Processing Files

Thomas Schwarz, SJ

Processing Files in Python

- We write strings to the file

```
with open('somefile.txt','wt') as f:
 f.write(str(500)+”\n")
```

- Redirect print

```
with open('somefile.txt','wt') as f:
 print(500, file = f)
```

Processing Files in Python

- Reading files
  - The read-instruction

```
string = inFile.read(10)
```

    reads ten bytes of the file

  - Read the entire file

```
with open('somefile.txt', 'rt') as f:
    data = f.read()
```

  - Variable data now contains the file contents as a string

Processing Files in Python

- Reading files
  - Read line by line

```
with open('somefile.txt', 'rt') as f:
    for line in f:
        #process line
```

More String Processing

- To process read lines:
  - `strip()` and its variants `lstrip(), rstrip()`
    - Remove white spaces (default) or list of characters from the beginning & end of the string
  - `split()` creates a list of words separated by white space (default)

```
"This is a sentence with many words in it.".split()
```

```
['This', 'is', 'a', 'sentence', 'with', 'many', 'words', 'in', 'it.']
```

Examples

- Finding all words over 13 letters long in “Alice in Wonderland”
  - Download from Project Gutenberg

```
import string

with open("alice.txt", "rt", encoding = "utf-8") as f:
    for line in f:
        for word in line.split():
            if len(word) > 13:
                print(word)
```

Examples

- Count the number of words and of lines in “Alice in Wonderland”
  - Read the file line by line
  - The number of words in a line is the length of line.split.

```
import string

line_counter = 0
word_counter = 0
with open("alice.txt", "rt", encoding = "utf-8") as f:
    for line in f:
        line_counter += 1
        word_counter += len(line.split())
print(line_counter, word_counter)
```

Problems with Line Endings

- ASCII code was developed when computers wrote to teleprinters.
  - A new line consisted of a carriage return followed or preceded by a line-feed.
- UNIX and windows choose to different encodings
  - Unix has just the newline character “\n”
  - Windows has the carriage return: “\r\n”
- By default, Python operates in “universal newline mode”
  - All common newline combinations are understood
  - Python writes new lines with the system default
- You could disable this mechanism by opening a file with the universal newline mode disabled by saying:
  - `open("filename.txt", newline='')`

Example:
Reading a CSV file

- Gujarat rainfall data from an India Meteorological Department Survey from 1901 - 2015
- First line contains column headers
- Each row contains entries separated by tabs
- Simple task: Extract a list of the annual rainfall per year

Gujarat Annual Rainfall

- Step one:
  - Open the file

```
with open('gurainfall.csv') as datafile:
```

  - Sometimes, you would have to give the full path to the filename

Gujarat Annual Rainfall

- Step 2:
  - Skip over the first line

```
with open('gurainfall.csv') as datafile:
        datafile.readline()
```

Gujarat Annual Rainfall

- Step 3: Extract the data we want
  - Look at the header column or look at the data directly
  - Get line for line, split on tab, and load the 15th element
  - Append to a `result`-list

```
result = [ ]
    with open('gurainfall.csv') as datafile:
        datafile.readline()
        for line in datafile:
            data = line.split('\t')
            result.append(float(data[14]))
```

Gujarat Annual Rainfall

- Step 4:
  - Put this into a nice function
    - ```
      def get_annual_rainfall( ):
          result = [ ]
      with open('gurainfall.csv') as datafile:
          datafile.readline()
          for line in datafile:
              data = line.split('\t')
              result.append(float(data[14]))
      return result
      ```
      ```

Gujarat Annual Rainfall

- Now that we have the data, we should do something
- Preview: Showing the data
  - Use `matplotlib.pyplot`
  - Needs to be imported: `pip3.10 import matplotlib`
  - Has a simple plotting function
  - But need to use plt.show( ) in order to see anything

Gujarat Annual Rainfall

- Plot needs the x-values and the y-values
  - Need to be an array (or something array-like)

```
import matplotlib.pyplot as plt

plt.plot(range(1901, 2016), get_annual_rainfall())
plt.show()
```

Gujarat Annual Rainfall

- Result: