dipakh21@gmail.com**1 Introduction & Components of Python:**

Understanding Python

Role of Python in AI and Data science

Installation and Working with Python

Types and Operation

Python Object Types-Number, Strings, Lists,

Dictionaries, Tuples, Files, User Defined Classes

Understanding python blocks

Python Program Flow Control

Conditional blocks using if, else and elif

Simple for loops in python

For loop using ranges, string, list and dictionaries

Use of while loops in python

Loop manipulation using pass, continue, break and else

Programming using Python conditional and loops block

===========================================================

**2 Python Functions, Modules & Packages:**

Function Basics-Scope, nested function, non-local

statements

Arguments Passing, Anonymous Function: lambda

Decorators and Generators

Module basic usage, namespaces, reloading modules.

Package: import basics

Python namespace packages

===========================================================

**3 Python Object Oriented Programming:**

Concept of class, object and instances, method call

Constructor, class attributes and destructors

Real time use of class in live projects

Inheritance, super class and overloading operators,

Static and class methods

Adding and retrieving dynamic attributes of classes

Programming using OOPS

============================================================

**4 Python Regular Expression:**

Powerful pattern matching and searching

Power of pattern searching using regex in python

Real time parsing of data using regex

Password, email, url validation using regular expression

Pattern finding programs using regular expression

=======================================================

**5 Python Multithreading and Exception Handling:**

Exception Handling

Avoiding code break using exception handling

Safe guarding file operation using exception handling

Handling and helping developer with error code

Programming using Exception handling

Multithreading

Understanding threads

Synchronizing the threads

Programming using multithreading

=============================================

**6 Python File Operation:**

Reading config files in python

Writing log files in python

Understanding read functions, read(), readline() and

readlines()

Understanding write

functions, write() and writelines()

Manipulating file pointer using seek

Programming using file operations

=======================================

**7 Python Database Interaction:**

SQL **Database** connection using python

**Creating** and **searching** tables

**Reading** and storing **config** information on database

Programming using database connections

======================================

**8 Python For Data Analysis:**

Numpy:

Introduction to numpy

Creating arrays, Using arrays and Scalars

Indexing Arrays, Array Transposition

Universal Array Function

*Array Input and Output*

**Pandas**:

What is ***pandas***? Where it is used?

Series in pandas, pandas DataFrames, Index objects,

ReIndex

Drop Entry, Selecting Entries

Data Alignment, Rank and Sort

Summary Statics, Missing Data, Index Hierarchy

Matplotlib:

Python for Data Visualization

Introduction to Matplotlib

Visualization Tools

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 |

## Getting the Data Type

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

### Example

Print the data type of the variable x:

x = 5  
print(type(x))

## 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) |

## 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))

## 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

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))

## 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))

## 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.

# Python Casting

## 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 rounding down to the previous whole number), 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

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'

# Python Strings

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

## String Literals

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

'hello' is the same as "hello".

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

### Example

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

[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.

## 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)

## 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 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)

## Negative Indexing

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

### Example

Get the characters from position 5 to position 1, starting the count from the end of the string:

b = "Hello, World!"  
print(b[-5:-2])

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

## 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)

## String Methods

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

### 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)

### 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)

### Example

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)

### 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)

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

## Check String

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

### Example

Check if the phrase "ain" is present in the following text:

txt = "The rain in Spain stays mainly in the plain"  
x = "ain" in txt  
print(x)

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

### Example

Check if the phrase "ain" is NOT present in the following text:

txt = "The rain in Spain stays mainly in the plain"  
x = "ain" not in txt  
print(x)

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

## String Concatenation

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

### Example

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)

## String Format

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

### Example

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)

## 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

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)

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/trypython.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) |

## String Methods

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

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [capitalize()](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 |
| [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() | 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() | 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 |

Answers:

a = " Hello, World! "

print(a.strip())

a = "Hello, World!"

print(a.lower())

a = "Hello, World!"

print(a.upper())

a = "Hello, World!"

print(a.replace("Helo","J"))

a = "Hello, World! and gondia"

print(a.split("!"))

txt = "The rain in Spain stays mainly in the plain"

x = "ain" in txt

print(x)

txt = "The rain in Spain stays mainly in the plain"

x = "ain" not in txt

print(x)

a = "Hello"

b = "Gondia"

c = a + b

print(c)

#if we want to add character and integer

age = 36

txt = "My name is John, and I am {}"

print(txt.format(age))

quantity = 3

itemno = 567

price = 49.95

myorder = "I want {} pieces of item {} for {} dollars."

myorder = "I want to pay {2} dollars for {0} pieces of item {1}."

print(myorder.format(quantity, itemno, price))

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

print(txt)

# Python Booleans

[❮ Previous](https://www.w3schools.com/python/python_strings.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

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)

## 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 evaluates 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 returns 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)

# 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.

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) |

## 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** |
| & | AND | Sets each bit to 1 if both bits are 1 |
| | | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |

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

print(thislist)

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

print(thislist[1])

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

print(thislist[-1])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:5])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[:4])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:])

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[-4:-1])

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

thislist[1] = "butterscotch"

print(thislist)

print("\n")

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

for x in thislist:

print(x)

print("\n")

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

if "apple" in thislist:

print("Yes, 'apple' is in the fruits list")

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

print(len(thislist))

print("\n")

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

thislist.append("orange")

print(thislist)

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

thislist.insert(1, "orange")

print(thislist)

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

thislist.remove("banana")

print(thislist)

thislist = ["apple", "banana", "cherry","Dassy"]

thislist.pop()

print(thislist)

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

del thislist[0]

print(thislist)

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

del thislist

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

thislist.clear()

print(thislist)

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

mylist = thislist.copy()

print(mylist)

print("copied successfully......")

