Lesson 6:
Files

Files

- Computers use files to store the data that has been processed in each code executions.
- Python provides many facilities for creating and accessing files.
- Each operating system (e.g., Windows, macOS…) has its own system for creating and accessing files.
- Python achieves operating system independence by accessing files through a **file handle** (a file identifier).

Open

- We must open a file before performing any action:

Parameters

```
nameHandle1 = open("File1.txt","w")
```

Name of the File Handle

Instruction to open

Opening mode

- The instruction `open` asks the operating system to create a file named `File1.txt`.
- The function returns a file handle (file identifier) for this file. The argument `"w"` indicates that the file will be opened for writing.

Open: opening modes

- There are different options for opening, depending on the purpose. The second parameter indicates the purpose: read (`r` or `r+`), write (`w, x, w+` or `x+`) or append (`a` ó `a+`).

```
nameHandle = open("File1.txt","<indicator>")
```

Indicator		Opening Mode		Pointer position
r	r+	Only reading.	and writing	At the beginning
w	w+	Only writing. If the file exists, it will overwrite it. If the file does not exist, it will create one.	and reading	At the beginning
x	x+	Only writing. Error `FileExistsError` if the file exists. If the file does not exist, it will create one.	and reading	At the beginning
a	a+	Add / append / concatenate content. If the file does not exist, it will create one.	and reading and writing	At the end

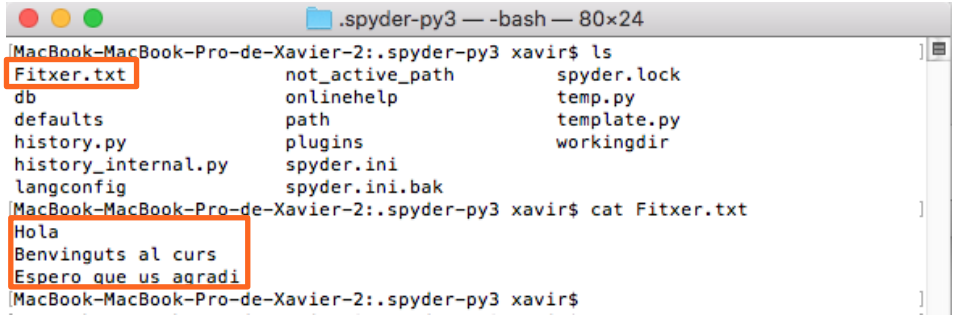
Write and Close

- The handle can be seen as an “object” (variable) that has associated funcions to manage the files.
- An associated function to the file handle is writing (`write`)

```
nameHandle = open("Fitxer.txt","w")
nameHandle.write("Hola\nBenvinguts al curs\n")
nameHandle.write("Espero que us agradi\n")
nameHandle.close()
```

In a string, the symbol "\" is a special character (escape sequence) that indicates that the next character must be treated in a special way. Here, **`\n`** indicates **new line**.

- The last instruction is for closing the file (`close`). Once closed, other programs will be able to access to their contents.
  - IMPORTANT. If we do not close the file, and the execution fails for whatever reason, the information in the file can be corrupted.



Read

- The instruction `read` is used to read a file. Here the argument of the function `open` is `r`:

```
nameHandle = open("Fitxer.txt","r")
print(nameHandle.read())
nameHandle.close()
```

```
In [7]: runfile('/Users/xa
Hola
Benvinguts al curs
Espero que us agradi
```

- Since Python treats files as a sequence of lines, we can use the instruction `for` to iterate over the file contents:

```
nameHandle = open("Fitxer.txt","r")
for tline in nameHandle:
    print(tline)
nameHandle.close()
```

```
In [9]: runfile('/Users/x
Hola

Benvinguts al curs

Espero que us agradi
```

- `tline` is a string, we can avoid `\n` by doing `tline[:-1]`

```
nameHandle = open("Fitxer.txt","r")
for tline in nameHandle:
    print(tline[:-1])
nameHandle.close()
```

```
In [10]: runfile('/Users/xa
Hola
Benvinguts al curs
Espero que us agradi
```

Readline

- The instruction `readline` allows to read one single line (until it finds the newline symbol `\n`)

```
nameHandle = open("Fitxer.txt","r")
print(nameHandle.readline())
nameHandle.close()
```

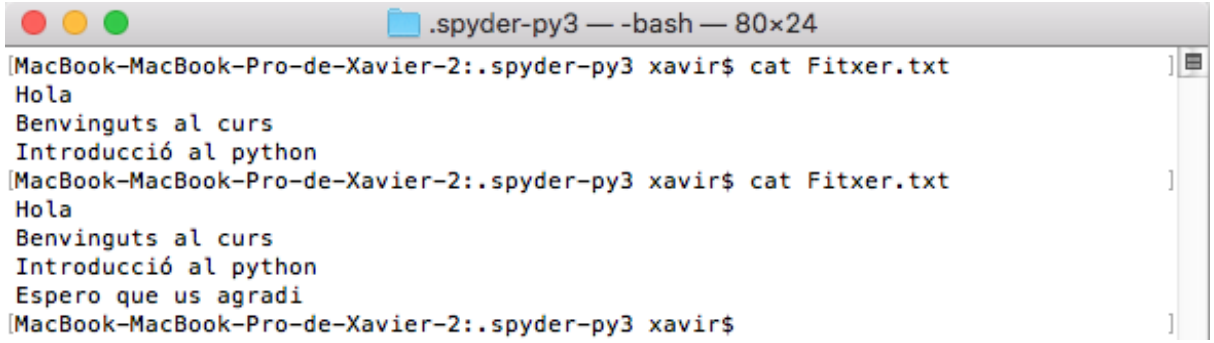
```
In [89]: runfile('/L
Hola
```

Append

- Whenever we call `open` with the parameter `w`, the file content is **lost** (overwritten).
- If we want to avoid this, we must use the parameter `a` (append).
- The new content will be added at the end of the file:

```
nameHandle = open("Fitxer.txt","w")
nameHandle.write("Hola\nBenvinguts al curs\n")
nameHandle.write("Introducció al python\n")
nameHandle.close()

nameHandle = open("Fitxer.txt","a")
nameHandle.write("Espero que us agradi\n")
nameHandle.close()
```



Tell and Seek

- Files are stored in a sequential manner. Whenever we write, we do it at the end of the file.
- To move the position of the pointer ("cursor") in the file, we can use the instruction `seek()` indicating the desired position.

```
""" Fitxer.txt
Linia1:1234567890\n     ← 18 characters (including return)
Linia2:1234567890\n        Note: \n is 1 character
Linia3:1234567890\n
"""
nameHandle = open("Fitxer.txt","r")
nameHandle.seek(19)
print(nameHandle.read(6))
nameHandle.close()
```

```
In [42]: runfile('
Linia2
```

- To know the position of the pointer in the file, we can use the instruction `tell()`

```
...
print(nameHandle.read(6))
print(nameHandle.tell())
nameHandle.close()
```

```
In [45]: runfile(
Linia2
25
```

with

- Since version 2.5, Python incorporates and "elegant" way to work with files that allows to work and automatically close them, without the need of calling `close` ().
- For this purpose, we add the instructions inside a block `with`:

```
""" Fitxer.txt
Linia1:1234567890\n
Linia2:1234567890\n
Linia3:1234567890\n
"""
with open("Fitxer.txt", "r") as nameHandle:
    cont = nameHandle.read()

print(cont)
```

```
In [94]: runfile('/Users/xa
Linia1:1234567890
Linia2:1234567890
Linia3:1234567890
```