String functions: String functions are available in python standard module. These are always available to use. Import statement is not required for using these functions.

Sr. No	Function with Example
11	

Python String isalnum() Method

Python isalnum() method checks whether the all characters of the string is alphanumeric or not. A character which is either a letter or a number is known as alphanumeric. It does not allow special chars even spaces.

Syntax: : isalnum()

Parameters: : No parameter is required.

Return: : It Return: s either True or False.

Python String isalnum() Method Example 1

```
# Python isalnum() function example

str = "Welcome"

str2 = str.isalnum()

print(str2)
```

Output:

```
True
```

Python String isalnum() Method Example 2

If there is a space anywhere in the string, it Return: s False. See the example below.

```
# Python isalnum() function example

str = "Welcome"

str2 = str.isalnum()

print(str2)
```

Output:

```
False
```

Python String isalnum() Method Example 3

```
# Python isalnum() function example

str = "Welcome123" # True
str3 = "Welcome 123" # False
```

```
str2 = str.isalnum()
str4 = str3.isalnum()

print(str2)
print(str4)
```

Output:

```
True
False
```

Python String isalnum() Method Example 4

It Return: s True even the string is full of digits. See the example.

```
str = "123456"
str2 = str.isalnum()
print(str2)
```

Output:True

12

Python String isalpha() Method

Python isalpha() method Return: s true if all characters in the string are alphabetic. It Return: s False if the characters are not alphabetic. It Return: s either True or False.

Syntax: isalpha()

Parameters: No parameter is required.

Return: It Return: s either True or False.

Let's see some examples of isalpha() method to understand it's functionalities.

Python String isalpha() Method Example 1

```
# Python isalpha() method example
str = "Dpssonepat"
str2 = str.isalpha()
print(str2)
```

Output:

```
True
```

Python String isalpha() Method Example 2

```
# Python isalpha() method example
str = "Welcome to the Dpssonepat"
str2 = str.isalpha()
print(str2)
```

Output:

```
False
```

Python String isalpha() Method Example 3

```
# Python isalpha() method example
str = "Dpssonepat"
if str.isalpha() == True:
```

```
    print("String contains alphabets")
else:
    print("Stringn contains other chars too.")
```

Output:

```
String contains alphabets
```

13

Python String isdecimal() Method

Python **isdecimal()** method checks whether all the characters in the string are decimal or not. Decimal characters are those have base 10. This method Return: s boolean either true or false.

Syntax: isdecimal()

Parameters: No parameter is required.

Return: It Return: s either True or False.

Let's see some examples of isdecimal() method to understand it's functionalities.

Python String isdecimal() Method Example 1

A simple example of isdecimal() method to check whether the string contains decimal value.

```
# Python isdecimal() method example

str = "Dpssonepat"
str2 = str.isdecimal()
print(str2)
```

Output:

```
False
```

Python String isdecimal() Method Example 2

Let's try to check floating value and see the outcome. It will Return: False if string is not decimal.

```
# Python isdecimal() method example

str = "123"     # True
str3 = "2.50"   # False

str2 = str.isdecimal()
str4 = str3.isdecimal()

print(str2)
print(str4)
```

Output:

```
True
False
```

Python String isdecimal() Method Example 3

Here, we are checking special chars also.

```
# Python isdecimal() method example
```

```
str = "123"     # True
str3 = "@#$"   # False

str2 = str.isdecimal()
str4 = str3.isdecimal()

print(str2)
print(str4)
```

Output:

```
True
False
```

14

Python String isdigit() Method

Python **isdigit()** method Return: s True if all the characters in the string are digits. It Return: s False if no character is digit in the string.

Syntax: **isdigit()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of isdigit() method to understand it's functionalities.

