Using external files in Python

CS101 lectures : part 5.1

Professor Lilian Besson

Mahindra École Centrale (School of Computer Science)

April 7th, 2015

Please contact me by email if needed: CS101@crans.org

Slides and examples will be uploaded on **Moodle**.

Overview of the content of this lecture

1. Presentation
2. Using files in Python (1 lecture)
   - Opening a file in Python : different modes
   - Reading and saving information to a file
   - Name of a file, closing a file (and other details)
   - Automatically closing a file thanks to the "with" statement
   - The JSON data format
   - Using JSON files in Python
3. Conclusion of this lecture

Stay focus, listen and take notes

If you talk or disturb my lectures, I will kick you out *immediately*. OK ?

Overview of the content of this lecture

1. Presentation
2. Using files in Python (1 lecture)
   - Opening a file in Python : different modes
   - Reading and saving information to a file
   - Name of a file, closing a file (and other details)
   - Automatically closing a file thanks to the "with" statement
   - The JSON data format
   - Using JSON files in Python
3. Conclusion of this lecture

Stay focus, listen and take notes

If you talk or disturb my lectures, I will kick you out *immediately*. OK ?

Do we really need to learn about files . . . ?

From the RAM to the hard-drive

"After using the RAM from January, now we use the hard-drive with Python!"

Why should we care about files ?

Files are extremely important for any kind of programming. Python provides an *easy way* to use files : a concise and meaningful syntax.

> *Remark:* *reading or saving to an external file will be useful for at least 4 of the 6 programming projects (FFT, Edge, Integrals to save an image or a plot, and Time Table to save the result).*

What kind of files ?

- **Today** we see how to **read** from and **write** to **a text file** or a JSON file.
- Next lectures, we will see how to *save a plot* (`savefig()` from the `matplotlib.pyplot` module), how to *read or save an image* (`imread()`, `imsave()` from the `matplotlib.image` module).

Do we really need to learn about files . . . ?

From the RAM to the hard-drive

"After using the RAM from January, now we use the hard-drive with Python!"

Why should we care about files ?

Files are extremely important for any kind of programming. Python provides an *easy way* to use files : a concise and meaningful syntax.

> *Remark:* *reading or saving to an external file will be useful for at least 4 of the 6 programming projects (FFT, Edge, Integrals to save an image or a plot, and Time Table to save the result).*

What kind of files ?

- **Today** we see how to **read** from and **write** to **a text file** or a JSON file.
- Next lectures, we will see how to *save a plot* (`savefig()` from the `matplotlib.pyplot` module), how to *read or save an image* (`imread()`, `imsave()` from the `matplotlib.image` module).

Do we really need to learn about files . . . ?

From the RAM to the hard-drive

"After using the RAM from January, now we use the hard-drive with Python!"

Why should we care about files ?

Files are extremely important for any kind of programming. Python provides an *easy way* to use files : a concise and meaningful syntax.

> *Remark:* *reading or saving to an external file will be useful for at least 4 of the 6 programming projects (FFT, Edge, Integrals to save an image or a plot, and Time Table to save the result).*

What kind of files ?

- **Today** we see how to **read** from and **write** to **a text file** or a JSON file.
- Next lectures, we will see how to *save a plot* (`savefig()` from the `matplotlib.pyplot` module), how to *read or save an image* (`imread()`, `imsave()` from the `matplotlib.image` module).

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: (*new!*)

- `r` is
- `w` is
- `a` is ,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is ??
- `w` is
- `a` is ,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.

  Remark: The file has to exists!.
- `w` is
- `a` is ,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.

  Remark: The file has to exists!.
- `w` is ??
- `a` is ,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.
  Remark: The file has to exists!.
- `w` is for *write* mode: content is written onto the file.
  (**Warning:** will erase the file!),
  Remark: The file will be created if it does not exist.
- `a` is ,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.
  Remark: The file has to exists!.
- `w` is for *write* mode: content is written onto the file.
  (**Warning:** will erase the file!),
  Remark: The file will be created if it does not exist.
- `a` is ??,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.
  Remark: The file has to exists!.
- `w` is for *write* mode: content is written onto the file.
  (**Warning:** will erase the file!),
  Remark: The file will be created if it does not exist.
- `a` is for *append* mode: content is written at the end of the file,
- `b` is .

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.
  Remark: The file has to exists!.
- `w` is for *write* mode: content is written onto the file.
  (**Warning:** will erase the file!),
  Remark: The file will be created if it does not exist.
- `a` is for *append* mode: content is written at the end of the file,
- `b` is ??.