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

mylist = list(thislist)

print(mylist)

print("Print successfully......")

list1 = ["a", "b" , "c"]

list2 = [1, 2, 3]

list3 = list1 + list2

print(list3)

list1 = ["a", "b" , "c"]

list2 = [1, 2, 3]

for x in list2:

list1.append(x)

print(list1)

list1 = ["a", "b" , "c"]

list2 = [1, 2, 3]

list1.extend(list2)

print(list1)

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

print(thislist)

# Python Dictionaries

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

## Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

### Example

Create and print a dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict)

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

## Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

### Example

Get the value of the "model" key:

x = thisdict["model"]

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

There is also a method called get() that will give you the same result:

### Example

Get the value of the "model" key:

x = thisdict.get("model")

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

## Change Values

You can change the value of a specific item by referring to its key name:

### Example

Change the "year" to 2018:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict["year"] = 2018

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

## Loop Through a Dictionary

You can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

### Example

Print all key names in the dictionary, one by one:

for x in thisdict:  
  print(x)

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

### Example

Print all values in the dictionary, one by one:

for x in thisdict:  
  print(thisdict[x])

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_loop2)

### Example

You can also use the values() function to return values of a dictionary:

for x in thisdict.values():  
  print(x)

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

### Example

Loop through both keys and values, by using the items() function:

for x, y in thisdict.items():  
  print(x, y)

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

## Check if Key Exists

To determine if a specified key is present in a dictionary use the in keyword:

### Example

Check if "model" is present in the dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
if "model" in thisdict:  
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

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

## Dictionary Length

To determine how many items (key-value pairs) a dictionary has, use the len() method.

### Example

Print the number of items in the dictionary:

print(len(thisdict))

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

## Adding Items

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

### Example

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict["color"] = "red"  
print(thisdict)

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

## Removing Items

There are several methods to remove items from a dictionary:

### Example

The pop() method removes the item with the specified key name:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.pop("model")  
print(thisdict)

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

### Example

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.popitem()  
print(thisdict)

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

### Example

The del keyword removes the item with the specified key name:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
del thisdict["model"]  
print(thisdict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_del2)

### Example

The del keyword can also delete the dictionary completely:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
del thisdict  
print(thisdict) #this will cause an error because "thisdict" no longer exists.

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_del3)

### Example

The clear() method empties the dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.clear()  
print(thisdict)

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

## Copy a Dictionary

You cannot copy a dictionary simply by typing dict2 = dict1, because: dict2 will only be a reference to dict1, and changes made in dict1 will automatically also be made in dict2.

There are ways to make a copy, one way is to use the built-in Dictionary method copy().

### Example

Make a copy of a dictionary with the copy() method:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
mydict = thisdict.copy()  
print(mydict)

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

Another way to make a copy is to use the built-in method dict().

### Example

Make a copy of a dictionary with the dict() method:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
mydict = dict(thisdict)  
print(mydict)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_copy2)

## Nested Dictionaries

A dictionary can also contain many dictionaries, this is called nested dictionaries.

### Example

Create a dictionary that contain three dictionaries:

myfamily = {  
  "child1" : {  
    "name" : "Emil",  
    "year" : 2004  
  },  
  "child2" : {  
    "name" : "Tobias",  
    "year" : 2007  
  },  
  "child3" : {  
    "name" : "Linus",  
    "year" : 2011  
  }  
}

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

Or, if you want to nest three dictionaries that already exists as dictionaries:

### Example

Create three dictionaries, than create one dictionary that will contain the other three dictionaries:

child1 = {  
  "name" : "Emil",  
  "year" : 2004  
}  
child2 = {  
  "name" : "Tobias",  
  "year" : 2007  
}  
child3 = {  
  "name" : "Linus",  
  "year" : 2011  
}  
  
myfamily = {  
  "child1" : child1,  
  "child2" : child2,  
  "child3" : child3  
}

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_dictionary_nested2)

## The dict() Constructor

It is also possible to use the dict() constructor to make a new dictionary:

### Example

thisdict = dict(brand="Ford", model="Mustang", year=1964)  
# note that keywords are not string literals  
# note the use of equals rather than colon for the assignment  
print(thisdict)

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

## Dictionary Methods

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [clear()](https://www.w3schools.com/python/ref_dictionary_clear.asp) | Removes all the elements from the dictionary |
| [copy()](https://www.w3schools.com/python/ref_dictionary_copy.asp) | Returns a copy of the dictionary |
| [fromkeys()](https://www.w3schools.com/python/ref_dictionary_fromkeys.asp) | Returns a dictionary with the specified keys and value |
| [get()](https://www.w3schools.com/python/ref_dictionary_get.asp) | Returns the value of the specified key |
| [items()](https://www.w3schools.com/python/ref_dictionary_items.asp) | Returns a list containing a tuple for each key value pair |
| [keys()](https://www.w3schools.com/python/ref_dictionary_keys.asp) | Returns a list containing the dictionary's keys |
| [pop()](https://www.w3schools.com/python/ref_dictionary_pop.asp) | Removes the element with the specified key |
| [popitem()](https://www.w3schools.com/python/ref_dictionary_popitem.asp) | Removes the last inserted key-value pair |
| [setdefault()](https://www.w3schools.com/python/ref_dictionary_setdefault.asp) | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| [update()](https://www.w3schools.com/python/ref_dictionary_update.asp) |  |

# 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

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.