Python String isdigit() Method Example 1

A simple example of digit testing using isdigit() method.

```
# Python isdigit() method example
str = '12345'
str2 = str.isdigit()
print(str2)
```

Output:

```
True
```

Python String isdigit() Method Example 2

It Return: s true only if all the characters are digit. See the example below.

```
# Python isdigit() method example

str = "12345"
str3 = "120-2569-854"

str2 = str.isdigit()
str4 = str3.isdigit()

print(str2)
print(str4)
```

Output:

```
True
False
```

Python String isdigit() Method Example 3

We can use it in programming to check whether a string contains digits or not.

```
# Python isdigit() method example

str = "123!@#$"
if str.isdigit() == True:
   print("String is digit")
else:
   print("String is not digit")
```

Output:

```
String is not digit
```

15

Python String isidentifier() Method

Python **isidentifier()** method is used to check whether a string is a valid identifier or not. It Return: s True if the string is a valid identifier otherwise Return: s False.
Python language has own identifier definition which is used by this method.

Syntax: **isidentifier()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of isidentifier() method to understand it's functionalities.

Python String isidentifier() Method Example 1

A simple Example to apply isidentifier() method, it Return: s True. See the example.

```
# Python isidentifier() method example

str = "abcdef"
str2 = str.isidentifier()
print(str2)
```

Output:

```
True
```

Python String isidentifier() Method Example 2

Here, we have created variety of identifiers. Some Return: s True and some False. See the example below.

```
# Python isidentifier() method example

str = "abcdef"
str2 = "20xyz"
str3 = "$abra"

str4 = str.isidentifier()
str5 = str2.isidentifier()
str6 = str3.isidentifier()
```

print(str4)
print(str5)
print(str6)

Output:

```
True
False
False
```

Python String isidentifier() Method Example 3

This method is useful in decision making therfore can be used with if statement.

```
# Python isidentifier() method example

str = "abcdef"

if str.isidentifier() == True:
   print("It is an identifier")
else:
   print("It is not identifier")
str2 = "$xyz"
if str2.isidentifier() == True:
   print("It is an identifier")
else:
   print("It is not identifier")
```

16

Python String islower() Method

Python string **islower()** method Return: s True if all characters in the string are in lowercase. It Return: s False if not in lowercase.

Syntax: **islower()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of islower() method to understand it's functionalities.

Python String islower() Method Example 1

A simple example to understand how to apply this method. It Return: s true, see the example below.

```
# Python islower() method example
str = "Dpssonepat"
str2 = str.islower()
print(str2)
```

Output: True

Python String islower() Method Example 2

It Return: s False if any single char found in other case than lowercase. See the example below.

```
# Python islower() method example
str = "Welcome To Dpssonepat"
str2 = str.islower()
```

print(str2)

Output: False

Python String islower() Method Example 3

String can have digits also and this method works in letters case and ignores digits. So it Return: s True, see the example below.

```
# Python islower() method example
str = "hi, my contact is 9896******"
str2 = str.islower()
print(str2)
```

Output: True

17

Python String isnumeric() Method

Python **isnumeric()** method checks whether all the characters of the string are numeric characters or not. It Return: s True if all the characters are true, otherwise Return: s False.
Numeric characters include digit characters and all the characters which have the Unicode numeric value property.

Syntax: **isnumeric()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of isnumeric() method to understand it's functionalities.

Python String isnumeric() Method Example 1

Here, a simple example is created to check the string is numeric or not.

```
# Python isnumeric() method example

str = "12345"
str2 = str.isnumeric()
print(str2)
```

Output: True

Python String isnumeric() Method Example 2

Let's test it on the non-numeric string, see it Return: s False.

```
# Python isnumeric() method example
str = "Dpssonepat12345"
str2 = str.isnumeric()
print(str2)
```

Output:

False

Python String isnumeric() Method Example 3

See a senario where and how we can apply isnumeric() method in python programming

```
# Python isnumeric() method example
str = "123452500" # True
if str.isnumeric() == True:
```

```
    print("Numeric")
else:
    print("Not numeric")
str2 = "123-4525-00" # False
if str2.isnumeric() == True:
    print("Numeric")
else:
    print("Not numeric")
```

Output:

```
Numeric
Not numeric
```

18

Python String isprintable() Method

Python **isprintable()** method Return: s True if all characters in the string are printable or the string is empty. It Return: s False if any character in the string is Nonprintable.

Syntax: isprintable()

Parameters: No parameter is required.

Return: It Return: s either True or False.

Let's see some examples of isprintable() method to understand it's functionalities.

