Plain text and text encoding

14 October 2024

Plain text and text encodings

What is Python and what can it do?

It's a free, open-source general purpose language. Nowadays the only version to use is Python 3 and the most up-to-date release is 3.13 (Python 2 is maintained on some systems for legacy reasons but new code should not be written in it).

Python can essentially do anything a computer can do, although it might not be the best choice for some things. It's a first choice for a lot of data science work and has become the main language for machine learning/AI.

Because of its breadth, there will always be areas of Python that are unfamiliar. If you know the fundamentals and have some practice, you'll be able to understand lots of Python code, even if the applications of it are different from what you're used to.

How do you run Python?

- Run a Python script (a text file with a `.py` extension) from the command line: `python myscript.py`
- From the built-in Python shell from the command line: type `python` and return and you'll get a prompt: this is an exploratory environment where you can try snippets
- From an installed shell like `iPython`: a more full featured version of the above
- From an installed program like Jupyter Notebook, which runs in a browser
- Online in Google Colab, Python Anywhere, GitHub Actions or many others

In Python it's easy to work with plain text. With `persuasion.txt` in the same folder as this notebook, we can read in the whole of Austen's *Persuasion* like this:

[ ]:
```
with open('persuasion.txt', 'r', encoding='utf-8') as f:
    persuasion = f.read()
```

If you're running this notebook on Colab, rather than from Jupyter Notebook on your laptop, you will need to do two extra things to read and write files: #### import and mount Drive like this:

```
from google.colab import drive

drive.mount('/content/drive')
```

open the revelant folder on Drive depending on where you saved the file, eg: `with open('drive/MyDrive/plain-text/persuasion.txt', 'r'): as fr:`

```
persuasion = f.read()
```

Now `persuasion` is a variable pointing at the full text of *Persuasion*. We can view the contents of a variable in Jupyter just by typing it in a code cell.

What usable information do we have here? Python has read and closed the file on your laptop so open it with a text editor and compare the output. What information can we get from plain text?

[ ]:
```
persuasion
```

Notice that this is fast. Plain text like this is very fast on a modern computer. But it's not very convenient for reading a novel.

We can see parts of the text using Python's *slice* notation. The numbers are character positions within the big text string which is `persuasion`. Try changing these to get a different slice.

[ ]:
```
persuasion[2000:3000]
```

How many characters are there in total? We can get this with the `len` function:

[ ]:
```
len(persuasion)
```

Does this seem reasonable?

The line endings are annoying. Here they're represented by `\n` (Unix line endings) but you will also see `\r\n` (Windows line endings).

We can get rid of them with Python's built-in `replace` method:

[ ]:
```
persuasion = persuasion.replace("\n", " ")
```

[ ]:
```
persuasion[1000:2000]
```

What else can we do with built-in methods? Jupyter will help you. Type `.?` after the variable you want to do something with (in this case `persuasion`). Then press tab. This will show the string methods available.

[ ]:
```
persuasion.
```

The methods available with `persuasion` depend on what type of thing `persuasion` is. We can find out with `type`. This is a very useful function in Python because it's easy to be under a misapprehension about what you're dealing with (which leads to a `type error`).

[ ]:
```
type(persuasion)
```

So `persuasion` is a string and we can use any of Python's built-in string methods on it. Let's try `find`. Anne Elliot is the principal character in *Persuasion* so let's look for *Anne*

[ ]:
```
persuasion.find("Anne")
```

Wait, what does `2133` refer to? You can get help on a particular method with a `?` and then shift + tab.

[ ]:
```
persuasion.find?
```

This is perhaps a bit abstruse but it means that *Anne* first occurs in our *Persuasion* string at character number 2133. Is that useful?

Let's try the `count` method. That seems like it should count the number of occurrences of something, but we can check.

[ ]:
```
persuasion.count?
```

[ ]:
```
persuasion.count("Anne")
```

We have to be careful with results like these. Does it mean that the person *Anne* is mentioned 496 times in *Persuasion*? Maybe, but is there a character called *Annette* or a trip to *Annecy* in France? Is there even another character called Anne who is not Anne Elliot?

[ ]:
```
persuasion.count("Anne Elliot")
```

[ ]:
```
persuasion.count("Miss Elliot")
```

Notice that when we get the length of *Persuasion* the syntax is different from when we count occurrences:

`len(persuasion)`

`persuasion.count("Anne")`

`len` is a *function* but `count` is a particular type of function, called a *method*.

Why do you get an error if you run `len` on an integer, eg `len(5)`? Why isn't the answer 1?

Group work

Keeping to the constraint of not importing any Python modules, just using built-in methods and functions, what else can you do with *Persuasion*?

- list of string methods with examples

Here are some ideas, but feel free to do your own thing: - divide up the text into equal chunks and look for distributions of words of phrases through the novel - get some context around a string you find using slice notation (yes this is very crude) - split the text on spaces using the `split` method. What are the pros and cons of this? - from the split text create a unique list of the strings and count - tediously remove the punctuation to improve the results of this - don't forget that you can use `persuasion.?` to find other string methods; are these useful?

Now import *Emma* from the same folder as *Persuasion*. You should have both texts in memory and so can do some comparative work using Python. Again, don't import anything, see how you can work withiin the constraint.

If you still have time. Find another text online and import that. Look for something which isn't a novel and see what difference it makes to what you can do with it. Possible sources for plain-text files are:

Gutenberg

Oxford Text Archive

Wikisource