د. عامر جعفر صادق
د. هاله حسن محمود
م.م امنه هيثم عبد اللطيف

Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `"hello"`.

You can display a string literal with the `print()` function:

Example

```
print("Hello")
print('Hello')
```

Output

```
Hello
Hello
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

Output

```
Hello
```

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.
However, Python does not have a character data type, a single character is simply a string with a length of 1.
Square brackets can be used to access elements of the string.

Example

Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

Output

```
e
```

Looping Through a String

Since strings are arrays, we can loop through the characters in a string, with a `for` loop.

Example

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

Output

```
b
a
n
a
n
a
```

String Length

To get the length of a string, use the `len()` function.

Example

The `len()` function returns the length of a string:

```
a = "Hello, World!"
print(len(a))
```

Output

13

Check String

To check if a certain phrase or character is present in a string, we can use the keyword in.

Example

Check if "free" is present in the following text:

```
txt = "The best things in life are free!"
print("free" in txt)
```

Output

True

Use it in an `if` statement:

Example

Print only if "free" is present:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

Output

Yes, 'free' is present.

Check if NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword `not in`.

Example

Check if "expensive" is NOT present in the following text:

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Output

True

Use it in an `if` statement:

Example

print only if "expensive" is NOT present:

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

Output

No, 'expensive' is NOT present.

Slicing Strings

You can return a range of characters by using the slice syntax. Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Output

llo

Example

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

Output

Hello

Slice To the End

By leaving out the *end* index, the range will go to the end:

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

Output

llo, World!

Negative Indexing

`

Example

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Output

orl

String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method	Description
capitalize()	Converts the first character to upper case
count()	Returns the number of times a specified value occurs in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalpha()	Returns True if all characters in the string are in the alphabet
isdigit()	Returns True if all characters in the string are digits
islower()	Returns True if all characters in the string are lower case
isupper()	Returns True if all characters in the string are upper case
join()	Converts the elements of an iterable into a string
lower()	Converts a string into lower case
replace()	Returns a string where a specified value is replaced with a specified value
split()	Splits the string at the specified separator, and returns a list
upper()	Converts a string into upper case

Modify Strings

Python has a set of built-in methods that you can use on strings.

Upper Case

Definition and Usage

The `upper()` method returns a string where all characters are in upper case.
Symbols and Numbers are ignored.

Syntax

```
string.upper()
```

Parameter Values

No parameters

Example

The `upper()` method returns the string in upper case:

```
a = "Hello, World!"
print(a.upper())
```

Output

HELLO, WORLD!

Lower Case

Definition and Usage

The `lower()` method returns a string where all characters are lower case.
Symbols and Numbers are ignored.

Syntax

```
string.lower()
```

Parameter Values

No parameters

Example

The `lower()` method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

Output

hello, world!

Replace String

Definition and Usage

The `replace()` method replaces a specified phrase with another specified phrase.

Note: *All* occurrences of the specified phrase will be replaced, if nothing else is specified.

Syntax

```
string.replace(oldvalue, newvalue, count)
```

Parameter Values

Parameter	Description
oldvalue	Required. The string to search for
newvalue	Required. The string to replace the old value with
count	Optional. A number specifying how many occurrences of the old value you want to replace. Default is all occurrences

Example

The `replace()` method replaces a string with another string:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Output

Jello, World!

Split String

The `split()` method returns a list where the text between the specified separator becomes the list items.

Example

The `split()` method splits the string into substrings if it finds instances of the separator:

```
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

Output

['Hello', ' World!']

String isdigit() Method

Definition and Usage

The `isdigit()` method returns True if all the characters are digits, otherwise False.
Exponents, like ², are also considered to be a digit.

Syntax

string.isdigit()

Parameter Values

No parameters.

Example

Check if all the characters in the text are digits:

```
txt = "50800"
x = txt.isdigit()
print(x)
```

Output

True

String isalpha() Method

Definition and Usage

The `isalpha()` method returns True if all the characters are alphabet letters (a-z).
Example of characters that are not alphabet letters: (space)!#%&? etc.

Syntax

```
string.isalpha()
```

Parameter Values

No parameters.

Example

Check if all the characters in the text are letters:

```
txt = "CompanyX"
x = txt.isalpha()
print(x)
```

Output

True

String capitalize() Method

Definition and Usage

The `capitalize()` method returns a string where the first character is upper case, and the rest is lower case.

Syntax

```
string.capitalize()
```

Parameter Values

No parameters

Example

Upper case the first letter in this sentence:

```
txt = "hello, and welcome to my world."
x = txt.capitalize()
print (x)
```

Output

Hello, and welcome to my world.

String split() Method

Definition and Usage

The `split()` method splits a string into a list.
You can specify the separator, default separator is any whitespace.

Note: When maxsplit is specified, the list will contain the specified number of elements *plus one*.

Syntax

```
string.split(separator, maxsplit)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator
maxsplit	Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

Example

Split a string into a list where each word is a list item:

```
txt = "welcome to the jungle"
x = txt.split()
print(x)
```

Output

['welcome', 'to', 'the', 'jungle']