Python String isprintable() Method Example 1

It is a simple example that prints True, if the string is printable. See the example below.

```
# Python isprintable() method example
str = "Hello, Dpssonepat"
str2 = str.isprintable()
print(str2)
```

Output:

```
True
```

Python String isprintable() Method Example 2

Here, we are using some other characters like newline and tabs in the string. See the results of the example.

```
# Python isprintable() method example
str = "Hello, Dpssonepat"
str2 = "Learn Java here\n"
str3 = "\t Python is a programming language"
str4 = str.isprintable()
str5 = str2.isprintable()
str6 = str3.isprintable()
print(str4)
print(str5)
print(str6)
```

Output:

```
True
False
False
```

Python String isprintable() Method Example 3

Testing different-different string values which contains special chars too. In case of special chars, method Return: s False.

```
# Python isprintable() method example

str = "Hello, Dpssonepat"
if str.isprintable() == True:
    print("It is printable")
else:
    print("Not printable")
str = "$Hello@Dpssonepat#"    # Special Chars
if str.isprintable() == True:
    print("It is printable")
else:
    print("Not printable")
str = "Hello\nDpssonepat"
if str.isprintable() == True:
    print("It is printable")
else:
    print("Not printable")
```

Output:

```
It is printable
It is printable
Not printable
```

19

Python String isspace() Method

Python **isspace()** method is used to check space in the string. It Return: a true if there are only whitespace characters in the string. Otherwise it Return: s false. Space, newline, and tabs etc are known as whitespace characters and are defined in the Unicode character database as **Other or Separator**.

Syntax: **isspace()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of isspace() method to understand it's functionalities.

Python String isspace() Method Example 1

```
# Python isspace() method example
str = " " # empty string
str2 = str.isspace()
print(str2)
```

Output:

```
True
```

Python String isspace() Method Example 2

```
# Python isspace() method example

str = "ab cd ef" # string contains spaces
str2 = str.isspace()
print(str2)
```

Output:

```
False
```

Python String isspace() Method Example 3

isspace() method Return: s true for all whitespaces like:

- ' ' - Space
- '\t' - Horizontal tab
- '\n' - Newline
- '\v' - Vertical tab
- '\f' - Feed
- '\r' - Carriage Return:

```
# Python isspace() method example
str = " " # string contains space
if str.isspace() == True:
   print("It contains space")
else:
   print("Not space")
str = "ab cd ef \n"
if str.isspace() == True:
   print("It contains space")
else:
   print("Not space")
str = "\t \r \n"
if str.isspace() == True:
   print("It contains space")
else:
   print("Not space")
```

Output:

```
It contains space
Not space
It contains space
```

20

Python String istitle() Method

Python **istitle()** method Return: s True if the string is a titlecased string. Otherwise Return: s False.

Syntax: **istitle()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of istitle() method to understand it's functionalities.

Python String istitle() Method Example 1

It is a simple example to understand

```
# Python istitle() method example

str = "Welcome To Dpssonepat"
str2 = str.istitle()
print(str2)
```

Python String istitle() Method Example 2

```
# Python istitle() method example
str = "Welcome To Dpssonepat" # True
str2 = str.istitle()
print(str2)
str = "hello Dpssonepat"    # False
str2 = str.istitle()
print(str2)
```

Output:

```
True
False
```

Python String istitle() Method Example 3

```
# Python istitle() method example
str = "ab cd ef"
if str.istitle() == True:
   print("In title case")
else:
   print("Not in tit0le case")
str = "Ab Cd Ef"
if str.istitle() == True:
   print("In title case")
else:
   print("Not in title case")
str = "1b 2d 3f"
if str.istitle() == True:
   print("In title case")
else:
   print("Not in title case")
```

Output:

```
Not in title case
In title case
Not in title case
```

21

Python String isupper() Method

Python **isupper()** method Return: s True if all characters in the string are in uppercase. It Return: s False if characters are not in uppercase.

Syntax: **isupper()**

Parameters: **No parameter is required.**

Return: **It Return: s either True or False.**

Let's see some examples of isupper() method to understand it's functionalities.

