CS 1110 Spring 2021, Assignment 1: Zoom or Room?

Updates to Assignment 1

The assignment itself, with corrections marked in orange, begins on the next page. On this "page 0", we also document the time, location, and nature of the updates, in reverse chronological order,

Updates:

- Friday Feb 26, 3:30pm: Typo on pg. 8: Function ~~`test_from_to()`~~ `test_tag_endi()` contains *at least* two bad test cases.

CS 1110 Spring 2021, Assignment 1: Write your own CourseGrab (Get open/closed/waitlist status from the registrar's live webpages)*

http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment1/a1.pdfhttp://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment1/a1.pdf

Navigating links in this pdf. Text in any shade of blue in this document is a clickable link.

Updates. Monitor announcements on Canvas[1].[2] (Non-Canvas access at `http://www.cs.cornell.edu/courses/cs1110/2021sp/announcements/archive.html`.)

Contents

1 Rules

1.1 Working with a Partner (You Can Have At Most One)

You may work alone or with exactly one other person.

If you are partnering, **the two of you must officially form a group on CMS BEFORE submitting. which will link your submission "portals"**. More details in Section 2.

Atom has a "Teletype" feature[3] enabling real-time collaboration. Staff members Jude Javillo, Jonathan Su, and Emily Parker recommend it for its "Google Docs"-like feel, but observe "one thing to note is that the person who

*Authors: Lillian Lee, with some instructions derived from previous assignments by Walker White and David Gries and formatting initially created by Stephen McDowell. Title idea from Aliva Das.

[1] `https://canvas.cornell.edu/courses/25213/announcements`

[2] Throughout, we include both footnotes and clickable links because the clickable links may not work for all readers. The URL in the footnotes can be copy-pasted into a browser as a last resort.

[3] `https://teletype.atom.io/`

shares the link for teletype will have the updated version but their partner won't be able to save it". Staff member Ben Rosenberg also recommends repl.it[4] .

If your partnership dissolves, see our Collaboration Policies' item on the "group divorce scenario[5]".

1.2 What Collaborations Are (Dis-)Allowed And How To Document Them

The full collaboration policy is on the course Academic Integrity page[6]. **Read it.**

As our advice on working with a partner[7] says, "We strongly advise against splitting the work; rather, the two of you should work together on all parts, or you, personally, won't get all the practice the assignment is meant to provide."

1.3 Python You Are NOT Allowed To Use In This Assignment

You may not use, and do not need, any Python constructs not yet covered by the labs, lectures, or posted lecture slides by the time this assignment was released.[8] We want you to demonstrate your skills with the Python we have taught so far.

2 Timeline and Deadlines

1. If you are partnering: well before Friday, Mar 5, follow our "How to form a group on CMS" instructions[9]. Both parties need to act on CMS[10] in order for the grouping to take effect.

   Once partnered on CMS, only one of you need submit on the partnership's behalf, but you can both submit multiple times. Whichever of you submits the latest before the deadline, that last submission will be what we grade for your group.

2. By 2pm Ithaca time on Friday, Mar 5, submit whatever you have done on `a1_first.py` and `policy_acknowledgment.py` to CMS[11]. Then, do steps 2 and 3 of our "Updating, verifying, and documenting assignment submission" instructions. It is OK if you haven't finished yet; CMS lets you update submissions until the final deadline.

3. By 11:59pm Ithaca time on Friday, Mar 5, make your final submission of `a1_first.py` and `policy_acknowledgment.py`, and save the prove-you-submitted screenshots.

4. Sometime on Saturday, Mar 6, we will transfer the CMS groupings you made in Deadline 1 to a "new" CMS assignment for `a1_second.py`, and open it for submission. (This implies that you cannot change partners for `a1_second.py`.)

5. By 2pm Ithaca time on Monday, Mar 8, submit whatever you have done on `a1_second.py` to CMS, and save the prove-you-submitted screenshots. It is OK if you haven't finished yet; CMS lets you update submissions until the final deadline.