## 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 are 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)

# Python Lambda

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

A lambda function is a small anonymous function.

A lambda function can take any number of arguments, but can only have one expression.

## Syntax

lambda arguments : expression

The expression is executed and the result is returned:

### Example

A lambda function that adds 10 to the number passed in as an argument, and print the result:

x = lambda a : a + 10  
print(x(5))

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

Lambda functions can take any number of arguments:

### Example

A lambda function that multiplies argument a with argument b and print the result:

x = lambda a, b : a \* b  
print(x(5, 6))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda2)

### Example

A lambda function that sums argument a, b, and c and print the result:

x = lambda a, b, c : a + b + c  
print(x(5, 6, 2))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_lambda3)

## Why Use Lambda Functions?

The power of lambda is better shown when you use them as an anonymous function inside another function.

Say you have a function definition that takes one argument, and that argument will be multiplied with an unknown number:

def myfunc(n):  
  return lambda a : a \* n

Use that function definition to make a function that always doubles the number you send in:

### Example

def myfunc(n):  
  return lambda a : a \* n  
  
mydoubler = myfunc(2)  
  
print(mydoubler(11))

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

Or, use the same function definition to make a function that always triples the number you send in:

### Example

def myfunc(n):  
  return lambda a : a \* n  
  
mytripler = myfunc(3)  
  
print(mytripler(11))

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

Or, use the same function definition to make both functions, in the same program:

### Example

def myfunc(n):  
  return lambda a : a \* n  
  
mydoubler = myfunc(2)  
mytripler = myfunc(3)  
  
print(mydoubler(11))  
print(mytripler(11))

# 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

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

### Example

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.

## 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.

# Python Classes and Objects

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

## Python Classes/Objects

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

## Create a Class

To create a class, use the keyword class:

### Example

Create a class named MyClass, with a property named x:

class MyClass:  
  x = 5

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class1)

## Create Object

Now we can use the class named MyClass to create objects:

### Example

Create an object named p1, and print the value of x:

p1 = MyClass()  
print(p1.x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class2)

## The \_\_init\_\_() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in \_\_init\_\_() function.

All classes have a function called \_\_init\_\_(), which is always executed when the class is being initiated.

Use the \_\_init\_\_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

### Example

Create a class named Person, use the \_\_init\_\_() function to assign values for name and age:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
p1 = Person("John", 36)  
  
print(p1.name)  
print(p1.age)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class3)

**Note:** The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

## Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

### Example

Insert a function that prints a greeting, and execute it on the p1 object:

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
  def myfunc(self):  
    print("Hello my name is " + self.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class4)

**Note:** The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

## The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

### Example

Use the words mysillyobject and abc instead of self:

class Person:  
  def \_\_init\_\_(mysillyobject, name, age):  
    mysillyobject.name = name  
    mysillyobject.age = age  
  
  def myfunc(abc):  
    print("Hello my name is " + abc.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class5)

## Modify Object Properties

You can modify properties on objects like this:

### Example

Set the age of p1 to 40:

p1.age = 40

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class6)

## Delete Object Properties

You can delete properties on objects by using the del keyword:

### Example

Delete the age property from the p1 object:

del p1.age

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class7)

## Delete Objects

You can delete objects by using the del keyword:

### Example

Delete the p1 object:

del p1

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_class8)

## The pass Statement

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

### Example

class Person:  
  pass

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

# Python Inheritance

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

## Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

**Parent class** is the class being inherited from, also called base class.

**Child class** is the class that inherits from another class, also called derived class.

## Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

### Example

Create a class named Person, with firstname and lastname properties, and a printname method:

class Person:  
  def \_\_init\_\_(self, fname, lname):  
    self.firstname = fname  
    self.lastname = lname  
  
  def printname(self):  
    print(self.firstname, self.lastname)  
  
#Use the Person class to create an object, and then execute the printname method:  
  
x = Person("John", "Doe")  
x.printname()

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

## Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

### Example

Create a class named Student, which will inherit the properties and methods from the Person class:

class Student(Person):  
  pass

**Note:** Use the pass keyword when you do not want to add any other properties or methods to the class.

Now the Student class has the same properties and methods as the Person class.

### Example

Use the Student class to create an object, and then execute the printname method:

x = Student("Mike", "Olsen")  
x.printname()

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

## Add the \_\_init\_\_() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the \_\_init\_\_() function to the child class (instead of the pass keyword).

**Note:** The \_\_init\_\_() function is called automatically every time the class is being used to create a new object.

### Example

Add the \_\_init\_\_() function to the Student class:

class Student(Person):  
  def \_\_init\_\_(self, fname, lname):  
    #add properties etc.

When you add the \_\_init\_\_() function, the child class will no longer inherit the parent's \_\_init\_\_() function.

**Note:** The child's \_\_init\_\_() function **overrides** the inheritance of the parent's \_\_init\_\_() function.

To keep the inheritance of the parent's \_\_init\_\_() function, add a call to the parent's \_\_init\_\_() function:

### Example

class Student(Person):  
  def \_\_init\_\_(self, fname, lname):  
    Person.\_\_init\_\_(self, fname, lname)

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

Now we have successfully added the \_\_init\_\_() function, and kept the inheritance of the parent class, and we are ready to add functionality in the \_\_init\_\_() function.

## Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

### Example

class Student(Person):  
  def \_\_init\_\_(self, fname, lname):  
    super().\_\_init\_\_(fname, lname)

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

By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

## Add Properties

### Example

Add a property called graduationyear to the Student class:

class Student(Person):  
  def \_\_init\_\_(self, fname, lname):  
    super().\_\_init\_\_(fname, lname)  
    self.graduationyear = 2019

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

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the \_\_init\_\_() function:

### Example

Add a year parameter, and pass the correct year when creating objects:

class Student(Person):  
  def \_\_init\_\_(self, fname, lname, year):  
    super().\_\_init\_\_(fname, lname)  
    self.graduationyear = year  
  
x = Student("Mike", "Olsen", 2019)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_inheritance_add_prop2)

## Add Methods

### Example

Add a method called welcome to the Student class:

class Student(Person):  
  def \_\_init\_\_(self, fname, lname, year):  
    super().\_\_init\_\_(fname, lname)  
    self.graduationyear = year  
  
  def welcome(self):  
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

# Python Iterators

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

## Python Iterators

An iterator is an object that contains a countable number of values.

An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.

Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods \_\_iter\_\_() and \_\_next\_\_().

## Iterator vs Iterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable containers which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

### Example

Return an iterator from a tuple, and print each value:

mytuple = ("apple", "banana", "cherry")  
myit = iter(mytuple)  
  
print(next(myit))  
print(next(myit))  
print(next(myit))

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

Even strings are iterable objects, and can return an iterator:

### Example

Strings are also iterable objects, containing a sequence of characters:

mystr = "banana"  
myit = iter(mystr)  
  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))  
print(next(myit))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator2)

## Looping Through an Iterator

We can also use a for loop to iterate through an iterable object:

### Example

Iterate the values of a tuple:

mytuple = ("apple", "banana", "cherry")  
  
for x in mytuple:  
  print(x)

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

### Example

Iterate the characters of a string:

mystr = "banana"  
  
for x in mystr:  
  print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_iterator_loop2)

The for loop actually creates an iterator object and executes the next() method for each loop.

## Create an Iterator

To create an object/class as an iterator you have to implement the methods \_\_iter\_\_() and \_\_next\_\_() to your object.

As you have learned in the [Python Classes/Objects](https://www.w3schools.com/python/python_classes.asp) chapter, all classes have a function called \_\_init\_\_(), which allows you to do some initializing when the object is being created.

The \_\_iter\_\_() method acts similar, you can do operations (initializing etc.), but must always return the iterator object itself.

The \_\_next\_\_() method also allows you to do operations, and must return the next item in the sequence.

### Example

Create an iterator that returns numbers, starting with 1, and each sequence will increase by one (returning 1,2,3,4,5 etc.):

class MyNumbers:  
  def \_\_iter\_\_(self):  
    self.a = 1  
    return self  
  
  def \_\_next\_\_(self):  
    x = self.a  
    self.a += 1  
    return x  
  
myclass = MyNumbers()  
myiter = iter(myclass)  
  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))  
print(next(myiter))

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

## StopIteration

The example above would continue forever if you had enough next() statements, or if it was used in a for loop.

To prevent the iteration to go on forever, we can use the StopIteration statement.

In the \_\_next\_\_() method, we can add a terminating condition to raise an error if the iteration is done a specified number of times:

### Example

Stop after 20 iterations:

class MyNumbers:  
  def \_\_iter\_\_(self):  
    self.a = 1  
    return self  
  
  def \_\_next\_\_(self):  
    if self.a <= 20:  
      x = self.a  
      self.a += 1  
      return x  
    else:  
      raise StopIteration  
  
myclass = MyNumbers()  
myiter = iter(myclass)  
  
for x in myiter:  
  print(x)

# Python Scope

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

A variable is only available from inside the region it is created. This is called **scope**.

## Local Scope

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

### Example

A variable created inside a function is available inside that function:

def myfunc():  
  x = 300  
  print(x)  
  
myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope1)

### Function Inside Function

As explained in the example above, the variable x is not available outside the function, but it is available for any function inside the function:

### Example

The local variable can be accessed from a function within the function:

def myfunc():  
  x = 300  
  def myinnerfunc():  
    print(x)  
  myinnerfunc()  
  
myfunc()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope2)

## Global Scope

A variable created in the main body of the Python code is a global variable and belongs to the global scope.

Global variables are available from within any scope, global and local.

### Example

A variable created outside of a function is global and can be used by anyone:

x = 300  
  
def myfunc():  
  print(x)  
  
myfunc()  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope3)

### Naming Variables

If you operate with the same variable name inside and outside of a function, Python will treat them as two separate variables, one available in the global scope (outside the function) and one available in the local scope (inside the function):

### Example

The function will print the local x, and then the code will print the global x:

x = 300  
  
def myfunc():  
  x = 200  
  print(x)  
  
myfunc()  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope4)

## Global Keyword

If you need to create a global variable, but are stuck in the local scope, you can use the global keyword.

The global keyword makes the variable global.

### Example

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

def myfunc():  
  global x  
  x = 300  
  
myfunc()  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_scope5)

Also, use the global keyword if you want to make a change to 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 = 300  
  
def myfunc():  
  global x  
  x = 200  
  
myfunc()  
  
print(x)

# Python Modules

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

## What is a Module?

Consider a module to be the same as a code library.

A file containing a set of functions you want to include in your application.