Python String isupper() Method Example 1

```
# Python isupper() method example

str = "WELCOME TO DPSSONEPAT"
str2 = str.isupper()
print(str2)
```

Output:

```
True
```

Python String isupper() Method Example 2

```
# Python isupper() method example
str = "WELCOME TO DPSSONEPAT"
str2 = str.isupper()
print(str2)
str3 = "Learn Java Here."
str4 = str3.isupper()
print(str4)
```

Output:

```
True
False
```

Python String isupper() Method Example 3

```
# Python isupper() method example
str = "WELCOME TO DPSSONEPAT"
str2 = str.isupper()
print(str2)
str3 = "WELCOME To DPSSONEPAT."
str4 = str3.isupper()
print(str4)
str5 = "123 @#$ -JAVA."
str6 = str5.isupper()
print(str6)
```

Output:

```
True
False
True
```

22

Python String ljust() Method

Python **ljust()** method left justify the string and fill the remaining spaces with fillchars. This method Return: s a new string justified left and filled with fillchars.

Syntax: **ljust(width,[fillchar])**

Parameters: width **: width of the given string.**

- **fillchar** : characters to fill the remaining space in the string. It is **optional**.

Return: **It Return: s a left justified string.**

Let's see some examples of ljust() method to understand it's functionalities.

Python String ljust() Method Example 1

```
# Python ljust() method example
str = 'Dpssonepat'
str = str.ljust(20)
print(str)
```

Output:

D	p	s	s	o	n	e	p	a	t										

Python String ljust() Method Example 2

```
# Python ljust() method example
str = 'Dpssonepat'
str = str.ljust(15,"$")
print(str)
```

Output:

D	p	s	s	o	n	e	p	a	t	$	$	$	$	$

23

Python String lower() Method

Python **lower()** method Return: s a copy of the string after converting all the characters into lowercase.

Syntax: **lower()**

Parameters: **No parameter.**

Return: **It Return: s a lowercase string.**

Let's see some examples of lower() method to understand it's functionalities.

Python String lower() Method Example 1

See a simple example in which string is converting into lowercase.

```
# Python lower() method example

str = "Dpssonepat"

str = str.lower()

print(str)
```

Output:

```
Dpssonepat
```

Python String lower() Method Example 2

String can be varried and has any case of letters. The method will Return: a new string of lowercase.

```
# Python lower() method example
```

```
str = "Welcome To Dpssonepat, WWW.DPSSONEPAT.COM"
str = str.lower()
print(str)
```

Output:

```
welcome to dpssonepat, www.dpssonepat.com
```

Python String lower() Method Example 3

We can test it whether it Return: ing lowercase by using if statement. See the example below.

```
# Python lower() method example
str = "DPSSONEPAT"
if str.lower() == "Dpssonepat":
   print("lowercase")
else:
   print("not lowercase")
```

Output:

```
lowercase
```

24

Python String lstrip() Method

Python **lstrip()** method is used to remove all leading characters from the string. It takes a char type parameter which is optional. If parameter is not provided, it removes all the leading spaces from the string.

Syntax: **lstrip([chars])**

Parameters: chars (optional) **: A list of chars**

Return: **It Return: s a string.**

Let's see some examples of **lstrip()** method to understand it's functionality

Python String lstrip() Method Example 1

A simple program to understand use of lstrip method. See how it removes leading spaces.

```
# Python lstrip() method example
str =  "  Dpssonepat  "
str2 = str.lstrip()
print(str)
print(str2)
```

Output:

```
          Dpssonepat
Dpssonepat
```

Python String lstrip() Method Example 2

Here, we are adding some special chars to the string and applying lstrip to remove leading chars.

```
# Python lstrip() method example


str =  "$$$$-Dpssonepat-$$$$"
str2 = str.lstrip('$')
print(str)
```

print(str2)

Output:

```
$$$$-Dpssonepat-$$$$
-Dpssonepat-$$$$
```

Python String lstrip() Method Example 3

```
# Python lstrip() method example
str =  "http://www.Dpssonepat.com"

str2 = str.lstrip('http://w.')
print(str)
print(str2)
```

Output:

```
http://www.Dpssonepat.com
Dpssonepat.com
```

25

Python String partition() Method

Python **partition()** method splits the string from the string specified in parameter. It splits the string from at the first occurrence of *parameter* and Return: s a tuple. The tuple contains the three parts before the separator, the separator itself, and the part after the separator.