6. By 11:59pm Ithaca time on Monday, Mar 8, make your final submission of `a1_second.py`, and save the prove-you-submitted screenshots.

The 2pm checkpoints on Friday, Mar 5 and Monday, Mar 8 provide you a chance to alert us during business hours of any submission problems. **Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto CMS on time** for whatever reason, including server delays stemming from many other students trying to submit at the same time as you.[12]

Of course, if some special circumstances arise, contact the instructor(s) immediately.

[4]https://repl.it/
[5]https://bit.ly/3sxzJdV
[6]http://www.cs.cornell.edu/courses/cs1110/2021sp/policies/cs1110integrity.html
[7]https://www.cs.cornell.edu/courses/cs1110/2021sp/resources/doing-assignments.html
[8]So, no `if`s, no loops, etc., even if for some reason you know what those are.
[9]https://bit.ly/3uyZDQr
[10]https://cmsx.cs.cornell.edu/web/auth/
[11]https://cmsx.cs.cornell.edu/web/auth/
[12]There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy.

2.1 Can we revise in response to grader feedback?

Stay focused on hitting the deadlines listed above. But yes, as long as you submit something for each of the three files by the given deadlines, you will have the chance to revise and resubmit, possibly multiple times! We'll talk more about this later.

3 Task Overview: Extracting Roster Information

When you look at a Cornell course roster page such as `https://classes.cornell.edu/browse/roster/SP21/class/CS/1110`, depicted in Figure 1, you can see which of the course's lectures and sections are open, closed, waitlisted, and so on; and, as of the pandemic, course listings now include modality, like "online" or "in person". It would be nice to have a more efficient way to look up such information and a more compact presentation.

In this assignment, you'll write functions that will plug into a program we wrote. When done, you'll be able to run that program, `query_roster.py`, on live Cornell Spring 2021 roster pages to have interactions like what's shown in Figure 2. (So when you're finished with A1, compare what `query_roster.py` outputs for you against what you see in this figure.)

The key to this process is that many webpages are really just big collections of special strings your browser displays using formatting information in those strings. You can view the underlying string for a given webpage by using the "view source" functionality of your browser.[13]

Figure 3 shows an excerpt of the source (underlying string) for the CS 1110 roster webpage. Green boxes highlight the locations of the information needed by `query_roster.py`: the "component" (lecture vs. section) is in the `data-ssr-component` box, the lecture/section number is in the `data-section` box, and so on.

3.1 The files you need

Create a new directory on your computer. Download and unzip into that directory this zip file.The contents are:

`policy_acknowledgment.py`
`a1_first.py`
`a1_second.py`
`query_roster.py`
`cornellasserts.py`
`roster_pages`, a folder of some downloaded roster webpages, as html strings.

We've written the entire program `query_roster.py` for you! But it won't work as expected yet, because it makes calls functions in file `a1_second.py` that are currently just "skeletons": just the function-definition headers and docstring specifications you'll need — **don't change those** — plus function bodies containing only the do-nothing command `pass`.

We've also given you a partially-completed testing file `a1_first.py` **It has some purposely erroneous test cases in it**; more on that later.

