Software Engineering 265
Software Development Methods
Summer 2019

Assignment 3

Due: Tuesday, July 23, 11:55 pm by submission via git
(no late submissions accepted)

Programming environment

For this assignment please ensure your work executes correctly on the Linux machines in ELW B238. You are welcome to use your own laptops and desktops for much of your programming; if you do this, give yourself a few days before the due date to iron out any bugs in the Python 3 program you have uploaded to the BSEng machines. (Bugs in this kind of programming tend to be platform specific, and something that works perfectly at home may end up crashing on a different hardware configuration.)

When evaluating your submission, the teaching team will use the environment established by the `setSENG265` command that is available on the lab workstations.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor (Zastre).** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact me as soon as possible. (Code-similarity analysis tools will be used to examine submitted programs.)

Objectives of this assignment

- Use more advanced features of Python 3, including user-defined classes and regular expressions.
- Use Git to manage changes in your source code and annotate the evolution of your solution with messages provided during commits.
- Test your code against the 15 provided test cases from assignment #1.

`calprint3.py`: A module with at least one class

You are to complete the implementation of the class `Calprint` which will be contained in the file named `calprint3.py`. Unlike the two previous assignments, however, your own code will be called by a test-driver program named `tester3.py` that is provided by the instructor.

The two methods of the class `Calprint` that must be implemented are:

- A constructor which takes a single parameter (i.e., the name of the ICS file)

- A method named `get_events_for_day` which takes a `datetime` object as its single parameter. If the day corresponding to the parameter has events (i.e., were in the ICS file whose named was given to the object's constructor), then the method returns a string with that days events formatted as required in previous assignments (including heading for date plus line of dashes). If the day corresponding to the parameter has no events, then `None` is returned by the method.

You are free to implement any other methods in `Calprint` necessary to complete your solution.

A call to `tester3.py` will used identical arguments to that from the previous assignment. For example:

```
./tester3.py --start=18/6/2019 --end=18/6/2019 --file=one.ics
```

is a typical use of the provider driver program.

A few more observations:

- You must have your solution in the `Calprint` class. As already mentioned, you are free to add any additional classes or methods needed for your solution as long as they are contained in the `calprint3.py` file. Note, however, that global variables or functions in `calprint3.py` are not permitted.

- You must use regular expressions in your solution, and not simply in one or two spots.

- Your `Calprint` class must not assume it is the only instance of `Calprint`. That is, the evaluators may test your solution with a program that instantiates two or more `Calprint` objects at the same time, and request events for dates where those dates may not be in chronological order. (Therefore do not be tempted to simply use `Calprint` as a wrapper that calls the `main()` function of your A#2 solution.)

- **You are not permitted to change any code in `tester3.py`.**
- You must **not use global variables**.
- You must **make good use of functional decomposition**. Phrased another way, your submitted work **must not** contain one or two giant functions where all of your program logic is concentrated.

Exercises for this assignment

1. Within your `git` repo ensure there is an "`a3/`" subdirectory. (For testing please use the files provided for assignment #1.) Your "`calprint3.py`" file must be located in this "`a3/`" directory. Note that a starter `calprint3.py` file (along with `tester3.py`) is available for you in the `/home/zastre/seng265/a3` directory.
2. Write your program. Amongst other tasks you will need to:
   - read text input from a file, line by line.
   - implement required methods for the solution, along with any other methods you believe are necessary.
   - extract substrings from lines produced when reading a file.
3. **Keep all of your code in one file for this assignment**. Everything you submit for credit must be in `calprint3.py`. Will we use our copy of `tester3py`.
4. Use the test files and listed test cases to guide your implementation effort. Refrain from writing the program all at once, and budget time to anticipate when "things go wrong".
5. For this assignment you can assume all test inputs will be well-formed (i.e., our teaching assistant will not test your submission for handling of input or for arguments containing errors). Assignment 4 may specify error-handling as part of the assignment.

What you must submit

- A single Python source file named "`calprint3.py`" within your `git` repository containing a solution to assignment #3.

Evaluation

Our grading scheme is relatively simple.

- "A" grade: A submission completing the requirements of the assignment which is well-structured and very clearly written. Regular expressions are used. All tests pass and therefore no extraneous output is produced.

- "B" grade: A submission completing the requirements of the assignment. `calprint3.py` can be used without any problems; that is, all tests pass and therefore no extraneous output is produced. Regular expressions are used.

- "C" grade: A submission completing most of the requirements of the assignment. `calprint3.py` runs with some problems. Regular expressions are used.

- "D" grade: A serious attempt at completing requirements for the assignment. `calprint3.py` runs with quite a few problems; some non-trivial tests pass. Regular expressions are not used.

- "F" grade: Either no submission given, or submission represents very little work, or no tests pass.