 CS 1110 Spring 2021, Assignment 3: Harvard Beats Yale, 29-29*

Besides providing practice with objects, lists, and for-loops, this assignment gives you experience with larger code bodies and more complexly inter-relating object structures. The most novel aspect of the work will probably not be the writing of the functions (novel though that may be), but understanding these structures.

You will not be submitting formal test cases (though you should do as much testing as you can). We supply you with some testing files and sample outputs, and have an intermediate submission deadline that involves you working out the desired output for a simple example.

Download the zip file of the files you will need: http://www.cs.cornell.edu/courses/cs1110/2021sp/assignments/assignment3/a3_skeleton.zip.

1 Exploring a Real-World College-Decisions Dataset

1.1 Motivation

Most of you were probably interested in the college application and admission process at some point. We can explore the outcomes of the 2019-2020 college-admissions cycle by looking at real survey data self-submitted by college applicants that is available[1] at a post to the subreddit r/ApplyingToCollege[2]. While this data has the usual issues of (significant) sample bias, noisy data entry, and so on, we hope you find it amusing and/or a challenge to your developing programming skills to complete a program that answers the following type of "showdown"[3] question:

> According to (biased and noisy and just from the year 2020 but still real-world) data, when a student gets into both college X and college Y, which are they more likely to go to?

Objects, lists, for-loops, and dictionaries are great tools for this task.

Contents

*Authors: Lillian Lee

[1] Sharing policy: "You may share and adapt this dataset if you give credit 'r/ApplyingToCollege/' [sic] and do not use the dataset for any commercial purposes (CC BY-SA 4.0)"

[2] https://www.reddit.com/r/ApplyingToCollege/comments/gucq7r/ra2c_2020_census_results_class_of_2024/

[3] Thanks to consultant Aliva Das for coming up with this terminology.

2 New Rules

1. A major goal of this assignment is practice with for-loops. Hence, for each of the functions we have you implement, *we reserve the right to assign no credit for code that isn't fundamentally based on an explicit for-loop if we asked for one, or that doesn't use the kind of for-loop in the way we ask for*, 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.

   This implies you may *not* use dictionaries in your code in this assignment.[4]

3. Unless the specification says otherwise, any function you complete must not change any objects given as arguments. *Changing a user's objects without notifying them is bad practice.*[5]

3 Previous Rules That Still Apply

See Sections 1.1-1.3 of Assignment 1[6] and Sections 3.1-3.2 of Assignment 2[7].

4 Timeline and Deadlines

(a) If you are partnering:[8] **do so *before* any submissions**, where **the first checkpoint is 2pm Ithaca time on Wed Mar 24**.

(b) By 2pm Ithaca time on Wed Mar 24, submit whatever you have done on `answer_to_testcase_q.txt` to CMS.[9] It is OK if you haven't finished working on it yet.[10]

(c) By **11:59pm (Ithaca time) on Wed Mar 24**, you **must** have made a first submission of `answer_to_testcase_q.txt` in order to be allowed to submit the other A3 files.[11] Again perform the aforementioned steps 1-3. *You don't have to have submitted any of the other files at that point.* Whether or not you make this deadline will be a small factor in your grade for this assignment.

    Groups cannot be formed or changed after submitting.[12]

(d) Some time the next day, we will change the due date on CMS to Sun Mar 28.

(e) By 2pm Ithaca time on Sun Mar 28, submit whatever you have done at that point on *all* A3 files to CMS, and perform the aforementioned steps 1-3. It is OK if you haven't finished them yet.

---

[4]Indeed, iteration over lists is not as Python-specific as the use of dictionaries is, so we really want you to practice with lists as much as possible in this assignment, as the more generic data type. We do acknowledge that dictionaries are useful for our task, and applied them ourselves in the code we provided you; but we want students to have all the concepts they need for A3 at release time.

[5]One might go as far as to say it's . . . objectionable?

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

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

[8]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` .

[9]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` .

