capitalize

Convert first character to upper case and rest lowercase

```
> name = "dwIGHt"
> name_new = name.capitalize()
> name_new
Dwight
```

casefold

Convert string to lower case. More aggresive than lower()

```
> txt = "OnE DoES Not SIMPLY waLK inTo MORdor"
> x = txt.casefold()
> x
one does not simply walk into mordor
```

center

Returns centred string using optional value as fill character. Space (" ") is default

```
> name = "Pam"
> name_new = name.center(9, "_")
> name_new
___Pam___
```

count

Return no of times value is in string. Optional parameters specify where to search in string

```
> advice = "Watch The Office"
> f_count = advice.count("f")
2
```

encode

Return encoded version of string. Optional parameter encoding & errors specify encoding to use and error method

```
> txt = "Mr Ståle"
> new = txt.encode('ascii', 'ignore')
> new
b'Mr Stle'
```

endswith

Return true if string ends with value

```
> name = "Michael"
> end = name.endswith("ael")
> end
True
```

expandtabs

Set tab size of string to specified number of whitespaces. Default is 8

```
> txt = "H\tT\tM\tL"
> x =  txt.expandtabs(3)
> x
H  T  M  L
```

find

Return position of value if found in string. -1 returned if string not found. 2 optional params specifying where to start & end search

```
> txt = "Pineapple on Pizza!"
> x = txt.find("a")
> x
4
```

format

Format values in string

```
> txt = "I {} The {}".format("love", "Office")
> txt
I love The Office

> txt = "I {1} The {0}".format("Office", "love")
> txt
I love The Office
```

format_map

Format values in string using map-based substitution

```
> kv = {'a' : 'Python', 'b' : 'like'}
> x = 'I {b} {a}'.format_map(kv)
> x
I like Python
```

index

Return position of value if found in string. Raise exception if not found. 2 optional params specifying where to start & end search

```
> txt = "Pineapple on Pizza!"
> x = txt.index("a")
> x
4
```

isalnum

Return true if all string characters are alphanumeric

```
> game = "Cyberpunk2077"
> x = game.isalnum()
> x
True
```

isalpha

Return true if all string characters are in alphabet

```
> name = "Andy123"
> is_alpha = name.isalpha()
> is_alpha
False
```

isascii

Return true if all string characters are ascii

```
> game = "Cyberpunk2077"
> is_ascii = game.isascii()
> is_ascii
True
```

isdecimal

Return true if all string characters are decimals (0-9). Can work on Unicode. Only supports decimals

```
> txt = "\u0033" # 3
> x = txt.isdecimal()
> x
True
```

isdigit

Return true if all string characters are digits

```
> value = "5000"
> is_digit = value.isdigit()
> is_digit
True
```

isidentifier

Return true if string is an identifer. These can only contain alphanumeric chars & underscores & can't start with numbers

```
> txt = "2077Cyber"
> x = txt.isidentifier()
> x
False
```

isnumeric

Return true if all characters are numeric

```
> txt = "2077"
> x = txt.isnumeric()
> x
True
```

isprintable

Return true if all characters are printable

```
> txt = 'Hey\nMy name is Aaron'
> x = txt.isprintable()
> x
False
```

islower

Return true if all characters are lower case

```
> name = "aragorn"
> is_lower = name.islower()
> is_lower
True
```

isspace

Return true if all characters are whitespaces

```
> txt = "    "
> x = txt.isspace()
> x
True
```

isupper

Return true if all characters are upper case

```
> txt = 'YOU SHALL NOT PASS!'
> x = txt.isupper()
> x
True
```

