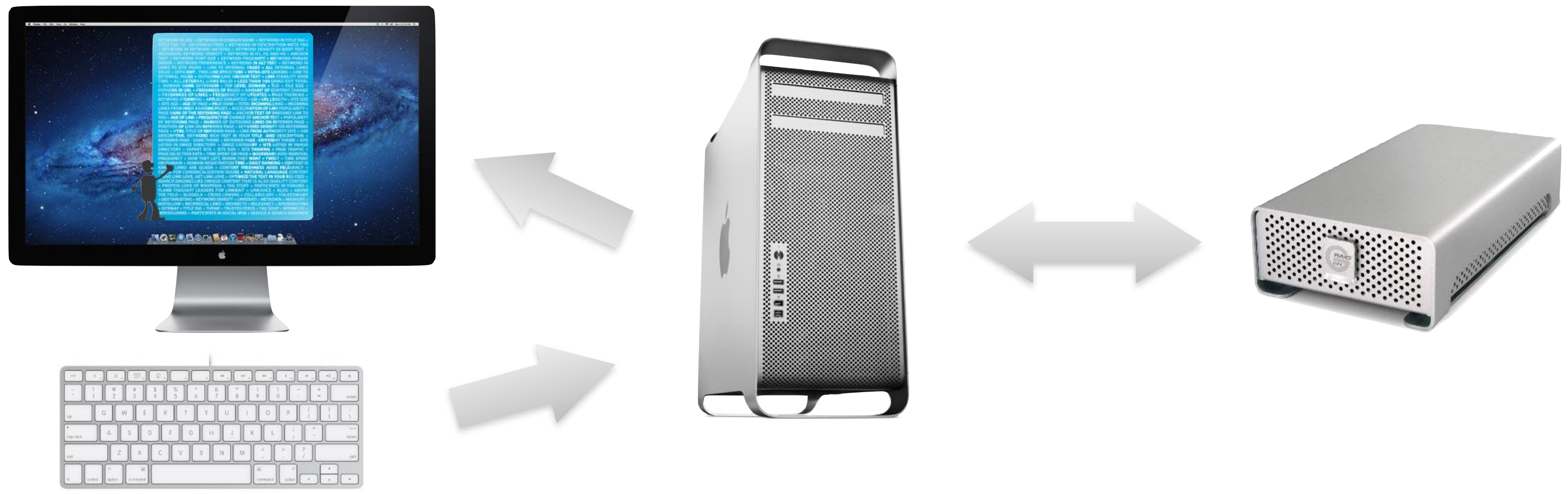
Table of contents

Introduction

- Up to now, our programs store their data in memory
  - We need non-volatile storage to permanently keep the information and retrieve it later