How to open a file in Python

When we have a file on our hard-drive, we can **open it with Python**.
The syntax for opening a file is to simply use the `open` function:

```
myfile = open("nameOfTheFile.txt", mode).
```

4 different modes can be used: *(new!)*

- `r` is for *read* mode: file is only readable.
  Remark: The file has to exists!.
- `w` is for *write* mode: content is written onto the file.
  (**Warning:** will erase the file!),
  Remark: The file will be created if it does not exist.
- `a` is for *append* mode: content is written at the end of the file,
- `b` is for *binary* mode: file is assumed as binary *(more complicated)*.

Different modes to open a file in Python

4 different modes can be used `r`, `w`, `a`, or `b`

Modes can be combined together:

- `'rw'` for reading and writing,
- `'ra'` for reading and appending,
- `'wb'` for writing to a binary file, etc

(more details on the Python documentation).

Using the `open` function can fail (and *raise an exception*)

The `open` will fail if the file is absent, or not readable:

- IOError: [Errno 2] No such file or directory: 'blabla.ext'
- IOError: [Errno 13] Permission denied: '/bin/sh'
- ... other exceptions are possible (some are platform dependant).

Different modes to open a file in Python

4 different modes can be used `r`, `w`, `a`, or `b`

Modes can be combined together:

- `'rw'` for reading and writing,
- `'ra'` for reading and appending,
- `'wb'` for writing to a binary file, etc

(more details on the Python documentation).

Using the `open` function can fail (and *raise an exception*)

The `open` will fail if the file is absent, or not readable:

- IOError: [Errno 2] No such file or directory: 'blabla.ext'
- IOError: [Errno 13] Permission denied: '/bin/sh'
- ... other exceptions are possible (some are platform dependant).

How to read from file in Python

Reading "manually" from a file : basic concept

It depends of the nature of the file:

- reading from a text file will be **line by line**, a line is a string.
- reading from a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To read the content of a text file, two approaches are possible:

- `list(myfile)` will return a list of strings, *read line by line.* Each string will contain the new-line character at its end (`'\n'`).
- `for line in myfile: ...` : yes, *we can loop over a file*, with a syntax as easy as this. Inside the body of the **for loop**, we can do whatever we want with this string `line`.

How to read from file in Python

Reading "manually" from a file : basic concept

It depends of the nature of the file:

- reading from a text file will be **line by line**, a line is a string.
- reading from a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To read the content of a text file, two approaches are possible:

- `list(myfile)` will return a list of strings, *read line by line.* Each string will contain the new-line character at its end (`'\n'`).
- `for line in myfile: ...` : yes, *we can loop over a file*, with a syntax as easy as this. Inside the body of the **for loop**, we can do whatever we want with this string `line`.

How to read from file in Python

Reading "manually" from a file : basic concept

It depends of the nature of the file:

- reading from a text file will be **line by line**, a line is a string.
- reading from a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To read the content of a text file, two approaches are possible:

- `list(myfile)` will return a list of strings, *read line by line.* Each string will contain the new-line character at its end (`'\n'`).
- `for line in myfile: ...` : yes, *we can loop over a file*, with a syntax as easy as this. Inside the body of the **for loop**, we can do whatever we want with this string `line`.

How to read from file in Python

Reading "manually" from a file : basic concept

It depends of the nature of the file:

- reading from a text file will be **line by line**, a line is a string.
- reading from a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To read the content of a text file, two approaches are possible:

- `list(myfile)` will return a list of strings, *read line by line.* Each string will contain the new-line character at its end (`'\n'`).
- `for line in myfile: ...` : yes, *we can loop over a file*, with a syntax as easy as this. Inside the body of the **for loop**, we can do whatever we want with this string `line`.

How to write to file in Python

Writing "manually" to a file : basic concept

It depends of the nature of the file:

- writing to a text file: we can write strings (and only strings).
- writing to a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To write some content of a text file, the content has to be a string, and then two approaches are possible:

- `myfile.write(a_string)` will write the string `a_string`.
  Warning: `myfile.flush()` or `myfile.close()` may be needed before the file on disk reflects the data written in Python.
- `myfile.writelines(seq_strings)` will write each string of `seq_strings`.
  Note that new-line symbols (`'\n'`) are **not** added.

How to write to file in Python

Writing "manually" to a file : basic concept

It depends of the nature of the file:

- writing to a text file: we can write strings (and only strings).
- writing to a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To write some content of a text file, the content has to be a string, and then two approaches are possible:

- `myfile.write(a_string)` will write the string `a_string`.
  Warning: `myfile.flush()` or `myfile.close()` may be needed before the file on disk reflects the data written in Python.
- `myfile.writelines(seq_strings)` will write each string of `seq_strings`.
  Note that new-line symbols (`'\n'`) are **not** added.

