What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

print("Hello, World!")

## Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have Python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

## Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_helloworld)

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

ADVERTISEMENT

## The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

From there you can write any python, including our hello world example from earlier in the tutorial:

C:\Users\Your Name>python  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>python  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print("Hello, World!")  
Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

# Python Syntax

[❮ Previous](https://www.w3schools.com/python/python_getstarted.asp)[Next ❯](https://www.w3schools.com/python/python_comments.asp)

## Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

>>> print("Hello, World!")  
Hello, World!

## On this page

[Execute Python Syntax](https://www.w3schools.com/python/python_syntax.asp#execute_python_syntax)[Python Indentation](https://www.w3schools.com/python/python_syntax.asp#python_indentation)[Python Variables](https://www.w3schools.com/python/python_syntax.asp#python_variables)[Python Comments](https://www.w3schools.com/python/python_syntax.asp#python_comments)[Exercises](https://www.w3schools.com/python/python_syntax.asp#exercises)

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users\Your Name>python myfile.py

## Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

if 5 > 2:  
  print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation)

Python will give you an error if you skip the indentation:

### Example

Syntax Error:

if 5 > 2:  
print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation_test)

The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

### Example

if 5 > 2:  
 print("Five is greater than two!")   
if 5 > 2:  
        print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation2)

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

### Example

Syntax Error:

if 5 > 2:  
 print("Five is greater than two!")  
        print("Five is greater than two!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_indentation2_error)

ADVERTISEMENT

## Python Variables

In Python, variables are created when you assign a value to it:

### Example

Variables in Python:

x = 5  
y = "Hello, World!"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_syntax_variables)

Python has no command for declaring a variable.

You will learn more about variables in the [Python Variables](https://www.w3schools.com/python/python_variables.asp) chapter.

## Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

### Example

Comments in Python:

#This is a comment.  
print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment)

Top of Form

## Test Yourself With Exercises

## Exercise:

Insert the missing part of the code below to output "Hello World".

("Hello World")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_syntax1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_getstarted.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_comments.asp)

# Python Comments

[❮ Previous](https://www.w3schools.com/python/python_syntax.asp)[Next ❯](https://www.w3schools.com/python/python_variables.asp)

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

## Creating a Comment

Comments starts with a #, and Python will ignore them:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

#This is a comment  
print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment1)

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

### Example

print("Hello, World!") #This is a comment

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment2)

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

### Example

#print("Hello, World!")  
print("Cheers, Mate!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment3)

ADVERTISEMENT

## Multiline Comments

Python does not really have a syntax for multiline comments.

To add a multiline comment you could insert a # for each line:

### Example

#This is a comment  
#written in  
#more than just one line  
print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment4)

Or, not quite as intended, you can use a multiline string.

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

### Example

"""  
This is a comment  
written in  
more than just one line  
"""  
print("Hello, World!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_comment5)

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

Top of Form

## Test Yourself With Exercises

## Exercise:

Comments in Python are written with a special character, which one?

This is a comment

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_comments1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_syntax.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_variables.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

# Python Variables

[❮ Previous](https://www.w3schools.com/python/python_comments.asp)[Next ❯](https://www.w3schools.com/python/python_variables_names.asp)

## Variables

Variables are containers for storing data values.

## Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x = 5  
y = "John"  
print(x)  
print(y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables1)

Variables do not need to be declared with any particular type, and can even change type after they have been set.

### Example

x = 4       # x is of type int  
x = "Sally" # x is now of type str  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables2)

## Casting

If you want to specify the data type of a variable, this can be done with casting.

### Example

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_casting)

ADVERTISEMENT

## Get the Type

You can get the data type of a variable with the type() function.

### Example

x = 5  
y = "John"  
print(type(x))  
print(type(y))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_type)

You will learn more about [data types](https://www.w3schools.com/python/python_datatypes.asp) and [casting](https://www.w3schools.com/python/python_casting.asp) later in this tutorial.

## Single or Double Quotes?

String variables can be declared either by using single or double quotes:

### Example

x = "John"  
# is the same as  
x = 'John'

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables7)

## Case-Sensitive

Variable names are case-sensitive.

### Example

This will create two variables:

a = 4  
A = "Sally"  
#A will not overwrite a

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_cs)

[❮ Previous](https://www.w3schools.com/python/python_comments.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_variables_names.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

# Python - Variable Names

[❮ Previous](https://www.w3schools.com/python/python_variables.asp)[Next ❯](https://www.w3schools.com/python/python_variables_multiple.asp)

## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Legal variable names:

myvar = "John"  
my\_var = "John"  
\_my\_var = "John"  
myVar = "John"  
MYVAR = "John"  
myvar2 = "John"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variable_names_legal)

### Example

Illegal variable names:

2myvar = "John"  
my-var = "John"  
my var = "John"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variable_names_error)

Remember that variable names are case-sensitive

ADVERTISEMENT

## Multi Words Variable Names

Variable names with more than one word can be difficult to read.

There are several techniques you can use to make them more readable:

## Camel Case

Each word, except the first, starts with a capital letter:

myVariableName = "John"

## Pascal Case

Each word starts with a capital letter:

MyVariableName = "John"

## Snake Case

Each word is separated by an underscore character:

my\_variable\_name = "John"

[❮ Previous](https://www.w3schools.com/python/python_variables.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_variables_multiple.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

ADVERTISEMENT

# Python Variables - Assign Multiple Values

[❮ Previous](https://www.w3schools.com/python/python_variables_names.asp)[Next ❯](https://www.w3schools.com/python/python_variables_output.asp)

## Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x, y, z = "Orange", "Banana", "Cherry"  
print(x)  
print(y)  
print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables8)

**Note:** Make sure the number of variables matches the number of values, or else you will get an error.

## One Value to Multiple Variables

And you can assign the same value to multiple variables in one line:

### Example

x = y = z = "Orange"  
print(x)  
print(y)  
print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables6)

## Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

### Example

Unpack a list:

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits  
print(x)  
print(y)  
print(z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_unpack)

Learn more about unpacking in our [Unpack Tuples](https://www.w3schools.com/python/python_tuples_unpack.asp) Chapter.

[❮ Previous](https://www.w3schools.com/python/python_variables_names.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_variables_output.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

## Output Variables

The Python print() function is often used to output variables.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x = "Python is awesome"  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_print)

In the print() function, you output multiple variables, separated by a comma:

### Example

x = "Python"  
y = "is"  
z = "awesome"  
print(x, y, z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables3)

You can also use the + operator to output multiple variables:

### Example

x = "Python "  
y = "is "  
z = "awesome"  
print(x + y + z)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables4)

Notice the space character after "Python " and "is ", without them the result would be "Pythonisawesome".

For numbers, the + character works as a mathematical operator:

### Example

x = 5  
y = 10  
print(x + y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables5)

In the print() function, when you try to combine a string and a number with the + operator, Python will give you an error:

### Example

x = 5  
y = "John"  
print(x + y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_test)

The best way to output multiple variables in the print() function is to separate them with commas, which even support different data types:

### Example

x = 5  
y = "John"  
print(x, y)

## Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create a variable outside of a function, and use it inside the function

x = "awesome"  
  
def myfunc():  
  print("Python is " + x)  
  
myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global)

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

### Example

Create a variable inside a function, with the same name as the global variable

x = "awesome"  
  
def myfunc():  
  x = "fantastic"  
  print("Python is " + x)  
  
myfunc()  
  
print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global2)

ADVERTISEMENT

## The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

### Example

If you use the global keyword, the variable belongs to the global scope:

def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global3)

Also, use the global keyword if you want to change a global variable inside a function.

### Example

To change the value of a global variable inside a function, refer to the variable by using the global keyword:

x = "awesome"  
  
def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_variables_global4)

[❮ Previous](https://www.w3schools.com/python/python_variables_output.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_variables_exercises.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

# Python - Variable Exercises

[❮ Previous](https://www.w3schools.com/python/python_variables_global.asp)[Next ❯](https://www.w3schools.com/python/python_datatypes.asp)

## Test Yourself With Exercises

Now you have learned a lot about variables, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

Top of Form

## Exercise:

Create a variable named carname and assign the value Volvo to it.

 = ""

Submit Answer »

Bottom of Form

Go to the Exercise section and test all of our Python Variable Exercises:

[Python Variable Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_variables1)

# Python Data Types

[❮ Previous](https://www.w3schools.com/python/python_variables_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_numbers.asp)

## Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | Dict |
| Set Types: | set, frozenset |
| Boolean Type: | Bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

## Getting the Data Type

You can get the data type of any object by using the type() function:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print the data type of the variable x:

x = 5  
print(type(x))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_type)

## Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** | **Try it** |
| x = "Hello World" | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str) |
| x = 20 | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int) |
| x = 20.5 | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float) |
| x = 1j | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex) |
| x = ["apple", "banana", "cherry"] | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list) |
| x = ("apple", "banana", "cherry") | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range) |
| x = {"name" : "John", "age" : 36} | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict) |
| x = {"apple", "banana", "cherry"} | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set) |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset) |
| x = True | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool) |
| x = b"Hello" | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview) |
| x = None | NoneType | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_nonetype) |

## Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** | **Try it** |
| x = str("Hello World") | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str2) |
| x = int(20) | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int2) |
| x = float(20.5) | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float2) |
| x = complex(1j) | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex2) |
| x = list(("apple", "banana", "cherry")) | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list2) |
| x = tuple(("apple", "banana", "cherry")) | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple2) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range2) |
| x = dict(name="John", age=36) | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict2) |
| x = set(("apple", "banana", "cherry")) | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set2) |
| x = frozenset(("apple", "banana", "cherry")) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset2) |
| x = bool(5) | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool2) |
| x = bytes(5) | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes2) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray2) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview2) |

Top of Form

## Test Yourself With Exercises

## Exercise:

The following code example would print the data type of x, what data type would that be?

x = 5

print(type(x))



Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_datatypes1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_variables_exercises.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_numbers.asp)

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

# Python Numbers

[❮ Previous](https://www.w3schools.com/python/python_datatypes.asp)[Next ❯](https://www.w3schools.com/python/python_casting.asp)

## Python Numbers

There are three numeric types in Python:

* int
* float
* complex

Variables of numeric types are created when you assign a value to them:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex

To verify the type of any object in Python, use the type() function:

### Example

print(type(x))  
print(type(y))  
print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers)

## Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

### Example

Integers:

x = 1  
y = 35656222554887711  
z = -3255522  
  
print(type(x))  
print(type(y))  
print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_int)

## Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

### Example

Floats:

x = 1.10  
y = 1.0  
z = -35.59  
  
print(type(x))  
print(type(y))  
print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_float)

Float can also be scientific numbers with an "e" to indicate the power of 10.

### Example

Floats:

x = 35e3  
y = 12E4  
z = -87.7e100  
  
print(type(x))  
print(type(y))  
print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_float2)

ADVERTISEMENT

## Complex

Complex numbers are written with a "j" as the imaginary part:

### Example

Complex:

x = 3+5j  
y = 5j  
z = -5j  
  
print(type(x))  
print(type(y))  
print(type(z))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_complex)

## Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:

### Example

Convert from one type to another:

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex  
  
#convert from int to float:  
a = float(x)  
  
#convert from float to int:  
b = int(y)  
  
#convert from int to complex:  
c = complex(x)  
  
print(a)  
print(b)  
print(c)  
  
print(type(a))  
print(type(b))  
print(type(c))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_convert)

**Note:** You cannot convert complex numbers into another number type.

## Random Number

Python does not have a random() function to make a random number, but Python has a built-in module called random that can be used to make random numbers:

### Example

Import the random module, and display a random number between 1 and 9:

import random  
  
print(random.randrange(1, 10))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_numbers_random)

In our [Random Module Reference](https://www.w3schools.com/python/module_random.asp) you will learn more about the Random module.

Top of Form

## Test Yourself With Exercises

## Exercise:

Insert the correct syntax to convert x into a floating point number.

x = 5

x = (x)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_numbers1)

Bottom of Form

# Python Casting

[❮ Previous](https://www.w3schools.com/python/python_numbers.asp)[Next ❯](https://www.w3schools.com/python/python_strings.asp)

## Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

* int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
* float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
* str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Integers:

x = int(1)   # x will be 1  
y = int(2.8) # y will be 2  
z = int("3") # z will be 3

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_casting_int)

### Example

Floats:

x = float(1)     # x will be 1.0  
y = float(2.8)   # y will be 2.8  
z = float("3")   # z will be 3.0  
w = float("4.2") # w will be 4.2

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_float)

### Example

Strings:

x = str("s1") # x will be 's1'  
y = str(2)    # y will be '2'  
z = str(3.0)  # z will be '3.0'

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string)

[❮ Previous](https://www.w3schools.com/python/python_numbers.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings.asp)

# Python Strings

[❮ Previous](https://www.w3schools.com/python/python_casting.asp)[Next ❯](https://www.w3schools.com/python/python_strings_slicing.asp)

## 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[Get your own Python Server](https://www.w3schools.com/spaces/)

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_literal)

## 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)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_var)

## Multiline Strings

You can assign a multiline string to a variable by using three quotes:

### Example

You can use three double quotes:

a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_multi)

Or three single quotes:

### Example

a = '''Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua.'''  
print(a)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_multi2)

**Note:** in the result, the line breaks are inserted at the same position as in the code.

ADVERTISEMENT

## 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])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string1)

## 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)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_string)

Learn more about For Loops in our [Python For Loops](https://www.w3schools.com/python/python_for_loops.asp) chapter.

## 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))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_len)

## 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)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_is_in)

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.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_if_in)

Learn more about If statements in our [Python If...Else](https://www.w3schools.com/python/python_conditions.asp) chapter.