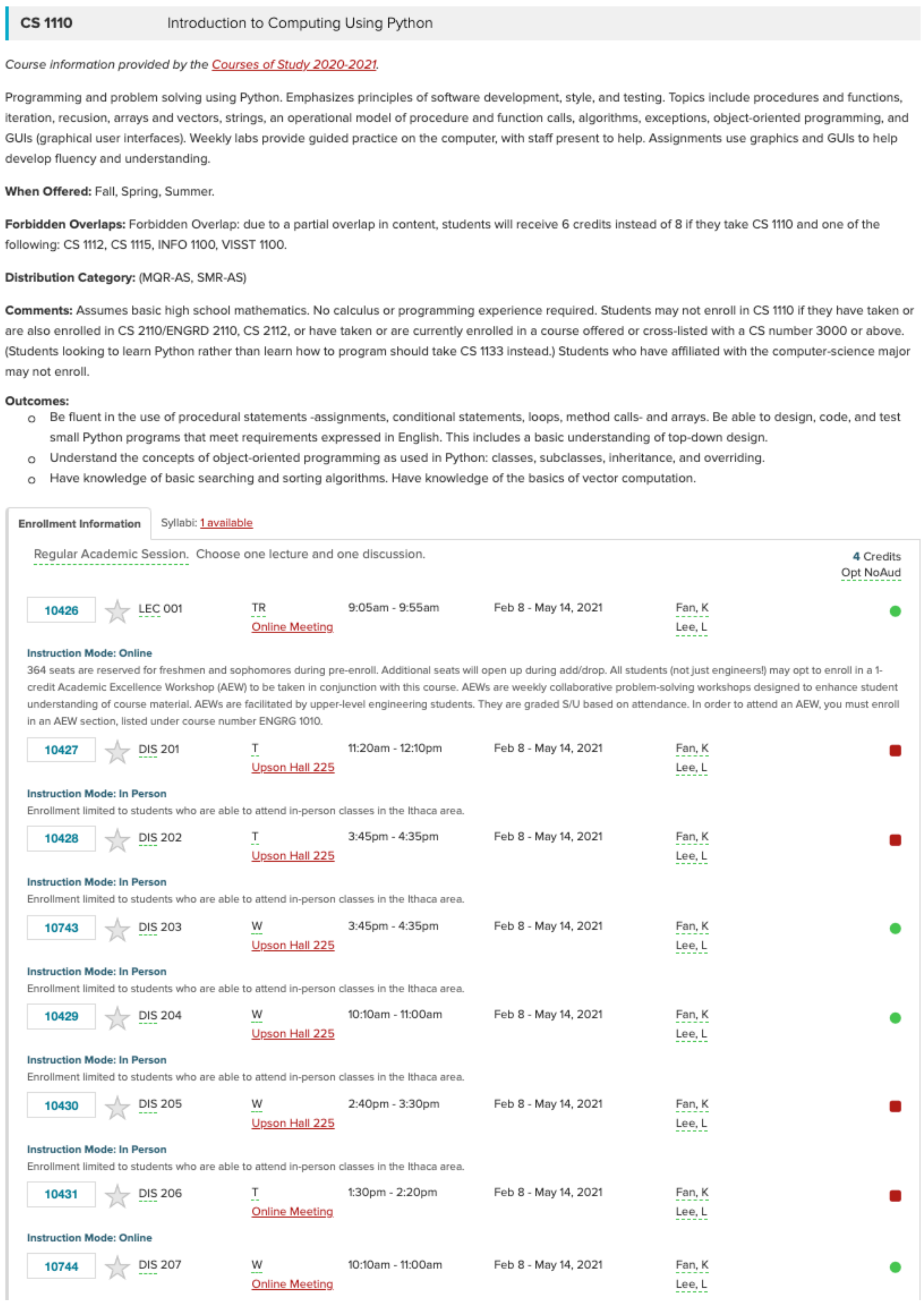
3.2 Desired/required function-call structure

- The program in file `query_roster.py` repeatedly calls `make_url()` and `report_section()` in module `a1_second`.
- Function `from_to` in `a1_second.py` should/must call "helper" function `tag_endi()`, potentially multiple times.[14]
- Function `report_section()` in `a1_second.py` should/must call "helper" function `from_to()`, explicitly or implicitly, potentially multiple times.
- The testing functions in `a1_first.py` should each call the corresponding functions from `a1_second.py` multiple times.

You are allowed to write your own helper functions, but if you do, you must (a) provide clear specification docstrings for them, and (b) provide adequate testing for them in `a1_first.py` .

[13]Chrome: View ≫ Developer ≫ View Source. Firefox: Tools ≫ Web Developer ≫ Page Source. Safari: Develop ≫ Show Page Source. Or, right-click or ctrl-click in the browser window often brings up a menu with an option to view page source.