Python

STRING METHODS

By @AbzAaron

istitle

Return true if string follows rules of a title (all words are lowercase except the first letter of each word)

```
> txt = "I Like Tech Twitter"
> x = txt.istitle()
> x
True
```

join

Join iterable elements to end of stiring

```
> morning = ("shower", "breakfast", "work")
> morning_join = " > ".join(morning)
> morning_join
shower > breakfast > work
```

ljust

Return string left justified. 2nd optional param specifies character to fill space. Default is space.

```
> txt = "JavaScript"
> x = txt.ljust(20, ".")
> x
JavaScript.........
```

lower

Convert string to lower case

```
> names = "JIm aNd DwIGht"
> lower_names = names.lower()
> lower_names
jim and dwight
```

lstrip

Same as strip but only leading chars

```
> txt = "     Frodo     "
> x = txt.lstrip()
> print("Hello", x, "!")
Hello Frodo     !
```

maketrans

Return translation table that can be used with translate().Check docs for more details on maketrans()

```
> txt = "Harry Cotter"
> tbl = txt.maketrans("C", "P")
> print(txt.translate(tbl))
Harry Potter
```

partition

Return tuple with string parted in 3 parts. Middle part is specified string. If not found, entire string stored in first part of tuple

```
> txt = 'HTML is a programming language?'
> x = txt.partition('programming')
> x
('HTML is a ', 'programming', ' language?')
```

replace

Return string where one value is replaced with another. Optional parameter specifies how many occurrences to replace

```
> txt = "I hate Jack Forge"
> txt2 = txt.replace("hate", "love")
> txt2
I love Jack Forge
```

rfind

Same as find but searches for last occurrence of string

```
> txt = "It's alive! It's alive!"
> x = txt.rfind("alive")
> x
17
```

rindex

Same as index but searches for last occurence of string

```
> txt = "It's alive! It's alive!"
> x = txt.index("alive")
> x
17
```

rjust

Return string right justified. 2nd optional param specifies character to fill space. Default is space.

```
> txt = "JavaScript"
> x = txt.rjust(20, ".")
> x
.........JavaScript
```

rpartition

Same as partition but searches for last occurence of string

```
> txt = "One for you and one for me."
> x = txt.rpartition("for")
> x
('One for you and one ', 'for', ' me.')
```

rsplit

Same as split but starts from the right

```
> txt = "No! No! No! No! No!"
> x = txt.rsplit("! ", 3)
> x
['No! No', 'No', 'No', 'No!']
```

rstrip

Same as strip but only trailing chars

```
> txt = "     Frodo     "
> x = txt.rstrip()
> print("Hello", x, "!")
Hello      Frodo !
```

split

Split string at whitespace and return list. 2 optional params allow you to specify separator and how many splits to do.

```
> txt = "No! No! No! No! No!"
> x = txt.split("! ", 3)
> x
['No', 'No', 'No', 'No! No!']
```

splitlines

Split string at linebreaks and return list. Optional param determines whether to include line break after split

```
> txt = "Hey there\nHow's your day going?"
> x = txt.splitlines(True)
> x
['Hey there\n', "How's your day going?"]
```

startswith

Return true if string starts with value

```
> name = "Michael"
> start = name.startswith("Mi")
> start
True
```

strip

Remove leading & trailing chars. Optional param specifies chars to remove as leading/trailing chars

```
> txt = "     Frodo     "
> x = txt.strip()
> print("Hello", x, "!")
Hello Frodo !
```

swapcase

Swap cases (e.g. lowercase become upper)

```
> txt = "e.t. PHONE HOME"
> x = txt.swapcase()
> x
E.T. phone home
```

title

Covert first character of each word to upper case

```
> txt = "may the force Be with YOU"
> x = txt.title()
> x
May The Force Be With You
```

translate

Returns a translated string using mapping table, or dictionary with ascii characters

```
> dict_ascii = {74 : 80, 105 : 97}
> txt = "Jim".translate(dict_ascii)
> txt
Pam
```

upper

Convert characters to uppercase

```
> names = "JIm aNd DwIGht"
> upper_names = names.upper()
> upper_names
JIM AND DWIGHT
```

zfill

Fills string with specified number of 0 values at start

```
> price = ".125"
> price_fill = price.zfill(6)
00.125
```

removeprefix

Return string without specified prefix. Only Python 3.9+

```
>> name = "Aaron"
>> name_new = name.removeprefix("Aar")
>> name_new
on
```

removesuffix

Return string without specified suffix. Only Python 3.9+

```
>> name = "Aaron"
>> name_new = name.removesuffix("on")
>> name_new
Aar
```

Consider Supporting Me!

@AbzAaron