####### Negative Indexing

Use negative indexes to start the slice from the end of the string:

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])
```

What will be the result of the following code:

```
x = 'Welcome'
print(x[3:5])
```

lcome

come

com

co

####### Python - Modify Strings

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

###Upper Case

Example:

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

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

###Lower Case

Example:

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

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

###Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

Example:

The strip() method removes any whitespace from the beginning or the end:

```
a = " Hello, World! "
```

```
print(a.strip()) # returns "Hello, World!"
```

Replace String

Example

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

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

###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!']
```

What is a correct syntax to print a string in upper case letters?

```
'Welcome'.upper()
```

'Welcome'.toUpper()

'Welcome'.toUpperCase()

String Concatenation

To concatenate, or combine, two strings you can use the + operator.

Merge variable a with variable b into variable c:

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Example:

To add a space between them, add a " ":

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

Python - Format - Strings

String Format

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

But we can combine strings and numbers by using f-strings or the format() method!

###F-Strings

F-String was introduced in Python 3.6, and is now the preferred way of formatting strings.

To specify a string as an f-string, simply put an f in front of the string literal, and add curly brackets {} as placeholders for variables and other operations.

Example:

Create an f-string:

```
age = 36
txt = f"My name is John, I am {age}"
print(txt)
```

################## String Methods #################

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isascii()	Returns True if all characters in the string are ascii characters
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string

ljust() Returns a left justified version of the string

lower() Converts a string into lower case

lstrip() Returns a left trim version of the string

maketrans() Returns a translation table to be used in translations

partition() Returns a tuple where the string is parted into three parts

replace() Returns a string where a specified value is replaced with a specified value

rfind()Searches the string for a specified value and returns the last position of where it was found

rindex() Searches the string for a specified value and returns the last position of where it was found

rjust() Returns a right justified version of the string

rpartition() Returns a tuple where the string is parted into three parts

rsplit() Splits the string at the specified separator, and returns a list

rstrip() Returns a right trim version of the string

split() Splits the string at the specified separator, and returns a list

splitlines() Splits the string at line breaks and returns a list

startswith() Returns true if the string starts with the specified value

strip() Returns a trimmed version of the string

swapcase() Swaps cases, lower case becomes upper case and vice versa

title() Converts the first character of each word to upper case

translate() Returns a translated string

upper() Converts a string into upper case

zfill() Fills the string with a specified number of 0 values at the beginning

Python Booleans

Booleans represent one of two values: True or False.

Boolean Values

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

Example:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

When you run a condition in an if statement, Python returns True or False:

Example:

Print a message based on whether the condition is True or False:

```
a = 200
b = 33

if b > a:
```

```
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Evaluate Values and Variables

The bool() function allows you to evaluate any value, and give you True or False in return,

Example:

Evaluate a string and a number:

```
print(bool("Hello"))
print(bool(15))
```

Example:

Evaluate two variables:

```
x = "Hello"
y = 15

print(bool(x))
print(bool(y))
```