CMSC330: Python Practicum

Chris Kauffman

Last Updated:
Thu Sep 21 09:16:47 AM EDT 2023

Logistics

Assignments

- Lecture Quiz 3 up later, due Tue
- Project 2 no late BUT +4 late tokens for all

Reading

Review only

Goals

- Wrap up FSMs
- Python in Practice

“Coding Day”

practicum (noun) *a practical section of a course of study*

According to what I’ve been told, it is typical in this course to include a few periods in which we just code something in a language to gain insight into the “flavor” of the language and see course techniques in practice.

A Quick Demo of Program Development

Spell Checker

- Spell check the contents of a text file
- Use dictionary words stored in another file
- Produce a new file with misspelled words identified or corrected
- Provides some module level functions that could be used by other apps
- Provides stand-alone command line run for simple usage

Techniques

- Interactive and Iterative development
- Docstrings and help features in Python
- Regex in action
- Building core logic introducing flexibility via first-class functions

Exercise: Mark Incorrectly Spelled Words

```
>> cat gettysburg.mispelled.txt
Four score and seven years agoo our pathers brought forth on this
continent, a new nation, conceived in Libertie, and dedicated to the
proposicion that all peops are created equal.

Abraham Lincoln
November 19, 1863

>> cat gettysburg.mispelled.txt.checked
Four score and seven years **agoo** our **pathers** brought forth on this
continent, a new nation, conceived in **Libertie**, and dedicated to the
**proposicion** that all **peops** are created equal.

**Abraham** **Lincoln**
**November** 19, 1863
```

Identify a general methodology for creating the "checked" file from the first file

Answers: Let's Code It

- Completed version of `spellcheck.py` is in the codepack
- Will demonstrate the general flow of how I reasoned about so that you can see how the code built outwards
- Constantly googling for "what's the python function for X" which is totally expected when there are *so many details. . .*

Interactive + Iterative Development

- Python allows reloading modules after function definition changes
- Use the `importlib` module; useful for interactive development[1]

```
>>> from importlib import reload
>>> import spellcheck as sc
>>> sc.spellcheck_file("gettysburg.txt")
...
# results don't look right
# edit code to improve, save and reload
>>> reload(spellcheck)
<module 'spellcheck' from 'spellcheck.py'>

>>> sc.spellcheck_file("gettysburg.txt")
...
# results look better, edit, save, reload, rerun
```

[1] Of course if you are using emacs you can fire up an inferior Python shell and use `C-c C-c` to send your modified buffer to the shell but I digress...

Alternative Designs and Extensions

- Make the checkers objects in a class hierarchy; initialize them with data, don't need to pass around `locals()` for context
- Add a checker with a "personal dictionary" for words like "Abraham" which aren't in the standard dictionary, allow saving / loading personal dictionary
- Adjust the file processing to avoid the "slurp"; read a line at a time as is more "Pythonic"; possibly adjust output as well so one never needs to store the entirety of input/output in memory