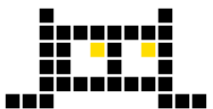


# STRINGS

1. Accessing a character in a string
2. Getting the length of a string
3. Creating a substring from existing string
4. Checking if a string is in another string
5. String methods



# 1. Accessing a character in a string

# The string is indexed (numbered). The index starts with 0 for the first character.

# Accessing the first character

```
breakfast = 'Spam and Eggs'
```

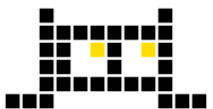
```
breakfast[0] # returns 'S'
```

```
breakfast[3] # 'm'
```

```
breakfast[4] # ' '
```

```
breakfast[-1] # 's'
```

```
breakfast[-4] # 'E'
```



## 2. Getting the length of a string

```
# The len() function returns the length of  
# a sequence (i.e. string, bytes, tuple, list or range) or  
# a collection (i.e. dictionary, set, frozen set)
```

```
breakfast = 'Spam and Eggs'  
len(breakfast)      # 13
```

```
full_name = 'Princess Diana of Themyscira, Daughter of Hippolyta'  
len(full_name)      # 51
```

```
hero_name = 'Wonder Woman'  
len(hero_name)      # 12
```

```
no_name = ''  
len(no_name)        # 0
```



### 3. Creating a substring from existing string

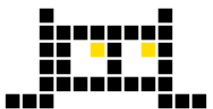
```
full_name = 'Princess Diana of Themyscira, Daughter of Hippolyta'
full_name[9:15]      # returns substring starting from index 9 of the string
                     # to 15 exclusive i.e. 'Diana'

# Substring from the beginning of the string to index 10 exclusive
full_name[:10]       # 'Princess D'

# Substring from index 25 to the end of the string
full_name[25:]       # 'ira, Daughter of Hippolyta'

# Substrings are strings, so they can be concatenated like any string
'full_name[42:] + '\s ' + full_name[30:39] + 'is' + full_name[8:14]

# Experiment: Test the above code on the Python shell to. What is the resultant string?
```



## 4. Checking if a string is in another string

```
full_name = 'Princess Diana of Themyscira, Daughter of Hippolyta'
```

```
'Diana' in full_name      # True  
'Hippocrates' in full_name # False  
'The' in full_name       # True
```



# 5. String methods

Strings are ***immutable***. It means that once created, the content of a string can not be modified. Notice that all string methods do not modify the original string. When a modification occurs, it is done on a copy of the string, which is then returned by the method.

The following are common string methods:

Method	Description	Returns
<code>capitalize()</code>	Returns a capitalized version of the string.	<code>str</code>
<code>casefold()</code>	Returns a version of the string suitable for caseless comparison.	<code>str</code>
<code>lower()</code>	Returns a copy of the string converted to lowercase.	<code>str</code>
<code>title()</code>	Returns a titlecased version of the string.	<code>str</code>
<code>swapcase()</code>	Returns a copy of the string with uppercase characters converted to lowercase and vice versa.	
<code>upper()</code>	Returns a copy of the string converted to uppercase.	<code>str</code>

## 5. String methods

Method	Description	Returns
<code>center(width[, fillchar])</code>	Returns the string centered and padded with optional fillchar. The total length of returned string is of length width. Default fillchar is a space.	str
<code>ljust(width[, fillchar])</code>	Returns left-justified string of length width. Padding is done using the specified fill character (default is a space).	str
<code>rjust(width[, fillchar])</code>	Returns right-justified string of length width. Padding is done using the specified fill character (default is a space).	str
<code>zfill (width)</code>	Pads the string with zeros on the left, to fill a field of the specified width. The string is never truncated.	str



## 5. String methods

Method	Description	Returns
<code>count(sub[,start[,end]])</code>	Returns the number of non-overlapping occurrences of substring <code>sub</code> in <code>string[start:end]</code> .	<code>int</code>
<code>find(sub[,start[,end]])</code>	Returns the lowest index in the string where substring <code>sub</code> is found, such that <code>sub</code> is contained within <code>[start:end]</code> . Returns <code>-1</code> on failure.	<code>int</code>
<code>index(sub[,start[,end]])</code>	Returns the lowest index in the string where substring <code>sub</code> is found, such that <code>sub</code> is contained within <code>[start:end]</code> . Raises <code>ValueError</code> when the substring is not found.	<code>int</code>
<code>rfind(sub[,start[,end]])</code>	Returns the highest index in the string where substring <code>sub</code> is found, such that <code>sub</code> is contained within <code>[start:end]</code> . Returns <code>-1</code> on failure.	<code>int</code>
<code>rindex(sub[,start[,end]])</code>	Returns the highest index in the string where substring <code>sub</code> is found, such that <code>sub</code> is contained within <code>[start:end]</code> . Raises <code>ValueError</code> when the substring is not found.	<code>int</code>





## 5. String methods

Method	Description	Returns
<code>endswith(suffix[, start[, end]])</code>	Returns True if the string ends with specified suffix, False otherwise. start and end are optional positions on the string on which suffix is compared. suffix can also be a tuple of strings to try.	bool
<code>startswith(prefix[, start[, end]])</code>	Returns True if the string starts with specified prefix, False otherwise. start and end are optional positions on the string on which prefix is compared. prefix can also be a tuple of strings to try.	bool