## 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)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_is_not_in)

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.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_if_not_in)

# Python - Slicing Strings

[❮ Previous](https://www.w3schools.com/python/python_strings.asp)[Next ❯](https://www.w3schools.com/python/python_strings_modify.asp)

## Slicing

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 your own Python Server](https://www.w3schools.com/spaces/)

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

b = "Hello, World!"  
print(b[2:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string2)

**Note:**The first character has index 0.

## Slice From the Start

By leaving out the start index, the range will start at the first character:

### Example

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

b = "Hello, World!"  
print(b[:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_slice_start)

ADVERTISEMENT

## 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:])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_slice_end)

## 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])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_negativeindex)

[❮ Previous](https://www.w3schools.com/python/python_strings.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings_modify.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

# Python - Modify Strings

[❮ Previous](https://www.w3schools.com/python/python_strings_slicing.asp)[Next ❯](https://www.w3schools.com/python/python_strings_concatenate.asp)

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

## Upper Case

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

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

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_upper)

## Lower Case

### Example

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

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_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!"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_strip)

ADVERTISEMENT

## Replace String

### Example

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

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_replace)

## 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!']

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_split)

Learn more about Lists in our [Python Lists](https://www.w3schools.com/python/python_lists.asp) chapter.

## String Methods

Learn more about String Methods with our [String Methods Reference](https://www.w3schools.com/python/python_ref_string.asp)

[❮ Previous](https://www.w3schools.com/python/python_strings_slicing.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings_concatenate.asp)

# Python - String Concatenation

[❮ Previous](https://www.w3schools.com/python/python_strings_modify.asp)[Next ❯](https://www.w3schools.com/python/python_strings_format.asp)

## String Concatenation

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

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Merge variable a with variable b into variable c:

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_concat)

### Example

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

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_concat2)

[❮ Previous](https://www.w3schools.com/python/python_strings_modify.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings_format.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

# Python - Format - Strings

[❮ Previous](https://www.w3schools.com/python/python_strings_concatenate.asp)[Next ❯](https://www.w3schools.com/python/python_strings_escape.asp)

## String Format

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

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format_error)

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

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

### Example

Use the format() method to insert numbers into strings:

age = 36  
txt = "My name is John, and I am {}"  
print(txt.format(age))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format1)

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

### Example

quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want {} pieces of item {} for {} dollars."  
print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format2)

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

### Example

quantity = 3  
itemno = 567  
price = 49.95  
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."  
print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_format3)

# Python - Escape Characters

[❮ Previous](https://www.w3schools.com/python/python_strings_format.asp)[Next ❯](https://www.w3schools.com/python/python_strings_methods.asp)

## Escape Character

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

You will get an error if you use double quotes inside a string that is surrounded by double quotes:

txt = "We are the so-called "Vikings" from the north."

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape_error)

To fix this problem, use the escape character \":

### Example

The escape character allows you to use double quotes when you normally would not be allowed:

txt = "We are the so-called \"Vikings\" from the north."

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape)

## Escape Characters

Other escape characters used in Python:

|  |  |  |
| --- | --- | --- |
| **Code** | **Result** | **Try it** |
| \' | Single Quote | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape2) |
| \\ | Backslash | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_backslash) |
| \n | New Line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_newline) |
| \r | Carriage Return | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_r) |
| \t | Tab | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_t) |
| \b | Backspace | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_string_b) |
| \f | Form Feed |  |
| \ooo | Octal value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_octal) |
| \xhh | Hex value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_hex) |

[❮ Previous](https://www.w3schools.com/python/python_strings_format.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings_methods.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

# Python - String Methods

[❮ Previous](https://www.w3schools.com/python/python_strings_escape.asp)[Next ❯](https://www.w3schools.com/python/python_strings_exercises.asp)

## String Methods

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

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case |
| [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case |
| [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string |
| [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string |
| [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string |
| [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value |
| [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string |
| [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found |
| [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string |
| format\_map() | Formats specified values in a string |
| [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found |
| [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric |
| [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet |
| [isascii()](https://www.w3schools.com/python/ref_string_isascii.asp) | Returns True if all characters in the string are ascii characters |
| [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals |
| [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits |
| [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier |
| [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case |
| [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric |
| [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable |
| [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces |
| [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title |
| [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case |
| [join()](https://www.w3schools.com/python/ref_string_join.asp) | Joins the elements of an iterable to the end of the string |
| [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string |
| [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case |
| [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string |
| [maketrans()](https://www.w3schools.com/python/ref_string_maketrans.asp) | Returns a translation table to be used in translations |
| [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts |
| [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value |
| [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found |
| [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string |
| [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts |
| [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list |
| [rstrip()](https://www.w3schools.com/python/ref_string_rstrip.asp) | Returns a right trim version of the string |
| [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a list |
| [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list |
| [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value |
| [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string |
| [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa |
| [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case |
| [translate()](https://www.w3schools.com/python/ref_string_translate.asp) | Returns a translated string |
| [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case |
| [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning |

[❮ Previous](https://www.w3schools.com/python/python_strings_escape.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_strings_exercises.asp)

# Python - String Exercises

[❮ Previous](https://www.w3schools.com/python/python_strings_methods.asp)[Next ❯](https://www.w3schools.com/python/python_booleans.asp)

## Test Yourself With Exercises

Now you have learned a lot about Strings, and how to use them in Python.

Are you ready for a test?

Try to insert the missing part to make the code work as expected:

Top of Form

## Test Yourself With Exercises

## Exercise:

Use the len method to print the length of the string.

x = "Hello World"

print()

Submit Answer »

Bottom of Form

Go to the Exercise section and test all of our Python Strings Exercises:

[Python String Exercises](https://www.w3schools.com/python/exercise.asp?filename=exercise_strings1)

[❮ Previous](https://www.w3schools.com/python/python_strings_methods.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_booleans.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

# Python Booleans

[❮ Previous](https://www.w3schools.com/python/python_strings_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_operators.asp)

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[Get your own Python Server](https://www.w3schools.com/spaces/)

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans1)

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")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans2)

## 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))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans3)

### Example

Evaluate two variables:

x = "Hello"  
y = 15  
  
print(bool(x))  
print(bool(y))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans4)

ADVERTISEMENT

## Most Values are True

Almost any value is evaluated to True if it has some sort of content.

Any string is True, except empty strings.

Any number is True, except 0.

Any list, tuple, set, and dictionary are True, except empty ones.

### Example

The following will return True:

bool("abc")  
bool(123)  
bool(["apple", "cherry", "banana"])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans5)

## Some Values are False

In fact, there are not many values that evaluate to False, except empty values, such as (), [], {}, "", the number 0, and the value None. And of course the value False evaluates to False.

### Example

The following will return False:

bool(False)  
bool(None)  
bool(0)  
bool("")  
bool(())  
bool([])  
bool({})

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans6)

One more value, or object in this case, evaluates to False, and that is if you have an object that is made from a class with a \_\_len\_\_ function that returns 0 or False:

### Example

class myclass():  
  def \_\_len\_\_(self):  
    return 0  
  
myobj = myclass()  
print(bool(myobj))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans7)

## Functions can Return a Boolean

You can create functions that returns a Boolean Value:

### Example

Print the answer of a function:

def myFunction() :  
  return True  
  
print(myFunction())

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_boolean_return)

You can execute code based on the Boolean answer of a function:

### Example

Print "YES!" if the function returns True, otherwise print "NO!":

def myFunction() :  
  return True  
  
if myFunction():  
  print("YES!")  
else:  
  print("NO!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_boolean_return2)

Python also has many built-in functions that return a boolean value, like the isinstance() function, which can be used to determine if an object is of a certain data type:

### Example

Check if an object is an integer or not:

x = 200  
print(isinstance(x, int))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_booleans8)

Top of Form

## Test Yourself With Exercises

## Exercise:

The statement below would print a Boolean value, which one?

print(10 > 9)



Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_booleans1)

Bottom of Form

# Python Operators

[❮ Previous](https://www.w3schools.com/python/python_booleans.asp)[Next ❯](https://www.w3schools.com/python/python_lists.asp)

## Python Operators

Operators are used to perform operations on variables and values.

In the example below, we use the + operator to add together two values:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

print(10 + 5)

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper)

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

## Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| + | Addition | x + y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_add) |
| - | Subtraction | x - y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_sub) |
| \* | Multiplication | x \* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mult) |
| / | Division | x / y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_div) |
| % | Modulus | x % y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod) |
| \*\* | Exponentiation | x \*\* y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_exp) |
| // | Floor division | x // y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_floordiv) |

## Python Assignment Operators

Assignment operators are used to assign values to variables:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Example** | **Same As** | **Try it** |
| = | x = 5 | x = 5 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass1) |
| += | x += 3 | x = x + 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass2) |
| -= | x -= 3 | x = x - 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass3) |
| \*= | x \*= 3 | x = x \* 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass4) |
| /= | x /= 3 | x = x / 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass5) |
| %= | x %= 3 | x = x % 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass6) |
| //= | x //= 3 | x = x // 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass7) |
| \*\*= | x \*\*= 3 | x = x \*\* 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass8) |
| &= | x &= 3 | x = x & 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass9) |
| |= | x |= 3 | x = x | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass10) |
| ^= | x ^= 3 | x = x ^ 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass11) |
| >>= | x >>= 3 | x = x >> 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass12) |
| <<= | x <<= 3 | x = x << 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_ass13) |

ADVERTISEMENT

## Python Comparison Operators

Comparison operators are used to compare two values:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| == | Equal | x == y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare1) |
| != | Not equal | x != y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare2) |
| > | Greater than | x > y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare4) |
| < | Less than | x < y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare5) |
| >= | Greater than or equal to | x >= y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare6) |
| <= | Less than or equal to | x <= y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_compare7) |

## Python Logical Operators

Logical operators are used to combine conditional statements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| and | Returns True if both statements are true | x < 5 and  x < 10 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical1) |
| or | Returns True if one of the statements is true | x < 5 or x < 4 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical2) |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_logical3) |

## Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| is | Returns True if both variables are the same object | x is y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_identity1) |
| is not | Returns True if both variables are not the same object | x is not y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_identity2) |

## Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| in | Returns True if a sequence with the specified value is present in the object | x in y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_membership1) |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_membership2) |

## Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** | **Try it** |
| & | AND | Sets each bit to 1 if both bits are 1 | x & y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_and) |
| | | OR | Sets each bit to 1 if one of two bits is 1 | x | y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_or) |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_xor) |
| ~ | NOT | Inverts all the bits | ~x | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_not) |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_left_shift) |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_right_shift) |

## Operator Precedence

Operator precedence describes the order in which operations are performed.

### Example

Parentheses has the highest precedence, meaning that expressions inside parentheses must be evaluated first:

print((6 + 3) - (6 + 3))

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_parentheses)

### Example

Multiplication \* has higher precedence than addition +, and therefor multiplications are evaluated before additions:

print(100 + 5 \* 3)

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_multiplication)

The precedence order is described in the table below, starting with the highest precedence at the top:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Try it** |
| () | Parentheses | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_parentheses) |
| \*\* | Exponentiation | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_exponent) |
| +x  -x  ~x | Unary plus, unary minus, and bitwise NOT | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_bitwise_not) |
| \*  /  //  % | Multiplication, division, floor division, and modulus | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_multiplication) |
| +  - | Addition and subtraction | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_subtraction) |
| <<  >> | Bitwise left and right shifts | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_shift) |
| & | Bitwise AND | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_bitwise_and) |
| ^ | Bitwise XOR | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_bitwise_xor) |
| | | Bitwise OR | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_bitwise_or) |
| ==  !=  >  >=  <  <=  is  is not  in  not in | Comparisons, identity, and membership operators | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_like) |
| Not | Logical NOT | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_not) |
| And | AND | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_and) |
| Or | OR | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_or) |

If two operators have the same precedence, the expression is evaluated from left to right.

### Example

Addition + and subtraction - has the same precedence, and therefor we evaluate the expression from left to right:

print(5 + 4 - 7 + 3)

[Run example »](https://www.w3schools.com/python/trypython.asp?filename=demo_precedence_same)

Top of Form

## Test Yourself With Exercises

## Exercise:

Multiply 10 with 5, and print the result.

print(10  5)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_operators1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_booleans.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_lists.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

# Python Lists

[❮ Previous](https://www.w3schools.com/python/python_operators.asp)[Next ❯](https://www.w3schools.com/python/python_lists_access.asp)

mylist = ["apple", "banana", "cherry"]

## List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create a List:

thislist = ["apple", "banana", "cherry"]  
print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list)

## List Items

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

## Ordered

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

**Note:** There are some [list methods](https://www.w3schools.com/python/python_lists_methods.asp) that will change the order, but in general: the order of the items will not change.

## Changeable

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

## Allow Duplicates

Since lists are indexed, lists can have items with the same value:

### Example

Lists allow duplicate values:

thislist = ["apple", "banana", "cherry", "apple", "cherry"]  
print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_duplicates)

ADVERTISEMENT

## List Length

To determine how many items a list has, use the len() function:

### Example

Print the number of items in the list:

thislist = ["apple", "banana", "cherry"]  
print(len(thislist))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_len)

## List Items - Data Types

List items can be of any data type:

### Example

String, int and boolean data types:

list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes)

A list can contain different data types:

### Example

A list with strings, integers and boolean values:

list1 = ["abc", 34, True, 40, "male"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes2)

## type()

From Python's perspective, lists are defined as objects with the data type 'list':

<class 'list'>

### Example

What is the data type of a list?

mylist = ["apple", "banana", "cherry"]  
print(type(mylist))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_datatypes3)

## The list() Constructor

It is also possible to use the list() constructor when creating a new list.

### Example

Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets  
print(thislist)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_constructor)

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**Set**](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.

\*Set items are unchangeable, but you can remove and/or add items whenever you like.

\*\*As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

[❮ Previous](https://www.w3schools.com/python/python_operators.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_lists_access.asp)

ADVERTISEMENT

#### [COLOR PICKER](https://www.w3schools.com/colors/colors_picker.asp)

[](https://www.w3schools.com/colors/colors_picker.asp)

ADVERTISEMENT

# Python - Access List Items

[❮ Previous](https://www.w3schools.com/python/python_lists.asp)[Next ❯](https://www.w3schools.com/python/python_lists_change.asp)

## Access Items

List items are indexed and you can access them by referring to the index number:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_access)

**Note:**The first item has index 0.

### Negative Indexing

Negative indexing means start from the end

-1 refers to the last item, -2 refers to the second last item etc.