[14]Such calls can be implicit if you use another helper function you write that calls `from_to()` .



CS 1110 Introduction to Computing Using Python

Course information provided by the Courses of Study 2020-2021.

Programming and problem solving using Python. Emphasizes principles of software development, style, and testing. Topics include procedures and functions, iteration, recusion, arrays and vectors, strings, an operational model of procedure and function calls, algorithms, exceptions, object-oriented programming, and GUIs (graphical user interfaces). Weekly labs provide guided practice on the computer, with staff present to help. Assignments use graphics and GUIs to help develop fluency and understanding.

When Offered: Fall, Spring, Summer.

Forbidden Overlaps: Forbidden Overlap: due to a partial overlap in content, students will receive 6 credits instead of 8 if they take CS 1110 and one of the following: CS 1112, CS 1115, INFO 1100, VISST 1100.

Distribution Category: (MQR-AS, SMR-AS)

Comments: Assumes basic high school mathematics. No calculus or programming experience required. Students may not enroll in CS 1110 if they have taken or are also enrolled in CS 2110/ENGRD 2110, CS 2112, or have taken or are currently enrolled in a course offered or cross-listed with a CS number 3000 or above. (Students looking to learn Python rather than learn how to program should take CS 1133 instead.) Students who have affiliated with the computer-science major may not enroll.

Outcomes:

- Be fluent in the use of procedural statements -assignments, conditional statements, loops, method calls- and arrays. Be able to design, code, and test small Python programs that meet requirements expressed in English. This includes a basic understanding of top-down design.
- Understand the concepts of object-oriented programming as used in Python: classes, subclasses, inheritance, and overriding.
- Have knowledge of basic searching and sorting algorithms. Have knowledge of the basics of vector computation.

Enrollment Information Syllabi: 1 available

Regular Academic Session. Choose one lecture and one discussion. **4** Credits Opt NoAud

10426	LEC 001	TR Online Meeting	9:05am - 9:55am	Feb 8 - May 14, 2021	Fan, K Lee, L	●

Instruction Mode: Online
364 seats are reserved for freshmen and sophomores during pre-enroll. Additional seats will open up during add/drop. All students (not just engineers!) may opt to enroll in a 1-credit Academic Excellence Workshop (AEW) to be taken in conjunction with this course. AEWs are weekly collaborative problem-solving workshops designed to enhance student understanding of course material. AEWs are facilitated by upper-level engineering students. They are graded S/U based on attendance. In order to attend an AEW, you must enroll in an AEW section, listed under course number ENGRG 1010.

10427	DIS 201	T Upson Hall 225	11:20am - 12:10pm	Feb 8 - May 14, 2021	Fan, K Lee, L	🟥

Instruction Mode: In Person
Enrollment limited to students who are able to attend in-person classes in the Ithaca area.

10428	DIS 202	T Upson Hall 225	3:45pm - 4:35pm	Feb 8 - May 14, 2021	Fan, K Lee, L	🟥

Instruction Mode: In Person
Enrollment limited to students who are able to attend in-person classes in the Ithaca area.

10743	DIS 203	W Upson Hall 225	3:45pm - 4:35pm	Feb 8 - May 14, 2021	Fan, K Lee, L	●

Instruction Mode: In Person
Enrollment limited to students who are able to attend in-person classes in the Ithaca area.

10429	DIS 204	W Upson Hall 225	10:10am - 11:00am	Feb 8 - May 14, 2021	Fan, K Lee, L	●

Instruction Mode: In Person
Enrollment limited to students who are able to attend in-person classes in the Ithaca area.

10430	DIS 205	W Upson Hall 225	2:40pm - 3:30pm	Feb 8 - May 14, 2021	Fan, K Lee, L	🟥

Instruction Mode: In Person
Enrollment limited to students who are able to attend in-person classes in the Ithaca area.