[10]The 2pm checkpoints provide you a chance to alert us if any problems arise. 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.

[11]We want you to have worked out a testcase ahead of time.

[12]Except for "group divorce" situations; see the course Policies page.

(f) By **11:59pm (Ithaca time) on Sun Mar 28**, make your final submission of all files, and perform the aforementioned steps 1-3.

(g) We will release solutions by Monday the 29th, since prelim 1 is on Tuesday the 30th.

5 Task

5.1 Goal

When you complete the code skeletons, you'll be able to run Python script `showdown_time.py` to choose a colleges datafile and query it about pairs of colleges.

We've provided sample outputs in the files in directory `sample_runs`, and you should consider these as representing test cases: does your code reproduce that output?

For example, here are the some of the contents of `real_data_output.txt`.[13] (We'll explain the first line later.) The colleges are in reverse order of number of applicants in the data.

```
# small_test is set to False, show_sd_internals set to True


[ljl2@utopia solution] python showdown_time.py
What college-info file in directory "data" should I use?
1: small_test1.txt
2: small_cornell_and_suny_test.txt
3: a2c_census2020_processed.txt
other: some other file in the "data" directory
Default is 1.

Your choice? 3
Here are the available college_names according to your menu_names() function.
0: UC-Los_Angeles   1: UC-Berkeley  2: Cornell_U    3: Stanford_U   4: Harvard_U    5: Yale_U   6: UC-San_Diego
[...]
123: Howard_U   124: USouth_Carolina    125: Smith_College  126: Pitzer_College 127: Skidmore_College
128: Bryn_Mawr_College
[...]
204: Wheaton_College    205: United_States_Military_Academy_at_West_Point   206: UArkansas  207: Brigham_Young_U
[...]
229: A_college_or_university_outside_of_the_United_States
230: None;_I_am_taking_a_gap_year/semester_and_will_enroll_in_college_in_2021.
231: A_college_or_university_not_listed.    232: None;_I_do_not_have_plans_to_attend_college_as_of_now.
233: Community_College  234: Liberty_U
Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 2 4

FYI, here is the internal showdown data
Cornell_U vs. Harvard_U
accepted at both: 383,404,453,548,888,1464,1731,1896,1902,1952,2665,1903
accepted at both, enrolled at Cornell_U: 1903
accepted at both, enrolled at Harvard_U: 383,404,453,548,888,1731,1902,1952,2665

Cornell_U vs. Harvard_U
Number of students accepted at both: 12
Number of such students who enrolled at one of them: 10
% who chose Cornell_U over Harvard_U: 10.0
% who chose Harvard_U over Cornell_U: 90.0

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 20 4

FYI, here is the internal showdown data
Massachusetts_Institute_of_Technology vs. Harvard_U
accepted at both: 128,221,404,453,472,888,1665,1801,1842,2189,2399,2901,1707,1896,2244,2375
accepted at both, enrolled at Massachusetts_Institute_of_Technology: 1707,1896,2244,2375
```