### Example

Print the last item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_negative_index)

### Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

### Example

Return the third, fourth, and fifth item:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range)

**Note:** The search will start at index 2 (included) and end at index 5 (not included).

Remember that the first item has index 0.

By leaving out the start value, the range will start at the first item:

### Example

This example returns the items from the beginning to, but NOT including, "kiwi":

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[:4])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range2)

By leaving out the end value, the range will go on to the end of the list:

### Example

This example returns the items from "cherry" to the end:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range3)

ADVERTISEMENT

### Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the list:

### Example

This example returns the items from "orange" (-4) to, but NOT including "mango" (-1):

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[-4:-1])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_range_negative)

## Check if Item Exists

To determine if a specified item is present in a list use the in keyword:

### Example

Check if "apple" is present in the list:

thislist = ["apple", "banana", "cherry"]  
if "apple" in thislist:  
  print("Yes, 'apple' is in the fruits list")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_list_in)

[❮ Previous](https://www.w3schools.com/python/python_lists.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_lists_change.asp)

# Python If ... Else

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_while_loops.asp)

## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

If statement:

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if2)

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

## Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

### Example

If statement, without indentation (will raise an error):

a = 33  
b = 200  
if b > a:  
print("b is greater than a") # you will get an error

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_error)

ADVERTISEMENT

## Elif

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

### Example

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_elif)

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

## Else

The else keyword catches anything which isn't caught by the preceding conditions.

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else)

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else2)

## Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

### Example

One line if statement:

if a > b: print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_short)

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

### Example

One line if else statement:

a = 2  
b = 330  
print("A") if a > b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short)

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

You can also have multiple else statements on the same line:

### Example

One line if else statement, with 3 conditions:

a = 330  
b = 330  
print("A") if a > b else print("=") if a == b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short2)

## And

The and keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, AND if c is greater than a:

a = 200  
b = 33  
c = 500  
if a > b and c > a:  
  print("Both conditions are True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_and)

## Or

The or keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, OR if a is greater than c:

a = 200  
b = 33  
c = 500  
if a > b or a > c:  
  print("At least one of the conditions is True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_or)

## Not

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

### Example

Test if a is NOT greater than b:

a = 33  
b = 200  
if not a > b:  
  print("a is NOT greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_not)

## Nested If

You can have if statements inside if statements, this is called nested if statements.

### Example

x = 41  
  
if x > 10:  
  print("Above ten,")  
  if x > 20:  
    print("and also above 20!")  
  else:  
    print("but not above 20.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_nested)

## The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

### Example

a = 33  
b = 200  
  
if b > a:  
  pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_pass)

Top of Form

## Test Yourself With Exercises

## Exercise:

Print "Hello World" if a is greater than b.

a = 50

b = 10

 a  b

print("Hello World")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_ifelse1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_exercises.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_while_loops.asp)

# Python If ... Else

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_exercises.asp)[Next ❯](https://www.w3schools.com/python/python_while_loops.asp)

## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

If statement:

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if2)

In this example we use two variables, a and b, which are used as part of the if statement to test whether b is greater than a. As a is 33, and b is 200, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

## Indentation

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

### Example

If statement, without indentation (will raise an error):

a = 33  
b = 200  
if b > a:  
print("b is greater than a") # you will get an error

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_error)

ADVERTISEMENT

## Elif

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

### Example

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_elif)

In this example a is equal to b, so the first condition is not true, but the elif condition is true, so we print to screen that "a and b are equal".

## Else

The else keyword catches anything which isn't caught by the preceding conditions.

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else)

In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

You can also have an else without the elif:

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else2)

## Short Hand If

If you have only one statement to execute, you can put it on the same line as the if statement.

### Example

One line if statement:

if a > b: print("a is greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_short)

## Short Hand If ... Else

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

### Example

One line if else statement:

a = 2  
b = 330  
print("A") if a > b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short)

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

You can also have multiple else statements on the same line:

### Example

One line if else statement, with 3 conditions:

a = 330  
b = 330  
print("A") if a > b else print("=") if a == b else print("B")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_else_short2)

## And

The and keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, AND if c is greater than a:

a = 200  
b = 33  
c = 500  
if a > b and c > a:  
  print("Both conditions are True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_and)

## Or

The or keyword is a logical operator, and is used to combine conditional statements:

### Example

Test if a is greater than b, OR if a is greater than c:

a = 200  
b = 33  
c = 500  
if a > b or a > c:  
  print("At least one of the conditions is True")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_or)

## Not

The not keyword is a logical operator, and is used to reverse the result of the conditional statement:

### Example

Test if a is NOT greater than b:

a = 33  
b = 200  
if not a > b:  
  print("a is NOT greater than b")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_not)

## Nested If

You can have if statements inside if statements, this is called nested if statements.

### Example

x = 41  
  
if x > 10:  
  print("Above ten,")  
  if x > 20:  
    print("and also above 20!")  
  else:  
    print("but not above 20.")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_nested)

## The pass Statement

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the pass statement to avoid getting an error.

### Example

a = 33  
b = 200  
  
if b > a:  
  pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_if_pass)

Top of Form

## Test Yourself With Exercises

## Exercise:

Print "Hello World" if a is greater than b.

a = 50

b = 10

 a  b

print("Hello World")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_ifelse1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_dictionaries_exercises.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_while_loops.asp)

# Python While Loops

[❮ Previous](https://www.w3schools.com/python/python_conditions.asp)[Next ❯](https://www.w3schools.com/python/python_for_loops.asp)

## Python Loops

Python has two primitive loop commands:

* while loops
* for loops

## The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print i as long as i is less than 6:

i = 1  
while i < 6:  
  print(i)  
  i += 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while)

**Note:** remember to increment i, or else the loop will continue forever.

The while loop requires relevant variables to be ready, in this example we need to define an indexing variable, i, which we set to 1.

## The break Statement

With the break statement we can stop the loop even if the while condition is true:

### Example

Exit the loop when i is 3:

i = 1  
while i < 6:  
  print(i)  
  if i == 3:  
    break  
  i += 1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_break)

ADVERTISEMENT

## The continue Statement

With the continue statement we can stop the current iteration, and continue with the next:

### Example

Continue to the next iteration if i is 3:

i = 0  
while i < 6:  
  i += 1  
  if i == 3:  
    continue  
  print(i)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_continue)

## The else Statement

With the else statement we can run a block of code once when the condition no longer is true:

### Example

Print a message once the condition is false:

i = 1  
while i < 6:  
  print(i)  
  i += 1  
else:  
  print("i is no longer less than 6")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_while_else)

Top of Form

## Test Yourself With Exercises

## Exercise:

Print i as long as i is less than 6.

i = 1

 i < 6

print(i)

i += 1

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_while_loops1)

Bottom of Form

[❮ Previous](https://www.w3schools.com/python/python_conditions.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_for_loops.asp)

## Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for)

The for loop does not require an indexing variable to set beforehand.

## Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

### Example

Loop through the letters in the word "banana":

for x in "banana":  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_string)

## The break Statement

With the break statement we can stop the loop before it has looped through all the items:

### Example

Exit the loop when x is "banana":

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)  
  if x == "banana":  
    break

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_break)

### Example

Exit the loop when x is "banana", but this time the break comes before the print:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    break  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_break2)

ADVERTISEMENT

## The continue Statement

With the continue statement we can stop the current iteration of the loop, and continue with the next:

### Example

Do not print banana:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    continue  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_continue)

## The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

### Example

Using the range() function:

for x in range(6):  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range)

Note that range(6) is not the values of 0 to 6, but the values 0 to 5.

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

### Example

Using the start parameter:

for x in range(2, 6):  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range2)

The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, **3**):

### Example

Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_range3)

## Else in For Loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

### Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

for x in range(6):  
  print(x)  
else:  
  print("Finally finished!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_else)

**Note:** The else block will NOT be executed if the loop is stopped by a break statement.

### Example

Break the loop when x is 3, and see what happens with the else block:

for x in range(6):  
  if x == 3: break  
  print(x)  
else:  
  print("Finally finished!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_else_break)

## Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

### Example

Print each adjective for every fruit:

adj = ["red", "big", "tasty"]  
fruits = ["apple", "banana", "cherry"]  
  
for x in adj:  
  for y in fruits:  
    print(x, y)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_nested)

## The pass Statement

for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

### Example

for x in [0, 1, 2]:  
  pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_pass)

Top of Form

## Test Yourself With Exercises

## Exercise:

Loop through the items in the fruits list.

fruits = ["apple", "banana", "cherry"]

 x  fruits

print(x)

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_for_loops1)

Bottom of Form

# Python Functions

[❮ Previous](https://www.w3schools.com/python/python_for_loops.asp)[Next ❯](https://www.w3schools.com/python/python_lambda.asp)

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

## Creating a Function

In Python a function is defined using the def keyword:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

def my\_function():  
  print("Hello from a function")

## Calling a Function

To call a function, use the function name followed by parenthesis:

### Example

def my\_function():  
  print("Hello from a function")  
  
**my\_function()**

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function)

## Arguments

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

### Example

def my\_function(**fname**):  
  print(fname + " Refsnes")  
  
my\_function(**"Emil"**)  
my\_function(**"Tobias"**)  
my\_function(**"Linus"**)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param)

Arguments are often shortened to args in Python documentations.

ADVERTISEMENT

## Parameters or Arguments?

The terms parameter and argument can be used for the same thing: information that are passed into a function.

From a function's perspective:

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

## Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

### Example

This function expects 2 arguments, and gets 2 arguments:

def my\_function(fname, lname):  
  print(fname + " " + lname)  
  
my\_function("Emil", "Refsnes")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args_n)

If you try to call the function with 1 or 3 arguments, you will get an error:

### Example

This function expects 2 arguments, but gets only 1:

def my\_function(fname, lname):  
  print(fname + " " + lname)  
  
my\_function("Emil")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args_error)

## Arbitrary Arguments, \*args

If you do not know how many arguments that will be passed into your function, add a \* before the parameter name in the function definition.

This way the function will receive a tuple of arguments, and can access the items accordingly:

### Example

If the number of arguments is unknown, add a \* before the parameter name:

def my\_function(\*kids):  
  print("The youngest child is " + kids[2])  
  
my\_function("Emil", "Tobias", "Linus")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_args)

Arbitrary Arguments are often shortened to \*args in Python documentations.

## Keyword Arguments

You can also send arguments with the key = value syntax.

This way the order of the arguments does not matter.

### Example

def my\_function(child3, child2, child1):  
  print("The youngest child is " + child3)  
  
my\_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_kwargs)

The phrase Keyword Arguments are often shortened to kwargs in Python documentations.

## Arbitrary Keyword Arguments, \*\*kwargs

If you do not know how many keyword arguments that will be passed into your function, add two asterisk: \*\* before the parameter name in the function definition.

This way the function will receive a dictionary of arguments, and can access the items accordingly:

### Example

If the number of keyword arguments is unknown, add a double \*\* before the parameter name:

def my\_function(\*\*kid):  
  print("His last name is " + kid["lname"])  
  
my\_function(fname = "Tobias", lname = "Refsnes")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_kwargs_n)

Arbitrary Kword Arguments are often shortened to \*\*kwargs in Python documentations.

## Default Parameter Value

The following example shows how to use a default parameter value.

If we call the function without argument, it uses the default value:

### Example

def my\_function(**country = "Norway"**):  
  print("I am from " + country)  
  
my\_function("Sweden")  
my\_function("India")  
my\_function()  
my\_function("Brazil")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param2)

## Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

E.g. if you send a List as an argument, it will still be a List when it reaches the function:

### Example

def my\_function(food):  
  for x in food:  
    print(x)  
  
fruits = ["apple", "banana", "cherry"]  
  
my\_function(fruits)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_param3)

## Return Values

To let a function return a value, use the return statement:

### Example

def my\_function(x):  
  **return 5 \* x**  
print(my\_function(3))  
print(my\_function(5))  
print(my\_function(9))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_return)

## The pass Statement

function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

### Example

def myfunction():  
  pass

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_function_pass)

## Recursion

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

In this example, tri\_recursion() is a function that we have defined to call itself ("recurse"). We use the k variable as the data, which decrements (-1) every time we recurse. The recursion ends when the condition is not greater than 0 (i.e. when it is 0).

To a new developer it can take some time to work out how exactly this works, best way to find out is by testing and modifying it.

### Example

Recursion Example

def tri\_recursion(k):  
  if(k > 0):  
    result = k + tri\_recursion(k - 1)  
    print(result)  
  else:  
    result = 0  
  return result  
  
print("\n\nRecursion Example Results")  
tri\_recursion(6)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_recursion)

Top of Form

## Test Yourself With Exercises

## Exercise:

Create a function named my\_function.

:

print("Hello from a function")

Submit Answer »

[Start the Exercise](https://www.w3schools.com/python/exercise.asp?filename=exercise_functions1)

Bottom of Form

# Python Arrays

[❮ Previous](https://www.w3schools.com/python/python_lambda.asp)[Next ❯](https://www.w3schools.com/python/python_classes.asp)

**Note:** Python does not have built-in support for Arrays, but [Python Lists](https://www.w3schools.com/python/python_lists.asp) can be used instead.

## Arrays

**Note:** This page shows you how to use LISTS as ARRAYS, however, to work with arrays in Python you will have to import a library, like the [NumPy library](https://www.w3schools.com/python/numpy/default.asp).

Arrays are used to store multiple values in one single variable:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create an array containing car names:

cars = ["Ford", "Volvo", "BMW"]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array1)

## What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

car1 = "Ford"  
car2 = "Volvo"  
car3 = "BMW"

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

## Access the Elements of an Array

You refer to an array element by referring to the index number.

### Example

Get the value of the first array item:

x = cars[0]

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array2)

### Example

Modify the value of the first array item:

cars[0] = "Toyota"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array3)

## The Length of an Array

Use the len() method to return the length of an array (the number of elements in an array).

### Example

Return the number of elements in the cars array:

x = len(cars)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array4)

**Note:** The length of an array is always one more than the highest array index.

ADVERTISEMENT

## Looping Array Elements

You can use the for in loop to loop through all the elements of an array.