10431	DIS 206	T Online Meeting	1:30pm - 2:20pm	Feb 8 - May 14, 2021	Fan, K Lee, L	🟥

Instruction Mode: Online

10744	DIS 207	W Online Meeting	10:10am - 11:00am	Feb 8 - May 14, 2021	Fan, K Lee, L	●

Figure 1: Screenshot of portion of the roster webpage for CS 1110 as of Feb 2021.

```
[ljl2@ushuaia assignment1] python query_roster.py
Enter "S" or "L".
"S": run on local roster-simulation files on your computer.
  (Limited, but avoids hundreds of CS 1110 students bothering the roster webserver frequently/simultaneously.
  Also useful if you have trouble accessing the Internet.)
"L": go ahead and use the live Cornell roster webpages.
Your choice? L
Enter a subject and course number, separated by a space, to look up in the Spring 2021 roster
(just hit return for "CS 1110").
Or, type "q" to quit:

Here are all components I found:
Lec 001 Open      Online
Dis 201 Closed    In person
Dis 202 Closed    In person
Dis 203 Open      In person
Dis 204 Open      In person
Dis 205 Closed    In person
Dis 206 Closed    Online
Dis 207 Open      Online
Dis 208 Closed    Online
Dis 209 Open      Online
Dis 210 Closed    Online
Dis 211 Closed    Online
Dis 212 Open      Online
Dis 213 Open      Online
Dis 214 Open      Online
Dis 215 Closed    Online
Dis 216 Open      Online
Dis 217 Open      Online
......

Enter another course name, like "CS 1110",  or "q" to quit: INFO 1300

Here are all components I found:
Lec 001 Open      Distance learning-asynchronous
Dis 201 Closed    Online
Dis 202 Open      Online
......

Enter another course name, like "CS 1110",  or "q" to quit: q
[ljl2@ushuaia assignment1]
```

Figure 2: An interaction with our program when the plug-in functions are correctly implemented.

```
class="tooltip-iws" data-toggle="popover" data-content="Add to Favorites"
aria-label="Add to Favorites" href="#"
 data-class-nbr="10426"  data-ssr-component="LEC"  data-section="001" ></a></span></p></li>
<li class="consent">&nbsp; </li><li class="meeting-pattern"><h5 class="hidden">
Meeting Pattern</h5><ul class="meetings
meetings-first"><li class="dates"><span class="pattern"><span
class="pattern-only"><span class="tooltip-iws" data-toggle="popover" data-content="Tues
&amp; Thurs">TR</span></span><time class="time">9:05am - 9:55am</time></span><a
class="facility-search" href="http://www.cornell.edu/about/maps/?q=Virtual#CUmap"
target="_blank" rel="nofollow">Online Meeting</a></li><li class="date-range"> Feb 8 -
May 14, 2021 </li><li class="instructors"><h5 class="hidden">Instructors</h5><p><span
class="tooltip-iws" data-toggle="popover" data-content="Kit-Yee Daisy Fan (kdf4)">Fan,
K</span></p><p><span class="tooltip-iws" data-toggle="popover" data-content="Lillian
Lee (ljl2)">Lee, L</span></p></li></ul></li><li class="open-status"><span
class="tooltip-iws" data-toggle="popover" data-content="Open"> [...]
class="fa fa-circle  open-status-open" ></span></span></li><li class="notes" title="Additional
Information"><h5 class="hidden">Additional Information</h5><p> [...]
 class="instr-mode">Instruction Mode: Online</span> <br> 364 seats are reserved for
freshmen and sophomores during pre-enroll. Additional seats will open up during
add/drop.
[...]
```

Figure 3:   The source (html) string for the CS 1110 roster webpage.

3.3 Your task: the one-line description (full descriptions in section 4 and section 5)

Fix and **complete** files `a1_first.py` and `a1_second.py`, following all directions given as comments starting "`STUDENTS`" in the `.py` files and all directions in this document.

4 Collaboration/Academic Integrity Policy Acknowledgment

Read our Collaboration and Academic Integrity Policies[15]. The aim of this exercise is that you understand points (1)-(4).