It Return: s an empty tuple having seperator only, if the seperator not found.

Syntax: **partition(sep)**

Parameters: sep**: A string parameter which separates the string.**

Return: **It Return: s a tuple, A 3-Tuple.**

Let's see some examples of **partition(sep)** method to understand it's functionality.

Python String partition() Method Example 1

First, let's see simple use of partition method.

```
# Python partition() method example

str = "Java is a programming language"

str2 = str.partition("is")

print(str2)
# when seperate from the start
str2 = str.partition("Java")
print(str2)
# when seperate is the end
str2 = str.partition("language")
print(str2)
# when seperater is a substring
```

```
str2 = str.partition("av")
print(str2)
```

Python String partition() Method Example 2

if the separator is not found, It Return: s a tuple containing string itself and two empty strings. See the example below.

```
# Python partition() method example

str = "Java is a programming language"

str2 = str.partition("not")

print(str2)
```

Output:

```
('Java is a programming language', '', '')
```

26

Python String replace() Method

Return: a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

Syntax: **replace(old,** new**[, count])**

Parameters: old **: An old string which will be replaced.**

new : New string which will replace the old string.

count : The number of times to process the replace.

Return: **It Return: s string**

Let's see some examples of **replace()** method to understand it's functionality.

Python String replace() Method Example 1

```
# Python replace() method example
str = "Java is a programming language"
str2 = str.replace("Java","C")

print("Old String: \n",str)
print("New String: \n",str2)
```

Output:

```
Old String:
 Java is a programming language
New String:
 C is a programming language
```

Python String replace() Method Example 2

```
# Python replace() method example

str = "Java C C# Java Php Python Java"
```

```
str2 = str.replace("Java","C#") # replaces all the occurences

print("Old String: \n",str)
print("New String: \n",str2)

str2 = str.replace("Java","C#",1) # replaces first occurance only

print("\n Old String: \n",str)
print("New String: \n",str2)
```

Output:

```
Old String:
 Java C C# Java Php Python Java
New String:
 C# C C# C# Php Python C#

 Old String:
 Java C C# Java Php Python Java
New String:
 C# C C# Java Php Python Java
```

Python String replace() Method Example 3

```
# Python replace() method example

str = "Apple is a fruit"

str2 = str.replace(str,"Tomato is also a fruit")

print(str2)
```

Output:

```
Tomato is also a fruit
```

27

Python String split() Method

Python **split()** method splits the string into a comma separated list. It separates string based on the separator delimiter. This method takes two Parameters: and both are optional. It is described below.

Syntax: **split(sep=None, maxsplit=-1)**

Parameters: sep**: A string parameter acts as a seperator.**

maxsplit: Number of times split perfome.

Return: **It Return: s a comma separated list.**

Let's see some examples of split() method to understand it's functionality.

Python String split() Method Example 1

This is a simple example to understand the usage of split() method. No parameter is given, by default whitespaces work as separator. See the example below.

```
# Python split() method example

str = "Java is a programming language"
str2 = str.split()
print(str)
print(str2)
```

Output:

```
Java is a programming language
['Java', 'is', 'a', 'programming', 'language']
```

Python String split() Method Example 2

Let's pass a parameter separator to the method, now it will separate the string based on the separator. See the example below.

```
# Python split() method example
str = "Java is a programming language"
str2 = str.split('Java')
print(str2)
```

Output:

```
['', ' is a programming language']
```

Python String rsplit() Method Example 3

The string is splited each time when a is occurred. See the example below.

```
# Python split() method example
str = "Java is a programming language"
str2 = str.split('a')
print(str)
print(str2)
```

Output:

```
Java is a programming language
['J', 'v', ' is ', ' progr', 'mming l', 'ngu', 'ge']
```

Python String split() Method Example 4

Along with separator, we can also pass maxsplit value. The maxsplit is used to set the number of times to split.

```
# Python split() method example
str = "Java is a programming language"
str2 = str.split('a',1)
print(str2)
  str2 = str.split('a',3)
print(str2)
```

Output:

```
['J', 'va is a programming language']
['J', 'v', ' is ', ' programming language']
```

28

Python String splitlines() Method

Python **splitlines()** method splits the string based on the lines. It breaks the string at line boundaries and Return: s a list of splitted strings. Line breakers can be a new line (\n), carriage Return: (\r) etc. A table of line breakers are given below which split the string.