### Example

Print each item in the cars array:

for x in cars:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array5)

## Adding Array Elements

You can use the append() method to add an element to an array.

### Example

Add one more element to the cars array:

cars.append("Honda")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array6)

## Removing Array Elements

You can use the pop() method to remove an element from the array.

### Example

Delete the second element of the cars array:

cars.pop(1)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array7)

You can also use the remove() method to remove an element from the array.

### Example

Delete the element that has the value "Volvo":

cars.remove("Volvo")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_array8)

**Note:** The list's remove() method only removes the first occurrence of the specified value.

## Array Methods

Python has a set of built-in methods that you can use on lists/arrays.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the first item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

**Note:** Python does not have built-in support for Arrays, but Python Lists can be used instead.

[❮ Previous](https://www.w3schools.com/python/python_lambda.asp)[Log in to track progress](https://www.w3schools.com/signup/?utm_source=classic&utm_medium=python_tutorial&utm_campaign=button_lower_navigation)[Next ❯](https://www.w3schools.com/python/python_classes.asp)

[**Skip to content**](https://www.freecodecamp.org/learn/scientific-computing-with-python/#content-start)

Top of Form

SearchSubmit search terms

Bottom of Form

Menu

**Scientific Computing with Python**

Python is one of the most popular, flexible programming languages today. You can use it for everything from basic scripting to machine learning.

In the Scientific Computing with Python Certification, you'll learn Python fundamentals like variables, loops, conditionals, and functions. Then you'll quickly ramp up to complex data structures, networking, relational databases, and data visualization.

**Courses**

**Python for Everybody**

Python for everybody is a free video course series that teaches the basics of using Python 3.

The courses were created by Dr. Charles Severance (also known as Dr. Chuck). He is a Clinical Professor at the University of Michigan School of Information, where he teaches various technology-oriented courses including programming, database design, and web development.

Collapse coursePython for Everybody

0/56, 0 of 56 challenges completed

**Scientific Computing with Python Projects**

Time to put your Python skills to the test. By completing these projects, you will demonstrate that you have a good foundational knowledge of Python and qualify for the Scientific Computing with Python Certification.

* [Arithmetic FormatterNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/arithmetic-formatter)
* [Time CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/time-calculator)
* [Budget AppNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/budget-app)
* [Polygon Area CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/polygon-area-calculator)
* [Probability CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/probability-calculator)

[Go to settings to claim your certificationScientific Computing with Python](https://www.freecodecamp.org/settings#certification-settings)

**Browse our other free certifications (we recommend doing these in order)**

**Earn free verified certifications with freeCodeCamp's core curriculum:**

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

**Trending Guides**

[**Skip to content**](https://www.freecodecamp.org/learn/2022/responsive-web-design/#content-start)

Top of Form

SearchSubmit search terms

Bottom of Form

Menu

**Responsive Web Design**

In this Responsive Web Design Certification, you'll learn the languages that developers use to build webpages: HTML (Hypertext Markup Language) for content, and CSS (Cascading Style Sheets) for design.

First, you'll build a cat photo app to learn the basics of HTML and CSS. Later, you'll learn modern techniques like CSS variables by building a penguin, and best practices for accessibility by building a quiz site.

Finally, you'll learn how to make webpages that respond to different screen sizes by building a photo gallery with Flexbox, and a magazine article layout with CSS Grid.

Note: Some browser extensions, such as ad-blockers and dark mode extensions can interfere with the tests. If you face issues, we recommend disabling extensions that modify the content or layout of pages, while taking the course.

**Courses**

**Learn HTML by Building a Cat Photo App, Not started**

HTML tags give a webpage its structure. You can use HTML tags to add photos, buttons, and other elements to your webpage.

In this course, you'll learn the most common HTML tags by building your own cat photo app.

[Start projectLearn HTML by Building a Cat Photo App](https://www.freecodecamp.org/learn/2022/responsive-web-design/learn-html-by-building-a-cat-photo-app/step-1)

**Learn Basic CSS by Building a Cafe Menu, Not started**

**Learn CSS Colors by Building a Set of Colored Markers, Not started**

**Learn HTML Forms by Building a Registration Form, Not started**

Certification Project

[**Survey FormCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-survey-form-project/build-a-survey-form)

This is one of the required projects to earn your certification.

For this project, you will build a survey form to collect data from your users.

**Learn the CSS Box Model by Building a Rothko Painting, Not started**

**Learn CSS Flexbox by Building a Photo Gallery, Not started**

**Learn Typography by Building a Nutrition Label, Not started**

**Learn Accessibility by Building a Quiz, Not started**

Certification Project

[**Tribute PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-tribute-page-project/build-a-tribute-page)

This is one of the required projects to earn your certification.

For this project, you will build a tribute page for a subject of your choosing, fictional or real.

**Learn More About CSS Pseudo Selectors By Building A Balance Sheet, Not started**

**Learn Intermediate CSS by Building a Cat Painting, Not started**

**Learn Responsive Web Design by Building a Piano, Not started**

Certification Project

[**Technical Documentation PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-technical-documentation-page-project/build-a-technical-documentation-page)

This is one of the required projects to earn your certification.

For this project, you will build a technical documentation page to serve as instruction or reference for a topic.

**Learn CSS Variables by Building a City Skyline, Not started**

**Learn CSS Grid by Building a Magazine, Not started**

Certification Project

[**Product Landing PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-product-landing-page-project/build-a-product-landing-page)

This is one of the required projects to earn your certification.

For this project, you will build a product landing page to market a product of your choice.

**Learn CSS Animation by Building a Ferris Wheel, Not started**

**Learn CSS Transforms by Building a Penguin, Not started**

Certification Project

[**Personal Portfolio WebpageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-personal-portfolio-webpage-project/build-a-personal-portfolio-webpage)

This is one of the required projects to earn your certification.

For this project, you will build your own personal portfolio page.

[Go to settings to claim your certificationResponsive Web Design](https://www.freecodecamp.org/settings#certification-settings)

**Browse our other free certifications (we recommend doing these in order)**

**Earn free verified certifications with freeCodeCamp's core curriculum:**

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

**JavaScript Algorithms and Data Structures**

While HTML and CSS control the content and styling of a page, JavaScript is used to make it interactive. In the JavaScript Algorithm and Data Structures Certification, you'll learn the fundamentals of JavaScript including variables, arrays, objects, loops, and functions.

Once you have the fundamentals down, you'll apply that knowledge by creating algorithms to manipulate strings, factorialize numbers, and even calculate the orbit of the International Space Station.

Along the way, you'll also learn two important programming styles or paradigms: Object Oriented Programming (OOP) and Functional Programming (FP).

Note: Some browser extensions, such as ad-blockers and script-blockers can interfere with the tests. If you face issues, we recommend disabling extensions that modify or block the content of pages while taking the course.

**Courses**

**Basic JavaScript**

JavaScript is a scripting language you can use to make web pages interactive. It is one of the core technologies of the web, along with HTML and CSS, and is supported by all modern browsers.

In this course, you'll learn fundamental programming concepts in JavaScript. You'll start with basic data structures like numbers and strings. Then you'll learn to work with arrays, objects, functions, loops, if/else statements, and more.

Collapse courseBasic JavaScript

0/113, 0 of 113 challenges completed

**ES6**

ECMAScript, or ES, is a standardized version of JavaScript. Because all major browsers follow this specification, the terms ECMAScript and JavaScript are interchangeable.

Most of the JavaScript you've learned up to this point was in ES5 (ECMAScript 5), which was finalized in 2009. While you can still write programs in ES5, JavaScript is constantly evolving, and new features are released every year.

ES6, released in 2015, added many powerful new features to the language. In this course, you'll learn these new features, including arrow functions, destructuring, classes, promises, and modules.

Expand courseES6

0/29, 0 of 29 challenges completed

**Regular Expressions**

Regular expressions, often shortened to "regex" or "regexp", are patterns that help programmers match, search, and replace text. Regular expressions are very powerful, but can be hard to read because they use special characters to make more complex, flexible matches.

In this course, you'll learn how to use special characters, capture groups, positive and negative lookaheads, and other techniques to match any text you want.

Expand courseRegular Expressions

0/33, 0 of 33 challenges completed

**Debugging**

Debugging is the process of going through your code, finding any issues, and fixing them.

Issues in code generally come in three forms: syntax errors that prevent your program from running, runtime errors where your code has unexpected behavior, or logical errors where your code doesn't do what you intended.

In this course, you'll learn how to use the JavaScript console to debug programs and prevent common issues before they happen.

Expand courseDebugging

0/12, 0 of 12 challenges completed

**Basic Data Structures**

Data can be stored and accessed in many ways. You already know some common JavaScript data structures — arrays and objects.

In this Basic Data Structures course, you'll learn more about the differences between arrays and objects, and which to use in different situations. You'll also learn how to use helpful JS methods like splice() and Object.keys() to access and manipulate data.

Expand courseBasic Data Structures

0/20, 0 of 20 challenges completed

**Basic Algorithm Scripting**

An algorithm is a series of step-by-step instructions that describe how to do something.

To write an effective algorithm, it helps to break a problem down into smaller parts and think carefully about how to solve each part with code.

In this course, you'll learn the fundamentals of algorithmic thinking by writing algorithms that do everything from converting temperatures to handling complex 2D arrays.

Expand courseBasic Algorithm Scripting

0/16, 0 of 16 challenges completed

**Object Oriented Programming**

OOP, or Object Oriented Programming, is one of the major approaches to the software development process. In OOP, objects and classes organize code to describe things and what they can do.

In this course, you'll learn the basic principles of OOP in JavaScript, including the this keyword, prototype chains, constructors, and inheritance.

Expand courseObject Oriented Programming

0/26, 0 of 26 challenges completed

**Functional Programming**

Functional Programming is another popular approach to software development. In Functional Programming, code is organized into smaller, basic functions that can be combined to build complex programs.

In this course, you'll learn the core concepts of Functional Programming including pure functions, how to avoid mutations, and how to write cleaner code with methods like .map() and .filter().

Expand courseFunctional Programming

0/24, 0 of 24 challenges completed

**Intermediate Algorithm Scripting**

Now that you know the basics of algorithmic thinking, along with OOP and Functional Programming, test your skills with the Intermediate Algorithm Scripting challenges.

Expand courseIntermediate Algorithm Scripting

0/21, 0 of 21 challenges completed

**JavaScript Algorithms and Data Structures Projects**

This is it — time to put your new JavaScript skills to work. These projects are similar to the algorithm scripting challenges you've done before – just much more difficult.

Complete these 5 JavaScript projects to earn the JavaScript Algorithms and Data Structures certification.

**Browse our other free certifications (we recommend doing these in order)**

**Earn free verified certifications with freeCodeCamp's core curriculum:**

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

**You can**[**make a tax-deductible donation here**](https://www.freecodecamp.org/donate)**.**

**Trending Guides**

**Scientific Computing with Python**

Python is one of the most popular, flexible programming languages today. You can use it for everything from basic scripting to machine learning.

In the Scientific Computing with Python Certification, you'll learn Python fundamentals like variables, loops, conditionals, and functions. Then you'll quickly ramp up to complex data structures, networking, relational databases, and data visualization.

**Courses**

**Python for Everybody**

Python for everybody is a free video course series that teaches the basics of using Python 3.

The courses were created by Dr. Charles Severance (also known as Dr. Chuck). He is a Clinical Professor at the University of Michigan School of Information, where he teaches various technology-oriented courses including programming, database design, and web development.

Collapse coursePython for Everybody

0/56, 0 of 56 challenges completed

**Scientific Computing with Python Projects**

Time to put your Python skills to the test. By completing these projects, you will demonstrate that you have a good foundational knowledge of Python and qualify for the Scientific Computing with Python Certification.

**Browse our other free certifications (we recommend doing these in order)**

**Earn free verified certifications with freeCodeCamp's core curriculum:**

**Earn free professional certifications:**

[(New) Foundational C# with Microsoft Certification](https://www.freecodecamp.org/learn/foundational-c-sharp-with-microsoft/)

**Prepare for the developer interview job search:**

[Coding Interview Prep](https://www.freecodecamp.org/learn/coding-interview-prep/)

[Project Euler](https://www.freecodecamp.org/learn/project-euler/)

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

import React from 'react';  
import {Text, View} from 'react-native';  
import {Header} from './Header';  
import {heading} from './Typography';  
  
const WelcomeScreen = () => (  
 <View>  
 <Header title="Welcome to React Native"/>  
 <Text style={heading}>Step One</Text>  
 <Text>  
 Edit App.js to change this screen and turn it  
 into your app.  
 </Text>  
 <Text style={heading}>See Your Changes</Text>  
 <Text>  
 Press Cmd + R inside the simulator to reload  
 your app’s code.  
 </Text>  
 <Text style={heading}>Debug</Text>  
 <Text>  
 Press Cmd + M or Shake your device to open the  
 React Native Debug Menu.  
 </Text>  
 <Text style={heading}>Learn</Text>  
 <Text>  
 Read the docs to discover what to do next:  
 </Text>  
 </View>  
);

operator = input("Enter an operator [+ - \* /]: ")

if operator in ('+', '-', '\*', '/'):

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

if operator == '+':

result = num1 + num2

print("Sum is " + str(result))

elif operator == '-':

result = num1 - num2

print("Difference is " + str(result))

elif operator == '\*':

result = num1 \* num2

print("Product is " + str(result))

elif operator == '/':

if num2 != 0:

result = num1 / num2

print("Division is " + str(result))

else:

print("Division by zero is not allowed.")

else:

print("Invalid operator. Please enter +, -, \*, or /.")

-------------------------------------------------------------------------------------------------------------------------------

|  |
| --- |
| **# git\_and\_github\_activity** |
|  |  |  |
|  |  | \*\*Activity Title: Introduction to Git and GitHub\*\* |
|  |  |  |
|  |  | \*\*Objective:\*\* In this activity, you will learn the basics of version control using Git and how to collaborate with others on GitHub. |
|  |  |  |
|  |  | \*\*Prerequisites:\*\* |
|  |  | - Create a GitHub account (if you don't have one already). |
|  |  | - Install Git on your local machine. |
|  |  |  |
|  |  | \*\*Instructions:\*\* |
|  |  |  |
|  |  | \*\*Step 1: Setting up a Local Git Repository\*\* |
|  |  |  |
|  |  | 1. Open a terminal or Git Bash. |
|  |  | 2. Navigate to the directory where you want to create your project. |
|  |  | 3. Initialize a new Git repository with the following command: |
|  |  | ``` |
|  |  | git init |
|  |  | ``` |
|  |  | 4. Create a new file (e.g., `pyfun.py`) and add some content to it. |
|  |  | 5. Add this file to your Git repository: |
|  |  | ``` |
|  |  | git add pyfun.py |
|  |  | ``` |
|  |  | 6. Commit your changes with a meaningful message: |
|  |  | ``` |
|  |  | git commit -m "Initial commit" |
|  |  | ``` |
|  |  |  |
|  |  | \*\*Step 2: Creating a GitHub Repository\*\* |
|  |  |  |
|  |  | 1. Go to your GitHub account and click on the '+' sign in the top right corner, then select "New Repository." |
|  |  | 2. Give your repository a name, choose public or private, and add a README file. |
|  |  | 3. Click "Create repository." |
|  |  |  |
|  |  | \*\*Step 3: Linking Local and Remote Repositories\*\* |
|  |  |  |
|  |  | 1. In your local terminal, add the remote repository URL you just created on GitHub: |
|  |  | ``` |
|  |  | git remote add origin <your-repo-url> |
|  |  | ``` |
|  |  | 2. Push your local repository to GitHub: |
|  |  | ``` |
|  |  | git push -u origin main |
|  |  | ``` |
|  |  |  |
|  |  | \*\*Step 4: Collaboration with Others\*\* |
|  |  |  |
|  |  | 1. Go to your GitHub repository on the GitHub website. |
|  |  | 2. Click on the "Settings" tab. |
|  |  | 3. Under "Manage access," click "Invite a collaborator" and invite a friend (using their GitHub username or email). |
|  |  | 4. Your collaborator should accept the invitation. |
|  |  | 5. Both you and your collaborator can now make changes to the project and collaborate using Git and GitHub. |
|  |  |  |
|  |  | \*\*Step 5: Making Changes and Pull Requests\*\* |
|  |  |  |
|  |  | 1. Make changes to your project locally and commit them. |
|  |  | 2. Push your changes to GitHub. |
|  |  | 3. In your GitHub repository, click on "New Pull Request" to suggest changes to the original repository. |
|  |  | 4. Review and merge the changes. |
|  |  |  |
|  |  | \*\*Step 6: Using Branches\*\* |
|  |  |  |
|  |  | 1. Create a new branch in your local repository: |
|  |  | ``` |
|  |  | git checkout -b new-feature |
|  |  | ``` |
|  |  | 2. Make changes in this branch, commit, and push them to GitHub. |
|  |  | 3. Create a pull request to merge the new branch into the master branch. |
|  |  |  |
|  |  | \*\*Step 7: Git and GitHub Best Practices\*\* |
|  |  |  |
|  |  | - Explore branching strategies (feature branches, hotfixes, etc.). |
|  |  | - Use meaningful commit messages. |
|  |  | - Regularly update your local repository with changes from the remote repository (`git pull`). |
|  |  | - Collaborate with your partner to practice working on the same codebase. |
|  |  |  |
|  |  | \*\*Conclusion:\*\* |
|  |  | Congratulations, you've completed this Git and GitHub activity for beginners! You've learned how to set up a Git repository, collaborate with others, and use essential Git and GitHub commands. |

 45 changes: 45 additions & 0 deletions45  [pyfun\_01/README.md](https://github.com/Pelino-Courses/pyfun-01-Obed200/commit/2ebb8c124b4e96ec2c3b2d1233edaecb76810ee6#diff-4a943a592e696bcb75bcb1ab01eb236aaeca4e40f36a885de3dbd59771e808ca)

