Strings

String is a collection of alphabets, words or other characters. It is one of the primitive data structures and are the building blocks for data manipulation. Python has a built-in string class named **str** . Python strings are "immutable" which means they cannot be changed after they are created.

Lets have some examples on strings:

# We can intialize a string in any of the following ways

s1 = 'this is a string'

s2 = "this is a string"

s3 = '''this is a string'''

s4 = """this is a string"""

print(s1)

print(s2)

print(s3)

print(s4)

'''

The capitalize() function returns a string with first letter capitalized and all other characters lowercased. It doesn't modify the original string.

'''

string = "python Is aN Object oriEnted Prog laNG"

capitalized\_string = string.capitalize()

print('Old String: ', string)

print('Capitalized String:', capitalized\_string)

'''

The center() method returns a string which is padded with the specified character.

center() Parameters

The center() method takes two arguments:

width - length of the string with padded characters

fillchar (optional) - padding character

The fillchar argument is optional. If it's not provided, space is taken as default argument.

The center() method returns a string padded with specified fillchar. It doesn't modify the original string.

'''

string = "Python Programming"

new\_string = string.center(43)

print("Centered String: ", new\_string)

#center() Method With \* fillchar

new\_string = string.center(30,'\*')

print("Centered String: ", new\_string)

'''

Python String casefold()

The casefold() method is an aggressive lower() method which convert strings to casefolded strings for caseless matching.

The casefold() method is removes all case distinctions present in a string. It is used for caseless matching, i.e. ignores cases when comparing.

'''

#Lowercase using casefold()

string = "PYTHON IS AWESome"

# print lowercase string

print("Lowercase string:", string.casefold())

#Comparison using casefold()

firstString = "hello dear"

secondString = "Hello Dear"

if firstString.casefold() == secondString.casefold():

    print('The strings are equal.')

else:

    print('The strings are not equal.')

'''

Python String count()

The string count() method returns the number of occurrences of a substring in the given string.

In simple words, count() method searches the substring in the given string and returns how many times the substring is present in it.

It also takes optional parameters start and end to specify the starting and ending positions in the string respectively.

The syntax of count() method is:

string.count(substring, start=..., end=...)

String count() Parameters

count() method only requires a single parameter for execution. However, it also has two optional parameters:

substring - string whose count is to be found.

start (Optional) - starting index within the string where search starts.

end (Optional) - ending index within the string where search ends.

Note: Index in Python starts from 0, not 1.

count() method returns the number of occurrences of the substring in the given string.

'''

#Count number of occurrences of a given substring

# define string

string = "Python is awesome, isn't it?"

substring = "is"

c = string.count(substring)

# print count

print("The count is:", c)

#Count number of occurrences of a given substring using start and end

string = "Python is awesome, isn't it?"

substring = "i"

# count after first 'i' and before the last 'i'

c = string.count(substring, 8, 25)

# print count

print("The count is:", c)

'''

Python String endswith()

The endswith() method returns True if a string ends with the specified suffix. If not, it returns False.

endswith() Parameters

The endswith() takes three parameters:

suffix - String or tuple of suffixes to be checked

start (optional) - Beginning position where suffix is to be checked within the string.

end (optional) - Ending position where suffix is to be checked within the string.

Return Value from endswith()

The endswith() method returns a boolean.

It returns True if strings ends with the specified suffix.

It returns False if string doesn't end with the specified suffix.

'''

#endswith() Without start and end Parameters

text = "Python is easy to learn."

result = text.endswith('to learn')

# returns False

print(result)

result = text.endswith('Python is easy to learn.')

# returns True

print(result)

#endswith() With start and end Parameters

text = "Python programming is easy to learn."

# start parameter: 7

# "programming is easy to learn." string is searched

result = text.endswith('learn.', 7)

print(result)

# Both start and end is provided

# start: 7, end: 26

# "programming is easy" string is searched

result = text.endswith('is', 7, 26)

# Returns False

print(result)

'''

#Passing Tuple to endswith()

It's possible to pass a tuple suffixes to the endswith() method in Python.

If the string ends with any item of the tuple, endswith() returns True. If not, it returns False

'''

#endswith() With Tuple Suffix

text = "programming is easy"

result = text.endswith(('programming', 'python'))

# prints False

print(result)

result = text.endswith(('python', 'easy', 'java'))

#prints True

print(result)

# With start and end parameter

# 'programming is' string is checked

result = text.endswith(('is', 'an'), 0, 14)

# prints True

print(result)

'''

Python String expandtabs()

The expandtabs() method returns a copy of string with all tab characters '\t' replaced with whitespace characters until the next multiple of tabsize parameter.

expandtabs() Parameters

The expandtabs() takes an integer tabsize argument. The default tabsize is 8.

Return Value from expandtabs()

The expandtabs() returns a string where all '\t' characters are replaced with whitespace characters until the next multiple of tabsize parameter.

'''

#expandtabs() With no Argument

str = 'xyz\t12345\tabc'

# no argument is passed

# default tabsize is 8

result = str.expandtabs()

print(result)

'''

How expandtabs() works in Python?

