

# Week 07 - support

on file handling i.e. read and write

# File:



<https://data-flair.training/blogs/python-file/>

# Python File i/o – Python Write to File and Python Read from File

BY DATAFLAIR

## 1. Python File i/o – Python Write to File and Python Read from File

An important concept in Python is File i/o. In this article, we will learn about File i/o in Python. We will learn how to write to a file and how to read from a file. We will also learn about the different file modes available in Python.

Reading and Writing to text files in Python

Reading and Writing to text files in Python

Why is Python the Best-Suited Programming Language for Machine Learning?

How to Start Learning Machine Learning?

12 Reasons Why You Should Learn Python in 2019

Python | Django News App

Python | Speech recognition on large audio files

Python | Training Image

## 2. Why is Python the Best-Suited Programming Language for Machine Learning?

A file is a collection of data that is stored on a storage device. It is used to store data permanently. To read from or write to a file, we use file i/o.

# GeeksforGeeks

A computer science portal for geeks

Google Custom Search

[Home](#)
[Algo](#)
[DS](#)
[Languages](#)
[Interview](#)
[Students](#)
[GATE](#)
[CS Subjects](#)
[Quizzes](#)

## Reading and Writing to text files in Python

Python provides inbuilt functions for creating, writing and reading files. There are two types of files that can be handled in python, normal text files and binary files (written in binary language, 0s and 1s).

- Text files:** In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character ('\n') in python by default.
- Binary files:** In this type of file, there is no terminator for a line and the data is stored after converting it into machine understandable binary language.

In this article, we will be focusing on opening, closing, reading and writing data in a text file.

### File Access Modes

Ad closed by

[Report this ad](#)

# What is a file?

Files on most modern file systems are composed of three main parts:

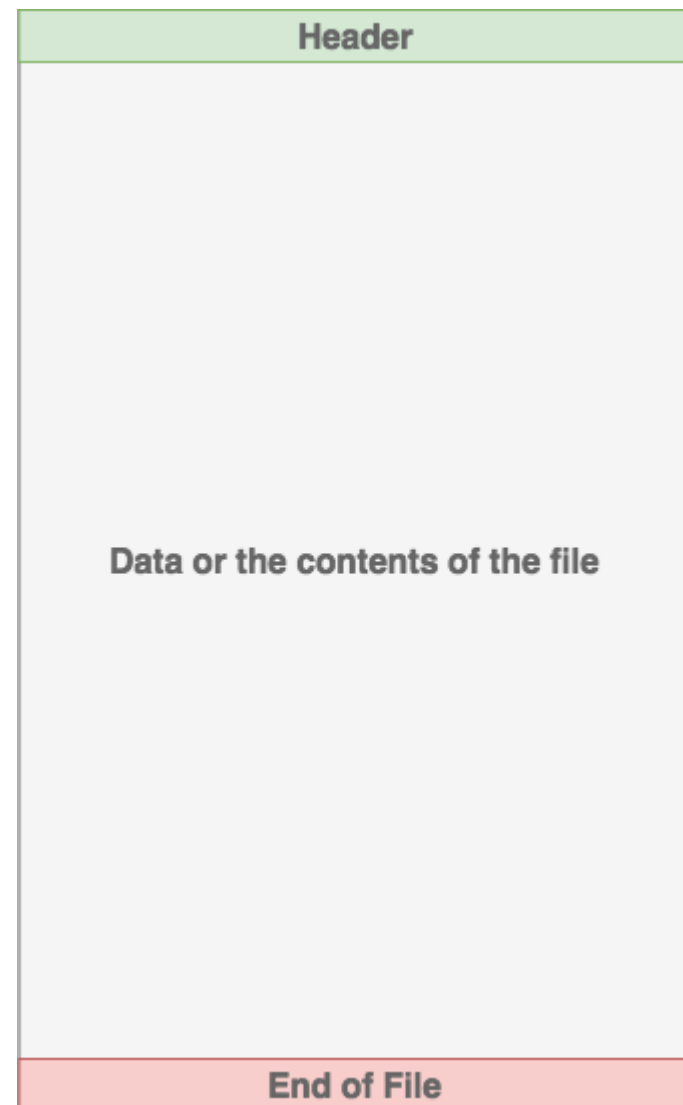
1. **Header:** metadata about the contents of the file (file name, size, type, and so on)
2. **Data:** contents of the file as written by the creator or editor
3. **End of file (EOF):** special character that indicates the end of the file

