# CS 1110 Spring 2021, Assignment 5: Looping Back to A3: A Class Act

**Updates:**

- Wed Apr 28, 6:30pm Ithaca time: Added Section 7.4 so as to mention that `sd_utilities_dict.menu_listing()` also needs to be reimplemented. (The skeleton code that was distributed already had a `STUDENTS:` instruction to change that function, so this update doesn't affect those files.)

# CS 1110 Spring 2021, Assignment 5: Looping Back to A3: A Class Act*

## 1 Overview

In this assignment, you'll use our new knowledge about (1) organizing code into classes, (2) dictionaries, and (3) while-loops to consider alterations/improvements to the A3 code. To reduce workload (it's been a rough semester), you'll only implement some of these changes; we've done some for you.

Download the zip file of the files you will need: http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment5/a5_skeleton.zip.

## Contents



## 2 Not-that-new Rules

1. A major goal of this assignment is to practice implementation with while-loops and dictionaries. Hence, we reserve the right to assign no credit for code that isn't fundamentally based on the way we ask you to implement it, even if the code fulfills the specification.

2. You may not use Python concepts not introduced in lecture or the corresponding materials by the time of this assignment's release.

## 3 Previous Rules That Still Apply. Remember To Acknowledge All Sources Of Help, Allowed Or Not.

Except you do not need to acknowledge the course staff (although it's not bad if you do).

*Authors: Lillian Lee, who was unable to think of a joke involving dictionaries for this assignment's title. I did once catch a friend out with the ol' "Did you know the word 'gullible' isn't in the dictionary?" gambit, though.

See Sections 1.1-1.3 of Assignment 1[1], Sections 3.1-3.2 of Assignment 2[2], and Section 2 item 3 of Assignment 3[3].

# 4 Timeline and Deadlines

(a) If you are partnering:[4] do so well *before* submitting.

(b) By 2pm Ithaca time on Wed May 5, submit whatever you have done at that point on to CMS.[5]

(c) By **11:59pm (Ithaca time) on Wed May 5**, make your final submission.[6]

# 5 Testing

All the code you'll write in this assignment should have the same functionality as what you wrote in A3. So, you only need to check that running `showdown_time_a5.py` produces the same results as running A3's `showdown_time.py`.

# 6 Tasks

## 6.1 Consider changing some non-method functions to methods.

Back during A3, you hadn't learned about methods, so A3's function `showdown_time.showdown_time()` did setup work for Showdown objects that should really be in the Showdown class's `__init__()` method. A5's code corrects this: see the body of `college_a5.Showdown.__init__()` and note that `showdown_time_a5.py` is more compact because it doesn't need to define a `showdown_time()` function.

Another change in the A5 code is that two non-method functions, `add_student()` and `was_accepted()`, have been converted to methods of class `college_a5.College`, and calls to them have been updated in all files.

But if you examine all the A5 files, you'll see that no other functions were converted to methods.

File `answers_to_code_design_qs.py` asks some questions about these design choices. **Answer those questions, placing your responses in the indicated places in the file.**

## 6.2 Use while-loops instead of for-loops where appropriate.

Function `process_line()` in both file sd_utilities_list.py and sd_utilities_dict.py does important data extraction to pull college data from a file. Here is its specification, and an implementation from A3 that we've updated[7] in A5 file sd_utilities_list.py.

---

[1] https://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment1/a1.pdf

[2] https://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment2/a2.pdf

[3] https://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment3/a3.pdf

[4] Reminder: Both parties need to act on CMS in order for the grouping to take effect. See the "How to form a group" instructions at https://www.cs.cornell.edu/courses/cs1110/2021sp/resources/cms.html .

[5] It is OK if you haven't finished the assignment yet; the 2pm checkpoint provides you a chance to alert us if any problems arise, and us to alert you if your submission seems to be missing and of the deadline that day. 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. There are no so-called "slipdays" and there is no "you get to submit late at the price of a late penalty" policy. Of course, if some special circumstances arise, contact the instructor(s) immediately.

[6] And, as usual, perform steps 1-3 in the "Updating, verifying, and documenting assignment submission" section of https://www.cs.cornell.edu/courses/cs1110/2021sp/resources/cms.html .

