

Assignment #12: Files

Master in Informatics and Computing Engineering
Programming Fundamentals
Instance: 2018/2019

Goals: File I/O

Pre-requirements (prior knowledge): see bibliography of Lecture #21

Rules: you may work with colleagues, however, each student must write and submit in Moodle his or her this assignment separately. Be sure to indicate with whom you have worked. We may run tools to detect plagiarism (e.g. duplicated code submitted)

Deadline: 8:00 Monday of the week after (24/12/2018)

Submission: to submit, first pack your files in a folder RE12, then compress it with zip to a file with name 2018xxxxx.zip (your_code.zip) and last (before the deadline) go to the Moodle activity (**you have only 2 attempts**)

1. wc

In Linux, there is a command called `wc` which displays general statistics about a given file. Write a function `wc(filename)` that given the string `filename` returns the tuple: (number of lines, number of words and number of characters) in the file.

Save the program as `wc.py`.

For example:

- Considering that in the current directory exists a text file [shakespeare.txt](#), then `wc("shakespeare.txt")` returns the tuple (14, 105, 611)
- Considering that in the current directory exists a text file [monty.txt](#), then `wc("monty.txt")` returns the tuple (15, 155, 856)

2. cut

In Linux, there is a command called `cut` which allows selecting a column(s) of a file given a delimiter. Write a function `cut(filename, delimiter, field)` that receives a `filename`, a `delimiter` character and `field` (which can be an integer or a list) as parameters and returns the corresponding column(s) (`field`) from the file as a string.

Save the program as `cut.py`.

For example, considering that in the current directory exists a file [data.csv](#) with the content:

1,2,3,4,5
6,7,8,9,10
11,12,13,14,15

- `cut("data.csv", ",", 2)` returns the string "2\n7\n12"
- `cut("data.csv", ",", [2,4])` returns the string "2,4\n7,9\n12,14"

3. Longest word in URL

Write a Python function `longest_word(url)` that given the string `url` returns the longest word in that Web resource that is also in the `/usr/share/dict/words`. The word comparison is case insensitive and a word can contain punctuation or other non-alphabetic characters.

Copy the the words file as `words` into the current working directory (if there is no `words` file in your system, you may get a 920 KiB [copy from here](#)).

If there are multiple words of the same length, return the first word in lexicographical order.

You can retrieve a Web resource by doing:

```
import urllib.request
response = urllib.request.urlopen('http://www.example.com/')
html = response.read().decode()
```

If the program takes too long to process the intersection with the dictionary, have a look at the [Set collection datatype](#).

Save the program in the file `longest_word.py`

For example:

- `longest_word("https://en.wikipedia.org/wiki/Monty_Python")` returns the string `"acknowledgement"`
- `longest_word("https://web.fe.up.pt/~jlopes/doku.php/teach/fpro/sheet")` returns the string `"classification"`

4. Sort by field

Write a function `sort_by_field(filename, field)` that, given a CSV (Comma Separated Values) file `filename` and a string `field`, sorts the lines of the CSV file by `field` in ascending order. In case of ties, keep the order of the original file.

The CSV file uses a comma to separate each specific data value (field). While the first line contains the name of each field, every subsequent line after that contains actual data associated to each field.

```
mail,name,surname
m.white@student.hathaway.edu,Maya,White
b.mwangi@hathaway.edu,Benjamin,Mwangi
e.nemec@hathaway.edu,Elizabeth,Nemec
...
```

Save the program in the file `sort_by_field.py`

For example, considering that in the current directory exists a text file [emails.txt](#):

- `sort_by_field("emails.txt", "surname")` returns a string with the same content of the file [emails_by_surname.txt](#)
- and `sort_by_field("emails.txt", "mail")` returns a string with the same content of the file [emails_by_email.txt](#)

5. Parse a tuple

Write a function `parse(filename)` that receives a `filename` string and parses the content of the file, which contains either parenthesis or integers, and returns the file structure as a nested tuple.

Save the program in the file `parse.py`.

For example:

- Considering that in the current directory exists a text file [tuple1.txt](#):

```
(  
  (  
    1  
    -2  
    (  
      5  
    )  
  )  
)
```

then `parse("tuple1.txt")` returns the tuple `((1, -2, (5,)),),)`

- Considering that in the current directory exists a text file [tuple2.txt](#) then `parse("tuple2.txt")` returns the tuple `((((100, -20), (-35, 46)), (21,), (50, (78, -89))),),)`

The end.

FPRO, 2018/19