Files

Files

file: logical entity consisting of a sequence of bytes, belonging to a system file in a secondary memory of a computer

File access types (the more basic):
- Sequential: traversal always from the beginning to the end
- Direct: direct access to specific positions

File contents types:
text file: textual data (bytes corresponding to printable characters)
binary file: binary codified data

STF: Sequential Text File
specific file type studied in this course

File structure

Generally, a file is structured in **lines** (**registers**):

- Fixed or variable length
- Ending with character line feed: ‘`\n`’
- Can be structured in several data fields

file with several lines of variable length, without structure

```
When forty winters shall besiege thy brow,
And dig deep trenches in thy beauty's field,
Thy youth's proud livery so gazed on now,
Will be a totter'd weed of small worth held:
Then being asked, where all thy beauty lies,
Where all the treasure of thy lusty days;
To say, within thine own deep sunken eyes,
Were an all-eating shame, and thriftless praise.
How much more praise deserv'd thy beauty's use,
If thou couldst answer 'This fair child of mine
Shall sum my count, and make my old excuse,'
Proving his beauty by succession thine!
This were to be new made when thou art old,
And see thy blood warm when thou feel'st it cold.

Sonet II. William Shakespeare
```

File structure

Generally, a file is structured in **lines** (**registers**):

- Fixed or variable length
- Ending with character line feed: '`\n`'
- Can be structured in several data fields

- file with several lines of variable length,
- structured in 4 data fields, separated by a comma and a space
- Includes two header lines with different structure

```
NIF,       Name,                  Time, I/O
----------------------------------------
11223344F, Ramon Pladevall Homs, 8:00, I
44337799P, Santiago Forns Cheng, 8:01, I
77722212M, Corneli Perez Simon, 8:33, I
11122233V, Anna Pi Fort, 9:30, I
44337799P, Santiago Forns Cheng, 10:31, O
```

STF and Python

file type (`io.TextIOWraper`). Type for managing files

external file: name (string) that designs the file in the file system (usual name)
internal file: name of the type `io.TextIOWraper` variable that designs the file within a Python function
channel (link): communication protocol between the external and internal files
Syntax to stablish this protocol:

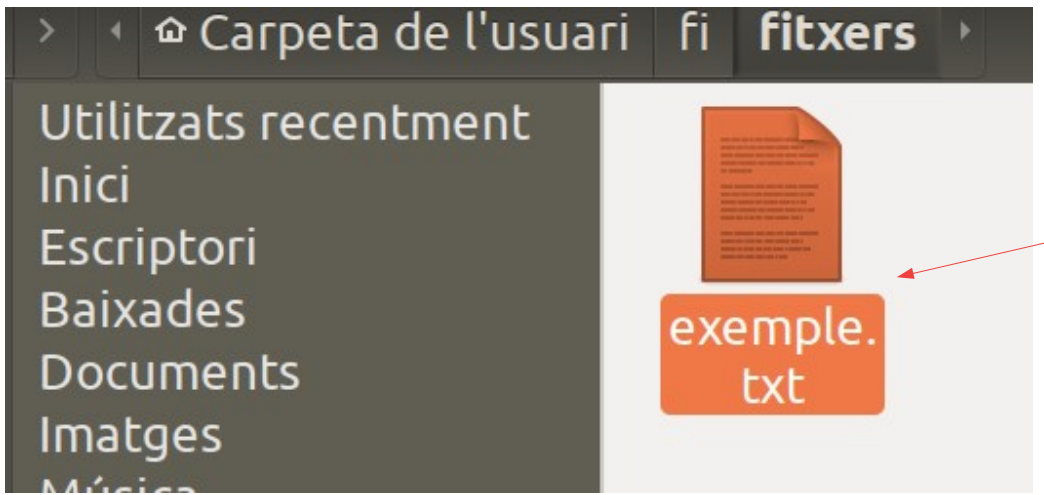
Reserved word (keyword)
Function of the builtins library
External file name (string)
Mode: '`r`' (read) or '`w`' (write)
Another reserved word
Internal file name (type `io.TextIOWraper` variable)
Referred to as type *file* from now on

```
with open(external_file_name, mode) as internal_file_name:
```

STF and Python

Mode:

- '`r`' (read). file must exist
- '`w`' (write). file is created. If there exists another file with this name, its contents is lost




```
with open('exemple.txt', 'r') as f:
```

external file

internal file

file extension
optional
txt: text file

Type file methods

Reading methods:

`f.read(size)`
reads a maximum of ‘size' characters from file f; returns a string

`f.readline(size)`
reads the following line of file f; returns a string

`f.readlines()`
reads the whole file f; returns a list of strings each corresponding to a line

Writing method:

`f.write(str)`
writes str (string) to file f; returns an integer: the str length

File traversal

```
for line in f:
    process line
```

`line:` each line of file f

Example:

file example.txt

must be in the working directory

file example.txt

```
NIF,       Name,                  Time, I/O
----------------------------------------
11223344F, Ramon Pladevall Homs, 8:00, I
44337799P, Santiago Forns Cheng, 8:01, I
77722212M, Corneli Perez Simon, 8:33, I
11122233V, Anna Pi Fort, 9:30, I
44337799P, Santiago Forns Cheng, 10:31, O
```

```
with open('example.txt', 'r') as f:
    f.readline()                # read 1st line but don't process it
    f.readline()                # read 2nd line but don't process it
    for line in f:              # read 3rd line and next lines until EOF (end of file)
       data = line.split(', ')  # split line into its 4 data fields
       print(data[0])           # show NIF
```

Processing a line

- Cleaning: remove extra special characters as space, line feed, …
  at the beginning or at the end: **`strip`** method **`(str)`**
- Splitting: split line into its data fields: **`split`** method
- Decoding: convert data fields to suitable types

file example.txt

```
NIF,       Name,                  Time, I/O
-----------------------------------------
11223344F, Ramon Pladevall Homs, 8:00, I
44337799P, Santiago Forns Cheng, 8:01, I
77722212M, Corneli Perez Simon, 8:33, I
11122233V, Anna Pi Fort, 9:30, I
44337799P, Santiago Forns Cheng, 10:31, O
```

Apply to lines from the 3rd of file example.txt:

- Remove line feed character
- Obtain 4 data fields
- Convert time to 2 integers

Processing a line

Example:

`line` is the 3rd line of file example.py: '`11223344F, Ramon Pladevall Homs, 8:00, E\n`'

```
>>> line = '11223344F, Ramon Pladevall Homs, 8:00, I\n'
>>> line = line.strip()
>>> line
'11223344F, Ramon Pladevall Homs, 8:00, I'
>>> data = line.split(', ')
>>> data
['11223344F', 'Ramon Pladevall Homs', '8:00', 'I']
>>> dni, name, time, IO = data
>>> dni, name, time, IO
('11223344F', 'Ramon Pladevall Homs', '8:00', 'I')
>>> hour, minute = time.split(':')
>>> hour, minute
('8', '00')
>>> hour, minute = int(hour), int(minute)
>>> hour, minute
(8, 0)
```

ending line feed chraracter removed

list with 4 data fields

unpacking

split and unpack `time`

convert to 2 integers