CMPT 120
Intro to CS & Programming I

WEEK 11 (Mar. 24-28)

— *Jérémie O. Lumbroso* —

Lecture 25:
File Input and Output

http://www.sfu.ca/~jlumbros/Courses/CMPT120/

Why File Input/Output?

- Up until now: terminal input/output
- It is useful to
  - store information
    - save progress of a computation
    - save result of an algorithm
    - save/log history of interaction
    - etc.
  - retrieve information
    - parameters of a computation
    - to resume a computation in progress
- There are various ways to store this information (databases, cloud, WebDAV, etc.), but files is one very classical way of doing it

Create a file, read it in Python, process its data

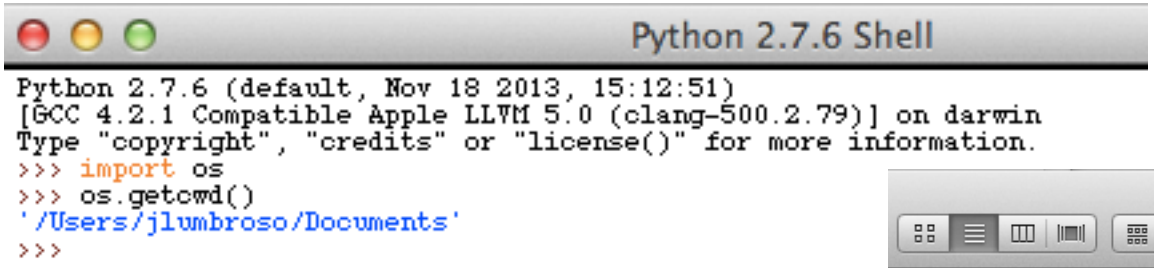
PART 1: READING A FILE

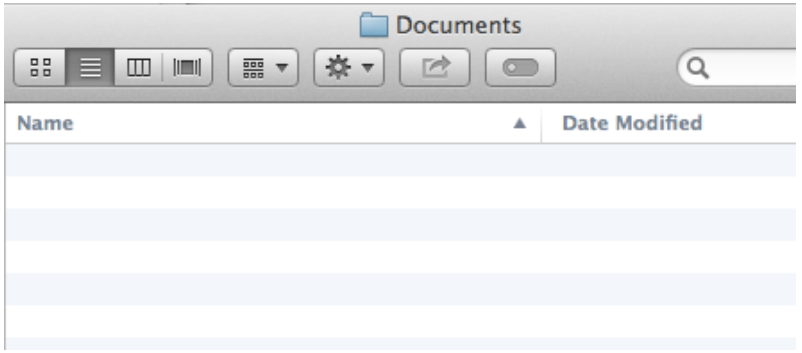
`os` module

- The `os` module (os for "operating system") provides convenient functions to interact with the operating system
- We will only use a few
  - `os.getcwd()` to get the **c**urrent **w**orking **d**irectory (the directory/folder in which Python is going to go looking for files that it has to open)
- For more information, as always, the doc: http://docs.python.org/2/library/os.html

Setting Up Our Environment

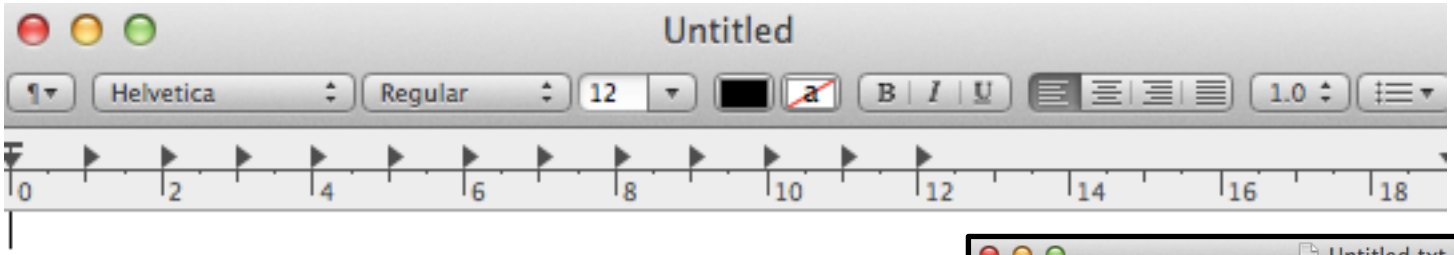
- Open IDLE
- Before anything, type `import os` (to let Python know you will be using the functions in the os module)
- Type `os.getcwd()` in the Python shell
- Open this folder in Explorer or Finder or whatever