[13] Except we've had to do some manual line-wrapping.

```
accepted at both, enrolled at Harvard_U: 128,221,404,453,472,888,1665,1801,1842,2399

Massachusetts_Institute_of_Technology vs. Harvard_U
Number of students accepted at both: 16
Number of such students who enrolled at one of them: 14
% who chose Massachusetts_Institute_of_Technology over Harvard_U: 28.57
% who chose Harvard_U over Massachusetts_Institute_of_Technology: 71.43

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 3 4

FYI, here is the internal showdown data
Stanford_U vs. Harvard_U
accepted at both:
↪ 670,678,771,956,1174,1358,1616,1665,1801,1902,1903,2244,2375,2665,2711,52,441,1579,1762,1771,2901
accepted at both, enrolled at Stanford_U: 52,441,1579,1762,1771,2901
accepted at both, enrolled at Harvard_U: 670,678,771,956,1174,1358,1616,1665,1801,1902,2665,2711

Stanford_U vs. Harvard_U
Number of students accepted at both: 21
Number of such students who enrolled at one of them: 18
% who chose Stanford_U over Harvard_U: 33.33
% who chose Harvard_U over Stanford_U: 66.67

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 0 6

FYI, here is the internal showdown data
UC-Los_Angeles vs. UC-San_Diego
accepted at both:
↪ 15,47,71,77,114,133,138,141,151,160,171,185,193,195,197,199,202,208,210,218,249,262,291,324,331,395,406,410,427,447
accepted at both, enrolled at UC-Los_Angeles:
↪ 51,81,277,290,317,341,455,459,538,596,631,675,698,761,777,797,808,855,872,884,911,935,952,973,1016,1035,1036,1043,1
accepted at both, enrolled at UC-San_Diego: 193,1475

UC-Los_Angeles vs. UC-San_Diego
Number of students accepted at both: 237
Number of such students who enrolled at one of them: 76
% who chose UC-Los_Angeles over UC-San_Diego: 97.37
% who chose UC-San_Diego over UC-Los_Angeles: 2.63

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 8 18

FYI, here is the internal showdown data
Columbia_U vs. New_York_U
accepted at both:
↪ 191,435,506,548,612,781,1174,1579,1670,2031,2232,2572,2699,2762,2763,2975,3015,530,800,1044,2462
accepted at both, enrolled at Columbia_U: 530,800,1044,2462
accepted at both, enrolled at New_York_U: 781

Columbia_U vs. New_York_U
Number of students accepted at both: 21
Number of such students who enrolled at one of them: 5
% who chose Columbia_U over New_York_U: 80.0
% who chose New_York_U over Columbia_U: 20.0

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): 11
↪ 74

FYI, here is the internal showdown data
UMichigan-Ann_Arbor vs. The_Ohio_State_U
accepted at both: 214,216,1059,1315,1995,2002,2131,2381,2564,3031,753,2012,2067,2158
accepted at both, enrolled at UMichigan-Ann_Arbor: 753,2012,2067,2158
accepted at both, enrolled at The_Ohio_State_U:
```

```
UMichigan-Ann_Arbor vs. The_Ohio_State_U
Number of students accepted at both: 14
Number of such students who enrolled at one of them: 4
% who chose UMichigan-Ann_Arbor over The_Ohio_State_U: 100.0
% who chose The_Ohio_State_U over UMichigan-Ann_Arbor: 0.0

One more round? Pick two numbers from the list of college names, separated by spaces (or 'q' to quit): q
Bye for now!
```

5.2 College-Data Files

5.2.1 Format

In the files supplying our data, each line represents the application choices and outcomes for a student.[14] Sample:

```
3  >> Boston_U : Accepted ## Northeastern_U : Accepted ## Northwestern_U : Accepted ## enr : Northwestern_U
```

The number before the `>>` is the *tag* for the student. Each `##` marker is preceded by a colon-separated pair consisting of a college name and an outcome. After the last `##` is the string "enr" and the college enrolled at. So, in the example above, student with tag 3 got into all three schools they applied to, and enrolled at Northeastern_U.[15]

Other outcomes in the data are "Rejected" or "Wait-listed".[16]

Assume there are no repeated tag numbers (although they may not be consecutive or in sorted order), and each student enrolled in only one "college".[17]

5.2.2 Small test case: `small_test1.txt`

For any two colleges X and Y, if you know which students — among those who were accepted to both and chose to enroll at either X or Y — enrolled at X, then you can compute what percent chose X over Y, and what percent chose Y over X.

Check your understanding by computing these percentages for the last few college pairs in the output shown in Section 5.1.

Demonstrate your understanding — and hence work out a test case — of the kind of computations you will need to do by filling out `answer_to_testcase_q.txt`.