## Create a Module

To create a module just save the code you want in a file with the file extension .py:

### Example

Save this code in a file named mymodule.py

def greeting(name):  
  print("Hello, " + name)

## Use a Module

Now we can use the module we just created, by using the import statement:

### Example

Import the module named mymodule, and call the greeting function:

import mymodule  
  
mymodule.greeting("Jonathan")

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module1)

**Note:** When using a function from a module, use the syntax: module\_name.function\_name.

## Variables in Module

The module can contain functions, as already described, but also variables of all types (arrays, dictionaries, objects etc):

### Example

Save this code in the file mymodule.py

person1 = {  
  "name": "John",  
  "age": 36,  
  "country": "Norway"  
}

### Example

Import the module named mymodule, and access the person1 dictionary:

import mymodule  
  
a = mymodule.person1["age"]  
print(a)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module2)

## Naming a Module

You can name the module file whatever you like, but it must have the file extension .py

## Re-naming a Module

You can create an alias when you import a module, by using the as keyword:

### Example

Create an alias for mymodule called mx:

import mymodule as mx  
  
a = mx.person1["age"]  
print(a)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module3)

## Built-in Modules

There are several built-in modules in Python, which you can import whenever you like.

### Example

Import and use the platform module:

import platform  
  
x = platform.system()  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_module4)

## Using the dir() Function

There is a built-in function to list all the function names (or variable names) in a module. The dir() function:

### Example

List all the defined names belonging to the platform module:

import platform  
  
x = dir(platform)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_module5)

**Note:** The dir() function can be used on all modules, also the ones you create yourself.

## Import From Module

You can choose to import only parts from a module, by using the from keyword.

### Example

The module named mymodule has one function and one dictionary:

def greeting(name):  
  print("Hello, " + name)  
  
person1 = {  
  "name": "John",  
  "age": 36,  
  "country": "Norway"  
}

### Example

Import only the person1 dictionary from the module:

from mymodule import person1  
  
print (person1["age"])

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_module6)

**Note:** When importing using the from keyword, do not use the module name when referring to elements in the module. Example: person1["age"], **not** ~~mymodule.person1["age"]~~

# Python Datetime

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

## Python Dates

A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

### Example

Import the datetime module and display the current date:

import datetime  
  
x = datetime.datetime.now()  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime1)

## Date Output

When we execute the code from the example above the result will be:

2020-03-25 00:18:04.660615

The date contains year, month, day, hour, minute, second, and microsecond.

The datetime module has many methods to return information about the date object.

Here are a few examples, you will learn more about them later in this chapter:

### Example

Return the year and name of weekday:

import datetime  
  
x = datetime.datetime.now()  
  
print(x.year)  
print(x.strftime("%A"))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime2)

## Creating Date Objects

To create a date, we can use the datetime() class (constructor) of the datetime module.

The datetime() class requires three parameters to create a date: year, month, day.

### Example

Create a date object:

import datetime  
  
x = datetime.datetime(2020, 5, 17)  
  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime3)

The datetime() class also takes parameters for time and timezone (hour, minute, second, microsecond, tzone), but they are optional, and has a default value of 0, (None for timezone).

## The strftime() Method

The datetime object has a method for formatting date objects into readable strings.

The method is called strftime(), and takes one parameter, format, to specify the format of the returned string:

### Example

Display the name of the month:

import datetime  
  
x = datetime.datetime(2018, 6, 1)  
  
print(x.strftime("%B"))

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

A reference of all the legal format codes:

|  |  |  |  |
| --- | --- | --- | --- |
| **Directive** | **Description** | **Example** | **Try it** |
| %a | Weekday, short version | Wed | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a) |
| %A | Weekday, full version | Wednesday | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_a2) |
| %w | Weekday as a number 0-6, 0 is Sunday | 3 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w) |
| %d | Day of month 01-31 | 31 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_d) |
| %b | Month name, short version | Dec | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b) |
| %B | Month name, full version | December | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_b2) |
| %m | Month as a number 01-12 | 12 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m) |
| %y | Year, short version, without century | 18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y) |
| %Y | Year, full version | 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_y2) |
| %H | Hour 00-23 | 17 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_h2) |
| %I | Hour 00-12 | 05 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_i2) |
| %p | AM/PM | PM | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_p) |
| %M | Minute 00-59 | 41 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_m2) |
| %S | Second 00-59 | 08 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_s2) |
| %f | Microsecond 000000-999999 | 548513 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_f) |
| %z | UTC offset | +0100 |  |
| %Z | Timezone | CST |  |
| %j | Day number of year 001-366 | 365 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_j) |
| %U | Week number of year, Sunday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_u2) |
| %W | Week number of year, Monday as the first day of week, 00-53 | 52 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_w2) |
| %c | Local version of date and time | Mon Dec 31 17:41:00 2018 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_c) |
| %x | Local version of date | 12/31/18 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x) |
| %X | Local version of time | 17:41:00 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_x2) |
| %% | A % character | % | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_datetime_strftime_percent) |

# Python JSON

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

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

## JSON in Python

Python has a built-in package called json, which can be used to work with JSON data.

### Example

Import the json module:

import json

## Parse JSON - Convert from JSON to Python

If you have a JSON string, you can parse it by using the json.loads() method.

The result will be a [Python dictionary](https://www.w3schools.com/python/python_dictionaries.asp).

