Department of Computer Science and Engineering
PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving
UE23CS151A

Lecture #42
String and it's Operations

By,
Prof. Sindhu R Pai,
Anchor, PCPS - 2023
Assistant Professor
Dept. of CSE, PESU

Verified by,
PCPS Team - 2023

Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Centers, Former Chairperson, CSE, PES University)
Prof. Chitra G M (Asst. Prof, Dept. of CSE, PCPS Anchor – 2022)

Functions/Methods of Strings in Python

As string is immutable, applying any function on it always returns a new string or returns a Boolean value.

Specific functions of Strings:

Use dir() function to **display the list of functions a type supports**. Can use help() on any of these to know its job and it' usage. Sample is shown below.

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
 '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '
__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne
__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__s
izeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endsw
ith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecim
al', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper'
, 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'repla
ce', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith'
, 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

```
>>> help(str.capitalize)
Help on method_descriptor:

capitalize(self, /)
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the rest lower
    case.

>>>
```

capitalize(): Returns a new string in which the first character will be upper case and all others will be in lowercase.

```
>>> s1 = "python ProgramMing class is from 2nd Sep, 2023"
>>> s2 = s1.capitalize()
>>> s1
'python ProgramMing class is from 2nd Sep, 2023'
>>> s2
'Python programming class is from 2nd sep, 2023'
>>>
```

title(): Returns a version of the string in which words start with uppercased characters and all remaining cased characters have lower case.

```
>>> s1
'python ProgramMing class is from 2nd Sep, 2023'
>>> s2 = s1.title()
>>> s2
'Python Programming Class Is From 2Nd Sep, 2023'
>>> s1
```

```
'python ProgramMing class is from 2nd Sep, 2023'
```

upper() and lower(): Returns a string in all uppercase and in all lower case respectively.

```
>>> s1
'python ProgramMing class is from 2nd Sep, 2023'
>>> s2 = s1.upper()
>>> s3 = s1.lower()
>>> s2
'PYTHON PROGRAMMING CLASS IS FROM 2ND SEP, 2023'
>>> s3
'python programming class is from 2nd sep, 2023'
>>>
```

swapcase(): Convert uppercase characters to lowercase and lowercase characters to uppercase and returns a new string.

```
>>> s1
'python ProgramMing class is from 2nd Sep, 2023'
>>> s2 = s1.swapcase()
>>> s2
'PYTHON pROGRAMmING CLASS IS FROM 2ND sEP, 2023'
>>> s1
'python ProgramMing class is from 2nd Sep, 2023'
>>>
```

isupper() and islower(): Returns a Boolean value based on whether all the cased characters in the string are in upper case or all in lower case respectively.

```
>>> s1 = "pyth on"
>>> s1.upper()
'PYTH ON'
>>> s1.isupper()
False
>>> s1.islower()
True
>>> s2 = "PY THON"
>>> s2.isupper()
True
>>> s2.islower()
False
>>>
```

```
s3 = "python 123"
>>> s3.islower()
True
>>> s4 = "pYThon 123"
>>> s4.islower()
False
>>> s4.isupper()
False
>>>
```

isdigit(): Returns True if all the characters in str are digits. Else returns False.

```
>>> s3
'python 123'
>>> s3.isdigit()
False
>>> s4 = "92345"
>>> s4.isdigit()
True
>>> s5 = "92 345"
>>> s5.isdigit()
False
>>>
```

isalpha(): Returns True if all characters in str are alphabetic, Else returns False.

```
>>> s1
'pyth on'
>>> s1.isalpha()
False
>>> s2 = "pythON"
>>> s2.isalpha()
True
>>> s3 = 'python123'
>>> s3.isalpha()
False
>>> s3 = 'python on'
>>> s3.isalpha()
False
>>>
```

isalnum(): Returns True if the string is an alpha-numeric string, False otherwise. A string is alpha-numeric if all characters in the string are either alphabets or numeric values.

```
>>> s1
'pyth on'
>>> s1.isalnum()
False
>>> s2 = 'python'
>>> s2.isalnum()
True
>>> s3 = "python123"
>>> s3.isalnum()
True
>>>
```

```
>>> s4 = "123"
>>> s4.alnum()
True
>>> s4 = "123 "
>>> s4.isalnum()
False
>>>
```

strip(): Returns a copy of the string with leading and trailing whitespace removed. Very useful when you are reading the data from the file using functions.

```
>>> s1 = "I love python        "   #2 tabs were given at the end
>>> len(s1)
15
>>> s2 = s1.strip()
>>> s2
'I love python'
>>> len(s2)
13
>>>
Try variations of it - > lstrip() and rstrip()
```

split(): Returns a list of the substrings in the string, using white space as the delimiter. White space character includes \\n \\r \\t \\f and spaces.

```
>>> s1 = "here is an example for splitting the given string"
>>> s1.split()
['here', 'is', 'an', 'example', 'for', 'splitting', 'the', 'given', 'string']
>>> s1
'here is an example for splitting the given string'
>>> s1.split("p")   # can you pass “\n”?
['here is an exam', 'le for s', 'litting the given string']
>>>
```

splitlines(): Returns a list of lines in the string, breaking at line boundaries.

```
>>> s1 = """here is an
... example for
... splittling"""
>>> s1
'here is an\nexample for\nsplittling'
>>> s1.splitlines()
['here is an', 'example for', 'splittling']
>>> s1
'here is an\nexample for\nsplittling'
>>>
```

join(): This function can take any iterable which has string elements. Concatenates the string elements from the iterable with the string with which the function join is called as shown here - > Example: **'.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'**

```
>>> s1 = {'1','3','2','8'}
 >>> s1
{'8', '1', '3', '2'}
>>> "abc".join(s1)
'8abc1abc3abc2'
>>> "abc".join("pqr")
'pabcqabcr'
>>> " ".join(["sindhu","r","pai"])
'sindhu r pai'
>>>
```

startswith() and endswith(): Return True if string starts/ends with the specified prefix in the argument, False otherwise

```
>>> s1 = "great job"
>>> s1.startswith("g")
True
>>> s1.startswith("G")
False
>>> s1.startswith(s1[0].upper())
False
>>> s1.endswith("b")
True
>>> s1.endswith("ob")
True
>>> s1.endswith("jb")
False
>>>
```

Few points to think!

- Can you convert the string to lowercase and assign it to the same variable?
- How do you split the given string into exactly 4 words?
- If you pass the dictionary to join function, how does it handle concatenation?
- Once you read all the data from the file, how will you find out the number of characters in each line?
- Think about copying only the numbers from one string to another.
- Try these functions - >find, rfind, count, index

-END-