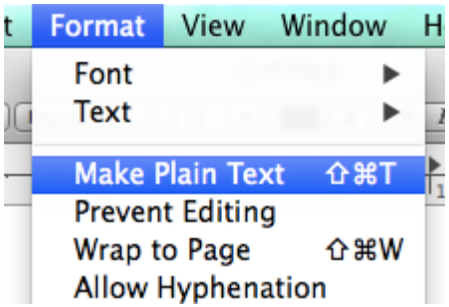


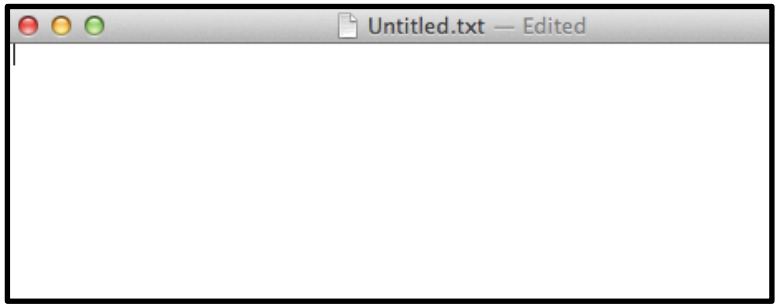


Manipulating a Plain Text File

- On Windows, use *Notepad* and save with .txt
- On Mac OS X, use *Text Edit* but be careful
  - by default, Text Edit will save in "Rich Text" with formatting information
  - go to the "Format" and select "Make Plain Text"







Create a File, and Read It

- In the folder given to you by `os.getcwd()`, create a file containing your date of birth in the following format `dd-mm-yyyy`, then a linebreak (press Enter after typing the date)
- Save the file as "`dob.txt`"
- In Python, type the following

```
>>> s = open("dob.txt").read()
>>> s
>>> print s
```

A Done

B No computer

New-Line Characters 

```
>>> s = open("dob.txt").read()
>>> s
'09-02-1987\n'    # on Windows, might be: '09-02-1987\r\n'
>>> print s
09-02-1987

>>>
```

- Line breaks are represented using the "new line character" coded by `"\n"` (backslash-n)
- Unix platforms (such as Linux or Mac OS X) only use `"\n"`
- Windows uses `"\r\n"`
- `"\r"` is character for "carriage return"

Cleaning Up a String

- The characters `"\r"` and `"\n"` and `"\t"` (tab) or `" "` (just a regular space) are all considered *white space*
- They are usually not very useful when they are
  - at the beginning of a string
  - at the end of a string (trailing spaces)
- Given a string in variable `mystring`, calling `mystring.strip()` returns the string with the white space at the beginning and the end removed

So…

```
>>> s = open("dob.txt").read()
>>> s
'09-02-1987\n'
>>> s.strip()
'09-02-1987'
```

Processing the Input

```
>>> s = open("dob.txt").read().strip()
```

- `s` is a string, for me, `s = '09-02-1987'`
- We can manipulate `s` as a string, i.e. `s[0:2]` will give '09', and so on
- We can also use the `s.split()` function

`split()` function

```
>>> s = open("dob.txt").read().strip()
>>> help(s.split)
Help on built-in function split:

split(...)
    S.split([sep [,maxsplit]]) -> list of strings

    Return a list of the words in the string S, using sep as the
    delimiter string.  If maxsplit is given, at most maxsplit
    splits are done. If sep is not specified or is None, any
    whitespace string is a separator and empty strings are removed
    from the result.
```

- This functions usually splits a string at the spaces, and returns a list of words
- But by giving it an argument, it can also split at other characters

```
>>> >>> "this is a phrase".split()
['this', 'is', 'a', 'phrase']
>>> "this      is a phrase".split()
['this', 'is', 'a', 'phrase']
>>> "this      is a phrase       ".split()
['this', 'is', 'a', 'phrase']

>>> "this is a phrase".split("h")
['t', 'is is a p', 'rase']

>>> s = open("dob.txt").read().strip()
>>> s.split("-")
['09', '02', '1987']
```

Could You Drink on Jan 1st ?

Write a function that does the following:

- Reading a date from "dob.txt"
- Clean the string and split it
- Looking at the year (and without making complicated calculations), determine how old you were on 01-01-2014

A Done

B Help!

Looking at the first steps of writing a file…

PART 2: WRITING A FILE

Open a File that Doesn't Exist

- We now want to save the result of the age calculation that we just did

```
>>> age = 21
>>> open("my-age.txt").write(age)
```

```
>>> age = 21
>>> open("age.txt").write(age)

Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    open("age.txt").write(age)
IOError: [Errno 2] No such file or directory: 'age.txt'
```

File in Write Mode

- When opening a file, we must tell Python what operations we are going to do on it (read or write, or both)
- By default, Python assumes weare going to: **read only** and a file that **exists**
- When we want to write a file, we must use a mode string to let Python know

```
open(filename, mode_string)
```

- where `mode_string` is either "r" or "w" (or more which we will see later

We Try Again…

```
>>> age = 21
>>> open("my-age.txt", "w").write(age)
```

```
>>> open("age.txt", "w").write(age)

Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    open("age.txt", "w").write(age)
TypeError: expected a character buffer object
```

- This still does not work, because Python expects us to write **strings** to the file
- We must convert any output to a string before writing it (for instance using the `str(...)` conversion function)

Write a Program

Using what you already wrote previously, write a program that

- Reads a date from "dob.txt"
- Calculates the age on 01-01-2014
- Writes this age in "age.txt"

A Done

B Help!

Pacing and Understanding

How well did you understand today?

A. Too easy, this lecture is way below my abilities

B. Everything went at a good pace, and I am fine

C. Too fast, but I will catch up on my own

D. Too fast, and I need you to slow down

E. I really do not think I can handle this