Top of Form

Bottom of Form

Top of Form

Bottom of Form

|  |  |  |
| --- | --- | --- |
|  |  | @@ -0,0 +1,45 @@ |
|  |  | \*\*Activity Title:\*\* PyFun 01 Challenge |
|  |  |  |
|  |  | \*\*Activity Description:\*\* |
|  |  | In this Python activity, learners will dive into the world of Python fundamentals by working on a series of challenges that revolve around variables and types, string manipulation, type conversion, and getting user input. |
|  |  | The goal is to improve their problem-solving skills and reinforce their understanding of these essential Python concepts. |
|  |  |  |
|  |  | \*\*Instructions:\*\* |
|  |  |  |
|  |  | \*\*Step 1: Variable Magic\*\* |
|  |  |  |
|  |  | Create a Python program that performs the following tasks: |
|  |  | - Define three variables: `num1`, `num2`, and `result`. |
|  |  | - Assign integer values to `num1` and `num2`. |
|  |  | - Calculate the sum of `num1` and `num2` and store it in `result`. |
|  |  | - Print the result in the following format: "The sum of [num1] and [num2] is [result]." |
|  |  |  |
|  |  | \*\*Step 2: String Shuffle\*\* |
|  |  |  |
|  |  | Write a Python program that does the following: |
|  |  | - Create a string variable `original\_string` with any text you like. |
|  |  | - Reverse the original string and store it in a new variable called `reversed\_string`. |
|  |  | - Print the reversed string. |
|  |  |  |
|  |  | \*\*Step 3: Type Transformer\*\* |
|  |  |  |
|  |  | Build a Python program that demonstrates the concept of type conversion. Follow these steps: |
|  |  | - Create a variable `num\_str` and assign a number in string format, e.g., "42". |
|  |  | - Convert `num\_str` into an integer and assign it to a new variable `num\_int`. |
|  |  | - Print `num\_int` and its data type. |
|  |  | - Create a float variable `num\_float` and assign it the value 3.14. |
|  |  | - Convert `num\_float` into a string and store it in `num\_str2`. |
|  |  | - Print `num\_str2` and its data type. |
|  |  |  |
|  |  | \*\*Step 4: User's Insights\*\* |
|  |  |  |
|  |  | Create an interactive Python program where the user is prompted to provide their name and age. |
|  |  | Then, the program should display a message tailored to the user's input. Here's how: |
|  |  | - Ask the user to enter their name. |
|  |  | - Ask the user to enter their age. |
|  |  | - Using the user's input, print a message like "Hello, [name]! You are [age] years old." |
|  |  |  |
|  |  | \*\*Bonus Challenge: String Slicer\*\* |
|  |  |  |
|  |  | For an extra challenge, modify the "String Shuffle" program from Step 2 to slice the original string into two parts, |
|  |  | reverse each part, and then join them back together. Print the final, reversed string.  **Responsive Web Design**  In this Responsive Web Design Certification, you'll learn the languages that developers use to build webpages: HTML (Hypertext Markup Language) for content, and CSS (Cascading Style Sheets) for design.  First, you'll build a cat photo app to learn the basics of HTML and CSS. Later, you'll learn modern techniques like CSS variables by building a penguin, and best practices for accessibility by building a quiz site.  Finally, you'll learn how to make webpages that respond to different screen sizes by building a photo gallery with Flexbox, and a magazine article layout with CSS Grid.  Note: Some browser extensions, such as ad-blockers and dark mode extensions can interfere with the tests. If you face issues, we recommend disabling extensions that modify the content or layout of pages, while taking the course.  **Courses**  **Learn HTML by Building a Cat Photo App, Not started**  HTML tags give a webpage its structure. You can use HTML tags to add photos, buttons, and other elements to your webpage.  In this course, you'll learn the most common HTML tags by building your own cat photo app.  [Start projectLearn HTML by Building a Cat Photo App](https://www.freecodecamp.org/learn/2022/responsive-web-design/learn-html-by-building-a-cat-photo-app/step-1)  **Learn Basic CSS by Building a Cafe Menu, Not started**  **Learn CSS Colors by Building a Set of Colored Markers, Not started**  **Learn HTML Forms by Building a Registration Form, Not started**  Certification Project  [**Survey FormCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-survey-form-project/build-a-survey-form)  This is one of the required projects to earn your certification.  For this project, you will build a survey form to collect data from your users.  **Learn the CSS Box Model by Building a Rothko Painting, Not started**  **Learn CSS Flexbox by Building a Photo Gallery, Not started**  **Learn Typography by Building a Nutrition Label, Not started**  **Learn Accessibility by Building a Quiz, Not started**  Certification Project  [**Tribute PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-tribute-page-project/build-a-tribute-page)  This is one of the required projects to earn your certification.  For this project, you will build a tribute page for a subject of your choosing, fictional or real.  **Learn More About CSS Pseudo Selectors By Building A Balance Sheet, Not started**  **Learn Intermediate CSS by Building a Cat Painting, Not started**  **Learn Responsive Web Design by Building a Piano, Not started**  Certification Project  [**Technical Documentation PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-technical-documentation-page-project/build-a-technical-documentation-page)  This is one of the required projects to earn your certification.  For this project, you will build a technical documentation page to serve as instruction or reference for a topic.  **Learn CSS Variables by Building a City Skyline, Not started**  **Learn CSS Grid by Building a Magazine, Not started**  Certification Project  [**Product Landing PageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-product-landing-page-project/build-a-product-landing-page)  This is one of the required projects to earn your certification.  For this project, you will build a product landing page to market a product of your choice.  **Learn CSS Animation by Building a Ferris Wheel, Not started**  **Learn CSS Transforms by Building a Penguin, Not started**  Certification Project  [**Personal Portfolio WebpageCertification Project**](https://www.freecodecamp.org/learn/2022/responsive-web-design/build-a-personal-portfolio-webpage-project/build-a-personal-portfolio-webpage)  This is one of the required projects to earn your certification.  For this project, you will build your own personal portfolio page.  [Go to settings to claim your certificationResponsive Web Design](https://www.freecodecamp.org/settings#certification-settings)  **Browse our other free certifications (we recommend doing these in order)**  **Earn free verified certifications with freeCodeCamp's core curriculum:**  freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)  Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.  Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.  **You can**[**make a tax-deductible donation here**](https://www.freecodecamp.org/donate)**.**  **Trending Guides**  [JS isEmpty Equivalent](https://www.freecodecamp.org/news/check-if-an-object-is-empty-in-javascript/)[Submit a Form with JS](https://www.freecodecamp.org/news/how-to-submit-a-form-with-javascript/)[Add to List in Python](https://www.freecodecamp.org/news/how-to-add-to-a-list-in-python-list-addition-tutorial/)[Grep Command in Linux](https://www.freecodecamp.org/news/grep-command-in-linux-usage-options-and-syntax-examples/)[String to Int in Java](https://www.freecodecamp.org/news/how-to-convert-a-string-to-an-integer-in-java/)[Add to Dict in Python](https://www.freecodecamp.org/news/adding-to-a-dict-in-python-how-to-add-to-a-dictionary/)[Java For Loop Example](https://www.freecodecamp.org/news/java-for-loop-example/)[Matplotlib Figure Size](https://www.freecodecamp.org/news/matplotlib-figure-size-change-plot-size-in-python/)[Database Normalization](https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/)[Nested Lists in Python](https://www.freecodecamp.org/news/list-within-a-list-in-python-initialize-a-nested-list/)[Coalesce SQL](https://www.freecodecamp.org/news/coalesce-sql-example-postgresql-and-sql-server-functions/)[Python join()](https://www.freecodecamp.org/news/python-join-how-to-combine-a-list-into-a-string-in-python/)[JS POST Request](https://www.freecodecamp.org/news/javascript-post-request-how-to-send-an-http-post-request-in-js/)[JS Type Checking](https://www.freecodecamp.org/news/javascript-type-checking-how-to-check-type-in-js-with-typeof/)[Read Python File](https://www.freecodecamp.org/news/how-to-read-a-file-line-by-line-in-python/)[SOLID Principles](https://www.freecodecamp.org/news/solid-design-principles-in-software-development/)[Sort a List in Java](https://www.freecodecamp.org/news/how-to-sort-a-list-in-java/)[For Loops in Python](https://www.freecodecamp.org/news/for-loops-in-python-with-example-code/)[JavaScript 2D Array](https://www.freecodecamp.org/news/javascript-2d-arrays/)[SQL CONVERT Function](https://www.freecodecamp.org/news/sql-convert-the-date-to-string-or-datetime-function-2/)[Rename Column in Pandas](https://www.freecodecamp.org/news/how-to-rename-a-column-in-pandas/)[Delete a File in Python](https://www.freecodecamp.org/news/how-to-delete-a-file-in-python-and-remove-a-directory-too/)[K-Nearest Neighbors Algo](https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/)[iferror Function in Excel](https://www.freecodecamp.org/news/iferror-function-in-excel-example/)[Remove From String Python](https://www.freecodecamp.org/news/remove-from-string-in-python-how-to-remove-characters-from-a-string/)[Create a File in Terminal](https://www.freecodecamp.org/news/how-to-make-a-file-in-linux-from-the-command-line-create-a-file-in-terminal/)[Clear Formatting in Excel](https://www.freecodecamp.org/news/how-to-clear-formatting-in-excel/)[Accounting Num Format Excel](https://www.freecodecamp.org/news/accounting-number-format-in-excel-how-to-apply-it-to-selected-cells/)[Check if File Exists Python](https://www.freecodecamp.org/news/how-to-check-if-a-file-exists-in-python/)[Iterate Over Dict in Python](https://www.freecodecamp.org/news/dictionary-iteration-in-python/)  [About](https://www.freecodecamp.org/news/about/)[Alumni Network](https://www.linkedin.com/school/free-code-camp/people/)[Open Source](https://github.com/freeCodeCamp/)[Shop](https://www.freecodecamp.org/shop/)[Support](https://www.freecodecamp.org/news/support/)[Sponsors](https://www.freecodecamp.org/news/sponsors/)[Academic Honesty](https://www.freecodecamp.org/news/academic-honesty-policy/)[Code of Conduct](https://www.freecodecamp.org/news/code-of-conduct/)[Privacy Policy](https://www.freecodecamp.org/news/privacy-policy/)[Terms of Service](https://www.freecodecamp.org/news/terms-of-service/)[Copyright Policy](https://www.freecodecamp.org/news/copyright-policy/)  Navigated to Responsive Web Design  **JavaScript Algorithms and Data Structures**  While HTML and CSS control the content and styling of a page, JavaScript is used to make it interactive. In the JavaScript Algorithm and Data Structures Certification, you'll learn the fundamentals of JavaScript including variables, arrays, objects, loops, and functions.  Once you have the fundamentals down, you'll apply that knowledge by creating algorithms to manipulate strings, factorialize numbers, and even calculate the orbit of the International Space Station.  Along the way, you'll also learn two important programming styles or paradigms: Object Oriented Programming (OOP) and Functional Programming (FP).  Note: Some browser extensions, such as ad-blockers and script-blockers can interfere with the tests. If you face issues, we recommend disabling extensions that modify or block the content of pages while taking the course.  **Courses**  **Basic JavaScript**  JavaScript is a scripting language you can use to make web pages interactive. It is one of the core technologies of the web, along with HTML and CSS, and is supported by all modern browsers.  In this course, you'll learn fundamental programming concepts in JavaScript. You'll start with basic data structures like numbers and strings. Then you'll learn to work with arrays, objects, functions, loops, if/else statements, and more.  Collapse courseBasic JavaScript  0/113, 0 of 113 challenges completed   * [Not PassedComment Your JavaScript Code](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comment-your-javascript-code) * [Not PassedDeclare JavaScript Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/declare-javascript-variables) * [Not PassedStoring Values with the Assignment Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/storing-values-with-the-assignment-operator) * [Not PassedAssigning the Value of One Variable to Another](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/assigning-the-value-of-one-variable-to-another) * [Not PassedInitializing Variables with the Assignment Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/initializing-variables-with-the-assignment-operator) * [Not PassedDeclare String Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/declare-string-variables) * [Not PassedUnderstanding Uninitialized Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understanding-uninitialized-variables) * [Not PassedUnderstanding Case Sensitivity in Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understanding-case-sensitivity-in-variables) * [Not PassedExplore Differences Between the var and let Keywords](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/explore-differences-between-the-var-and-let-keywords) * [Not PassedDeclare a Read-Only Variable with the const Keyword](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/declare-a-read-only-variable-with-the-const-keyword) * [Not PassedAdd Two Numbers with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/add-two-numbers-with-javascript) * [Not PassedSubtract One Number from Another with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/subtract-one-number-from-another-with-javascript) * [Not PassedMultiply Two Numbers with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/multiply-two-numbers-with-javascript) * [Not PassedDivide One Number by Another with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/divide-one-number-by-another-with-javascript) * [Not PassedIncrement a Number with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/increment-a-number-with-javascript) * [Not PassedDecrement a Number with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/decrement-a-number-with-javascript) * [Not PassedCreate Decimal Numbers with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/create-decimal-numbers-with-javascript) * [Not PassedMultiply Two Decimals with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/multiply-two-decimals-with-javascript) * [Not PassedDivide One Decimal by Another with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/divide-one-decimal-by-another-with-javascript) * [Not PassedFinding a Remainder in JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/finding-a-remainder-in-javascript) * [Not PassedCompound Assignment With Augmented Addition](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/compound-assignment-with-augmented-addition) * [Not PassedCompound Assignment With Augmented Subtraction](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/compound-assignment-with-augmented-subtraction) * [Not PassedCompound Assignment With Augmented Multiplication](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/compound-assignment-with-augmented-multiplication) * [Not PassedCompound Assignment With Augmented Division](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/compound-assignment-with-augmented-division) * [Not PassedEscaping Literal Quotes in Strings](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/escaping-literal-quotes-in-strings) * [Not PassedQuoting Strings with Single Quotes](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/quoting-strings-with-single-quotes) * [Not PassedEscape Sequences in Strings](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/escape-sequences-in-strings) * [Not PassedConcatenating Strings with Plus Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/concatenating-strings-with-plus-operator) * [Not PassedConcatenating Strings with the Plus Equals Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/concatenating-strings-with-the-plus-equals-operator) * [Not PassedConstructing Strings with Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/constructing-strings-with-variables) * [Not PassedAppending Variables to Strings](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/appending-variables-to-strings) * [Not PassedFind the Length of a String](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/find-the-length-of-a-string) * [Not PassedUse Bracket Notation to Find the First Character in a String](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-bracket-notation-to-find-the-first-character-in-a-string) * [Not PassedUnderstand String Immutability](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understand-string-immutability) * [Not PassedUse Bracket Notation to Find the Nth Character in a String](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-bracket-notation-to-find-the-nth-character-in-a-string) * [Not PassedUse Bracket Notation to Find the Last Character in a String](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-bracket-notation-to-find-the-last-character-in-a-string) * [Not PassedUse Bracket Notation to Find the Nth-to-Last Character in a String](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-bracket-notation-to-find-the-nth-to-last-character-in-a-string) * [Not PassedWord Blanks](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/word-blanks) * [Not PassedStore Multiple Values in one Variable using JavaScript Arrays](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/store-multiple-values-in-one-variable-using-javascript-arrays) * [Not PassedNest one Array within Another Array](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/nest-one-array-within-another-array) * [Not PassedAccess Array Data with Indexes](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/access-array-data-with-indexes) * [Not PassedModify Array Data With Indexes](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/modify-array-data-with-indexes) * [Not PassedAccess Multi-Dimensional Arrays With Indexes](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/access-multi-dimensional-arrays-with-indexes) * [Not PassedManipulate Arrays With push Method](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/manipulate-arrays-with-push) * [Not PassedManipulate Arrays With pop Method](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/manipulate-arrays-with-pop) * [Not PassedManipulate Arrays With shift Method](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/manipulate-arrays-with-shift) * [Not PassedManipulate Arrays With unshift Method](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/manipulate-arrays-with-unshift) * [Not PassedShopping List](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/shopping-list) * [Not PassedWrite Reusable JavaScript with Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/write-reusable-javascript-with-functions) * [Not PassedPassing Values to Functions with Arguments](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/passing-values-to-functions-with-arguments) * [Not PassedReturn a Value from a Function with Return](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/return-a-value-from-a-function-with-return) * [Not PassedGlobal Scope and Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/global-scope-and-functions) * [Not PassedLocal Scope and Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/local-scope-and-functions) * [Not PassedGlobal vs. Local Scope in Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/global-vs--local-scope-in-functions) * [Not PassedUnderstanding Undefined Value returned from a Function](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understanding-undefined-value-returned-from-a-function) * [Not PassedAssignment with a Returned Value](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/assignment-with-a-returned-value) * [Not PassedStand in Line](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/stand-in-line) * [Not PassedUnderstanding Boolean Values](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/understanding-boolean-values) * [Not PassedUse Conditional Logic with If Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-conditional-logic-with-if-statements) * [Not PassedComparison with the Equality Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-equality-operator) * [Not PassedComparison with the Strict Equality Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-strict-equality-operator) * [Not PassedPractice comparing different values](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/practice-comparing-different-values) * [Not PassedComparison with the Inequality Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-inequality-operator) * [Not PassedComparison with the Strict Inequality Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-strict-inequality-operator) * [Not PassedComparison with the Greater Than Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-greater-than-operator) * [Not PassedComparison with the Greater Than Or Equal To Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-greater-than-or-equal-to-operator) * [Not PassedComparison with the Less Than Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-less-than-operator) * [Not PassedComparison with the Less Than Or Equal To Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparison-with-the-less-than-or-equal-to-operator) * [Not PassedComparisons with the Logical And Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparisons-with-the-logical-and-operator) * [Not PassedComparisons with the Logical Or Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/comparisons-with-the-logical-or-operator) * [Not PassedIntroducing Else Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/introducing-else-statements) * [Not PassedIntroducing Else If Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/introducing-else-if-statements) * [Not PassedLogical Order in If Else Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/logical-order-in-if-else-statements) * [Not PassedChaining If Else Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/chaining-if-else-statements) * [Not PassedGolf Code](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/golf-code) * [Not PassedSelecting from Many Options with Switch Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/selecting-from-many-options-with-switch-statements) * [Not PassedAdding a Default Option in Switch Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/adding-a-default-option-in-switch-statements) * [Not PassedMultiple Identical Options in Switch Statements](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/multiple-identical-options-in-switch-statements) * [Not PassedReplacing If Else Chains with Switch](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/replacing-if-else-chains-with-switch) * [Not PassedReturning Boolean Values from Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/returning-boolean-values-from-functions) * [Not PassedReturn Early Pattern for Functions](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/return-early-pattern-for-functions) * [Not PassedCounting Cards](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/counting-cards) * [Not PassedBuild JavaScript Objects](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/build-javascript-objects) * [Not PassedAccessing Object Properties with Dot Notation](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-object-properties-with-dot-notation) * [Not PassedAccessing Object Properties with Bracket Notation](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-object-properties-with-bracket-notation) * [Not PassedAccessing Object Properties with Variables](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-object-properties-with-variables) * [Not PassedUpdating Object Properties](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/updating-object-properties) * [Not PassedAdd New Properties to a JavaScript Object](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/add-new-properties-to-a-javascript-object) * [Not PassedDelete Properties from a JavaScript Object](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/delete-properties-from-a-javascript-object) * [Not PassedUsing Objects for Lookups](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/using-objects-for-lookups) * [Not PassedTesting Objects for Properties](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/testing-objects-for-properties) * [Not PassedManipulating Complex Objects](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/manipulating-complex-objects) * [Not PassedAccessing Nested Objects](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-nested-objects) * [Not PassedAccessing Nested Arrays](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/accessing-nested-arrays) * [Not PassedRecord Collection](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/record-collection) * [Not PassedIterate with JavaScript While Loops](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/iterate-with-javascript-while-loops) * [Not PassedIterate with JavaScript For Loops](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/iterate-with-javascript-for-loops) * [Not PassedIterate Odd Numbers With a For Loop](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/iterate-odd-numbers-with-a-for-loop) * [Not PassedCount Backwards With a For Loop](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/count-backwards-with-a-for-loop) * [Not PassedIterate Through an Array with a For Loop](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/iterate-through-an-array-with-a-for-loop) * [Not PassedNesting For Loops](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/nesting-for-loops) * [Not PassedIterate with JavaScript Do...While Loops](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/iterate-with-javascript-do---while-loops) * [Not PassedReplace Loops using Recursion](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/replace-loops-using-recursion) * [Not PassedProfile Lookup](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/profile-lookup) * [Not PassedGenerate Random Fractions with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/generate-random-fractions-with-javascript) * [Not PassedGenerate Random Whole Numbers with JavaScript](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/generate-random-whole-numbers-with-javascript) * [Not PassedGenerate Random Whole Numbers within a Range](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/generate-random-whole-numbers-within-a-range) * [Not PassedUse the parseInt Function](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-the-parseint-function) * [Not PassedUse the parseInt Function with a Radix](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-the-parseint-function-with-a-radix) * [Not PassedUse the Conditional (Ternary) Operator](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-the-conditional-ternary-operator) * [Not PassedUse Multiple Conditional (Ternary) Operators](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-multiple-conditional-ternary-operators) * [Not PassedUse Recursion to Create a Countdown](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-recursion-to-create-a-countdown) * [Not PassedUse Recursion to Create a Range of Numbers](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/basic-javascript/use-recursion-to-create-a-range-of-numbers)   **ES6**  ECMAScript, or ES, is a standardized version of JavaScript. Because all major browsers follow this specification, the terms ECMAScript and JavaScript are interchangeable.  Most of the JavaScript you've learned up to this point was in ES5 (ECMAScript 5), which was finalized in 2009. While you can still write programs in ES5, JavaScript is constantly evolving, and new features are released every year.  ES6, released in 2015, added many powerful new features to the language. In this course, you'll learn these new features, including arrow functions, destructuring, classes, promises, and modules.  Expand courseES6  0/29, 0 of 29 challenges completed  **Regular Expressions**  Regular expressions, often shortened to "regex" or "regexp", are patterns that help programmers match, search, and replace text. Regular expressions are very powerful, but can be hard to read because they use special characters to make more complex, flexible matches.  In this course, you'll learn how to use special characters, capture groups, positive and negative lookaheads, and other techniques to match any text you want.  Expand courseRegular Expressions  0/33, 0 of 33 challenges completed  **Debugging**  Debugging is the process of going through your code, finding any issues, and fixing them.  Issues in code generally come in three forms: syntax errors that prevent your program from running, runtime errors where your code has unexpected behavior, or logical errors where your code doesn't do what you intended.  In this course, you'll learn how to use the JavaScript console to debug programs and prevent common issues before they happen.  Expand courseDebugging  0/12, 0 of 12 challenges completed  **Basic Data Structures**  Data can be stored and accessed in many ways. You already know some common JavaScript data structures — arrays and objects.  In this Basic Data Structures course, you'll learn more about the differences between arrays and objects, and which to use in different situations. You'll also learn how to use helpful JS methods like splice() and Object.keys() to access and manipulate data.  Expand courseBasic Data Structures  0/20, 0 of 20 challenges completed  **Basic Algorithm Scripting**  An algorithm is a series of step-by-step instructions that describe how to do something.  To write an effective algorithm, it helps to break a problem down into smaller parts and think carefully about how to solve each part with code.  In this course, you'll learn the fundamentals of algorithmic thinking by writing algorithms that do everything from converting temperatures to handling complex 2D arrays.  Expand courseBasic Algorithm Scripting  0/16, 0 of 16 challenges completed  **Object Oriented Programming**  OOP, or Object Oriented Programming, is one of the major approaches to the software development process. In OOP, objects and classes organize code to describe things and what they can do.  In this course, you'll learn the basic principles of OOP in JavaScript, including the this keyword, prototype chains, constructors, and inheritance.  Expand courseObject Oriented Programming  0/26, 0 of 26 challenges completed  **Functional Programming**  Functional Programming is another popular approach to software development. In Functional Programming, code is organized into smaller, basic functions that can be combined to build complex programs.  In this course, you'll learn the core concepts of Functional Programming including pure functions, how to avoid mutations, and how to write cleaner code with methods like .map() and .filter().  Expand courseFunctional Programming  0/24, 0 of 24 challenges completed  **Intermediate Algorithm Scripting**  Now that you know the basics of algorithmic thinking, along with OOP and Functional Programming, test your skills with the Intermediate Algorithm Scripting challenges.  Expand courseIntermediate Algorithm Scripting  0/21, 0 of 21 challenges completed  **JavaScript Algorithms and Data Structures Projects**  This is it — time to put your new JavaScript skills to work. These projects are similar to the algorithm scripting challenges you've done before – just much more difficult.  Complete these 5 JavaScript projects to earn the JavaScript Algorithms and Data Structures certification.   * [Palindrome CheckerNot Passed](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/javascript-algorithms-and-data-structures-projects/palindrome-checker) * [Roman Numeral ConverterNot Passed](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/javascript-algorithms-and-data-structures-projects/roman-numeral-converter) * [Caesars CipherNot Passed](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/javascript-algorithms-and-data-structures-projects/caesars-cipher) * [Telephone Number ValidatorNot Passed](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/javascript-algorithms-and-data-structures-projects/telephone-number-validator) * [Cash RegisterNot Passed](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/javascript-algorithms-and-data-structures-projects/cash-register)   [Go to settings to claim your certificationJavaScript Algorithms and Data Structures](https://www.freecodecamp.org/settings#certification-settings)  **Browse our other free certifications (we recommend doing these in order)**  **Earn free verified certifications with freeCodeCamp's core curriculum:**  [Responsive Web Design Certification](https://www.freecodecamp.org/learn/2022/responsive-web-design/)  [JavaScript Algorithms and Data Structures Certification](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/)  [Front End Development Libraries Certification](https://www.freecodecamp.org/learn/front-end-development-libraries/)  [Data Visualization Certification](https://www.freecodecamp.org/learn/data-visualization/)  [Relational Database Certification](https://www.freecodecamp.org/learn/relational-database/)  [Back End Development and APIs Certification](https://www.freecodecamp.org/learn/back-end-development-and-apis/)  [Quality Assurance Certification](https://www.freecodecamp.org/learn/quality-assurance/)  [Scientific Computing with Python Certification](https://www.freecodecamp.org/learn/scientific-computing-with-python/)  [Data Analysis with Python Certification](https://www.freecodecamp.org/learn/data-analysis-with-python/)  [Information Security Certification](https://www.freecodecamp.org/learn/information-security/)  [Machine Learning with Python Certification](https://www.freecodecamp.org/learn/machine-learning-with-python/)  [College Algebra with Python Certification](https://www.freecodecamp.org/learn/college-algebra-with-python/)  **Earn free professional certifications:**  [(New) Foundational C# with Microsoft Certification](https://www.freecodecamp.org/learn/foundational-c-sharp-with-microsoft/)  **Prepare for the developer interview job search:**  [Coding Interview Prep](https://www.freecodecamp.org/learn/coding-interview-prep/)  [Project Euler](https://www.freecodecamp.org/learn/project-euler/)  freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)  Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.  Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.  **You can**[**make a tax-deductible donation here**](https://www.freecodecamp.org/donate)**.**  **Trending Guides**  [JS isEmpty Equivalent](https://www.freecodecamp.org/news/check-if-an-object-is-empty-in-javascript/)[Submit a Form with JS](https://www.freecodecamp.org/news/how-to-submit-a-form-with-javascript/)[Add to List in Python](https://www.freecodecamp.org/news/how-to-add-to-a-list-in-python-list-addition-tutorial/)[Grep Command in Linux](https://www.freecodecamp.org/news/grep-command-in-linux-usage-options-and-syntax-examples/)[String to Int in Java](https://www.freecodecamp.org/news/how-to-convert-a-string-to-an-integer-in-java/)[Add to Dict in Python](https://www.freecodecamp.org/news/adding-to-a-dict-in-python-how-to-add-to-a-dictionary/)[Java For Loop Example](https://www.freecodecamp.org/news/java-for-loop-example/)[Matplotlib Figure Size](https://www.freecodecamp.org/news/matplotlib-figure-size-change-plot-size-in-python/)[Database Normalization](https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/)[Nested Lists in Python](https://www.freecodecamp.org/news/list-within-a-list-in-python-initialize-a-nested-list/)[Coalesce SQL](https://www.freecodecamp.org/news/coalesce-sql-example-postgresql-and-sql-server-functions/)[Python join()](https://www.freecodecamp.org/news/python-join-how-to-combine-a-list-into-a-string-in-python/)[JS POST Request](https://www.freecodecamp.org/news/javascript-post-request-how-to-send-an-http-post-request-in-js/)[JS Type Checking](https://www.freecodecamp.org/news/javascript-type-checking-how-to-check-type-in-js-with-typeof/)[Read Python File](https://www.freecodecamp.org/news/how-to-read-a-file-line-by-line-in-python/)[SOLID Principles](https://www.freecodecamp.org/news/solid-design-principles-in-software-development/)[Sort a List in Java](https://www.freecodecamp.org/news/how-to-sort-a-list-in-java/)[For Loops in Python](https://www.freecodecamp.org/news/for-loops-in-python-with-example-code/)[JavaScript 2D Array](https://www.freecodecamp.org/news/javascript-2d-arrays/)[SQL CONVERT Function](https://www.freecodecamp.org/news/sql-convert-the-date-to-string-or-datetime-function-2/)[Rename Column in Pandas](https://www.freecodecamp.org/news/how-to-rename-a-column-in-pandas/)[Delete a File in Python](https://www.freecodecamp.org/news/how-to-delete-a-file-in-python-and-remove-a-directory-too/)[K-Nearest Neighbors Algo](https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/)[iferror Function in Excel](https://www.freecodecamp.org/news/iferror-function-in-excel-example/)[Remove From String Python](https://www.freecodecamp.org/news/remove-from-string-in-python-how-to-remove-characters-from-a-string/)[Create a File in Terminal](https://www.freecodecamp.org/news/how-to-make-a-file-in-linux-from-the-command-line-create-a-file-in-terminal/)[Clear Formatting in Excel](https://www.freecodecamp.org/news/how-to-clear-formatting-in-excel/)[Accounting Num Format Excel](https://www.freecodecamp.org/news/accounting-number-format-in-excel-how-to-apply-it-to-selected-cells/)[Check if File Exists Python](https://www.freecodecamp.org/news/how-to-check-if-a-file-exists-in-python/)[Iterate Over Dict in Python](https://www.freecodecamp.org/news/dictionary-iteration-in-python/)  **Scientific Computing with Python**  Python is one of the most popular, flexible programming languages today. You can use it for everything from basic scripting to machine learning.  In the Scientific Computing with Python Certification, you'll learn Python fundamentals like variables, loops, conditionals, and functions. Then you'll quickly ramp up to complex data structures, networking, relational databases, and data visualization.  **Courses**  **Python for Everybody**  Python for everybody is a free video course series that teaches the basics of using Python 3.  The courses were created by Dr. Charles Severance (also known as Dr. Chuck). He is a Clinical Professor at the University of Michigan School of Information, where he teaches various technology-oriented courses including programming, database design, and web development.  Collapse coursePython for Everybody  0/56, 0 of 56 challenges completed   * [Not PassedIntroduction: Why Program?](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/introduction-why-program) * [Not PassedIntroduction: Hardware Architecture](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/introduction-hardware-architecture) * [Not PassedIntroduction: Python as a Language](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/introduction-python-as-a-language) * [Not PassedIntroduction: Elements of Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/introduction-elements-of-python) * [Not PassedVariables, Expressions, and Statements](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/variables-expressions-and-statements) * [Not PassedIntermediate Expressions](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/intermediate-expressions) * [Not PassedConditional Execution](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/conditional-execution) * [Not PassedMore Conditional Structures](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/more-conditional-structures) * [Not PassedPython Functions](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/python-functions) * [Not PassedBuild your own Functions](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/build-your-own-functions) * [Not PassedLoops and Iterations](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/loops-and-iterations) * [Not PassedIterations: Definite Loops](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/iterations-definite-loops) * [Not PassedIterations: Loop Idioms](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/iterations-loop-idioms) * [Not PassedIterations: More Patterns](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/iterations-more-patterns) * [Not PassedStrings in Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/strings-in-python) * [Not PassedIntermediate Strings](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/intermediate-strings) * [Not PassedReading Files](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/reading-files) * [Not PassedFiles as a Sequence](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/files-as-a-sequence) * [Not PassedPython Lists](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/python-lists) * [Not PassedWorking with Lists](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/working-with-lists) * [Not PassedStrings and Lists](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/strings-and-lists) * [Not PassedPython Dictionaries](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/python-dictionaries) * [Not PassedDictionaries: Common Applications](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/dictionaries-common-applications) * [Not PassedDictionaries and Loops](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/dictionaries-and-loops) * [Not PassedThe Tuples Collection](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/the-tuples-collection) * [Not PassedComparing and Sorting Tuples](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/comparing-and-sorting-tuples) * [Not PassedRegular Expressions](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/regular-expressions) * [Not PassedRegular Expressions: Matching and Extracting Data](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/regular-expressions-matching-and-extracting-data) * [Not PassedRegular Expressions: Practical Applications](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/regular-expressions-practical-applications) * [Not PassedNetworking with Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-with-python) * [Not PassedNetworking Protocol](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-protocol) * [Not PassedNetworking: Write a Web Browser](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-write-a-web-browser) * [Not PassedNetworking: Text Processing](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-text-processing) * [Not PassedNetworking: Using urllib in Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-using-urllib-in-python) * [Not PassedNetworking: Web Scraping with Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/networking-web-scraping-with-python) * [Not PassedUsing Web Services](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/using-web-services) * [Not PassedWeb Services: XML](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-xml) * [Not PassedWeb Services: XML Schema](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-xml-schema) * [Not PassedWeb Services: JSON](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-json) * [Not PassedWeb Services: Service Oriented Approach](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-service-oriented-approach) * [Not PassedWeb Services: APIs](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-apis) * [Not PassedWeb Services: API Rate Limiting and Security](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/web-services-api-rate-limiting-and-security) * [Not PassedPython Objects](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/python-objects) * [Not PassedObjects: A Sample Class](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/objects-a-sample-class) * [Not PassedObject Lifecycle](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/object-lifecycle) * [Not PassedObjects: Inheritance](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/objects-inheritance) * [Not PassedRelational Databases and SQLite](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/relational-databases-and-sqlite) * [Not PassedMake a Relational Database](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/make-a-relational-database) * [Not PassedRelational Database Design](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/relational-database-design) * [Not PassedRepresenting Relationships in a Relational Database](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/representing-relationships-in-a-relational-database) * [Not PassedRelational Databases: Relationship Building](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/relational-databases-relationship-building) * [Not PassedRelational Databases: Join Operation](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/relational-databases-join-operation) * [Not PassedRelational Databases: Many-to-many Relationships](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/relational-databases-many-to-many-relationships) * [Not PassedVisualizing Data with Python](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/visualizing-data-with-python) * [Not PassedData Visualization: Page Rank](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/data-visualization-page-rank) * [Not PassedData Visualization: Mailing Lists](https://www.freecodecamp.org/learn/scientific-computing-with-python/python-for-everybody/data-visualization-mailing-lists)   **Scientific Computing with Python Projects**  Time to put your Python skills to the test. By completing these projects, you will demonstrate that you have a good foundational knowledge of Python and qualify for the Scientific Computing with Python Certification.   * [Arithmetic FormatterNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/arithmetic-formatter) * [Time CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/time-calculator) * [Budget AppNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/budget-app) * [Polygon Area CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/polygon-area-calculator) * [Probability CalculatorNot Passed](https://www.freecodecamp.org/learn/scientific-computing-with-python/scientific-computing-with-python-projects/probability-calculator)   [Go to settings to claim your certificationScientific Computing with Python](https://www.freecodecamp.org/settings#certification-settings)  **Browse our other free certifications (we recommend doing these in order)**  **Earn free verified certifications with freeCodeCamp's core curriculum:**  [Responsive Web Design Certification](https://www.freecodecamp.org/learn/2022/responsive-web-design/)  [JavaScript Algorithms and Data Structures Certification](https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/)  [Front End Development Libraries Certification](https://www.freecodecamp.org/learn/front-end-development-libraries/)  [Data Visualization Certification](https://www.freecodecamp.org/learn/data-visualization/)  [Relational Database Certification](https://www.freecodecamp.org/learn/relational-database/)  [Back End Development and APIs Certification](https://www.freecodecamp.org/learn/back-end-development-and-apis/)  [Quality Assurance Certification](https://www.freecodecamp.org/learn/quality-assurance/)  [Scientific Computing with Python Certification](https://www.freecodecamp.org/learn/scientific-computing-with-python/)  [Data Analysis with Python Certification](https://www.freecodecamp.org/learn/data-analysis-with-python/)  [Information Security Certification](https://www.freecodecamp.org/learn/information-security/)  [Machine Learning with Python Certification](https://www.freecodecamp.org/learn/machine-learning-with-python/)  [College Algebra with Python Certification](https://www.freecodecamp.org/learn/college-algebra-with-python/)  **Earn free professional certifications:**  [(New) Foundational C# with Microsoft Certification](https://www.freecodecamp.org/learn/foundational-c-sharp-with-microsoft/)  **Prepare for the developer interview job search:**  [Coding Interview Prep](https://www.freecodecamp.org/learn/coding-interview-prep/)  [Project Euler](https://www.freecodecamp.org/learn/project-euler/)  freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charitable organization (United States Federal Tax Identification Number: 82-0779546)  Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.  Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.  **You can**[**make a tax-deductible donation here**](https://www.freecodecamp.org/donate)**.**  **Trending Guides**  [JS isEmpty Equivalent](https://www.freecodecamp.org/news/check-if-an-object-is-empty-in-javascript/)[Submit a Form with JS](https://www.freecodecamp.org/news/how-to-submit-a-form-with-javascript/)[Add to List in Python](https://www.freecodecamp.org/news/how-to-add-to-a-list-in-python-list-addition-tutorial/)[Grep Command in Linux](https://www.freecodecamp.org/news/grep-command-in-linux-usage-options-and-syntax-examples/)[String to Int in Java](https://www.freecodecamp.org/news/how-to-convert-a-string-to-an-integer-in-java/)[Add to Dict in Python](https://www.freecodecamp.org/news/adding-to-a-dict-in-python-how-to-add-to-a-dictionary/)[Java For Loop Example](https://www.freecodecamp.org/news/java-for-loop-example/)[Matplotlib Figure Size](https://www.freecodecamp.org/news/matplotlib-figure-size-change-plot-size-in-python/)[Database Normalization](https://www.freecodecamp.org/news/database-normalization-1nf-2nf-3nf-table-examples/)[Nested Lists in Python](https://www.freecodecamp.org/news/list-within-a-list-in-python-initialize-a-nested-list/)[Coalesce SQL](https://www.freecodecamp.org/news/coalesce-sql-example-postgresql-and-sql-server-functions/)[Python join()](https://www.freecodecamp.org/news/python-join-how-to-combine-a-list-into-a-string-in-python/)[JS POST Request](https://www.freecodecamp.org/news/javascript-post-request-how-to-send-an-http-post-request-in-js/)[JS Type Checking](https://www.freecodecamp.org/news/javascript-type-checking-how-to-check-type-in-js-with-typeof/)[Read Python File](https://www.freecodecamp.org/news/how-to-read-a-file-line-by-line-in-python/)[SOLID Principles](https://www.freecodecamp.org/news/solid-design-principles-in-software-development/)[Sort a List in Java](https://www.freecodecamp.org/news/how-to-sort-a-list-in-java/)[For Loops in Python](https://www.freecodecamp.org/news/for-loops-in-python-with-example-code/)[JavaScript 2D Array](https://www.freecodecamp.org/news/javascript-2d-arrays/)[SQL CONVERT Function](https://www.freecodecamp.org/news/sql-convert-the-date-to-string-or-datetime-function-2/)[Rename Column in Pandas](https://www.freecodecamp.org/news/how-to-rename-a-column-in-pandas/)[Delete a File in Python](https://www.freecodecamp.org/news/how-to-delete-a-file-in-python-and-remove-a-directory-too/)[K-Nearest Neighbors Algo](https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/)[iferror Function in Excel](https://www.freecodecamp.org/news/iferror-function-in-excel-example/)[Remove From String Python](https://www.freecodecamp.org/news/remove-from-string-in-python-how-to-remove-characters-from-a-string/)[Create a File in Terminal](https://www.freecodecamp.org/news/how-to-make-a-file-in-linux-from-the-command-line-create-a-file-in-terminal/)[Clear Formatting in Excel](https://www.freecodecamp.org/news/how-to-clear-formatting-in-excel/)[Accounting Num Format Excel](https://www.freecodecamp.org/news/accounting-number-format-in-excel-how-to-apply-it-to-selected-cells/)[Check if File Exists Python](https://www.freecodecamp.org/news/how-to-check-if-a-file-exists-in-python/)[Iterate Over Dict in Python](https://www.freecodecamp.org/news/dictionary-iteration-in-python/)  import React from 'react'; import {Text, View} from 'react-native'; import {Header} from './Header'; import {heading} from './Typography';  const WelcomeScreen = () => (  <View>  <Header title="Welcome to React Native"/>  <Text style={heading}>Step One</Text>  <Text>  Edit App.js to change this screen and turn it  into your app.  </Text>  <Text style={heading}>See Your Changes</Text>  <Text>  Press Cmd + R inside the simulator to reload  your app’s code.  </Text>  <Text style={heading}>Debug</Text>  <Text>  Press Cmd + M or Shake your device to open the  React Native Debug Menu.  </Text>  <Text style={heading}>Learn</Text>  <Text>  Read the docs to discover what to do next:  </Text>  </View> );  operator = input("Enter an operator [+ - \* /]: ")  if operator in ('+', '-', '\*', '/'):  num1 = float(input("Enter the first number: "))  num2 = float(input("Enter the second number: "))    if operator == '+':  result = num1 + num2  print("Sum is " + str(result))  elif operator == '-':  result = num1 - num2  print("Difference is " + str(result))  elif operator == '\*':  result = num1 \* num2  print("Product is " + str(result))  elif operator == '/':  if num2 != 0:  result = num1 / num2  print("Division is " + str(result))  else:  print("Division by zero is not allowed.")  else:  print("Invalid operator. Please enter +, -, \*, or /.")  Example Data Type Try it  x = "Hello World" str [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str)  x = 20 int [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int)  x = 20.5 float [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float)  x = 1j complex [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex)  x = ["apple", "banana", "cherry"] list [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list)  x = ("apple", "banana", "cherry") tuple [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple)  x = range(6) range [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range)  x = {"name" : "John", "age" : 36} dict [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict)  x = {"apple", "banana", "cherry"} set [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set)  x = frozenset({"apple", "banana", "cherry"}) frozenset [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset)  x = True bool [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool)  x = b"Hello" bytes [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes)  x = bytearray(5) bytearray [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray)  x = memoryview(bytes(5)) memoryview [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview)  x = None NoneType [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_nonetype)  To fix this problem, use the escape character \": Example The escape character allows you to use double quotes when you normally would not be allowed:  txt = "We are the so-called \"Vikings\" from the north."  [Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape) Escape Characters Other escape characters used in Python:   |  |  |  | | --- | --- | --- | | **Code** | **Result** | **Try it** | | \' | Single Quote | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_escape2) | | \\ | Backslash | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_backslash) | | \n | New Line | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_newline) | | \r | Carriage Return | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_r) | | \t | Tab | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_t) | | \b | Backspace | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_string_b) | | \f | Form Feed |  | | \ooo | Octal value | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_octal) | | \xhh | Hex value |  |  |  |  | | --- | --- | | **Method** | **Description** | | [capitalize()](https://www.w3schools.com/python/ref_string_capitalize.asp) | Converts the first character to upper case | | [casefold()](https://www.w3schools.com/python/ref_string_casefold.asp) | Converts string into lower case | | [center()](https://www.w3schools.com/python/ref_string_center.asp) | Returns a centered string | | [count()](https://www.w3schools.com/python/ref_string_count.asp) | Returns the number of times a specified value occurs in a string | | [encode()](https://www.w3schools.com/python/ref_string_encode.asp) | Returns an encoded version of the string | | [endswith()](https://www.w3schools.com/python/ref_string_endswith.asp) | Returns true if the string ends with the specified value | | [expandtabs()](https://www.w3schools.com/python/ref_string_expandtabs.asp) | Sets the tab size of the string | | [find()](https://www.w3schools.com/python/ref_string_find.asp) | Searches the string for a specified value and returns the position of where it was found | | [format()](https://www.w3schools.com/python/ref_string_format.asp) | Formats specified values in a string | | format\_map() | Formats specified values in a string | | [index()](https://www.w3schools.com/python/ref_string_index.asp) | Searches the string for a specified value and returns the position of where it was found | | [isalnum()](https://www.w3schools.com/python/ref_string_isalnum.asp) | Returns True if all characters in the string are alphanumeric | | [isalpha()](https://www.w3schools.com/python/ref_string_isalpha.asp) | Returns True if all characters in the string are in the alphabet | | [isascii()](https://www.w3schools.com/python/ref_string_isascii.asp) | Returns True if all characters in the string are ascii characters | | [isdecimal()](https://www.w3schools.com/python/ref_string_isdecimal.asp) | Returns True if all characters in the string are decimals | | [isdigit()](https://www.w3schools.com/python/ref_string_isdigit.asp) | Returns True if all characters in the string are digits | | [isidentifier()](https://www.w3schools.com/python/ref_string_isidentifier.asp) | Returns True if the string is an identifier | | [islower()](https://www.w3schools.com/python/ref_string_islower.asp) | Returns True if all characters in the string are lower case | | [isnumeric()](https://www.w3schools.com/python/ref_string_isnumeric.asp) | Returns True if all characters in the string are numeric | | [isprintable()](https://www.w3schools.com/python/ref_string_isprintable.asp) | Returns True if all characters in the string are printable | | [isspace()](https://www.w3schools.com/python/ref_string_isspace.asp) | Returns True if all characters in the string are whitespaces | | [istitle()](https://www.w3schools.com/python/ref_string_istitle.asp) | Returns True if the string follows the rules of a title | | [isupper()](https://www.w3schools.com/python/ref_string_isupper.asp) | Returns True if all characters in the string are upper case | | [join()](https://www.w3schools.com/python/ref_string_join.asp) | Joins the elements of an iterable to the end of the string | | [ljust()](https://www.w3schools.com/python/ref_string_ljust.asp) | Returns a left justified version of the string | | [lower()](https://www.w3schools.com/python/ref_string_lower.asp) | Converts a string into lower case | | [lstrip()](https://www.w3schools.com/python/ref_string_lstrip.asp) | Returns a left trim version of the string | | [maketrans()](https://www.w3schools.com/python/ref_string_maketrans.asp) | Returns a translation table to be used in translations | | [partition()](https://www.w3schools.com/python/ref_string_partition.asp) | Returns a tuple where the string is parted into three parts | | [replace()](https://www.w3schools.com/python/ref_string_replace.asp) | Returns a string where a specified value is replaced with a specified value | | [rfind()](https://www.w3schools.com/python/ref_string_rfind.asp) | Searches the string for a specified value and returns the last position of where it was found | | [rindex()](https://www.w3schools.com/python/ref_string_rindex.asp) | Searches the string for a specified value and returns the last position of where it was found | | [rjust()](https://www.w3schools.com/python/ref_string_rjust.asp) | Returns a right justified version of the string | | [rpartition()](https://www.w3schools.com/python/ref_string_rpartition.asp) | Returns a tuple where the string is parted into three parts | | [rsplit()](https://www.w3schools.com/python/ref_string_rsplit.asp) | Splits the string at the specified separator, and returns a list | | [split()](https://www.w3schools.com/python/ref_string_split.asp) | Splits the string at the specified separator, and returns a l | | [splitlines()](https://www.w3schools.com/python/ref_string_splitlines.asp) | Splits the string at line breaks and returns a list | | [startswith()](https://www.w3schools.com/python/ref_string_startswith.asp) | Returns true if the string starts with the specified value | | [strip()](https://www.w3schools.com/python/ref_string_strip.asp) | Returns a trimmed version of the string | | [swapcase()](https://www.w3schools.com/python/ref_string_swapcase.asp) | Swaps cases, lower case becomes upper case and vice versa | | [title()](https://www.w3schools.com/python/ref_string_title.asp) | Converts the first character of each word to upper case | | [translate()](https://www.w3schools.com/python/ref_string_translate.asp) | Returns a translated string | | [upper()](https://www.w3schools.com/python/ref_string_upper.asp) | Converts a string into upper case | | [zfill()](https://www.w3schools.com/python/ref_string_zfill.asp) | Fills the string with a specified number of 0 values at the beginning | |

