chsh_files

May 19, 2022

1 Cheat Sheet on files

Cheat sheet on files.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 change the encoding of a file

[2]:
```
with open("essai.txt", "w", encoding="latin-1") as f:
    f.write("ée\nàà")
```

[3]:
```
from ensae_projects.datainc import change_encoding
change_encoding("essai.txt", "essai.utf8.txt", enc1="latin-1", enc2="utf-8")
```

[3]:
```
1
```

[4]:
```
with open("essai.utf8.txt", "r", encoding="utf8") as f:
    s = f.read()
print(s)
```

```
ée
àà
```

1.2 select a subset of columns from a tsv files

[5]:
```
import pyensae.datasource
%load_ext pyensae
files = pyensae.datasource.download_data("OnlineNewsPopularity.zip",
                                website="http://archive.ics.uci.edu/ml/
 ↪machine-learning-databases/00332/")
files[1]
```

[5]:
```
'OnlineNewsPopularity/OnlineNewsPopularity.csv'
```

[6]:
```
%head OnlineNewsPopularity/OnlineNewsPopularity.csv -n 2
```

[6]:
```
<IPython.core.display.HTML object>
```

[7]:
```
from ensae_projects.datainc import enumerate_text_lines
def clean_column_name(s):
    return s.strip()
bigfile = enumerate_text_lines("OnlineNewsPopularity/OnlineNewsPopularity.csv",
                               encoding="utf-8", header=True, quotes_as_str=False,
                               sep=",",
                               clean_column_name=clean_column_name, fLOG=print)
```

[8]:
```
res = list(map(lambda row: {"LDA_00": row["LDA_00"], "title_sentiment_polarity":
 ↪row["title_sentiment_polarity"]},
           bigfile))
len(res)
```

[8]:
```
39644
```

[9]:
```
import pandas
df = pandas.DataFrame(res)
df.head()
```

[9]:

	LDA_00	title_sentiment_polarity
0	0.500331204081	-0.1875
1	0.799755687423	0.0
2	0.217792288518	0.0
3	0.0285732164707	0.0
4	0.0286328101715	0.136363636364

1.3 look at the head or tail of a file

We use magic commands %head and %tail.

[10]:
```
%load_ext pyensae
```

```
The pyensae extension is already loaded. To reload it, use:
  %reload_ext pyensae
```

[11]:
```
%head essai.txt -n 1 -s ignore
```

[11]:
```
<IPython.core.display.HTML object>
```

[12]:
```
%tail essai.txt -n 1 -s ignore
```

[12]:
```
<IPython.core.display.HTML object>
```

1.4 select lines of a flat file based on a regular expression

[13]:
```
%load_ext pyensae
```

```
The pyensae extension is already loaded. To reload it, use:
  %reload_ext pyensae
```

[14]:
```
%grep essai.utf8.txt .*é.*
```

[14]:
```
<IPython.core.display.HTML object>
```

More complex, we extract all lines containing a substring and we add the header to the file to make it look like a dataframe. We do that usually when we cannot load a big file into memory with pandas due to the lack of memory. This code relies on magic command grep and function enumerate_grep.

[15]:
```
import pandas
import pyensae
df = pandas.DataFrame([dict(name="Dupré", first_name="Xavier"),
                       dict(name="Dupré", first_name="Sloane")])
df.to_csv("data.txt", encoding="utf8", index=False)
%head data.txt
```

[15]:
```
<IPython.core.display.HTML object>
```

[16]:
```
raw = %grep data.txt Xavier --raw
```

[17]:
```
raw
```

[17]:
```
'Xavier,Dupré\n'
```

[18]:
```
header = %head data.txt -n 1 --raw
header
```

[18]:
```
'first_name,name\n'
```

[19]:
```
with open("data_xavier.txt", "w", encoding="utf8") as f:
    f.write(header)
    f.write(raw)

%head data_xavier.txt
```

[19]:
```
<IPython.core.display.HTML object>
```

[20]:
```
pandas.read_csv("data_xavier.txt")
```

[20]:
```
  first_name   name
0     Xavier  Dupré
```

[21]:
```

```