Open `policy_acknowledgment.py`. Insert your NetID(s) and the date into the header comments.

Paste into the file, between the first set of three double-quotes and the series of dots, the lines from the collaboration policy starting with "Until all students' . . ." and ending with "acknowledge the course staff".

Change all the pronouns appropriately so that the subjects of all the relevant sentences are in the first person, not the second person (implicit or explicit). In other words, "you" statements and imperatives should be changed to "I" or "we" statements.

Below the series of dots but above the second set of three double-quotes, write down any questions you have about policies on the aforementioned webpage; we would be more than happy to clarify anything you are wondering about! (OK to not have any questions).

Save the file and submit it.

[15] `http://www.cs.cornell.edu/courses/cs1110/2021sp/policies/cs1110integrity.html`

5 Iterative Development (How to Work Through the Assignment)

We said in subsection 3.2 that there are dependencies between the functions you will write. The wisest course of action: write *and test* the basic functions first before moving on to the functions that build on that basis.

Hence, develop and test the functions in `a1_second.py` one at a time, in order. For each function, do the following.

1. **Carefully read the specification for the function.** Backquotes are used to visually distinguish variable names, like this: `` `tag` ``. We don't always use angle brackets the way we do in lecture because html strings often themselves contain angle brackets.

```
def make_url(part1, part2, part3):
    """Returns: new string of the form part1/part2/part3

    Preconditions: part1, part2, and part3 are all nonempty strings.

    The intent, although not a precondition, is:
        part1 is like "https://classes.cornell.edu/browse/roster/SP21/class"
        part2 is like "CS"
        part3 is like "1110"
    and the returned string would be
        https://classes.cornell.edu/browse/roster/SP21/class/CS/1110
    """
```

```
def tag_endi(tag, text):
    """Returns: the index of the end of the first occurrence of `tag` in `text`

    Preconditions:
        `text` [str]: contains at least one instance of `tag`
        `tag`  [str]: length > 0

    Examples:
        tag_endi("+", "ab+c") --> 2
        tag_endi('<name "intro">','faith <name "intro"> hope charity') --> 19
    """
```

```
def from_to(start, end, text):
    """Returns substring of `text` occurring between the 1st occurrence of
    `start` and the first following occurrence of `end`.

    Preconditions:
    `text` [str]: length > 0.
    `start` and `end` [str]: both non-empty and occur in `text`.
    At least one `end` appears after a `start` in `text`.

    Examples:
    from_to('+', '!', '+a+b+c!+4def+5') ---> 'a+b+c'
    from_to('(', ')', 'good job :) good example foo(0) ') --> '0'
    t = '<li style="color:purple">python</li><span>the < is intentional</span>'
    from_to('<span>','</span>', t') ---> "the < is intentional"
    """
```

```
def report_section(s):
    """Returns string of the form
        '<component type> <section number> <open status> <mode>'.
    The component type, mode, and open status should have the first letter
    capitalized, and all other letters lower-case.  The section number should be
    exactly as in s with respect to capitalization, spacing, and so on.

    See a1_first.test_report_section() for examples.

    Preconditions:
    `s` is a string of the following form, where CT, SN, OS, and MODE
    indicates word(s) without quotation marks or angle brackets ("<" or ">"),
    and "..." stands for anything:

        ...data-ssr-component="CT" ... data-section="SN" ...
        open-status-OS" ... class="instr-mode">Instruction Mode: MODE</span> ...

    BUT where the following substrings occur exactly once:
        'data-ssr-component="'
        'data-section="'
        'open-status-'
        'class="instr-mode">Instruction Mode:'
    """
```

2. **Fix the bad test cases in `a1_first.py`.** We've given you a number of test cases, so you have examples to look at, but planted bad ones to ensure you carefully read the given function specifications.