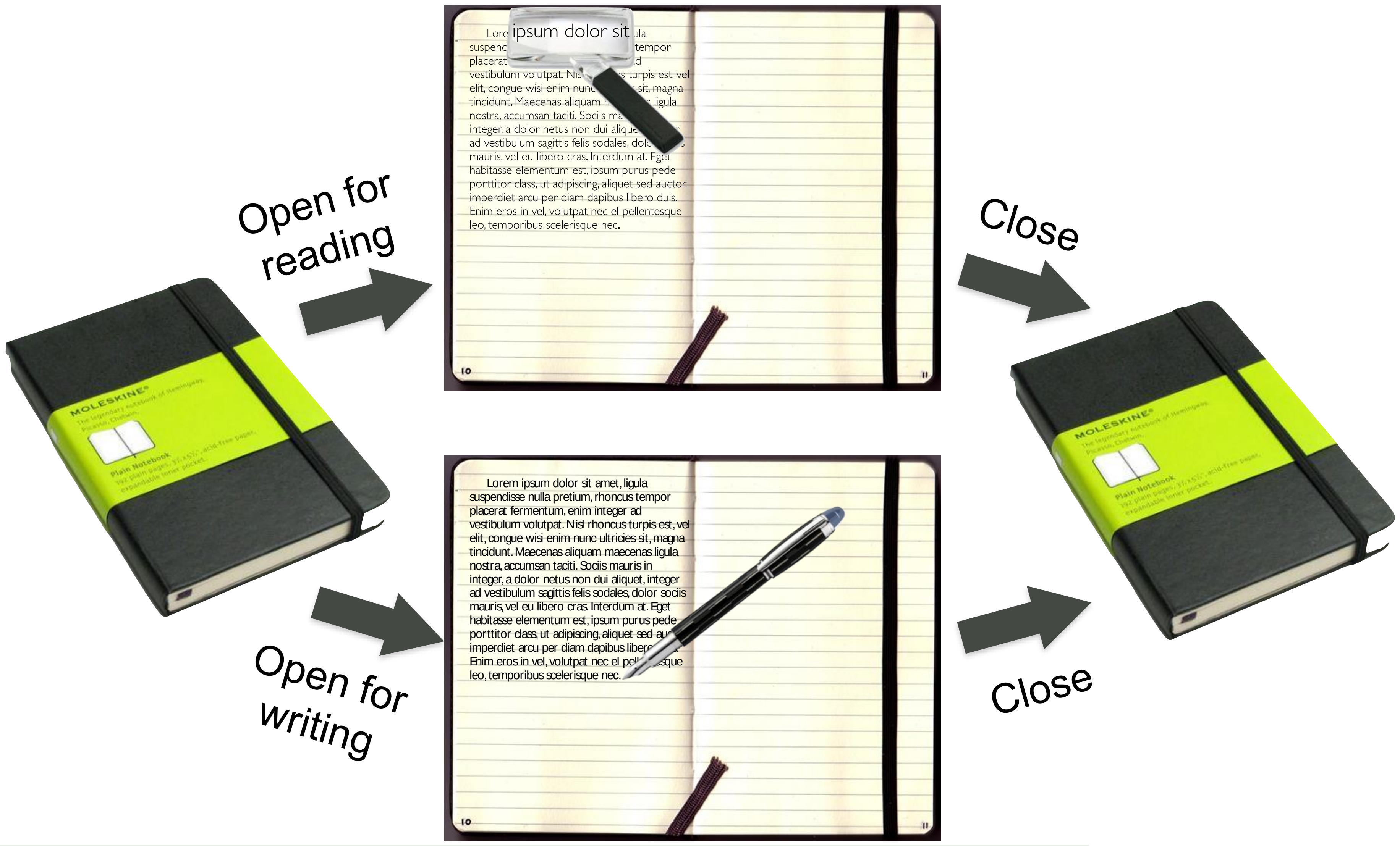


- **A file is a byte sequence that is kept in a permanent storage device**

Using files

- In Python, a file is categorized as either a text or binary, and the difference between the wo file types is important.
  - **Text files** are structured as a sequence of lines, where each line includes a sequence of characters.
  - Each line is determined with a special character, called the EOL or End of Line character. There are several types, but the most common is the new line character (`\n`).

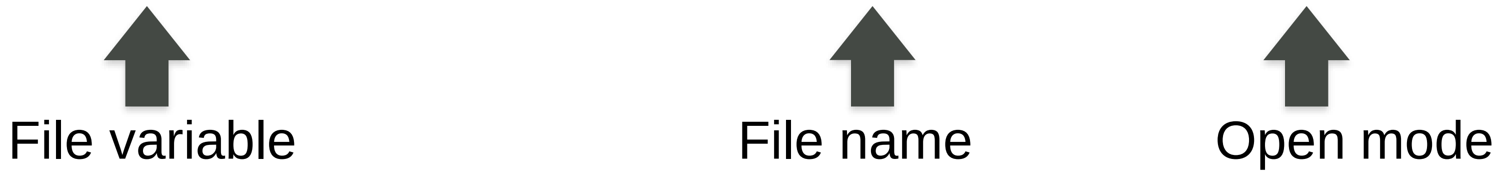
Using files



Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet, integer ad vestibulum sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed au imperdiet arcu per diam dapibus libero Enim eros in vel, volutpat nec el pell esque leo, temporibus scelerisque nec.

Opening a file

```
f = open("myfile.txt", "r")
```



- **Open modes:** read, write, append.
  - **"r" → Open for reading (default value)**. If the file does not exist, an error is launched.
  - **"w" → Open for writing**. If the file does not exist, a new one is created. In any other case, its content is overwritten.
  - **"a" → Open for appending**. If the file does not exist, a new one is created. In any other case, it is opened with write permissions and new information is appended at the end.