This method splits on the given line boundaries.

Representation	Description
\n	Line Feed
\r	Carriage Return:
\r\n	Carriage Return: + Line Feed
\v or \x0b	Line Tabulation
\f or \x0c	Form Feed
\x1c	File Separator
\x1d	Group Separator
\x1e	Record Separator
\x85	Next Line (C1 Control Code)
\u2028	Line Separator
\u2029	Paragraph Separator

Syntax: **splitlines([keepends])**

Parameters: keepends**: It is a boolean value which can be either True or False. It is** optional**.**

Return: **It Return: s a comma separated list of lines.**

Let's see some examples of splitlines() method to understand it's functionality.

Python String splitlines() Method Example 1

```
# Python splitlines() method example

str = "Java is a programming language"

str2 = str.splitlines() # Return: s a list having single element
print(str)
print(str2)
str = "Java \n is a programming \r language"
str2 = str.splitlines() # Return: s a list having splitted elements
print(str2)
```

Output:

```
Java is a programming language
['Java is a programming language']
['Java ', ' is a programming ', ' language']
```

Python String splitlines() Method Example 2

Passing True to the method which causes to include line breakers into the list of string. See the example below.

```
# Python splitlines() method example
```

```
str = "Java \n is a programming \r language"

str2 = str.splitlines(True) # Return: s a list having splitted elements
print(str2)
```

Output:

```
['Java \n', ' is a programming \r', ' language']
```

Python String splitlines() Method Example 3

```
# Python splitlines() method example

str = "Java \n is a programming \r language for \r\n  software development"

str2 = str.splitlines() # Return: s a list having splitted elements

print(str2)
# getting back list to string
print("".join(str2)) # now it does not contain any line breaker character
```

Output:

```
['Java ', ' is a programming ', ' language for ', '  software development']
Java  is a programming  language for   software development
```

29

Python String startswith() Method

Python startswith() method Return: s either True or False. It Return: s True if the string starts with the prefix, otherwise False. It takes two Parameters: start and end. Start is a starting index from where searching starts and end index is where searching stops.

Syntax: **startswith(prefix[, start[, end]])**

Parameters:

prefix : A string which is to be checked.

start : Start index from where searching starts.

end : End index till there searching performs.

Both start and end are optional Parameters: .

Return: **It Return: s boolean value either True or False.**

Let's see some examples of startswith() method to understand it's functionality.

Python String startswith() Method Example 1

Let's first create a simple example which prints True if the string starts with the prefix.

```
# Python String startswith() method
# Declaring variable
str = "Hello Dpssonepat"
```

```
str2 = str.startswith("Hello")

print (str2)
```

Output:

```
True
```

Python String startswith() Method Example 2

If the string does not start with prefix, the method Return: s False. See the example below

```
# Python String startswith() method
# Declaring variable
str = "Hello Dpssonepat"

str2 = str.startswith("Java") # False

print (str2)
```

Output:

```
False
```

Python String startswith() Method Example 3

This method takes three Parameters: . The start and end index are optional. Here, we are passing start index only.

```
# Python String startswith() method
# Declaring variable
str = "Hello Dpssonepat"

str2 = str.startswith("Java",6)

print (str2)
```

Output:

```
True
```

Python String startswith() Method Example 4

It Return: s true if the string lie between start and end index starts from the prefix. An example is created to describe the process.

```
# Python String startswith() method
# Declaring variable
str = "Hello Dpssonepat"

str2 = str.startswith("Java",6,10)

print (str2)
str2 = str.startswith("Java",8,12)

print (str2)
```

Output:

True
False

30

Python String swapcase() Method

Python swapcase() method converts case of the string characters from uppercase to lowercase and vice versa. It does not require any parameter and Return: s a string after case conversion.

Syntax: **swapcase()**

Parameters: **No Parameter**

Return: **It Return: s a string.**

Let's see some examples of swapcase() method to understand it's functionality.