### Example

Convert from JSON to Python:

import json  
  
# some JSON:  
x =  '{ "name":"John", "age":30, "city":"New York"}'  
  
# parse x:  
y = json.loads(x)  
  
# the result is a Python dictionary:  
print(y["age"])

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

## Convert from Python to JSON

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

### Example

Convert from Python to JSON:

import json  
  
# a Python object (dict):  
x = {  
  "name": "John",  
  "age": 30,  
  "city": "New York"  
}  
  
# convert into JSON:  
y = json.dumps(x)  
  
# the result is a JSON string:  
print(y)

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

You can convert Python objects of the following types, into JSON strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None

### Example

Convert Python objects into JSON strings, and print the values:

import json  
  
print(json.dumps({"name": "John", "age": 30}))  
print(json.dumps(["apple", "bananas"]))  
print(json.dumps(("apple", "bananas")))  
print(json.dumps("hello"))  
print(json.dumps(42))  
print(json.dumps(31.76))  
print(json.dumps(True))  
print(json.dumps(False))  
print(json.dumps(None))

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

When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

### Example

Convert a Python object containing all the legal data types:

import json  
  
x = {  
  "name": "John",  
  "age": 30,  
  "married": True,  
  "divorced": False,  
  "children": ("Ann","Billy"),  
  "pets": None,  
  "cars": [  
    {"model": "BMW 230", "mpg": 27.5},  
    {"model": "Ford Edge", "mpg": 24.1}  
  ]  
}  
  
print(json.dumps(x))

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

## Format the Result

The example above prints a JSON string, but it is not very easy to read, with no indentations and line breaks.

The json.dumps() method has parameters to make it easier to read the result:

### Example

Use the indent parameter to define the numbers of indents:

json.dumps(x, indent=4)

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

You can also define the separators, default value is (", ", ": "), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values:

### Example

Use the separators parameter to change the default separator:

json.dumps(x, indent=4, separators=(". ", " = "))

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

## Order the Result

The json.dumps() method has parameters to order the keys in the result:

### Example

Use the sort\_keys parameter to specify if the result should be sorted or not:

json.dumps(x, indent=4, sort\_keys=True)

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

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

# Python RegEx

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

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

## RegEx Module

Python has a built-in package called re, which can be used to work with Regular Expressions.

Import the re module:

import re

## RegEx in Python

When you have imported the re module, you can start using regular expressions:

### Example

Search the string to see if it starts with "The" and ends with "Spain":

import re  
  
txt = "The rain in Spain"  
x = re.search("^The.\*Spain$", txt)

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

## RegEx Functions

The re module offers a set of functions that allows us to search a string for a match:

|  |  |
| --- | --- |
| **Function** | **Description** |
| [findall](https://www.w3schools.com/python/python_regex.asp#findall) | Returns a list containing all matches |
| [search](https://www.w3schools.com/python/python_regex.asp#search) | Returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match anywhere in the string |
| [split](https://www.w3schools.com/python/python_regex.asp#split) | Returns a list where the string has been split at each match |
| [sub](https://www.w3schools.com/python/python_regex.asp#sub) | Replaces one or many matches with a string |

## Metacharacters

Metacharacters are characters with a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| [] | A set of characters | "[a-m]" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta1) |
| \ | Signals a special sequence (can also be used to escape special characters) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta2) |
| . | Any character (except newline character) | "he..o" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta3) |
| ^ | Starts with | "^hello" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta4) |
| $ | Ends with | "world$" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta5) |
| \* | Zero or more occurrences | "aix\*" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta6) |
| + | One or more occurrences | "aix+" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta7) |
| {} | Exactly the specified number of occurrences | "al{2}" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta8) |
| | | Either or | "falls|stays" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_meta9) |
| () | Capture and group |  |  |

## Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

|  |  |  |  |
| --- | --- | --- | --- |
| **Character** | **Description** | **Example** | **Try it** |
| \A | Returns a match if the specified characters are at the beginning of the string | "\AThe" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq1) |
| \b | Returns a match where the specified characters are at the beginning or at the end of a word | r"\bain" r"ain\b" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2) [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq2-2) |
| \B | Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word | r"\Bain" r"ain\B" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3) [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq3-2) |
| \d | Returns a match where the string contains digits (numbers from 0-9) | "\d" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq4) |
| \D | Returns a match where the string DOES NOT contain digits | "\D" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq5) |
| \s | Returns a match where the string contains a white space character | "\s" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq6) |
| \S | Returns a match where the string DOES NOT contain a white space character | "\S" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq7) |
| \w | Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character) | "\w" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq8) |
| \W | Returns a match where the string DOES NOT contain any word characters | "\W" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq9) |
| \Z | Returns a match if the specified characters are at the end of the string | "Spain\Z" | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_seq10) |

## Sets

A set is a set of characters inside a pair of square brackets [] with a special meaning:

|  |  |  |
| --- | --- | --- |
| **Set** | **Description** | **Try it** |
| [arn] | Returns a match where one of the specified characters (a, r, or n) are present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set1) |
| [a-n] | Returns a match for any lower case character, alphabetically between a and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set2) |
| [^arn] | Returns a match for any character EXCEPT a, r, and n | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set3) |
| [0123] | Returns a match where any of the specified digits (0, 1, 2, or 3) are present | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set4) |
| [0-9] | Returns a match for any digit between 0 and 9 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set5) |
| [0-5][0-9] | Returns a match for any two-digit numbers from 00 and 59 | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set6) |
| [a-zA-Z] | Returns a match for any character alphabetically between a and z, lower case OR upper case | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set7) |
| [+] | In sets, +, \*, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_set8) |

## The findall() Function

The findall() function returns a list containing all matches.