That is, here are the complete contents of college-data file `small_test1.txt`:

```
11  >> A : Accepted ## B : Rejected ## D : Accepted ## E : Accepted ## enr : A
10  >> A : Accepted ## B : Rejected ## D : Accepted ## enr : D
20  >> A : Accepted ## B : Rejected ## D: Accepted ## E : Accepted ## enr : A
21  >> A : Wait-listed ## E : Accepted ## enr : E
```

Assuming that file is the data source, open file `answer_to_testcase_q.txt` in Atom and replace the "`???`"s in that file with the correct answers, following the pattern established in Section 5.1.

For reference, here are the relevant parts of `answer_to_testcase_q.txt`:

```
A vs. D
accepted at both: ???
accepted at both, enrolled at A: ???
accepted at both, enrolled at D: ???
A vs. D
Number of students accepted at both: ???
Number of such students who enrolled at one of them: ???
% who chose A over D: ???
% who chose D over A: ???
```

[14] Thanks to CS/IS Professor Jon Kleinberg for providing us with an initial pre-processed version of the A2C 2020 Census results. The numbers for the top-ten enrolled-at colleges according to the pre-processed version are close to but not quite consistent with those given at https://www.reddit.com/r/ApplyingToCollege/comments/gucq7r/ra2c_2020_census_results_class_of_2024/. This may be because of different ways of matching variant spellings of college names, or different ways of handling waitlisted students.

[15] Do note that the data in `a2c_census2020_processed.txt` has not been completely cleaned. But for the purposes of this assignment, don't clean it further. For example, "California_Polytechnic_State_U-San_Luis_**Obsipo**" should really be "California_Polytechnic_State_U-San_Luis_**Obsipo**", but just leave it alone.

[16] We converted "Accepted;_Wait-listed and "Rejected;_Wait-listed" to "Accepted" and "Rejected", respectively. We discarded lines containing an outcome "Accepted;_Rejected".

[17] We treat "A_college_or_university_not_listed", "None;_I_am_taking_a_gap_year/semester_and_will_enroll_in_college_in_2021.", "California_State_U_(All)" and the like as if each were a unique college.

5.3 Specifications And Calling Structure For The Functions You Need To Complete

The most important, top-level function is in file `showdown_time.py`:

```
def showdown_time(c1name, c2name, colleges):
    """Return Showdown object representing the showdown comparison between the
    College with name `c1name` and the College wih name c2name from data source
    `colleges`.

    Returns None if `colleges` doesn't contain a college named c1name or c2name.
    """
```

Computing showdown information requires knowing which students were accepted to the colleges in question. Hence, you should (and are required) to use, and hence complete, this helper function, in file `college.py`:

```
def was_accepted(tag, c):
    """Returns True if the student with tag number `tag` was accepted at
    College c; returns False otherwise.

    (For A3, a waitlisted student is not considered accepted even if they
    eventually made it off the waitlist.)

    Preconditions:
        tag is an int.
        c is a College object (not None).
    """
```

Also, two of the arguments to the main function are the names of the colleges to compare, given as strings. The following helper function returns the actual College object that a name refers to, so we can look up its data. All the available College objects are assumed to be stored in a list so we can access them. In file `sd_utilities.py`:

```
    name `n`.
    """
    pass # STUDENTS: implement this function. You must make effective use of a
         # for-loop.  You may NOT use dictionaries.




def menu_listing(colleges):
```

Someone who wants to use our main function `showdown_time()` has to specify which colleges to compare. But there are many ways the user could specify a college (e.g., "Cornell" vs. "Cornell University" vs. "CU") as well as misspell a canonical name should there be one. To minimize user data-entry mistakes, we'll present the user with a phone-menu-like listing of the options ("Para Cornell, oprima dos"[18]). In file `sd_utilities.py`:

```
    The numbered-name items are separated by a tab (\t).

    Example: for the list colleges_from_file('small_test1.txt'), the output
    is
        '0: B\t1: A\t2: D\t3: E'
    which prints out as
        0: B    1: A    2: D    3: E

    There is no newline ('\n') or whitespace at the end of the returned string.
```

[18]Thanks to consultants Dylan Castillo and Ben Rosenberg for the Spanish check, and no thanks to Google Translate.

```
    Precondition: `colleges` is a list of Colleges (possibly empty.)
    """
    pass # STUDENTS: implement this. Your solution must be based on a for-loop
         # using the `range()` function.
         # Be careful about not having your output end with a tab
```

It would be reasonable to work on these functions in the following order: `was_accepted()`, `college_named()`, `menu_listing()`, `showdown_time()`; but there are other reasonable orderings, too.