[7] The change: adjust for `add_student` being changed to a College method.

```
1  def process_line(sline, clist):
2      """ Add the tag of sline to the appropriate list (accepted_enrolled,
3      ..., or waitlist) of each College in clist; if sline names a college
4      not already represented in clist, a corresponding College is added
5      to clist.
6
7      Preconditions: sline is a string formatted as if it were a line from a data file
8      described in the CS1110 Spring 2021 A3 handout.
9
10     sline does not have trailing newlines.
11
12     clist is a list of Colleges, possibly empty."""
13
14     tag = int(sline[:sline.index(">>")].strip())
15
16     # name of college student enrolled at
17     senrolled_cn = sline[sline.rindex(":")+1:].strip()
18
19     outcomes = sline[sline.index(">>") + len(">>"):sline.rindex('##')].split("##")
20
21     for outcome in outcomes:
22         oc = outcome.split(":")
23         scname= oc[0].strip() # name of a college supplied by student
24         decision = oc[1].strip()
25         found_scname = False # have not found scname in clist yet
26         found_enrolled = False # have not found enrolled yet (for non-college options)
27         for c in clist:
28             if c.name == scname:
29                 found_scname = True
30                 # place tag in correct status list for c
31                 c.add_student(tag, decision, senrolled_cn)
32             if c.name == senrolled_cn:
33                 found_enrolled = True
34         if not found_scname:
35             new_c = college_module.College(scname)
36             new_c.add_student(tag, decision, senrolled_cn)
37             clist.append(new_c)
38             if senrolled_cn == scname:
39                 # now the enrolled school is in clist
40                 found_enrolled = True
41     if not found_enrolled:
42         # apparently "enrolled" in something they didn't apply to, like "gap year"
43         new_c = college_module.College(senrolled_cn)
44         new_c.add_student(tag, decision, senrolled_cn)
45         clist.append(new_c)
```

*Read the code above carefully to determine how it works.* The text below assumes you understand the code well.

The for-loop in lines 27-33 does unnecessary work, because it always continues through the entire list `clist` even if a College with name `scname` and a College with name `senrolled_cn` are found "early" in the list. Instead, while-loops are a good mechanism for "early termination" of a loop — where the loop stops as soon as some target has been found or a certain condition has been met.

Rewrite the body of `sd_utilities_list.process_line()` as directed in the file to replace the for-loop over `clist` with one or more while-loops that don't go over unnecessary parts of `clist`.[8]

[8] On my computer, there's a slight but noticeable speed-up compared to the A3 implementation.

#### 6.2.1 Answer the while-loop question in `answers_to_code_design_qs.py`.

#### 6.2.2 Optional challenge (just for those interested, no extra credit)

Move as much of the computation out of the `for outcome in outcomes` for-loop as possible, even if this means more than one while-loop. Efficiency is best served by having as little nesting of loops as possible.

**Hint:** the fact that lines 32-33, which are checking for the College that a student enrolled at, are inside the `for outcome in outcomes` for-loop is an inelegant hack done for the sake of efficiency:[9] since line 27 is already going through all of `clist` anyway, I tucked the search for the enrolled college in `clist` there.

# 7 Rewrite the showdown utilities to use dictionaries instead of lists

Dictionaries of Colleges, where the keys are College names, turn out to be more convenient than lists of Colleges for the task A3 addresses.[10] To prove this to yourself, we've provided you file sd_utilities_dict.py, which is where you'll write dictionary-based re-implementations of some functions in file sd_utilities_list.py. We already wrote `sd_utilities_dict.colleges_from_file()` for you, to serve as a model.

## 7.1 Swap in module sd_utilities_dict for sd_utilities_list in showdown_time_a5.py

To make sure that showdown_time_a5.py uses your dictionary-based code rather than your list code, follow the instruction comment in line 17 of that file to `import sd_utilities_dict as sdu` instead of module sd_utilities_list.

## 7.2 Finish a dictionary-based implementation of `sd_utilities_dict.process_line()`

One advantage you should find: you shouldn't need any nested loops at all!

## 7.3 Implement `sd_utilities_dict.college_named()`

This implementation also becomes simpler and loopless.[11]

## 7.4 Revise the implementation of `sd_utilities_dict.menu_listing()`

Hint: there are at least two ways to do this; try to think of one that is simpler (so that the conversion to a dictionary is an improvement, not a degradation, of the code).

[9] Conceptually, what college the student enrolled at has nothing to do with all the other colleges the student applied to.

[10] Indeed, dictionaries are so useful for many purposes that you might have noticed that we already used them in the code we gave you for A3 and A4. Take a look if you're interested in more examples of using dictionaries.

[11] Indeed, with dictionaries, one might not even bother defining function `college_named()` at all.