Python - Strings Operations

String object supports lot of methods. Lets explore some of them here.

In [ ]:

```
dir(str)  # list the methods of string
```

String methods doed not change the origianl string. If the change needs to be captured, it has to be assigned back to some variable.

In [ ]:

```
my_string = "Python is beautiful!"
```

In [ ]:

```
my_string.lower()  #converts to lowercase
```

In [ ]:

```
my_string  # original string is not altered
```

In [ ]:

```
lower_string = my_string.lower() # lower the string and assign the new string to a variable
lower_string
```

String methods

lower() - converts the every character of string in lower case

In [ ]:

```
my_string.lower()
```

upper() - converts the every character of string in upper case

In [ ]:

```
my_string.upper()
```

In [ ]:

```
stmt = "Chennai Super Kings are going to win IPL this time."
print("Original stmt : ",stmt)

mod_stmt = stmt.upper()
print("Modified stmt : ",mod_stmt)

mod_stmt = stmt.lower()
print("Modified stmt again : ",mod_stmt)
```

islower() - determines whether the character is in lowercase or not

In [ ]:

```
my_string.islower()
```

isupper() - determines whether the character is in uppercase or not

In [ ]:

```
my_string.isupper()
```

count() - counts the number of occurances of characters in string

In [ ]:

```
my_string.count('t')
```

index - returns the index of given set of characters

In [ ]:

```
my_string.index('t')
```

In [ ]:

```
my_string.index('fu')
```

isalpha() - determines whether a character of string is letter or not

In [ ]:

```
my_string[0].isalpha()
```

isdigit() - determines whether a character of string is digit or not

In [ ]:

```
my_string[0].isdigit()
```

isnumeric() - returns true if the string contains all number values in it

In [ ]:

```
my_string.isnumeric()
```

In [ ]:

```
num_string = "123"
num_string.isnumeric()
```

String stripping

Sometimes the strings comes with white spaces attached at both ends. The characters from the left and right side of string can be removed with the strip function.

In [ ]:

```
stmt = "Chennai Super Kings are going to win IPL this time.    "
print(stmt)
print(stmt.rstrip())  #remove the empty spaces at right side
```

In [ ]:

```
stmt = "     Chennai Super Kings are going to win IPL this time."
print(stmt)
print(stmt.lstrip())  #remove the empty spaces at left side
```

In [ ]:

```
stmt = "     Chennai Super Kings are going to win IPL this time.   "
print(stmt)
print(stmt.strip())  #remove the empty spaces from both side
```

In [ ]:

```
stmt = "$$$$Chennai Super Kings are going to win IPL this time."
print(stmt)
print(stmt.lstrip("$"))  #Splitting character can also be specified
```

Substrings

The strings which are part of string are substrings. For example, 'beautiful' is substring of string 'python is beautiful'. There are several functions to deal with substrings.

find(string_to_be_searched) - returns the index of place where the substring is present otherwise -1

In [ ]:

```
my_string = 'python is beautiful'
```

In [ ]:

```
my_string.find('is')
```

In [ ]:

```
my_string.find('are')
```

startswith(string_to_be_serached) - returns True if string starts with given substring, othrewise False

In [ ]:

```
my_string.startswith('python')
```

In [ ]:

```
my_string.startswith('Python')
```

endswith(string_to_be_serached) - returns True if string starts with given substring, othrewise False

In [ ]:

```
my_string.endswith('beautiful')
```

In [ ]:

```
my_string.endswith('Beautiful')
```

replace(value1, value2) - replaces each occurance of string value1 with string value2

In [ ]:

```
my_string.replace('t', '#')
```

String Splitting

In [ ]:

```
help(str.split)
```

In [ ]:

```
my_string = "Python is beautiful!"
```

In [ ]:

```
my_string.split()   #split using default delimiter i.e. white space
```

In [ ]:

```
splitted_string = my_string.split()   #split using default delimiter i.e. white space
```

In [ ]:

```
type(splitted_string)  # is list of values
```

In [ ]:

```
splitted_string[0]  # aceess first part of splitted string
```

In [ ]:

```
splitted_string[1]  # aceess second part of splitted string
```

In [ ]:

```
splitted_string[2]  # aceess third part of splitted string
```

In [ ]:

```
my_string.split(' is ')   #split using user defined delimiter i.e. ' is '
```

Exercise

Q1. Ask user to input two strings - first string is statement in which the second string needs to be looked upon. Then using the string methods determine whether second string is present in first string or not. Inform the user about the result

In [ ]:

```
#Try this out
```

Q2. Ask the user to input a string , then output the same string in lowercase, uppercase and reverse manner.

In [ ]:

```
#Try this out
```

Q3. Write a code snippet thta asks a user for their name and print the name in following pattern **B Bi Bil Bill**

In [ ]:

```
#Try this out
```

In [ ]:

```
```