How to write to file in Python

Writing "manually" to a file : basic concept

It depends of the nature of the file:

- writing to a text file: we can write strings (and only strings).
- writing to a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To write some content of a text file, the content has to be a string, and then two approaches are possible:

- `myfile.write(a_string)` will write the string `a_string`.
  Warning: `myfile.flush()` or `myfile.close()` may be needed before the file on disk reflects the data written in Python.
- `myfile.writelines(seq_strings)` will write each string of `seq_strings`.
  Note that new-line symbols (`'\n'`) are **not** added.

How to write to file in Python

Writing "manually" to a file : basic concept

It depends of the nature of the file:

- writing to a text file: we can write strings (and only strings).
- writing to a binary file *is more complicated. There is no general case.* Next lectures will show some examples (binary data and images).

The easy way: list of lines, or a for loop

To write some content of a text file, the content has to be a string, and then two approaches are possible:

- `myfile.write(a_string)` will write the string `a_string`.
  Warning: `myfile.flush()` or `myfile.close()` may be needed before the file on disk reflects the data written in Python.
- `myfile.writelines(seq_strings)` will write each string of `seq_strings`.
  Note that new-line symbols (`'\n'`) are **not** added.

A file object has other methods, and some attributes

Now that you now OOP...

`open()` returns an *object of type* `file` (ie. an instance of the class `file`).

Other methods and three attributes for the `file` class

With a file object `myfile`, you can use:

- `myfile.name` to get the (relative) *path of the file*.
- `myfile.mode` is the mode with which you opened the file (`'r'`, `'w'` etc).
- `myfile.close()` to ask Python to close the file. (But there is a nice syntax trick that allows us to never have close a file ourself).
- `myfile.closed` to know if the file *is closed or not* (not really useful).
- `myfile.flush()` will flush the internal I/O buffer (updates the file by performing all the pending I/O operations).

A file object has other methods, and some attributes

Now that you now OOP...

`open()` returns an *object of type* `file` (ie. an instance of the class `file`).

Other methods and three attributes for the `file` class

With a file object `myfile`, you can use:

- `myfile.name` to get the (relative) *path of the file*.
- `myfile.mode` is the mode with which you opened the file (`'r'`, `'w'` etc).
- `myfile.close()` to ask Python to close the file. (But there is a nice syntax trick that allows us to never have close a file ourself).
- `myfile.closed` to know if the file *is closed or not* (not really useful).
- `myfile.flush()` will flush the internal I/O buffer (updates the file by performing all the pending I/O operations).

A file object has other methods, and some attributes

Now that you now OOP...

`open()` returns an *object of type* `file` (ie. an instance of the class `file`).

Other methods and three attributes for the `file` class

With a file object `myfile`, you can use:

- `myfile.name` to get the (relative) *path of the file*.
- `myfile.mode` is the mode with which you opened the file (`'r'`, `'w'` etc).
- `myfile.close()` to ask Python to close the file. (But there is a nice syntax trick that allows us to never have close a file ourself).
- `myfile.closed` to know if the file *is closed or not* (not really useful).
- `myfile.flush()` will flush the internal I/O buffer (updates the file by performing all the pending I/O operations).

A file object has other methods, and some attributes

Now that you now OOP...

`open()` returns an *object of type* `file` (ie. an instance of the class `file`).

Other methods and three attributes for the `file` class

With a file object `myfile`, you can use:

- `myfile.name` to get the (relative) *path of the file*.
- `myfile.mode` is the mode with which you opened the file (`'r'`, `'w'` etc).
- `myfile.close()` to ask Python to close the file. (But there is a nice syntax trick that allows us to never have close a file ourself).
- `myfile.closed` to know if the file *is closed or not* (not really useful).
- `myfile.flush()` will flush the internal I/O buffer (updates the file by performing all the pending I/O operations).

Using the `with` block statement

Concept and syntax? (*new!*)

The syntax is like this:

```
with open("file.ext", mode) as myfile:
    # inside this indented block
    # we can use the file with the variable myfile.
# After the indented block, the file has been closed
```

Inside that block, the file is opened and available, and after the block, Python will close the file *automatically* (and delete the variable).

Let us count how many times the word "file" is present in these slides?

```
with open("slides.tex", 'r') as f:
    total = sum( l.count("file") for l in f )
    print total, "occurences of 'file' in", f.name
```

Using the `with` block statement

Concept and syntax? (*new!*)

The syntax is like this:

```
with open("file.ext", mode) as myfile:
    # inside this indented block
    # we can use the file with the variable myfile.
# After the indented block, the file has been closed
```

Inside that block, the file is opened and available, and after the block, Python will close the file *automatically* (and delete the variable).