   We guarantee that:

   - there are no (intentional) errors in the "expected answers" in `test_make_url()` or `test_report_section()`.
   - Function ~~`test_from_to()`~~ `test_tag_endi()` contains *at least* two bad test cases.[16]

   Here's how to fix:

   - For a test case where the "expected answer" is wrong, add the comment `# ... STUDENT-FIXED ERROR ...` under the comment that numbers the test case; comment out the incorrect `assert_equals` call; and add the fixed call below, like this:

```
    # 0. first input has multiple spaces in it
    # ... STUDENT-FIXED ERROR ...            <--- added
    result = a1_second.some_wonderful_function('s t a', 'st')
    # cornellasserts.assert_equals('', result)    <--- commented out
    cornellasserts.assert_equals('a', result)     <--- fix added
```

   - For a test case where the situation should not have been tested, add the comment `# ... STUDENT-DELETED CASE ...` under the comment that numbers the test case; add a comment giving your reasoning, and comment out the entire test case, like this:

```
    # 1110. first input is an int
    # ... STUDENT-DELETED CASE ...           <--- added
    # ... REASON: violates precondition      <--- added
    # result = a1_second.brilliant_function(2, 3)     <--- commented out
    # cornellasserts.assert_equals('3', result)    <--- commented out
```

[16]To be clear: maybe there are just two, maybe there are three, maybe there are more. . .

3. **Add missing representative test cases for that function in the appropriate place in `test_tag_endi()` and `test_from_to()`.**[17] You want cases that represent valid inputs, but exhibit different aspects of the problem the function is trying to solve, so that using your suite of test cases can catch different types of bugs.

   For each test case you add, include in a comment a short justification of what the test case represents.

4. **Write the function body in `a1_second.py`.**[18]

5. **Run python on the script `a1_first.py`.**[19] If errors are revealed with the function you're currently working on, fix them and re-test.

6 Grand Finale

If you're convinced that your code is correct, you should be able to run Python on the file `query_roster.py`[20] and reproduce the interaction in Figure 2!

7 Code Cleanup

Before submitting, ensure your code obeys the following.[21]

- Lines are short enough (~80 characters) that horizontal scrolling is not necessary.
- You have indented with spaces, not tabs.
- Functions are separated from each other by at least two blank lines.
- You have removed any debugging `print` statements.
- You have removed all `pass` statements.
- You have removed "instruction" comments, such as "`# IMPLEMENT THIS FUNCTION`".
- If you added any helper functions, these have good docstring specifications and you have put sufficient testing code for your functions in `a1_second.py` .

8 Pre-Submission Checklist and What to Submit

The files to submit to CMS[22] are `policy_acknowledgment.py`, `a1_first.py`, and `a1_second.py`.[23]

Make sure the following are all true before you submit.

1. You've changed the header comments in all files to list the entire set of people and sources that contributed to the code.
2. You (and your partner) have included your NetIDs in the header of all files.
3. The date in the header comments has been changed to when the files were last edited.

[17] We have constructed enough test cases for you for `test_make_url()` and `test_report_section()`, so you don't need to add any more. You're welcome.

[18] Sanity checks:

- If the specification says to return something, you need a `return` statement returning something of the correct type.
- Double-check that if the instructions said to call a certain helper, that you did indeed use that helper function.
- The functions in `query_roster.py` do some string processing, so you may find it useful to look in that file for inspiration/examples. But it is definitely not necessary to do so, and if you do choose to check that file out, don't be intimidated by the Python in there that you don't know (yet).

[19] At the command prompt, *not* the `>>>` Python interactive prompt, enter `python a1_first.py` .

[20] At the command-shell prompt, enter `python query_roster.py` .

[21] These requirements up speed up the process of reading/grading hundreds of files.

[22] `https://cmsx.cs.cornell.edu/web/auth/`

[23] Do not submit any files with the extension/suffix `.pyc`. It will help to set the preferences in your operating system so that extensions always appear.

4. You have set your CMS notifications settings to receive email regarding grade changes, and regarding group invitations.

5. (reminder) If working with a partner, you have grouped on CMS. (One has invited on CMS, and the other has accepted on CMS.)