Software Engineering 265
Software Development Methods
Fall 2021

Assignment 4

Due: December 6, 11:55 pm submitted to the assignment 04 page in brightspace
(no late submissions accepted)

Read these assignment instructions carefully.

Programming environment

For this assignment you must ensure your work executes correctly on the virtual machines you installed as part of Assignment 0 subsequently referred to as *Senjhalla*. This is our "reference platform". This same environment is used by the teaching team when evaluating your submitted work.

All test files for this assignment are available on the SENG server at **/seng265work/A4/** Use the **scp** command to copy these files into your Senjhalla

Any programming done outside of Senjhalla might not work during evaluation, and thus result in 0 marks for the assignment.

Individual work

Like all the other assignments, Assignment 4 is to be completed by each individual student (i.e., no group work, no copying of code, your own work only). You are encouraged to discuss aspects of the problem with fellow students which is part of the learning process. However, sharing of code fragments is strictly forbidden. Note: Code-similarity analysis tools are used to examine submitted work of all students. Plagiarism of any form is strictly forbidden.

Objectives of this assignment

- Use more advanced features of Python 3, including user-defined classes and regular expressions.
- Test your code against the provided test cases.

process_cal4.py: A module with at least one class

You are to complete the implementation of the class *process_cal* which will be contained in the file named *process_cal4.py*. Unlike the three previous assignments, however, your own code will be called by a test-driver program named *tester4.py* that is provided by the teaching team.

The below ***two methods*** of the class *process_cal* **must be implemented exactly as described below**:

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

- A method named *get_events_for_day*
    - Parameter: a *datetime* object
    - This method takes the datetime object and checks to see if there are any events for that date in the ICS file received by the constructor and if found, returns a string with the day's events
        - If the day corresponding to the datetime object parameter has events (i.e., there were events in the ICS file whose named was given to the object's constructor), then the method returns a string with that day's events formatted as required in previous assignments.
        - If the day corresponding to the datetime object parameter has no events, then *None* is returned by the method.

You are encouraged to implement additional classes and methods in the class *process_cal* as necessary for your solution and to reflect the problem decomposition.

A call to *tester4.py* will use identical arguments to that from the previous assignment. For example:

```
./tester4.py --start=2021/01/24 --end=2021/4/15 --file=one.ics
```

is a typical use of the provided driver program.

A few more observations:

- Your solution **must be completely contained within** the *process_cal* class (which itself is completely contained in *process_cal4.py*). **You are not permitted to use any global variables in *process_cal4.py*.** You are encouraged to implement additional classes and methods in the *process_cal4.py* file as necessary for your solution and to reflect the problem decomposition. Feel free to use elements of your Assignment 2 solution.

- Your *process_cal* class must not assume that it is the only instance of *process_cal*. That is, the evaluators may test your solution with a program that instantiates several *process_cal* objects, and then requests events for dates that are not in chronological order. **Do not output to *stdout* (or any other output stream) within your *process_cal* class. Do not simply use *process_cal* as a wrapper that then calls the *main* function of your Assignment 2 solution.**

- **You must use regular expressions** as part of your solution. Use methods of **Python re module. The re module is already part of the python installation in senjhalla. However, in order to use it, you will need to import the module into your file with "import re"**

- **You are not permitted to change any code in *tester4.py*.** In fact, that particular Python file specifically disables the default *stdout* stream when invoking methods on *process_cal* objects.

- **The problem decomposition must be reflected in your program solution**. Thus, a couple of unwieldy methods will not result in full marks even if the program works.

- The Bash command **Diff** will be used to determine success and failure of tests.

- **Assessment criteria**

    - **Docstrings are required at all levels: package, module, class, method.**
    - Documentation and commenting: The purpose of documentation is for anyone other than yourself (with coding knowledge) can review your program and quickly understand how it works. In terms of marking, documentation is not a large part of the mark, but it will be factored into the overall quality assessment.
    - Problem decomposition: Quality coding requires the good use of classes, attributes, and methods. Code that relies on few large functions is not a good solution. Typically, a good program has a main function that does some basic tasks and calls other functions, that do most of the work. A solution that passes all tests, but contains all code in one or two large functions will not be given a grade better than a B.
    - Debugging / Comment artifacts: You must submit a clean file with no residual commented lines of code or unintended text.
    - Quality of solution: Markers will access the submission for logical and functional quality of the solution. Some examples that would result in a reduction of grade: solutions that read the input files several times, solutions which represent the data in inappropriate data structures, solutions which scale unreasonably with the size of the input.

Instructions

1. 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;
    - return lists of strings corresponding to the format of the human-readable version of the calendar.

2. **Keep all of your code in one file for this assignment**. Everything you submit for credit must be in *process_cal4.py*. We will use our copy of *tester4.py* when evaluating student submissions.

3. 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".

4. For this assignment you can assume all test inputs will be well-formed (i.e., our markers will not test your submission for handling of input or for arguments containing errors).

What you must submit

- Submit a single Python source file named *"process_cal4.py"* to the Assignment 4 page in Brightspace. Brightspace is the only acceptable way of submission.

Evaluation

To determine a grade for your submission, markers will make a qualitative assessment and will look at how many and which tests pass and whether the assignments requirements in these instructions are followed

- "A" grade: This is a good solution that follows assignment requirements, is well-structured and very clearly written and does not have any overall quality issues. All tests pass and therefore no extraneous output is produced. However, it lacks the programming and logical skill to be an outstanding solution. Outstanding solutions will be given an A+ and solutions with minor issues will be given an A-.

- "B" grade: This is a submission that passes all the tests, but has significant quality issues. Depending on the amount of qualitative issues, the marker may give a B+, B or B- grade. A solution that passes all tests, but contains all code in one or two large functions will not be given a grade better than a B.

- "C" grade: This is a submission that represents a proper effort. It meets most of the assignment requirements, but fails some tests. Depending on the number of tests passed and which tests pass and a qualitative assessment, a grade of C or C+ is given.
    - A submission that does not use regular expressions cannot get a grade higher than C
    - Submissions that use global variables cannot get a grade higher than C+

- "D" grade: A serious attempt at completing requirements for the assignment. The submitted solution runs with quite a few problems. Some non-trivial tests pass.

- "F" grade:

    - Plagiarized or copied code, 0 marks
    - No submission, 0 marks

- Submission does not run, 0 marks
- Submissions that fail all tests and show very poor to no effort (as evaluated by the marker), 0 marks
- User-defined classes not used, 0 marks
- Using the class to simply call your Assignment 2 solution, 0 marks
- Submissions that fail all tests but represent a sincere effort (as evaluated by the marker) may be given a few marks