Let us count how many times the word "file" is present in these slides?

```
with open("slides.tex", 'r') as f:
    total = sum( l.count("file") for l in f )
    print total, "occurences of 'file' in", f.name
```

The JSON data interchange standard format (1/2)

About JSON (cf. http://json.org/)

*"**JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write, and easy for machines to parse and generate."*
"JSON is a text format that is completely language independent but uses conventions that are familiar to programmers used to C, C++, Java, JavaScript, Python, and many others."
"These properties make JSON an ideal data-interchange language."

If you are interested, please work on that part by yourself!

The JSON data interchange standard format (2/2)

A widely used standard...

JSON is now the standard format for exchanging data text-files on networks and Internet. It is used by the majority of websites and webservices that provide API. Example: `wakatime.com/api`.

2 structures in JSON (*new!*)

JSON is built on *two structures*:

- A collection of name/value pairs : like a Python **dictionary**,
- An ordered list of values : like a Python **list**.

The JSON data interchange standard format (2/2)

A widely used standard...

JSON is now the standard format for exchanging data text-files on networks and Internet. It is used by the majority of websites and webservices that provide API. Example: `wakatime.com/api`.

2 structures in JSON *(new!)*

JSON is built on *two structures*:

- A collection of name/value pairs : like a Python **dictionary**,
- An ordered list of values : like a Python **list**.

How to (and why) use JSON files in Python?

In Python? The `json` module from the standard library *(new!)*

One example, coming from the student database browser I showed previously:

```
import json

with open("students.json", 'r') as f:
    # f is a text file, opened in 'read'-only mode
    students = json.load(f)
    # we load the database of student
    # as a Python dictionary (students is a dict)

# This simply prints the database
print json.dumps()

# But this is more readable
print json.dumps(students, sort_keys=True, indent=4,↩
    separators=(',', ': '))
```

How to (and why) use JSON files in Python?

4 functions in that module `json` (*new!*)

We can write a Python object to a string/file, or read an object from a string/file:

⟵ `json.load` (or `json.loads`) de-serialize from a file (or a string) to a Python object,

⟶ `json.dump` (or `json.dumps`) serialize a Python object to a file (or a string),

See the Python documentation for more details, and more examples: docs.python.org/2/library/json.html.

Two other Python modules to save and load data to files

- The `pickle` module: save data in a text-file format.
  Advantage: easily readable or editable by a human.
  Drawback: not all Python objects are compatible.
- The `marshal` module: save data in binary format (Python specific format), for efficient reading and writing.

How to (and why) use JSON files in Python?

4 functions in that module `json` (*new!*)

We can write a Python object to a string/file, or read an object from a string/file:

⟵ `json.load` (or `json.loads`) de-serialize from a file (or a string) to a Python object,

⟶ `json.dump` (or `json.dumps`) serialize a Python object to a file (or a string),

See the Python documentation for more details, and more examples: docs.python.org/2/library/json.html.

Two other Python modules to save and load data to files

- The `pickle` module: save data in a text-file format.
  Advantage: easily readable or editable by a human.
  Drawback: not all Python objects are compatible.
- The `marshal` module: save data in binary format (Python specific format), for efficient reading and writing.

Quick sum-up about **files** in Python

About files, **we just saw**: *new!*

- Why should we use files in a (Python) program?
- How to open files in Python (whatever if they are text, JSON, image or binary),
- Several ways to read data from a file, and write Python values to a file,
- How to close a file, or open it nicely with a "`with` block",
- Using the `json` module to manipulate JSON files (Work on that by yourself!).

More could have been studied...

- With more time, we could have presented some binary formats, or explain how to download files from or send files to the Internet, etc.
- *Files encoding is another issue*, that you might have to work on by yourself if it arises during your project *(please contact us if needed)*.

Quick sum-up about **files** in Python

About files, **we just saw**: *new!*

- Why should we use files in a (Python) program?
- How to open files in Python (whatever if they are text, JSON, image or binary),
- Several ways to read data from a file, and write Python values to a file,
- How to close a file, or open it nicely with a "`with` block",
- Using the `json` module to manipulate JSON files (Work on that by yourself!).

More could have been studied...

- With more time, we could have presented some binary formats, or explain how to download files from or send files to the Internet, etc.
- *Files encoding is another issue*, that you might have to work on by yourself if it arises during your project *(please contact us if needed)*.

Thanks for listening to this lecture about files

Any question?

Reference websites

- www.MahindraEcoleCentrale.edu.in/portal : **MEC Moodle**,
- the IntroToPython.org website,
- and the Python documentation at docs.python.org/2/,

Want to know more?

↪ practice by yourself with these websites !

↪ contact us (e-mail, *flying pigeons*, Moodle etc) if needed,

↪ or consult the *"Python in Easy Steps"* book on Python.

Next 3 lectures: **scientific computations and plotting with Python** (by me again)