**Activity Title:** PyFun 01 Challenge

**Activity Description:** In this Python activity, learners will dive into the world of Python fundamentals by working on a series of challenges that revolve around variables and types, string manipulation, type conversion, and getting user input. The goal is to improve their problem-solving skills and reinforce their understanding of these essential Python concepts.

**Instructions:**

**Step 1: Variable Magic**

Create a Python program that performs the following tasks:

* Define three variables: num1, num2, and result.
* Assign integer values to num1 and num2.
* Calculate the sum of num1 and num2 and store it in result.
* Print the result in the following format: "The sum of [num1] and [num2] is [result]."

**Step 2: String Shuffle**

Write a Python program that does the following:

* Create a string variable original\_string with any text you like.
* Reverse the original string and store it in a new variable called reversed\_string.
* Print the reversed string.

**Step 3: Type Transformer**

Build a Python program that demonstrates the concept of type conversion. Follow these steps:

* Create a variable num\_str and assign a number in string format, e.g., "42".
* Convert num\_str into an integer and assign it to a new variable num\_int.
* Print num\_int and its data type.
* Create a float variable num\_float and assign it the value 3.14.
* Convert num\_float into a string and store it in num\_str2.
* Print num\_str2 and its data type.

**Step 4: User's Insights**

Create an interactive Python program where the user is prompted to provide their name and age. Then, the program should display a message tailored to the user's input. Here's how:

* Ask the user to enter their name.
* Ask the user to enter their age.
* Using the user's input, print a message like "Hello, [name]! You are [age] years old."

**Bonus Challenge: String Slicer**

For an extra challenge, modify the "String Shuffle" program from Step 2 to slice the original string into two parts, reverse each part, and then join them back together. Print the final, reversed string.

**it\_2023\_python\_functions\_activity**

**Assignment Title: Guess the Number Game**

**Objective:** Reinforce understanding of control structures (if statements, loops) and functions in Python.

**Instructions:**

1. **Introduction:**
   * Provide a brief introduction to the assignment, explaining that students will be implementing a "Guess the Number" game in Python.
2. **Clone the Repository:**
   * Clone the assignment repository to their local machines.

git clone <repository-url>

1. **Code Skeleton:**
   * Inside the repository, there's a Python file (guess\_the\_number.py) containing the initial code skeleton.
2. import random
3. def generate\_random\_number():
4. # Code to generate a random number
5. def take\_user\_guess():
6. # Code to take user input for guessing
7. def check\_guess(random\_number, user\_guess):
8. # Code to check if the guess is correct
9. def main():
10. # Main program logic
11. if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Task Description:**
   * Break down the task into smaller steps:
     + Complete the generate\_random\_number function.
     + Implement the take\_user\_guess function to handle user input.
     + Write the logic for the check\_guess function to provide feedback.
     + Create the main game logic inside the main function.
2. **Challenge (Optional):**
   * Limit the number of guesses.
   * Add a scoring system.
3. **Commit and Push:**
   * Commit your changes and push them to the GitHub repository.
4. git add guess\_the\_number.py
5. git commit -m "Completed Guess the Number Game"

git push origin main

**Resources:**

1. [Python Random Module](https://www.geeksforgeeks.org/python-random-module/)
2. [Python Functions](https://www.geeksforgeeks.org/python-functions/)

**Happy Coding**