6 Classes College and Showdown

What are these College and Showdown objects? They're used to store the info we need in a convenient way, and file `college.py` defines these classes. We haven't discussed classes in detail yet, but here's all you need to know about these two classes.

A College object has five attributes (see `college.py` for more specifics):

- `name`, like `"Cornell_U"`.
- `accepted_enrolled`: list of tags (which are ints, no repeats) for the applicants who were accepted by this College *and* enrolled there.
- `accepted_not_enrolled`: similar, except its for the applicants who were accepted by this college but enrolled somewhere else.
- `rejected`: similar, but for the rejected applicants.
- `waitlisted`: similar, but for the waitlisted applicants.

A new College object is created with a call like `College("A")`; all the applicant lists start out empty.

Showdown objects represent a comparison between two colleges, and have the following attributes, according to `college.py`:

```
c1: A College object.
c2: A different College object. (Although we don't check that c1 != c2).

accepted_at_both: list of tags of students accepted by both c1 and c2.
enrolled_at_c1: list of tags of students *accepted at both* who enrolled at c1.
enrolled_at_c2: list of tags of students *accepted at both* who enrolled at c2.
```

A Showdown object is created with a call like `Showdown(x, y, xylist, xlist, ylist)`, where `x` and `y` are College objects, and `xylist`, `xlist`, and `ylist` are *meant* to be the correct lists for the new Showdown object's `accepted_at_both`, `enrolled_at_c1`, and `enrolled_at_c2` attributes.

6.1 How to Test

It is fine and expected that you will write `menu_listing()` and then `showdown_time()` *last*, and that you'll test by comparing their output against the outputs we provided you in `sample_runs`.

For testing the other functions and also, to some degree, `menu_listing()`, you can run the Python script `helper_tests.py`.

6.2 Grand Finale

There are two variables in `showdown_time.py` meant to control how it behaves.

```
#STUDENTS: when you're sure your code is done,
# change these variables to False for full functionality!

small_test = True  # Whether to just use colleges 1 (A) and 2 (D)
                   # from small_test1.txt
show_sd_internals = True # Whether to show who was accepted, waitlisted,
                         # etc for the colleges in the showdown
```

When you've completed all the code to your satisfaction, change those lines to set the first variables to `False`. Then, you can run Python on `showdown_time.py` and reproduce the results in the files in `sample_runs`, or do some exploring of the data in `a2c_census2020_processed.txt`. Let us know what you find interesting or "ex/acceptional"!

7 Advice

7.1 Plan your program logic on paper beforehand. (We needed to!)

The idea is to get the logic of "when to put which objects where" straight, and outlined in natural language, straight before trying to translate your logic into Python.

First advantage: you can separate out logical mistakes from programming errors.

Second advantage: we course staff will be able to help you significantly faster if we can first see your human-language outline of your plan first.

7.2 Avoiding bugs; debugging hints

1. Many bugs are caused by unintentionally changing the semantics of a variable. Pick informative variable names and/or comment what your intents are. Make sure you update variable values correctly when the situation changes.

2. Section 13.10 of the text ("...especially if you are working on a hard bug") is good advice.

3. Only implement a little bit at a time and test incessantly. It's normal to let many tests fail for code you haven't implemented yet, as long as what you *are* working on is getting closer to functioning.[19]

   You don't want an uncaught bug early one messing up a lot of things downstream.

   Add temporary print statement to check your partial progress as necessary.

4. You may be able to use Python Tutor to visualize what your code is doing.

7.3 Navigating complex files

1. Atom lets you "fold up" parts of code, such as function bodies, to temporarily hide them. Look for a little down-pointing arrow-head in the lefthand "gutter" of a code window, and click on it.