There are three different categories of file objects:

- Text files
- Buffered binary files
- Raw binary files

Most of the focus

<https://realpython.com/read-write-files-python/>



## Python Tutorial

[Python 3 - About](#)[Python 3 - Installation](#)[Python 3 - Environment](#)[Python 3 - Hello World](#)[Python 3 - Variables](#)[Python 3 - User Input](#)[Python 3 - Strings](#)[Python 3 - Lists](#)[Python 3 - Tuples](#)[Python 3 - Dictionary](#)[Python 3 - Numbers](#)[Python 3 - Operators](#)[Python 3 - If Statements](#)[Python 3 - While Loops](#)[Python 3 - For Loops](#)[Python 3 - Functions](#)[Python 3 - Modules](#)[Python 3 - Classes](#)[Python 3 - Read/Write Files](#)

## Python Reference

[Python 3 - Operators](#)[Python 3 - Escape Sequences](#)<http://python-ds.com/python-read-write-files>[Python 3 - String Methods](#)

# Python Read Write Files

[← Python Classes](#)[About Python →](#)

Python has various methods for performing operations against files. Here's an overview.

## Open a File

When you read or write a file, the first thing you need to do is open it (or create it). Python provides the `open()` method that is used to open a file. It also creates the file if it doesn't already exist.

The `open()` syntax is like this:

```
open(filename, mode)
```

It takes two parameters. The first one provides the name of the file, and the second parameter specifies the mode to be used. A mode of `r` means "read", `a` means "append". You can append the mode with `b` to specify binary mode. You can also do stuff like `r+` to make it read



## Create a File

Here's an example of creating a new file:

```
# Create the file in 'write' mode
f = open("hello.txt", "w")

# Write some text to the file
f.write("Hello World!")

# Close the file
f.close()
```

## Read a File

Now that we've created a file and added some content to it, we can read it. Here's how:

```
# Open the file in 'read' mode
f = open("hello.txt", "r")

# Put the contents of the file into a variable
f_contents = f.read()

# Close the file
f.close()

# Print the file's contents
print(f_contents)
```

# Need to close or use with...

- Close is to leave the file alone

```
reader = open('dog_breeds.txt')  
reader.close()
```

- Using with the close is automatic – after doing what is inside with

```
with open('dog_breeds.txt') as reader:  
    # Further file processing goes here
```

<https://realpython.com/read-write-files-python/>

# “With ... as ...”

## Read

```
with open('data.txt', 'r') as f:  
    data = f.read()
```

## Write

```
with open('data.txt', 'w') as f:  
    data = 'some data to be written to the file'  
    f.write(data)
```

<https://realpython.com/working-with-files-in-python/>



## File Modes in Python

Mode	Description
'r'	This is the default mode. It Opens file for reading.
'w'	This Mode Opens file for writing. If file does not exist, it creates a new file. If file exists it truncates the file.
'x'	Creates a new file. If file already exists, the operation fails.
'a'	Open file in append mode. If file does not exist, it creates a new file.
't'	This is the default mode. It opens in text mode.
'b'	This opens in binary mode.
'+'	This will open a file for reading and writing (updating)

<https://www.guru99.com/reading-and-writing-files-in-python.html>

# Writing and reading...

**#writing something in a new file “guru99.txt”**

```
f= open("guru99.txt","w+")  
for i in range(10):  
    f.write("This is line %d\r\n" % (i+1))  
f.close()
```

**#Open the file back and read the contents**

```
f=open("guru99.txt", "r")  
#option 1: reads all as a string  
contents =f.read()  
print (contents)  
#option 2 : readlines reads the individual line into a list  
fl =f.readlines()  
for x in fl:  
    print(x)
```

<https://www.guru99.com/reading-and-writing-files-in-python.html>

# Read a Single Line

You can use `readline()` to read a single line in the file:

```
f = open("hello.txt", "r")
line1 = f.readline()
line2 = f.readline()
print("Line 1:", line1)
print("Line 2:", line2)
f.close()
```

## Looping over the Lines

You can use a `for` loop to loop over each line in the file. Here's how that works:

```
f = open("hello.txt", "r")
for line in f:
    print("Line:", line, end="")
f.close()
```