Reading a file

- Most programming languages offer several ways to get information from a file.
- If you need to extract a string that contains all (or a number of) characters in the file, you can use the following method:

Characters read

```
c = f.read()
```

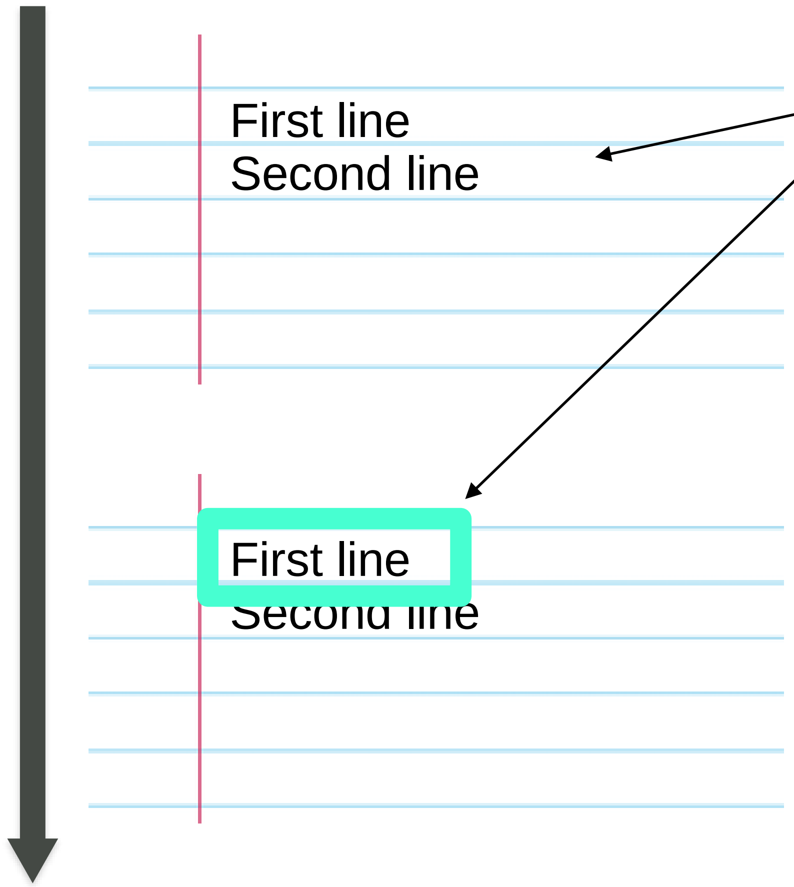
File variable

- The file is read in a sequential way
- It returns the whole file (or the number of characters specified as parameter)
- Next call will return next characters

Reading a file

- `f.readline()` will read from a file line by line (rather than pulling the entire file in at once). Basically, it will read a single line from the file and return a string containing characters up to `\n`. Subsequent calls to `readline()` will return successive lines.
- `f.readlines()` returns the complete the complete file as a list of strings each separated by `\n`.

Reading a file: example



```
>>> f = open("myfile.txt", "r")
>>> line = f.readline()
>>> print(line)
First line

>>> f.close()
```

IMPORTANT: The carriage return is also read and it is stored in the `line` variable, which is also displayed by `print`.

Writing a file

- Most programming languages offer one (or more) functions to add information from a file.

