this is string example .... wow!! ! "
 >>> str.partition('s')
 (' thi', 's', ' is string example .... wow!! ! ')
```

18. **split(sep)**

Return a list of the words from the string using sep as the delimiter string. If sep is not specified or None, any whitespace string is a separator.

```
>>> str='this is my book'
>>> str.split()
['this', 'is', 'my', 'book']
```

19. **swapcase()**

Return a copy of the string with reversed character case.

```
>>> str="This is string example .... WOW!!! "
 >>> str.swapcase()
 'tHIS IS STRING EXAMPLE ... .wow! !! '
```

20. **join(separater)**

Return a string in which the string elements have been joined by a separator.

```
>>> str="hello"
>>> t='*'
>>> t.join(str)
'h*e*l*l*o'
```

21. **replace(old,new)**

Return a string with all occurrences of sub-string old replaced by new.

```
>>> str="this is string example.... wow!! ! this is really string"
 >>> str.replace("is ","was")
 'thwas was string example.... wow!! ! thwas was really string'
```

Prepared by Rani Sastry, PGT
AECS-4, Mumbai