<http://python-ds.com/python-read-write-files>



# Read Multiple Lines as a List

You can use the `readlines()` or the `list()` methods to return all lines in a list. Like this:

```
# Using 'readlines()'
f = open("hello.txt", "r")
f_content = f.readlines()
print(f_content)
f.close()

# Using 'list()'
f = open("hello.txt", "r")
f_content = list(f)
print(f_content)
f.close()
```

# Reading

Method	What It Does
<code>.read(size=-1)</code>	This reads from the file based on the number of <code>size</code> bytes. If no argument is passed or <code>None</code> or <code>-1</code> is passed, then the entire file is read.
<code>.readline(size=-1)</code>	This reads at most <code>size</code> number of characters from the line. This continues to the end of the line and then wraps back around. If no argument is passed or <code>None</code> or <code>-1</code> is passed, then the entire line (or rest of the line) is read.
<code>.readlines()</code>	This reads the remaining lines from the file object and returns

```
>>> f = open('dog_breeds.txt')
>>> f.readlines() # Returns a list object
['Pug\n', 'Jack Russell Terrier\n', 'English Springer Spaniel\n', 'German Shepherd\n']
```

```
>>> with open('dog_breeds.txt', 'r') as reader:
>>>     # Read & print the entire file
>>>     print(reader.read())
```

<https://realpython.com/read-write-files-python/>

# Writing... ( text files)

Method	What It Does
<code>.write(string)</code>	This writes the string to the file.
<code>.writelines(seq)</code>	This writes the sequence to the file. No line endings are appended to each sequence item. It's up to you to add the appropriate line ending(s).

```
with open('dog_breeds.txt', 'r') as reader:
    # Note: readlines doesn't trim the line endings
    dog_breeds = reader.readlines()

with open('dog_breeds_reversed.txt', 'w') as writer:
    # Alternatively you could use
    # writer.writelines(reversed(dog_breeds))

    # Write the dog breeds to the file in reversed order
    for breed in reversed(dog_breeds):
        writer.write(breed)
```

<https://realpython.com/read-write-files-python/>



# Work with two files

```
d_path = 'dog_breeds.txt'
d_r_path = 'dog_breeds_reversed.txt'
with open(d_path, 'r') as reader, open(d_r_path, 'w') as writer:
    dog_breeds = reader.readlines()
    writer.writelines(reversed(dog_breeds))
```

# Some notes on formats

## Don't Re-Invent the Snake

There are common situations that you may encounter while working with files. Most of these cases can be handled using other modules. Two common file types you may need to work with are `.csv` and `.json`. *Real Python* has already put together some great articles on how to handle these:

- [Reading and Writing CSV Files in Python](#)
- [Working With JSON Data in Python](#)

Additionally, there are built-in libraries out there that you can use to help you:

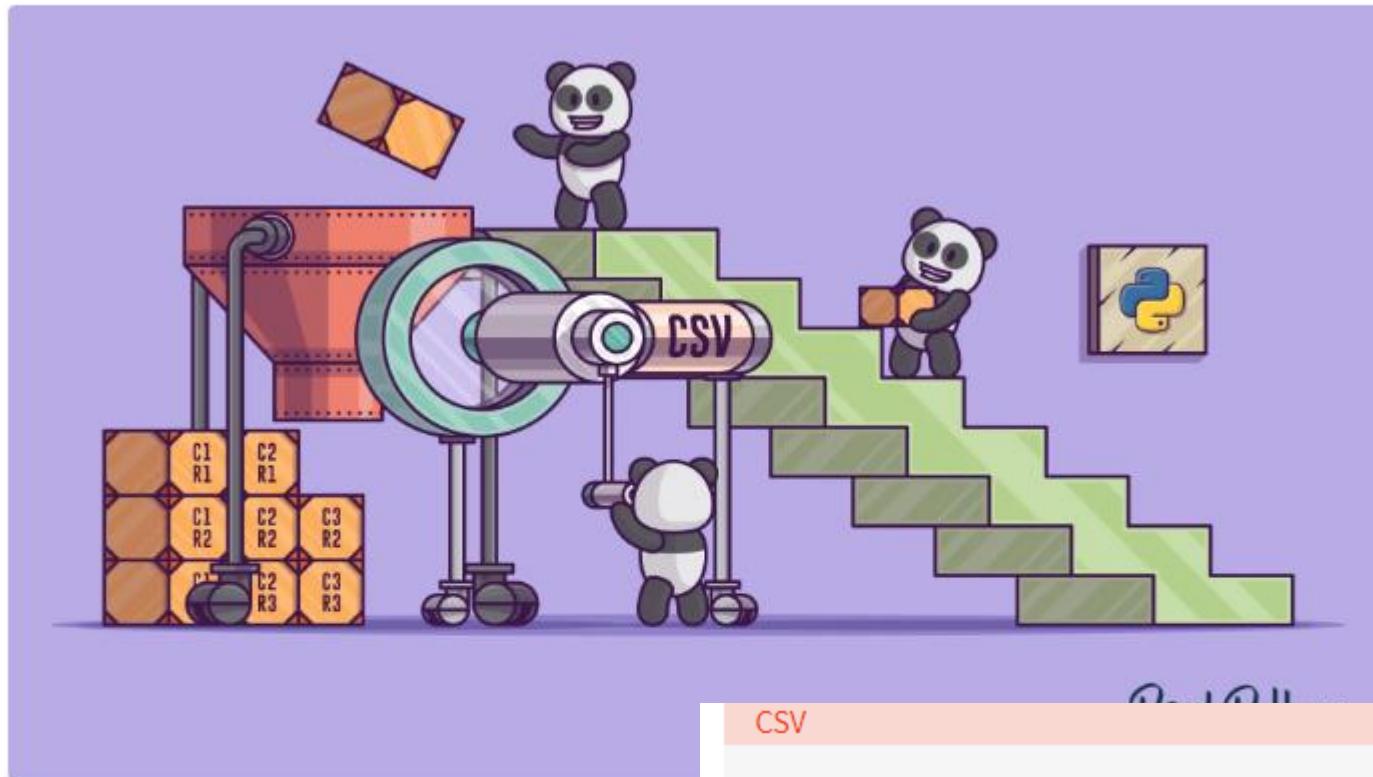
- `wave`: read and write WAV files (audio)
- `aifc`: read and write AIFF and AIFC files (audio)
- `sunau`: read and write Sun AU files
- `tarfile`: read and write tar archive files
- `zipfile`: work with ZIP archives
- `configparser`: easily create and parse configuration files
- `xml.etree.ElementTree`: create or read XML based files
- `msilib`: read and write Microsoft Installer files
- `plistlib`: generate and parse Mac OS X `.plist` files

There are plenty more out there. Additionally there are even more third party tools available on PyPI. Some popular ones are the following:

- `PyPDF2`: PDF toolkit
- `xlwings`: read and write Excel files
- `Pillow`: image reading and manipulation

<https://realpython.com/read-write-files-python/>  
<https://realpython.com/python-json/>  
<https://realpython.com/python-csv/>





# Reading and Writing

by Jon Fincher  57 Comments  data-science

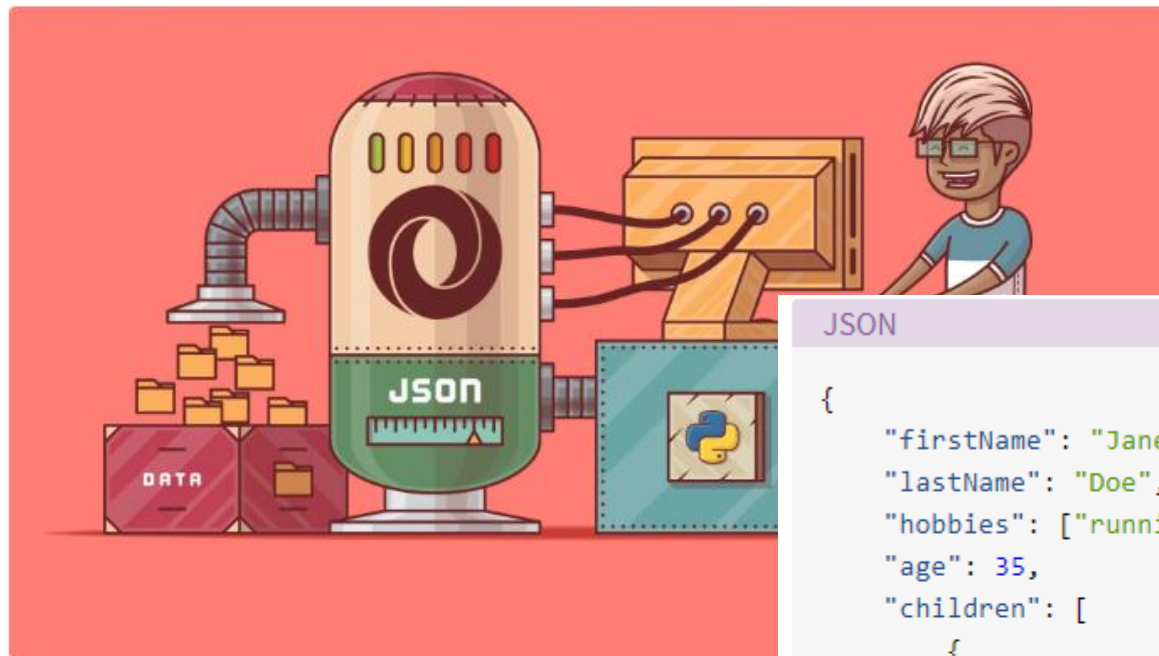
[Tweet](#)
[Share](#)
[Email](#)

## CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

## Table of Contents

<https://realpython.com/python-csv/>



# Working With JSON Data in Python

by Lucas Lofaro 34 Comments intermediate python

[Tweet](#) [Share](#) [Email](#)

## Table of Contents

- A (Very) Brief History of JSON
- Look, it's JSON!
- Python Supports JSON Natively!
  - A Little Vocabulary
  - Converting to JSON

<https://realpython.com/python-json/>

- Some Useful Keyword Arguments

## JSON

```
{
  "firstName": "Jane",
  "lastName": "Doe",
  "hobbies": ["running", "sky diving", "singing"],
  "age": 35,
  "children": [
    {
      "firstName": "Alice",
      "age": 6
    },
    {
      "firstName": "Bob",
      "age": 8
    }
  ]
}
```



# Processing lines

# strip

The **strip()** method returns a copy of the string in which all chars have been stripped from the beginning and the end of the string (default whitespace characters).

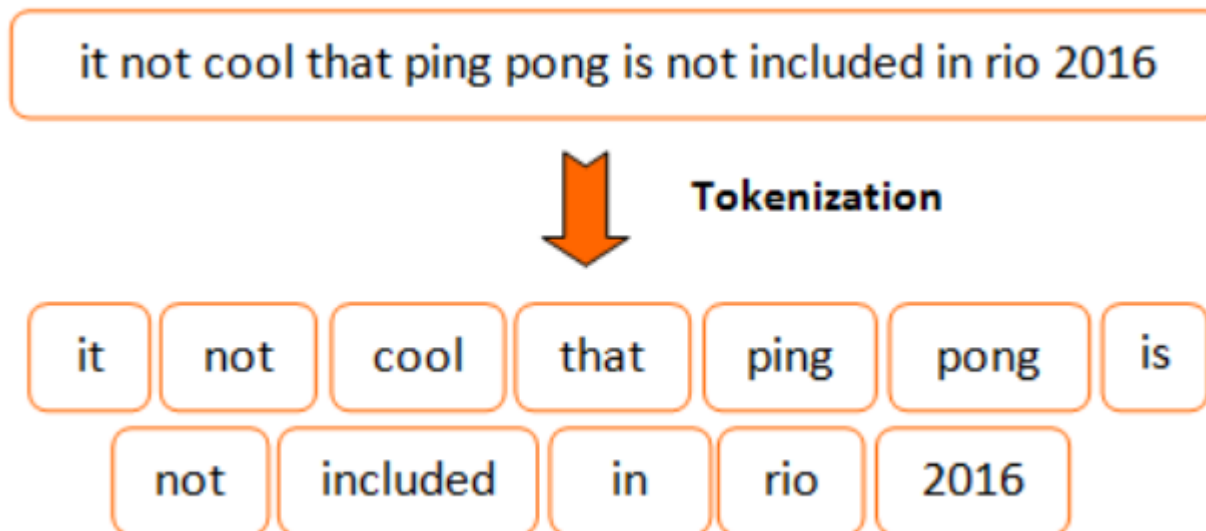
```
#!/usr/bin/python3

str = "*****this is string example....wow!!!*****"
print (str.strip( '*' ))
```

```
this is string example....wow!!!
```

# split

The **split()** method returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally limiting the number of splits to num.