### Example

Print a list of all matches:

import re  
  
txt = "The rain in Spain"  
x = re.findall("ai", txt)  
print(x)

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

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

### Example

Return an empty list if no match was found:

import re  
  
txt = "The rain in Spain"  
x = re.findall("Portugal", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_findall2)

## The search() Function

The search() function searches the string for a match, and returns a [Match object](https://www.w3schools.com/python/python_regex.asp#matchobject) if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

### Example

Search for the first white-space character in the string:

import re  
  
txt = "The rain in Spain"  
x = re.search("\s", txt)  
  
print("The first white-space character is located in position:", x.start())

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

If no matches are found, the value None is returned:

### Example

Make a search that returns no match:

import re  
  
txt = "The rain in Spain"  
x = re.search("Portugal", txt)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_search2)

## The split() Function

The split() function returns a list where the string has been split at each match:

### Example

Split at each white-space character:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt)  
print(x)

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

You can control the number of occurrences by specifying the maxsplit parameter:

### Example

Split the string only at the first occurrence:

import re  
  
txt = "The rain in Spain"  
x = re.split("\s", txt, 1)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_split2)

## The sub() Function

The sub() function replaces the matches with the text of your choice:

### Example

Replace every white-space character with the number 9:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt)  
print(x)

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

You can control the number of replacements by specifying the count parameter:

### Example

Replace the first 2 occurrences:

import re  
  
txt = "The rain in Spain"  
x = re.sub("\s", "9", txt, 2)  
print(x)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_regex_sub2)

## Match Object

A Match Object is an object containing information about the search and the result.

**Note:** If there is no match, the value None will be returned, instead of the Match Object.

### Example

Do a search that will return a Match Object:

import re  
  
txt = "The rain in Spain"  
x = re.search("ai", txt)  
print(x) #this will print an object

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

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.  
.string returns the string passed into the function  
.group() returns the part of the string where there was a match

### Example

Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.span()**)

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

### Example

Print the string passed into the function:

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.string**)

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

### Example

Print the part of the string where there was a match.

The regular expression looks for any words that starts with an upper case "S":

import re  
  
txt = "The rain in Spain"  
x = re.search(r"\bS\w+", txt)  
print(**x.group()**)

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

**Note:** If there is no match, the value None will be returned, instead of the Match Object.

# Python PIP

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

## What is PIP?

PIP is a package manager for Python packages, or modules if you like.

**Note:** If you have Python version 3.4 or later, PIP is included by default.

## What is a Package?

A package contains all the files you need for a module.

Modules are Python code libraries you can include in your project.

## Check if PIP is Installed

Navigate your command line to the location of Python's script directory, and type the following:

### Example

Check PIP version:

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version

## Install PIP

If you do not have PIP installed, you can download and install it from this page: <https://pypi.org/project/pip/>

## Download a Package

Downloading a package is very easy.

Open the command line interface and tell PIP to download the package you want.

Navigate your command line to the location of Python's script directory, and type the following:

### Example

Download a package named "camelcase":

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip install camelcase

Now you have downloaded and installed your first package!

## Using a Package

Once the package is installed, it is ready to use.

Import the "camelcase" package into your project.

### Example

Import and use "camelcase":

import camelcase  
  
c = camelcase.CamelCase()  
  
txt = "hello world"  
  
print(c.hump(txt))

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_camelcase)

## Find Packages

Find more packages at <https://pypi.org/>.

## Remove a Package

Use the uninstall command to remove a package:

### Example

Uninstall the package named "camelcase":

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip uninstall camelcase

The PIP Package Manager will ask you to confirm that you want to remove the camelcase package:

Uninstalling camelcase-02.1:  
  Would remove:  
    c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camecase-0.2-py3.6.egg-info  
    c:\users\Your Name\appdata\local\programs\python\python36-32\lib\site-packages\camecase\\*  
Proceed (y/n)?

Press y and the package will be removed.

## List Packages

Use the list command to list all the packages installed on your system:

### Example

List installed packages:

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip list

Result:

Package         Version  
-----------------------  
camelcase       0.2  
mysql-connector 2.1.6  
pip             18.1  
pymongo         3.6.1  
setuptools      39.0.1

# Python JSON

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

JSON is a syntax for storing and exchanging data.

JSON is text, written with JavaScript object notation.

## JSON in Python

Python has a built-in package called json, which can be used to work with JSON data.

### Example

Import the json module:

import json

## Parse JSON - Convert from JSON to Python

If you have a JSON string, you can parse it by using the json.loads() method.

The result will be a [Python dictionary](https://www.w3schools.com/python/python_dictionaries.asp).

### Example

Convert from JSON to Python:

import json  
  
# some JSON:  
x =  '{ "name":"John", "age":30, "city":"New York"}'  
  
# parse x:  
y = json.loads(x)  
  
# the result is a Python dictionary:  
print(y["age"])

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

## Convert from Python to JSON

If you have a Python object, you can convert it into a JSON string by using the json.dumps() method.

### Example

Convert from Python to JSON:

import json  
  
# a Python object (dict):  
x = {  
  "name": "John",  
  "age": 30,  
  "city": "New York"  
}  
  
# convert into JSON:  
y = json.dumps(x)  
  
# the result is a JSON string:  
print(y)

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

You can convert Python objects of the following types, into JSON strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None

### Example

Convert Python objects into JSON strings, and print the values:

import json  
  
print(json.dumps({"name": "John", "age": 30}))  
print(json.dumps(["apple", "bananas"]))  
print(json.dumps(("apple", "bananas")))  
print(json.dumps("hello"))  
print(json.dumps(42))  
print(json.dumps(31.76))  
print(json.dumps(True))  
print(json.dumps(False))  
print(json.dumps(None))

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

When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

### Example

Convert a Python object containing all the legal data types:

import json  
  
x = {  
  "name": "John",  
  "age": 30,  
  "married": True,  
  "divorced": False,  
  "children": ("Ann","Billy"),  
  "pets": None,  
  "cars": [  
    {"model": "BMW 230", "mpg": 27.5},  
    {"model": "Ford Edge", "mpg": 24.1}  
  ]  
}  
  
print(json.dumps(x))

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

## Format the Result

The example above prints a JSON string, but it is not very easy to read, with no indentations and line breaks.

The json.dumps() method has parameters to make it easier to read the result:

### Example

Use the indent parameter to define the numbers of indents:

json.dumps(x, indent=4)

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

You can also define the separators, default value is (", ", ": "), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values:

### Example

Use the separators parameter to change the default separator:

json.dumps(x, indent=4, separators=(". ", " = "))

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

## Order the Result

The json.dumps() method has parameters to order the keys in the result:

### Example

Use the sort\_keys parameter to specify if the result should be sorted or not:

json.dumps(x, indent=4, sort\_keys=True)

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

# Python Try Except

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

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

## Exception Handling

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

### Example

The try block will generate an exception, because x is not defined:

try:  
  print(x)  
except:  
  print("An exception occurred")

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

Since the try block raises an error, the except block will be executed.

Without the try block, the program will crash and raise an error:

### Example

This statement will raise an error, because x is not defined:

print(x)

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

## Many Exceptions

You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error:

### Example

Print one message if the try block raises a NameError and another for other errors:

try:  
  print(x)  
except NameError:  
  print("Variable x is not defined")  
except:  
  print("Something else went wrong")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except2)

## Else

You can use the else keyword to define a block of code to be executed if no errors were raised:

### Example

In this example, the try block does not generate any error:

try:  
  print("Hello")  
except:  
  print("Something went wrong")  
else:  
  print("Nothing went wrong")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except3)

## Finally

The finally block, if specified, will be executed regardless if the try block raises an error or not.

### Example

try:  
  print(x)  
except:  
  print("Something went wrong")  
finally:  
  print("The 'try except' is finished")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except4)

This can be useful to close objects and clean up resources:

### Example

Try to open and write to a file that is not writable:

try:  
  f = open("demofile.txt")  
  f.write("Lorum Ipsum")  
except:  
  print("Something went wrong when writing to the file")  
finally:  
  f.close()

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_try_except5)

The program can continue, without leaving the file object open.

## Raise an exception

As a Python developer you can choose to throw an exception if a condition occurs.

To throw (or raise) an exception, use the raise keyword.

### Example

Raise an error and stop the program if x is lower than 0:

x = -1  
  
if x < 0:  
  raise Exception("Sorry, no numbers below zero")

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

The raise keyword is used to raise an exception.

You can define what kind of error to raise, and the text to print to the user.

### Example

Raise a TypeError if x is not an integer:

x = "hello"  
  
if not type(x) is int:  
  raise TypeError("Only integers are allowed")

# Python User Input

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

## User Input

Python allows for user input.

That means we are able to ask the user for input.

The method is a bit different in Python 3.6 than Python 2.7.

Python 3.6 uses the input() method.

Python 2.7 uses the raw\_input() method.

The following example asks for the username, and when you entered the username, it gets printed on the screen:

### Python 3.6

username = input("Enter username:")  
print("Username is: " + username)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_user_input3)

### Python 2.7

username = raw\_input("Enter username:")  
print("Username is: " + username)

[Run Example »](https://www.w3schools.com/python/showpython.asp?filename=demo_user_input2)

Python stops executing when it comes to the input() function, and continues when the user has given some input.

# Python String Formatting

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

To make sure a string will display as expected, we can format the result with the format() method.

## String format()

The format() method allows you to format selected parts of a string.

Sometimes there are parts of a text that you do not control, maybe they come from a database, or user input?

To control such values, add placeholders (curly brackets {}) in the text, and run the values through the format() method:

### Example

Add a placeholder where you want to display the price:

price = 49  
txt = "The price is {} dollars"  
print(txt.format(price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting1)

You can add parameters inside the curly brackets to specify how to convert the value:

### Example

Format the price to be displayed as a number with two decimals:

txt = "The price is {:.2f} dollars"

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting2)

Check out all formatting types in our [String format() Reference](https://www.w3schools.com/python/ref_string_format.asp).

## Multiple Values

If you want to use more values, just add more values to the format() method:

print(txt.format(price, itemno, count))

And add more placeholders:

### Example

quantity = 3  
itemno = 567  
price = 49  
myorder = "I want {} pieces of item number {} for {:.2f} dollars."  
print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting3)

## Index Numbers

You can use index numbers (a number inside the curly brackets {0}) to be sure the values are placed in the correct placeholders:

### Example

quantity = 3  
itemno = 567  
price = 49  
myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."  
print(myorder.format(quantity, itemno, price))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting4)

Also, if you want to refer to the same value more than once, use the index number:

### Example

age = 36  
name = "John"  
txt = "His name is {1}. {1} is {0} years old."  
print(txt.format(age, name))

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_string_formatting5)

## Named Indexes

You can also use named indexes by entering a name inside the curly brackets {carname}, but then you must use names when you pass the parameter values txt.format(carname = "Ford"):

### Example

myorder = "I have a {carname}, it is a {model}."  
print(myorder.format(carname = "Ford", model = "Mustang"))