Data 100 Debugging Guide

Yash Dave Lillian Weng

Table of contents

About

This text offers pointers for keyboard shortcuts or common mistakes that accompany the coursework in the Spring 2024 Edition of the UC Berkeley course Data 100: Principles and Techniques of Data Science.

Inspiration for this guide was taken from the UC San Diego course DSC 10: Principles of Data Science and their debugging guide.

If you spot any typos or would like to suggest any changes, please email us at **data100.instructors@berkeley.edu**

1 Jupyter Shortcuts

> **Note**
>
> **Note**: if you're using a MacBook, please replace `ctrl` with `cmd`.

1.1 Shortcuts for Cells

For the following commands, make sure you're in command mode. You can enter this mode by pressing `esc`.

- `a`: create a cell above
- `b`: create a cell below
- `dd`: delete current cell
- `m`: convert a cell to markdown (text cell)
- `y`: convert a cell to code

1.2 Running Cells

For individual cells,

- `ctrl` + `return`: run the current cell
- `shift` + `return`: run the current cell and move to the next cell

To run all cells in a notebook:

- In the menu bar on the left, click `Run`. From here, you have several options. The ones we use most commonly are:
  - `Run All Above Selected Cell`: this runs every cell above the selected cell
  - `Run Selected Cell and All Below`: this runs the selected cell and all cells below
  - `Run All`: this runs every cell in the notebook from top-to-bottom

1.3 Restarting Kernel

In the menu bar on the left, click `Kernel`. From here, you have several options. The ones we use most commonly are:

- `Restart Kernel...`
- `Restart Kernel and Run up to Selected Cell`
- `Restart Kernel and Run All Cells`

2 Datahub

2.0.1 I can't edit a cell

We set some cells to read-only mode prevent accidental modification. To make the cell writeable,

1. Click the cell
2. Click setting on the top right corner
3. Under "Common Tools", you can toggle between "Editable" (can edit the cell) and "Read-Only" (cannot edit the cell)

2.0.2 I can't export my assignment as a PDF due to a `LatexFailed` error

Occasionally when running the `grader.export(run_tests=True)` cell at the end of the notebook, you run into an error where the PDF failed to generate:

Converting a Jupyter notebook to a PDF involves formatting some of the markdown text in LaTeX. However, this process will fail if your free response answers have (unresolved) LaTeX characters like `\n`, `$`, or `$$`. If you're short on time, your best bet is to take screenshots of your free response answers and submit them to Gradescope. If you have more time and would like the Datahub-generated PDF, please remove any special LaTeX characters from your free response answers.

2.0.3 My text cell looks like code

If you double-click on a text (markdown) cell, it'll appear in its raw format. To fix this, simply run the cell. If this doesn't fix the problem, check out the commonly asked question below.

2.0.4 My text cell changed to a code cell / my code cell changed to a text cell

Sometimes, a text (markdown) cell was changed to a code cell, or a code cell can't be run because it's been changed to a text (markdown) or raw cell. To fix this, toggle the desired cell type in the top bar.

2.0.5 Why does running a particular cell cause my kernel to die?

If one particular cell seems to cause your kernel to die, this is likely because the computer is trying to use more memory than it has available. For instance: your code is trying to create a gigantic array. To prevent the entire server from crashing, the kernel will "die". This is an indication that there is a mistake in your code that you need to fix.

3 Autograder and Gradescope

> **Note**
>
> **Citation**: many of these common questions were taken and modified from the UC San Diego course DSC 10: Principles of Data Science and their debugging guide.

3.1 Autograder

While

3.1.1 Why do I get an error saying "`grader is not defined`"?

If it has been a while since you've worked on an assignment, the kernel will shut itself down to preserve memory. When this happens, all of your variables are forgotten, including the grader. That's OK. The easiest way to fix this is by restarting your kernel and rerunning all the cells. To do this, in the top left menu, click `Kernel` -> `Restart and Run All Cells`.

3.1.2 I'm positive I have the right answer, but the test fails. Is there a mistake in the test?

While you might see the correct answer displayed as the result of the cell, chances are your solution isn't being stored in the answer variable. Make sure you are assigning the result to the answer variable and that there are no typos in the variable name. Finally, restart your kernel and run all the cells in order: `Kernel` -> `Restart and Run All Cells`.

3.1.3 Why does `grader.check_all()` fail if all previous tests passed?

This can happen if you "overwrite" a variable that is used in a question. For instance, if Question 1 asks you to store your answer in a variable named `stat` and, later on in the notebook, you change the value of `stat`, the test after Question 1 will pass, but the test at the end of the notebook will fail. It is good programming practice to give your variables informative names and to avoid repeating the same variable name for more than one purpose.

3.1.4 Why does a notebook test fail now when it passed before, and I didn't change my code?

You probably ran your notebook out of order. Re-run all previous cells in order, which is how your code will be graded.

3.1.5 I accidentally deleted something in a cell that was provided to me – how do I get it back?

Suppose you're working on Lab 5. One solution is to go directly to DataHub and rename your lab05 folder to something else, like lab05-old. Then, click the Lab 5 link on the course website again, and it'll bring you to a brand-new version of Lab 5. You can then copy your work from your old Lab 5 to this new one, which should have the original version of the assignment.

Alternatively, you can access this public repo and navigate to a blank copy of the assignment you were working on. In the case of Lab 5 for example, the notebook would be located at `lab/lab05/lab05.ipynb`. You can then check and copy over the contents of the deleted cell into a new cell in your existing notebook.

3.2 Gradescope

When submitting to Gradescope, there are often unexpected errors that make students lose more points than expected. Thus, it is imperative that you stay on the submission page until the autograder finishes running, and the results are displayed.

3.2.1 Why did a Gradescope test fail when all the Jupyter notebook's tests passed?

This can happen if you're running your notebook's cells out of order. The autograder runs your notebook from top-to-bottom. If you're defining a variable at the bottom of your notebook and using it at the top, the Gradescope autograder will fail because it doesn't recognize the variable when it encounters it.

This is why we recommend going into the top left menu and clicking `Kernel`` ->Restart->Run All“. The autograder “forgets” all of the variables and runs the notebook from top-to-bottom like the Gradescope autograder does. This will highlight any issues.

Find the first cell that raises an error. Make sure that all of the variables used in that cell have been defined above that cell, and not below.

3.2.2 My autograder keeps running/timed out

If your Gradescope submission page has been stuck running on this page for a while:

or if it times out:

it means that the Gradescope autograder failed to execute in the expected amount of time. This could be due to an inefficiency in your code or a problem on Gradescope's end, so we recommend resubmitting and letting the autograder rerun. It is your responsibility to ensure that the autograder runs properly, and, if it still fails, to follow up by making a private Ed post.