[https://www.tutorialspoint.com/python3/string\\_split.htm](https://www.tutorialspoint.com/python3/string_split.htm)

# split

The **split()** method returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally limiting the number of splits to num.

```
['this', 'is', 'string', 'example....wow!!!']  
['th', 's is string example....wow!!!']  
['this is string example....', 'o', '!!!']
```

```
#!/usr/bin/python3  
  
str = "this is string example....wow!!!"  
print (str.split( ))  
print (str.split('i',1))  
print (str.split('w'))
```

[https://www.tutorialspoint.com/python3/string\\_split.htm](https://www.tutorialspoint.com/python3/string_split.htm)

# Typical processing

```
with open( ...filename ...) as f:
```

```
    # get line in f
```

```
    for line in f:
```

```
        # take out “empty” in beginning and end - strip
```

```
        # and split in list using space (default) p
```

```
        words = line.strip().split()
```

```
        # do something...
```

```
import sys

src = sys.argv[1]      # The input source file
dst = sys.argv[2]      # The file to output to

with open(src, 'r') as fin, open(dst, 'a') as fout:      # Open in append mode

    student = fin.readline().strip()      # Strip the newline character
    while student:

        student_data = student.split()      # ['Liam', '84'] for example

        name = student_data[0]
        mark = int(student_data[1])

        if mark < 40:
            grade = "FAIL"
        else:
            grade = "PASS"

        fout.write('{:s} {:s}\n'.format(name, grade))      # Write to the output file

        student = fin.readline().strip()      # Get the next student
```

[https://www.slitherintopython.com/book/chapter\\_10/chapter\\_10.html](https://www.slitherintopython.com/book/chapter_10/chapter_10.html)







## File I/O

### ? Overview

Teaching: 10 min

Exercises: 5 min

#### Questions

- How can I read data from a file?

### Use open to open file

- Arguments are a path and:
  - 'r' for reading
  - 'w' for writing (immediately erases existing file)
  - 'a' for appending
- Result is an object with methods for reading:
  - `file.read()` reads the entire file.
  - `file.read(N)` reads up to that many characters.
- Close files with `file.close()`.

```
reader = open('myfile.txt', 'r')
data = reader.read()
```

# thispointer.com

Python

Pandas

C++11 Tutorials

C++ Tutorials

STL

Multi

Design Patterns

java

Datastructure ▾

Subscribe

[Home](#) » [FileHandling](#) » [Python](#) » You are reading »

# 5 Different ways to read a file line by line in Python

👤 Varun

📅 September 29, 2018

📁 FileHandling, Python

💬 No Comment

<https://thispointer.com/5-different-ways-to-read-a-file-line-by-line-in-python/>

<https://swcarpentry.github.io/python-second-language/12-file-io/>

Python.



Suppose we have a file named `data.txt` in same directory as our python script. Let's see how to read its contents line by line.

# Handling directories & files

Just some examples... explore the links

## File & Directory Related Methods

There are three important sources, which provide a wide range of utility methods to handle and manipulate files & directories on Windows and Unix operating systems. They are as follows –

- File Object Methods  : The *file* object provides functions to manipulate files.
- OS Object Methods  : This provides methods to process files as well as directories.

[https://www.tutorialspoint.com/python/python\\_files\\_io.htm](https://www.tutorialspoint.com/python/python_files_io.htm)

# Paths & directories

Details out of scope



<https://data-flair.training/blogs/python-directory/>  
<https://realpython.com/read-write-files-python/>

# Handling directories

```
>>> os.listdir('my_directory/')
['sub_dir_c', 'file1.py', 'sub_dir_b', 'file3.txt', 'file2.csv', 'sub_dir']
```

```
from pathlib import Path

entries = Path('my_directory/')
for entry in entries.iterdir():
    print(entry.name)
```

```
sub_dir_c
file1.py
sub_dir_b
file3.txt
file2.csv
sub_dir
```

```
import os

# List all files in a directory using os.listdir
basepath = 'my_directory/'
for entry in os.listdir(basepath):
    if os.path.isfile(os.path.join(basepath, entry)):
        print(entry)
```

```
my_directory/
|
├── sub_dir/
|   ├── bar.py
|   └── foo.py
|
├── sub_dir_b/
|   └── file4.txt
|
├── sub_dir_c/
|   ├── config.py
|   └── file5.txt
|
├── file1.py
├── file2.csv
└── file3.txt
```

<https://realpython.com/working-with-files-in-python/>

# Handling directories

```
import os

# List all subdirectories using os.listdir
basepath = 'my_directory/'
for entry in os.listdir(basepath):
    if os.path.isdir(os.path.join(basepath, entry)):
        print(entry)
```