The expandtabs() method keeps track of the current cursor position.

The position of first '\t' character in the above program is 3. And, the tabsize is 8 (if argument is not passed).

The expandtabs() character replaces the '\t' with whitespace until the next tab stop. The position of '\t' is 3 and the first tab stop is 8. Hence, the number of spaces after 'xyz' is 5.

The next tab stops are the multiples of tabsize. The next tab stops are 16, 24, 32 and so on.

Now, the position of second '\t' character is 13. And, the next tab stop is 16. Hence, there are 3 spaces after '12345'.

'''

#expandtabs() With Different Argument

str = "xyz\t12345\tabc"

print('Original String:', str)

# tabsize is set to 2

print('Tabsize 2:', str.expandtabs(2))

# tabsize is set to 3

print('Tabsize 3:', str.expandtabs(3))

# tabsize is set to 4

print('Tabsize 4:', str.expandtabs(4))

# tabsize is set to 5

print('Tabsize 5:', str.expandtabs(5))

# tabsize is set to 6

print('Tabsize 6:', str.expandtabs(6))

'''

The default tabsize is 8. The tab stops are 8, 16 and so on. Hence, there is 5 spaces after 'xyz' and 3 after '12345' when you print the original string.

When you set the tabsize to 2. The tab stops are 2, 4, 6, 8 and so on. For 'xyz', the tab stop is 4, and for '12345', the tab stop is 10. Hence, there is 1 space after 'xyz' and 1 space after '12345'.

When you set the tabsize to 3. The tab stops are 3, 6, 9 and so on. For 'xyz', the tab stop is 6, and for '12345', the tab stop is 12. Hence, there are 3 spaces after 'xyz' and 1 space after '12345'.

When you set the tabsize to 4. The tab stops are 4, 8, 12 and so on. For 'xyz', the tab stop is 4 and for '12345', the tab stop is 12. Hence, there is 1 space after 'xyz' and 3 spaces after '12345'.

When you set the tabsize to 5. The tab stops are 5, 10, 15 and so on. For 'xyz', the tab stop is 5 and for '12345', the tab stop is 15. Hence, there are 2 spaces after 'xyz' and 5 spaces after '12345'.

When you set the tabsize to 6. The tab stops are 6, 12, 18 and so on. For 'xyz', the tab stop is 6 and for '12345', the tab stop is 12. Hence, there are 3 spaces after 'xyz' and 1 space after '12345'.

'''

'''

Python String index()

The index() method returns the index of a substring inside the string (if found). If the substring is not found, it raises an exception.

index() Parameters

The index() method takes three parameters:

sub - substring to be searched in the string str.

start and end(optional) - substring is searched within str[start:end]

Return Value from index()

If substring exists inside the string, it returns the lowest index in the string where substring is found.

If substring doesn't exist inside the string, it raises a ValueError exception.

The index() method is similar to find() method for strings.

The only difference is that find() method returns -1 if the substring is not found, whereas index() throws an exception.

'''

#index() With Substring argument Only

sentence = 'Python programming is fun.'

result = sentence.index('is fun')

print("Substring 'is fun':", result)

result = sentence.index('Java')

print("Substring 'Java':", result)

#index() With start and end Arguments

sentence = 'Python programming is fun.'

# Substring is searched in 'gramming is fun.'

print(sentence.index('ing', 10))

# Substring is searched in 'gramming is '

print(sentence.index('g is', 10, -4))

# Substring is searched in 'programming'

print(sentence.index('fun', 7, 18))

'''

Python String lstrip()

The lstrip() method returns a copy of the string with leading characters removed (based on the string argument passed).

The lstrip() removes characters from the left based on the argument (a string specifying the set of characters to be removed).

lstrip() Parameters

chars (optional) - a string specifying the set of characters to be removed.

If the chars argument is not provided, all leading whitespaces are removed from the string.

Return Value from lstrip()

The lstrip() returns a copy of the string with leading characters stripped.

All combinations of characters in the chars argument are removed from the left of the string until first mismatch.

'''

#Working of lstrip()

random\_string = '   this is good '

# Leading whitepsace are removed

print(random\_string.lstrip())

# Argument doesn't contain space

# No characters are removed.

print(random\_string.lstrip('sti'))

print(random\_string.lstrip('s ti'))

website = 'https://www.firstman.com/'

print(website.lstrip('htps:/.'))

#Working of rstrip()

random\_string = ' this is good'

# Leading whitepsace are removed

print(random\_string.rstrip())

# Argument doesn't contain 'd'

# No characters are removed.

print(random\_string.rstrip('si oo'))

print(random\_string.rstrip('sid oo'))

website = 'www.programiz.com/'

print(website.rstrip('m/.'))

#Python String strip()

#The strip() method returns a copy of the string with both leading and trailing characters removed (based on the string argument passed).

#Working of strip()

string = ' xoxo love xoxo   '

# Leading whitepsace are removed

print(string.strip())

print(string.strip(' xoxoe'))

# Argument doesn't contain space

# No characters are removed.

print(string.strip('sti'))

string = 'android is awesome'

print(string.strip('an'))