## 5. String methods

Method	Description	Returns
<code>isalnum()</code>	Returns True if all characters in the string are alphanumeric, and there is at least one character in the string. False otherwise.	bool
<code>isalpha()</code>	Returns True if all characters in the string are alphabetic and there is at least one character in the string. False otherwise.	bool
<code>isdecimal()</code>	Returns True if there are only decimal characters in the string. False otherwise.	bool
<code>isdigit()</code>	Returns True if all characters in the string are digits and there is at least one character in the string. False otherwise.	bool
<code>isidentifier()</code>	Returns True if the string is a valid identifier according to the language definition. Use <code>keyword.iskeyword()</code> to test for reserved identifiers such as "def" and "class".	bool
<code>islower()</code>	Returns True if all cased characters in the string are lowercase and there is at least one cased character in the string. False otherwise.	bool



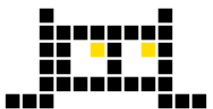
## 5. String methods

Method	Description	Returns
<code>isnumeric()</code>	Returns True if there are only numeric characters in the string, False otherwise.	bool
<code>isprintable()</code>	Returns True if all characters in the string are considered printable in <code>repr()</code> or the string is empty. False otherwise.	bool
<code>isspace()</code>	Returns True if all characters in the string are whitespace and there is at least one character in the string. False otherwise.	bool
<code>istitle()</code>	Returns True if the string is a titlecased string and there is at least one character in it, i.e. upper- and titlecase characters may only follow uncased characters and lowercase characters only cased ones. Returns False otherwise.	bool
<code>isupper()</code>	Returns True if all cased characters in the string are uppercase and there is at least one cased character in it. False otherwise.	bool



## 5. String methods

Method	Description	Returns
<code>strip([chars])</code>	Returns a copy of the string with leading and trailing whitespace removed. If <code>chars</code> is given and not <code>None</code> , remove characters in <code>chars</code> instead.	<code>str</code>
<code>lstrip([chars])</code>	Returns a copy of the string with leading whitespace removed. If <code>chars</code> is given and not <code>None</code> , remove characters in <code>chars</code> instead.	<code>str</code>
<code>rstrip([chars])</code>	Returns a copy of the string with trailing whitespace removed. If <code>chars</code> is given and not <code>None</code> , remove characters in <code>chars</code> instead.	<code>str</code>
<code>expandtabs(tabsize=8)</code>	Returns a copy of the string where all tab characters are expanded using spaces. If <code>tabsize</code> is not given, a tab size of 8 characters is assumed.	<code>str</code>
<code>replace(old, new[, count])</code>	Returns a copy of the string with all occurrences of substring <code>old</code> replaced by <code>new</code> . If the optional argument <code>count</code> is given, only the first <code>count</code> occurrences are replaced.	<code>str</code>



## 5. String methods

Method	Description	Returns
<code>split(sep=None, maxsplit=-1)</code>	Returns a list of words in the string, using <code>sep</code> as the delimiter string. If <code>maxsplit</code> is given, at most <code>maxsplit</code> splits are done. If <code>sep</code> is not specified or is <code>None</code> , any whitespace string is a separator and empty strings are removed from the result.	List of strings
<code>rsplit(sep=None, maxsplit=-1)</code>	Returns a list of words in the string, using <code>sep</code> as the delimiter string, starting at the end of the string and working to the front. If <code>maxsplit</code> is given, at most <code>maxsplit</code> splits are done. If <code>sep</code> is not specified, any whitespace string is a separator.	List of strings
<code>splitlines([keepends])</code>	Returns a list of lines in the string, breaking at line boundaries. Line breaks are not included in the resulting list unless <code>keepends</code> is given and true.	List of strings
<code>join(iterable)</code>	Returns a string which is the concatenation of the strings in the iterable. The separator between elements is the string.	str



## 5. String methods

Method	Description	Returns
<code>format(*args, **kwargs)</code>	Returns a formatted version of the string, using substitutions from args and kwargs. The substitutions are identified by braces ('{' and '}').	str
<code>format_map(mapping)</code>	Return a formatted version of the string, using substitutions from mapping. The substitutions are identified by braces ('{' and '}').	str



## 5. String methods

Method	Description	Returns
<code>translate(table)</code>	Returns a copy of the string in which each character has been mapped through the given translation table. The table must implement lookup/indexing via <code>__getitem__</code> , for instance a dictionary or list, mapping Unicode ordinals to Unicode ordinals, strings, or None. If this operation raises <code>LookupError</code> , the character is left untouched. Characters mapped to None are deleted.	<code>str</code>
<code>partition(sep)</code>	Searches for the separator <code>sep</code> in the string, and returns the part before it, the separator itself, and the part after it. If the separator is not found, returns the string and two empty strings.	<code>(head, sep, tail)</code>
<code>rpartition(sep)</code>	Search for the separator <code>sep</code> in the string, starting at the end of <code>S</code> , and returns the part before it, the separator itself, and the part after it. If the separator is not found, returns two empty strings and the string.	<code>(head, sep, tail)</code>



## 5. String methods

Method	Description	Returns
<code>encode(encoding='utf-8', errors='strict')</code>	Encodes the string using the codec registered for encoding. Default encoding is 'utf-8'. errors may be given to set a different error handling scheme. Default is 'strict' meaning that encoding errors raise a <code>UnicodeEncodeError</code> . Other possible values are 'ignore', 'replace' and 'xmlcharrefreplace' as well as any other name registered with <code>codecs.register_error</code> that can handle <code>UnicodeEncodeErrors</code> .	bytes

