Problem Set 3

Due Wednesday Sep. 27, 10 am

Comments

- This covers material in Unit 5..
- It's due at 10 am (Pacific) on September 27, both submitted as a PDF to Gradescope as well as committed to your GitHub repository.
- Please see PS1 for formatting and attribution requirements.

Problems

1. In this problem we'll create a little time-saving hack as a way to get practice with Python classes.

   Suppose I want to be lazy and when I type `"q"` in Python, Python should quit (i.e., I don't want to have to type `quit()`). Write a bit of Python code to achieve this.

   Hints: (a) What specifically happens if I type the name of an object? (b) You will probably want this code to be in an `#| eval:false` chunk, since it if it successful, its operation will cause Python to quit and presumably your document will not render.

2. Let's investigate the structure of the `pandas` package to get some experience with the structure of a large Python package and with how `import` and the `__init__.py` file(s) are used. You'll need to go into the Pandas source code (see Unit 5). Note that the main `__init__.py` and the `__init__.py` files in the subpackages/submodules are complicated, and I'm not expecting you to understand everything about them. Also note that the following cases involve functions, classes, and class methods. Be sure to be clear to say which of those it is and in the method case(s), make sure you're clear on what class the method is part of and any class inheritance structure. Import `pandas` and then consider the following questions:

   a. Consider `pandas.core.config_init.is_terminal`. What namespace is it (or its class) in? What file/module is `get_terminal` in? Is it a function, class, or a class method (and if a class method what is the class)? What namespace is it (or its class) in when you run `import pandas`? Describe how it is imported by discussing the relevant statement(s) in the relevant `__init__.py` file(s).
   b. Consider `pandas.read_csv`. What namespace is it (or its class) in? What file/module is `read_csv` in? Is it a function, class, or a class method (and if a class method what is the class)? Describe how it is imported by discussing the relevant statement(s) in the relevant `__init__.py` file(s).

c. Consider `pandas.arrays.BooleanArray`. What namespace is it (or its class) in? What file/module is `BooleanArray` in? Is it a function, class, or a class method (and if a class method what is the class)? Describe how it is imported by discussing the relevant statement(s) in the relevant `__init__.py` file(s).
d. Consider `pandas.DataFrame.to_csv`. What namespace is it (or its class) in? What file/module is `to_csv` in? Is it a function, class, or a class method (and if a class method what is the class)? Describe how it is imported by discussing the relevant statement(s) in the relevant `__init__.py` file(s).

Hints: (1) `grep -R <pattern> <directory>` will search all files within a directory recursively. (2) As you work on this, you may want to be able to modify one or more of the `__init__.py` files to better understand what is happening (e.g., by commenting out a line of code or adding a print statement). A good way to do this is to create a Conda environment in which pandas is installed, so you isolate any changes you make, e.g., `conda create -n test_env python=3.11 pandas`. Then you can edit code files in the environment and when you start Python and import pandas, you should see the effects of your changes. Alternatively, you could use the debugger to set breakpoint(s) in an `__init__.py` file. (3) Or you might create your own small toy package to experiment and see how things work with nested `__init__.py` files and various ways to use `import`.

3. The goal of this problem is two-fold: first to give you practice with regular expressions and string processing and the second to have you thinking about writing well-structured, readable code (similar to question 4 of PS1). The website https://www.presidency.ucsb.edu/documents/presidential-documents-archive-guidebook/annual-messages-congress-the-state-the-union#axzz265cEKp1a has the text from all of the State of the Union speeches by US presidents. (These are the annual speeches in which the president speaks to Congress to "report" on the situation in the country.) Your task is to process the information and produce data on the speeches for the years 1900 to present. Note that while I present the problem below as subparts (a)-(i), your solution does NOT need to be divided into subparts in the same way. Your solution should do all of the downloading and processing from within Python so that your operations are self-contained and reproducible.

   You can choose to use either a functional programming approach or an object-oriented approach, or possibly something that mixes the two. I strongly recommend that you use the approach that you are *less* familiar with so as to gain more experience. Please think about writing short, modular functions or methods, and operating in a vectorized manner (e.g., using `map` or possibly list comprehension). Think carefully about how to structure your objects to store the speech information so that the structure works well with your functions/methods.

   Given you've already worked on webscraping, I'm providing some initial code for processing the main landing page in `ps/ps3start.py` in the course repository, and an example for getting the text of a speech. Also note that you will want to distinguish regular dashes (i.e., the hyphen in a hyphenated work like "team-building") from a long ("em") dash that separates clauses (unicode `U+2014`) – syntax from Unit 2 Section 5 may be helpful.

   a. Extract the URLs for the individual speeches from the URL above. Then use that information to read each speech into Python.

   b. For each speech, extract the body of the speech.

c. Convert the text so that all text that was not spoken by the president (e.g., Laughter and Applause) is stripped out.

d. Extract the words and sentences from each speech as lists of strings, one element per sentence and one element per word. Note that there are probably some special cases here. Try to deal with as much as you can, but we're not expecting perfection.

e. For each speech count the number of words and characters and compute the average word length.

f. Count the following words or word stems: I, we, America{,n}, democra{cy,tic}, republic, Democrat{,ic}, Republican, free{,dom}, war, God [not including God bless], God {B,b}bless, {Jesus, Christ, Christian}, and any others that you think would be interesting.

g. The result of all of this activity should be well-structured data object(s) containing the information about the speeches.

h. Make some basic plots that show how the variables have changed over time. Your response here does not have to be extensive but should illustrate what you would do if you were to proceed on to do extensive exploratory data analysis. If you're not comfortable plotting in Python (that is the case for me) and you prefer to move the relevant data to R to make the plots, that is fine.

i. Extra credit: Do some additional research and/or additional thinking to come up with additional variables that quantify speech in interesting ways. Do some plotting that illustrates how the speeches have changed over time.

4. Now sketch out a design for a functional programming (FP) approach (if your solution to problem 3 used OOP) or an OOP approach (if your solution to problem 3 used functional programming). If you're designing an OOP approach, decide what the classes would be and the fields and methods of those classes. If you're designing a FP approach, decide what the functions would be and what inputs/output they would use. **To be clear, you do not have to write any of the code for the methods/classes/functions; the idea is just to design the code.** As your response in the OOP case, for each class, please provide a bulleted list of methods and bulleted list of fields and for each item briefly comment what the purpose is. Or in the FP case, for each function, provide a bulleted list of inputs and output and briefly comment on the purpose of each function.