<https://realpython.com/working-with-files-in-python/>  
<https://www.programiz.com/python-programming/directory>  
<http://net-informations.com/python/file/directory.htm>  
[https://www.tutorialspoint.com/python/os\\_file\\_methods.htm](https://www.tutorialspoint.com/python/os_file_methods.htm)





LEARN PYTHON

programming language

## Python Basic Tutorial

- Python - Home
- Python - Overview
- Python - Environment Setup
- Python - Basic Syntax
- Python - Variable Types
- Python - Basic Operators
- Python - Decision Making
- Python - Loops
- Python - Numbers
- Python - Strings
- Python - Lists
- Python - Tuples
- Python - Dictionary
- Python - Date & Time
- Python - Functions
- Python - Modules

## PYTHON TUTORIAL

- Python Introduction
- Python Flow Control
- Python Functions
- Python Datatypes
- Python Files
  - Python File Operation
  - Python Directory
  - Python Exception
  - Exception Handling
  - User-defined Exception
  - Take Quiz

- Python Object & Class
- Python Advanced Topics
- Python Date and time

**START YOUR  
PYTHON CAREER  
NOW.**

Download Now!

# Python File Methods

Advertisements

Ad closed by Google

Report this ad

Why this ad? ▸

## Python File I/O

In this article, you'll learn about Python file operations. More specifically, opening a file, reading from it, writing into it, closing it and various file methods you should be aware of.



**LOOKING TO LEARN PROGRAMMING?**  
Start your programming journey with Programiz **AT NO COST.**



### What is a file?

File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).

Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.

When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order.

1. Open a file
2. Read or write (perform operation)
3. Close the file

### How to open a file?

This function returns a file object,  
we use it to read or write to the file accordingly.

#### Contents

- What is a file?
- How to open a file?
- How to close a file Using Python?
- How to write to File Using Python?
- How to read files in Python?
- Python File Methods

[https://www.tutorialspoint.com/python/file\\_methods.htm](https://www.tutorialspoint.com/python/file_methods.htm)

<https://www.programiz.com/python-programming/file-operation>



LEARN PYTHON

programming language

## Python Basic Tutorial

- Python - Home
- Python - Overview
- Python - Environment Setup
- Python - Basic Syntax
- Python - Variable Types
- Python - Basic Operators
- Python - Decision Making
- Python - Loops
- Python - Numbers
- Python - Strings
- Python - Lists
- Python - Tuples
- Python - Dictionary
- Python - Date & Time
- Python - Functions

## PYTHON TUTORIAL

- Python Introduction
- Python Flow Control
- Python Functions
- Python Datatypes
- Python Files
  - Python File Operation
  - Python Directory
  - Python Exception
  - Exception Handling
  - User-defined Exception
  - Take Quiz
- Python Object & Class
- Python Advanced Topics
- Python Date and time

**START YOUR  
PYTHON CAREER  
NOW.**

<https://www.programiz.com/python-programming/directory>  
[https://www.tutorialspoint.com/python/os\\_file\\_methods.htm](https://www.tutorialspoint.com/python/os_file_methods.htm)

# Python OS File/Directory Methods

Advertisements

**Robot Advisor Gestão de Carteiras DWS**  
 Invista desde 50€/mês  
 Serviço de Gestão de Carteiras prestado pela DWS. DWS Investimentos S.A. é uma entidade com sede no Luxemburgo, supervisionada pela CSSF e membro do Sistema de Indemnização.

## Python Directory and Files Management

In this article, you'll learn about file and directory management in Python, i.e. creating a directory, renaming it, listing all directories and working with them.

**LOOKING TO LEARN PROGRAMMING?**  
 Start your programming journey with Programiz **AT NO COST.**

## What is Directory in Python?

If there are a large number of files to handle in your Python program, you can arrange your code within different directories to make things more manageable.

A directory or folder is a collection of files and sub directories. Python has the `os` module, which provides us with many useful methods to work with directories (and files as well).

## Get Current Directory

We can get the present working directory using the `getcwd()` method.

This method returns the current working directory in the form of a string. We can also use the `getcwdb()` method to get it as bytes object.

```
>>> import os
>>> os.getcwd()
'C:\\Program Files\\PyScripter'
5.
>>> os.getcwdb()
7. b'C:\\Program Files\\PyScripter'
```

### Contents

- What is Directory in Python?
- Get Current Directory
- Changing Directory
- List Directories and Files
- Making a New Directory
- Renaming a Directory or a File
- Removing Directory or File



# The END