7.4 Worked examples of for-loops and object manipulation

For inspiration and models, besides the lecture materials, there are solved A3s from previous semesters in the "Archive" section of our assignment advice and archive page[20] and solutions to previous exam questions at our exams page[21]

Function `add_student()` that we completed for you `college.py` is an example of handling College objects.

[19](Ha.)
[20]https://www.cs.cornell.edu/courses/cs1110/2021sp/resources/doing-assignments.html
[21]https://www.cs.cornell.edu/courses/cs1110/2021sp/exams

A For `a2c_census2020_processed.txt`: Alphabetical Mapping of Colleges to Menu Numbers

We provide this alphabetized list of colleges from `a2c_census2020_processed.txt` together with their menu numbers to make it easier to look up colleges of interest. For instance, you can easily see that to compare Harvard against Harvey Mudd, you should enter the menu-item numbers 4 and 88 when running `showdown_time.py`. Cornell is number 3. (Meaning it was the 3rd most-applied-to college!)

But: warning: it is possible that your computer might assign different menu numbers than what is listed here, as ties in number of applications might be broken differently than the way Prof. Lee's did.

```
231: A_college_or_university_not_listed.
229: A_college_or_university_outside_of_the_United_States
67: American_U
49: Amherst_College
68: Arizona_State_U
169: Auburn_U
147: Babson_College
73: Barnard_College
121: Bates_College
118: Baylor_U
193: Bentley_U
228: Bob_Jones_U
42: Boston_College
26: Boston_U
69: Bowdoin_College
116: Brandeis_U
207: Brigham_Young_U
9: Brown_U
128: Bryn_Mawr_College
138: Bucknell_U
70: Cal_Poly_Pomona
47: California_Institute_of_Technology
32: California_Polytechnic_State_U-San_Luis_Obsipo
65: California_State_U_(All)
99: Carleton_College
23: Carnegie_Mellon_U
35: Case_Western_Reserve_U
142: Chapman_U
227: Claflin_Unversity
87: Claremont_McKenna_College
192: Clarkson_U
145: Clemson_U
63: Colby_College
110: Colgate_U
208: College_of_Charleston
157: College_of_the_Holy_Cross
167: Colorado_College
109: Colorado_School_of_Mines
150: Colorado_State_U
8: Columbia_U
233: Community_College
2: Cornell_U
27: Dartmouth_College
141: Davidson_College
75: Drexel_U
13: Duke_U
174: Elon_U
209: Embry_Riddle
170: Emerson_College
30: Emory_U
183: Fairfield_U
224: Fisk_U
181: Florida_Institute_of_Technology
98: Florida_State_U
43: Fordham_U
144: Franklin_and_Marshall_College
53: George_Washington_U
41: Georgetown_U
19: Georgia_Institute_of_Technology
201: Gettysburg_College
182: Gonzaga_U
78: Grinnell_College
211: Gustavus_Adolphus_College
105: Hamilton_College
4: Harvard_U
88: Harvey_Mudd_College
103: Haverford_College
123: Howard_U
196: Illinois_Institute_of_Technology
86: Indiana_U-Bloomington
159: Iowa_State_U
155: James_Madison_U
28: Johns_Hopkins_U
219: Kansas_State_U
106: Kenyon_College
131: Lafayette_College
93: Lehigh_U
168: Lewis_and_Clark_College
234: Liberty_U
115: Loyola_Marymount_U
130: Macalester_College
20: Massachusetts_Institute_of_Technology
97: McGill_U
119: Miami_U_(OH)
96: Michigan_State_U
80: Middlebury_College
191: Mississippi_State_U
215: Morehouse_College
143: Mount_Holyoke_College
217: New_Mexico_State_U
223: New_Mexico_Tech
18: New_York_U
230: None;_I_am_taking_a_gap_year/semester_and_will_enroll_in_college_i
232: None;_I_do_not_have_plans_to_attend_college_as_of_now.
92: North_Carolina_State_U
12: Northeastern_U
14: Northwestern_U
82: Notre_Dame_U
107: Oberlin_College
114: Occidental_College
166: Ohio_U
212: Oklahoma_State_U
152: Oregon_State_U
50: Penn_State
160: Pepperdine_U
126: Pitzer_College
44: Pomona_College
10: Princeton_U
186: Providence_College
38: Purdue_U
108: Reed_College
79: Rensselaer_Polytechnic_Institute
188: Rhode_Island_School_of_Design
180: Rhodes_College
29: Rice_U
81: Rochester_Institute_of_Technology
161: Rose-Hulman_Institute_of_Technology
48: Rutgers_U
```