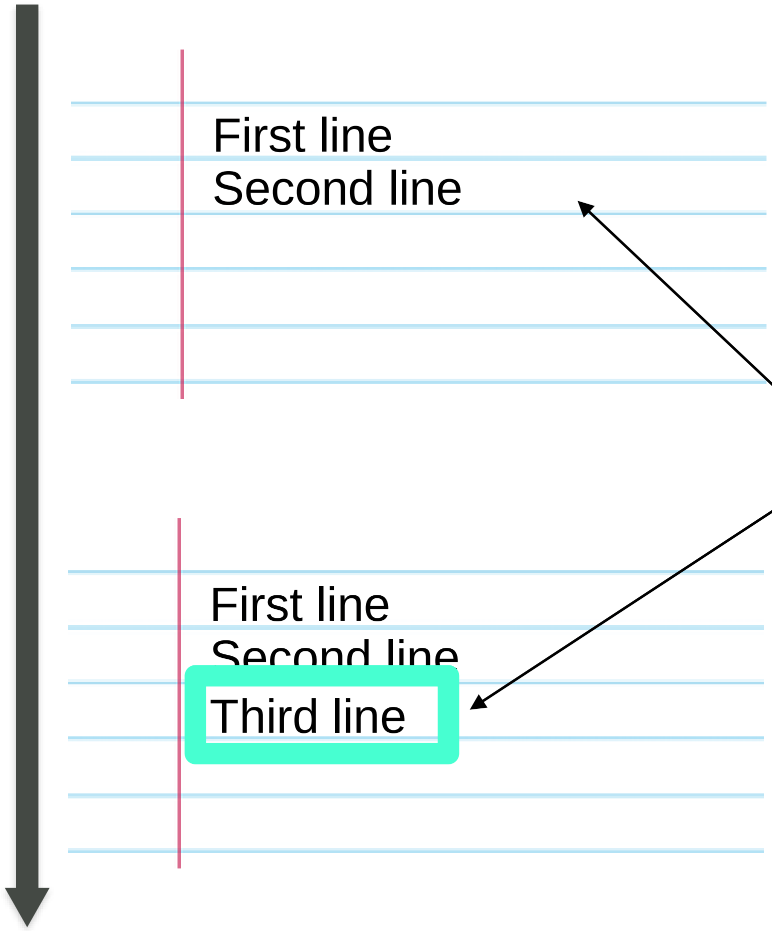
Characters to be added to the file

```
f.write(characters)
```

File variable

- Open mode **w**: Writes a string (character-oriented file) in the actual position of a file.
- Open mode **a**: Points to the EOF and add the characters there.

Writing a file (appending): example



```
>>> f = open("myfile.txt", "a")
>>> f.write("Third line\n")
>>> f.close()
```

We have to explicitly provide the EOL character in order to start a new line.

Closing a file

- The file cannot be accessed anymore unless it is opened again.

```
f.close()
```

↑

File variable

Example

- Python can read a whole file with just one instruction:

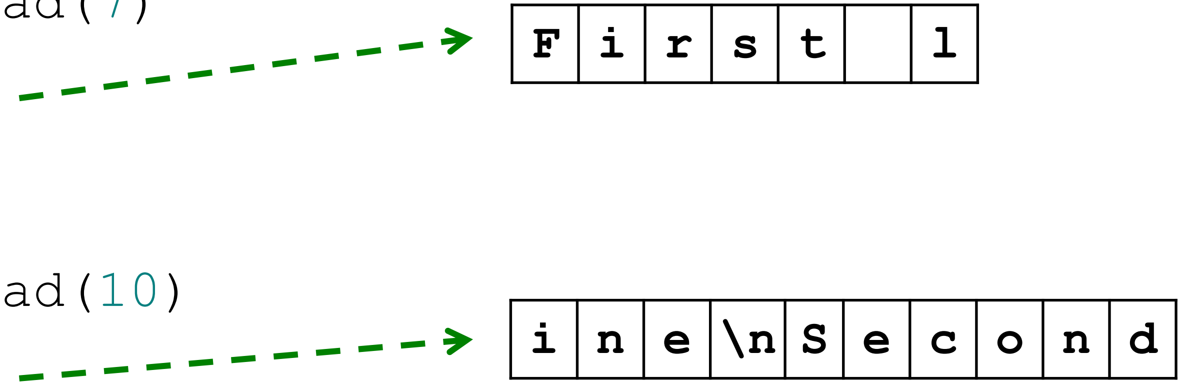
```
>>> f = open("myfile.txt", "r")
>>> all = f.read()
>>> print(all)
First line
Second line

>>> f.close()
```

F	i	r	s	t		l	i	n	e	\n	S	e	c	o	n	d		l	i	n	e	\n

Example

- Instead of reading full lines, we can also indicate the number of bytes we want to read.

```
>>> f = open("myfile.txt", "r")
>>> letters = f.read(7)
>>> print(letters)
First l
>>> letters = f.read(10)
>>> print(letters)
ine
Second
>>> f.close()
```



Examples

- A for loop to read a file line by line:

```
f = open("myfile.txt","r")
lines = f.readlines()
f.close()
for l in lines:
    print(l)
```

```
f = open("myfile.txt","r")
for l in f:
    print(l)
f.close()
```

- We can also read the file until we find the EOF:

```
f = open("myfile.txt","r")
line = f.readline()
while line != "":
    print(line)
    line = f.readline()
f.close()
```

```
[Output]
First line

Second line
```