**Python String In-Build Function**

**capitalize()**

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 AWesome."

capitalized\_string = string.capitalize()

print('Old String: ', string)

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

## 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.

string = "Python is awesome"

new\_string = string.center(50)

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

## center() Method With \* fillchar

string = "Python is awesome"

new\_string = string.center(40, '\*')

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

o/p- Centered String: \*\*\*\*\*\*\*\*\*\*\*Python is awesome\*\*\*\*\*\*\*\*\*\*\*\*

**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.

# define string

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

substring = "is"

count = string.count(substring)

# print count

print("The count is:", count)

o/p-> The count is: 2

# define string

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

substring = "i"

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

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

# print count

print("The count is:", count)

o/p-> The count is: 1

# **Python String endswith()**

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

text = "Python is easy to learn."

result = text.endswith('to learn')

# returns False

print(result)

result = text.endswith('to learn.')

# returns True

print(result)

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

# returns True

print(result)

o/p->

False

True

True

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)

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

# returns True

print(result)

o/p->

True

False

True

# **String encode()**

Python 3.0, [strings](https://www.programiz.com/python-programming/string) are stored as Unicode, i.e. each character in the string is represented by a code point. So, each string is just a sequence of Unicode code points.

For efficient storage of these strings, the sequence of code points are converted into set of bytes. The process is known as **encoding**.

There are various encodings present which treats a string differently. The popular encodings being **utf-8**, **ascii**, etc.

Using string's encode() method, you can convert unicoded strings into any [encodings supported by Python](http://docs.python.org/3/library/codecs.html#standard-encodings). By default, Python uses **utf-8** encoding.

# unicode string

string = 'pythön!'

# print string

print('The string is:', string)

# default encoding to utf-8

string\_utf = string.encode()

# print result

print('The encoded version is:', string\_utf)

The string is: pythön!

The encoded version is: b'pyth\xc3\xb6n!'

# **String find()**

The find() method returns the index of first occurrence of the substring (if found). If not found, it returns -1.

The find() method returns an integer value.

* If substring exists inside the string, it returns the index of first occurence of the substring.
* If substring doesn't exist inside the string, it returns -1.

quote = 'Let it be, let it be, let it be'

result = quote.find('let it')

print("Substring 'let it':", result)

result = quote.find('small')

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

# How to use find()

if (quote.find('be,') != -1):

print("Contains substring 'be,'")

else:

print("Doesn't contain substring")

Substring 'let it': 11

Substring 'small': -1

Contains substring 'be,'

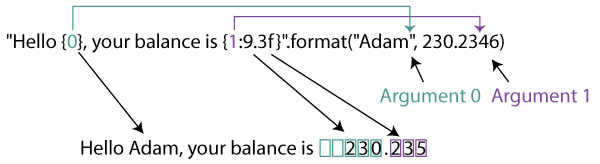
**String format() Parameters**

format() method takes any number of parameters. But, is divided into two types of parameters:

* **Positional parameters** - list of parameters that can be accessed with index of parameter inside curly braces {index}
* **Keyword parameters** - list of parameters of type key=value, that can be accessed with key of parameter inside curly braces {key}

The format() method returns the formatted string.

The format() reads the type of arguments passed to it and formats it according to the format codes defined in the string.



Here, Argument 0 is a string "Adam" and Argument 1 is a floating number 230.2346.

**Note:**Argument list starts from 0 in Python.

The string "Hello {0}, your balance is {1:9.3f}" is the template string. This contains the format codes for formatting.

# default arguments

print("Hello {}, your balance is {}.".format("Adam", 230.2346))

# positional arguments

print("Hello {0}, your balance is {1}.".format("Adam", 230.2346))

# keyword arguments

print("Hello {name}, your balance is {blc}.".format(name="Adam", blc=230.2346))

# mixed arguments

print("Hello {0}, your balance is {blc}.".format("Adam", blc=230.2346))

Hello Adam, your balance is 230.2346.

Hello Adam, your balance is 230.2346.

Hello Adam, your balance is 230.2346.

Hello Adam, your balance is 230.2346.

### Simple number formatting

# integer arguments

print("The number is:{:d}".format(123))

# float arguments

print("The float number is:{:f}".format(123.4567898))

# octal, binary and hexadecimal format

print("bin: {0:b}, oct: {0:o}, hex: {0:x}".format(12))

The number is: 123

The number is:123.456790

bin: 1100, oct: 14, hex: c

**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](https://www.programiz.com/python-programming/methods/string/find).

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

sentence = 'Python programming is fun.'

result = sentence.index('is fun')

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

result = sentence.index('Java')

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

Substring 'is fun': 19

Traceback (most recent call last):

File "...", line 6, in

result = sentence.index('Java')

ValueError: substring not found

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))

15

17

Traceback (most recent call last):

File "...", line 10, in

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

ValueError: substring not found

# **String isalnum()**

The isalnum() method returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.

name = "M234onica"

print(name.isalnum())

# contains whitespace

name = "M3onica Gell22er "

print(name.isalnum())

name = "Mo3nicaGell22er"

print(name.isalnum())

name = "133"

print(name.isalnum())

True

False

True

True