46: SUNYs
175: Saint_Louis_U
66: San_Diego_State_U
85: San_Jose_State_U
76: Santa_Clara_U
214: Sarah_Lawrence_College
122: Scripps_College
177: Seattle_U
225: Sewanee
127: Skidmore_College
125: Smith_College
156: Southern_Methodist_U
189: Spelman_College
202: St._John's_College
158: St._Olaf_College
3: Stanford_U
95: Stevens_Institute_of_Technology
45: Swarthmore_College
100: Syracuse_U
111: Temple_U
71: Texas_A&M_U
185: Texas_Christian_U
190: Texas_Tech_U
194: The_Cooper_Union_for_the_Advancement_of_Science_and_Art
74: The_Ohio_State_U
112: The_UAlabama
171: Trinity_College
179: Trinity_U
36: Tufts_U
54: Tulane_U
220: Tuskegee_U
203: UAlabama-Birmingham
221: UAlaska_Fairbanks
113: UArizona
206: UArkansas
1: UC-Berkeley
22: UC-Davis
17: UC-Irvine
0: UC-Los_Angeles
137: UC-Merced
104: UC-Riverside
6: UC-San_Diego
16: UC-Santa_Barbara
39: UC-Santa_Cruz
24: UChicago
173: UCincinnati
91: UColorado-Boulder
102: UConnecticut
135: UDelaware
153: UDenver
62: UFlorida
89: UGeorgia
172: UHawaii_at_Manoa
34: UIllinois_at_Urbana-Champaign
139: UIowa
178: UKansas
187: UKentucky
163: UMaine
59: UMaryland-College_Park
57: UMassachusetts-Amherst
84: UMiami
11: UMichigan-Ann_Arbor
52: UMinnesota-Twin_Cities
199: UMissouri-Columbia
216: UMontana
176: UNebraska-Lincoln
195: UNevada-Reno
210: UNew_Hampshire
198: UNew_Mexico
40: UNorth_Carolina_at_Chapel_Hill
226: UNorth_Dakota
60: UNotre_Dame
149: UOklahoma
120: UOregon
7: UPennsylvania
58: UPittsburgh
162: UPortland
197: URhode_Island
129: URichmond
56: URochester
154: USan_Diego
133: USan_Francisco
124: USouth_Carolina
222: USouth_Dakota
15: USouthern_California
151: UTennessee
33: UTexas-Austin
94: UTexas_at_Dallas
140: UUtah
136: UVermont
31: UVirginia
37: UWashington
55: UWisconsin
218: UWyoming
132: U_at_Buffalo
205: United_States_Military_Academy_at_West_Point
148: Uthe_Pacific
21: Vanderbilt_U
83: Vassar_College
90: Villanova_U
146: Virginia_Commonwealth_U
77: Virginia_Tech
117: Wake_Forest_U
134: Washington_&_Lee_U
164: Washington_State_U
25: Washington_U_in_St._Louis
72: Wellesley_College
64: Wesleyan_U
200: West_Virginia_U
204: Wheaton_College
165: Whitman_College
213: Whitworth_U
61: William_&_Mary
51: Williams_College
101: Worcester_Polytechnic_Institute
184: Xavier_U
5: Yale_U