Python String swapcase() Method Example 1

It is a simple example to convert uppercase to lowercase using the swapcase() method.

```
# Python String swapcase() method
# Declaring variable
str = "HELLO DPSSONEPAT"

str2 = str.swapcase()

print (str2)
```

Output:

```
hello Dpssonepat
```

Python String swapcase() Method Example 2

This example describes conversion from lowercase to uppercase.

```
# Python String swapcase() method
# Declaring variable
str = "hello Dpssonepat"

str2 = str.swapcase()

print (str2)
```

Output:

```
HELLO DPSSONEPAT
```

Python String swapcase() Method Example 3

If the string is in camelcase, the output of this method also would be in camelcase. All the lowercase will be converted to uppercase and vice versa.

```
# Python String swapcase() method
# Declaring variable
str = "Hello DpSsOnepat"

str2 = str.swapcase()

print (str2)
```

	Output: hELLO dPsSoNEPAT
31	## Python String title() Method Python title() method is used to convert the string into the title-case i.e., The string soNepaT will be converted to Sonepat. **Syntax:** **title()** **Parameters:** **No Parameters:** **Return:** **It Return: s a string.** Let's see some examples of title() method to understand it's functionality. ### Python String title() Method Example 1 str = "hello dPS sonEpat" result=str.title() print (result) Output: Hello Dps Sonepat

33	## Python String upper() Method
	
	Python upper() method converts all the character to uppercase and Return: s a uppercase string.
	
	Syntax: **upper()**
	
	Parameters: **No Parameters:**
	
	Return: **It Return: s a string.**
	
	Let's see some examples of upper() method to understand it's functionality.
	
	### Python String upper() Method Example 1
	
	First see a simple example of upper() method. This method Return: s a uppercase string.
	# Python upper() method
	# Declaring table and variables
	str = "Hello Dpssonepat"
	str2 = str.upper()
	print(str2)
	
	Output:HELLO DPSSONEPAT
	
	### Python String upper() Method Example 2
	
	See, how can we use this method in programming. The below example converts all the elements of the list into uppercase. See the example.
	# Python upper() method
	# Declaring variables

```
list = ["irfan","sohan","mohan"]
for  l in list:
   print(l.upper())
```

Output:

```
 IRFAN
SOHAN
MOHAN
```

Python String upper() Method Example 3

An example that converts all the string into uppercase which starts with vowels. See the example below.

```
# Python upper() method
# Declaring variables
names = ["irfan","sohan","aman","mohan"]
vowels = ['a','e','i','o','u']
for  l in names:
   for v in vowels:
      if(l.startswith(v)):
         print(l.upper())
```

Output:

```
IRFAN
AMAN
```

34

Python String zfill() Method

Python zfill() method fills the string at left with 0 digit to make a string of length width. It Return: s a string contains a sign prefix + or - before the 0 digit.

It Return: s original length string if the width is less than string length.

Syntax: **zfill(width)**

Parameters: width **: length of the string.**

Return: **It Return: s a string.**

Let's see some examples of zfill() method to understand it's functionality.

Python String zfill() Method Example 1

Let's see an example which has width greater than string length. It Return: s a new string containing 0 to the left and total length is equal to width.

```
# Python zfill(width) method
# Declaring variables
text = "Zfill Example"

str2 = text.zfill(20)

print(str2)
```

Output:

```
0000000Zfill Example
```

Python String zfill() Method Example 2

Here, we are passing width less than the string length therefore it Return: s the original string without filling zeros to the left.

```
# Python zfill(width) method
# Declaring variables
text = "Zfill Example"

str2 = text.zfill(5)

print(str2)
```

Output:

```
Zfill Example
```

Python String zfill() Method Example 3

This example describes the sign either + or - includes before the zero fills to the left of the string. See every value is filled after prefixing (+ or -) sign.

```
# Python zfill(width) method
# Declaring variables
val = "+100"
val2 = "-200"
val3 = "--Rohan--"

str2 = val.zfill(10)
str3 = val2.zfill(10)
str4 = val3.zfill(10)



print(str2)
print(str3)
print(str4)
```

Output:

```
+000000100
-000000200
-0-Rohan--
```