str.capitalize()
Return a copy of the string with its first character capitalized and the rest lowercased.

Changed in version 3.8: The first character is now put into titlecase rather than uppercase. This means that characters like digraphs will only have their first letter capitalized, instead of the full character.

str.count(sub[, start[, end]])
Return the number of non−overlapping occurrences of substring sub in the range [start, end]. Optional arguments start and end are interpreted as in slice notation.

str.endswith(suffix[, start[, end]])
Return True if the string ends with the specified suffix, otherwise return False. suffix can also be a tuple of suffixes to look for. With optional start, test beginning at that position. With optional end, stop comparing at that position.

str.find(sub[, start[, end]])
Return the lowest index in the string where substring sub is found within the slice s[start:end]. Optional arguments start and end are interpreted as in slice notation. Return −1 if sub is not found.

Note The find() method should be used only if you need to know the position of sub. To check if sub is a substring or not, use the in operator:

```
>>>
>>> 'Py' in 'Python'
True
```

str.index(sub[, start[, end]])
Like find(), but raise ValueError when the substring is not found.

str.isalnum()
Return True if all characters in the string are alphanumeric and there is at least one character, False otherwise. A character c is alphanumeric if one of the following returns True: c.isalpha(), c.isdecimal(), c.isdigit(), or c.isnumeric().

str.isalpha()
Return True if all characters in the string are alphabetic and there is at least one character, False otherwise.

str.isdecimal()
Return True if all characters in the string are decimal characters and there is at least one character, False otherwise. Decimal characters are those that can be used to form numbers in base 10, e.g. U+0660, ARABIC−INDIC DIGIT ZERO.

str.isdigit()

Return True if all characters in the string are digits and there is at least one character, False otherwise. Digits include decimal characters and digits that need special handling, such as the compatibility superscript digits. This covers digits which cannot be used to form numbers in base 10, like the Kharosthi numbers. Formally, a digit is a character that has the property value Numeric_Type=Digit or Numeric_Type=Decimal.

str.islower()
Return True if all cased characters 4 in the string are lowercase and there is at least one cased character, False otherwise.

str.isnumeric()
Return True if all characters in the string are numeric characters, and there is at least one character, False otherwise. Numeric characters include digit characters, and all characters that have the Unicode numeric value property, e.g. U+2155, VULGAR FRACTION ONE FIFTH. Formally, numeric characters are those with the property value Numeric_Type=Digit, Numeric_Type=Decimal or Numeric_Type=Numeric.

str.isspace()
Return True if there are only whitespace characters in the string and there is at least one character, False otherwise.

str.isupper()
Return True if all cased characters 4 in the string are uppercase and there is at least one cased character, False otherwise.

```
>>>
'BANANA'.isupper()
True
'banana'.isupper()
False
'baNana'.isupper()
False
' '.isupper()
False
```

str.join(iterable)
Return a string which is the concatenation of the strings in iterable. A TypeError will be raised if there are any non-string values in iterable, including bytes objects. The separator between elements is the string providing this method.

str.replace(old, new[, count])
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.

str.rfind(sub[, start[, end]])

```
Return the highest index in the string where substring sub is found, such that
    sub is contained within s[start:end]. Optional arguments start and end
    are interpreted as in slice notation. Return -1 on failure.
```

```
my_int = 42
my_str = ”the answer to life the universe and everything”
my_float = 3.14
print(f”{my_int} is {my_str}, not {my_float}”)
```

---

x in s True if an item of s is equal to x, else False
x not in s False if an item of s is equal to x, else True
s + t the concatenation of s and t
s * n or n * s n shallow copies of s concatenated
s[i] ith item of s, origin 0
s[i:j] slice of s from i to j
s[i:j:k] slice of s from i to j with step k
len(s) length of s
min(s) smallest item of s
max(s) largest item of s
s.index(x[, i[, j]]) index of the first occurrence of x in s (at or after index i and before index j)
s.count(x) total number of occurrences of x in s
s[i] = x item i of s is replaced by x
s[i:j] = t slice of s from i to j is replaced by the contents of the iterable t
del s[i:j] same as s[i:j] = []
s[i:j:k] = t the elements of s[i:j:k] are replaced by those of t
del s[i:j:k] removes the elements of s[i:j:k] from the list
s.append(x) appends x to the end of the sequence (same as s[len(s):len(s)] = [x])
s.clear() removes all items from s (same as del s[:])
s.copy() creates a shallow copy of s (same as s[:])
s.extend(t) extends s with the contents of t (same as s[len(s):len(s)] = t)
s.insert(i, x) inserts x into s at the index given by i (same as s[i:i] = [x])
s.pop([i]) retrieves the item at i and also removes it from s
s.remove(x) remove the first item from s where s[i] == x
s.reverse() reverses the items of s in place