The Journal of Open Source Education

nbgrader: A Tool for Creating and Grading Assignments in the Jupyter Notebook

Project Jupyter[1], Douglas Blank[7], David Bourgin[1], Alexander Brown[8], Matthias Bussonnier[1], Jonathan Frederic[2], Brian Granger[3], Thomas L. Griffiths[1], Jessica Hamrick[4], Kyle Kelley[9], M Pacer[9], Logan Page[5], Fernando Pérez[1], Benjamin Ragan-Kelley[6], Jordan W. Suchow[1], and Carol Willing[3]

1 University of California, Berkeley **2** Google Inc. **3** Cal Poly, San Luis Obispo **4** DeepMind **5** University of Pretoria **6** Simula Research Laboratory **7** Bryn Mawr College **8** Lafayette College **9** Netflix, Inc.

DOI: 10.21105/jose.00032

Software

- Review
- Repository
- Archive

Submitted: 06 October 2018
Published: 08 October 2018

Note: Authors on this paper are listed in alphabetical order.

Summary

nbgrader is a flexible tool for creating and grading assignments in the Jupyter notebook (Kluyver et al., 2016). nbgrader allows instructors to create a single, master copy of an assignment, including tests and canonical solutions. From the master copy, a student version is generated without the solutions, thus obviating the need to maintain two separate versions. nbgrader also automatically grades submitted assignments by executing the notebooks and storing the results of the tests in a database. After auto-grading, instructors can manually grade free responses and provide partial credit using the *formgrader* notebook extension. Finally, instructors can use nbgrader to leave personalized feedback for each student's submission, including comments as well as detailed error information.

nbgrader can also be used with JupyterHub (Jupyter Development Team, 2018), which is a centralized, server-based installation that manages user logins and management of Jupyter notebooks. When used with JupyterHub, nbgrader provides additional workflow functionality, covering the entire grading process. After creating an assignment, instructors can distribute it to students, who can then fetch a copy of the assignment directly through the notebook server interface. Students can submit their completed assignment through the same interface, making it available for instructors. After students submit their assignments, instructors can collect the assignments with a single command and use the auto-grading functionality in the normal way.

Since its conception in September 2014, nbgrader has been used in a number of educational contexts, including courses at UC Berkeley, Cal Poly, University of Pretoria, University of Edinburgh, Northeastern University, Central Connecticut State University, KTH Royal Institute of Technology, CU Boulder, University of Amsterdam, George Washington University, Texas A&M, Bryn Mawr College, Lafayette College, and University of Maryland; and, as of May 2018, over 10,000 nbgrader-based notebooks exist on GitHub. In addition to its core functionality, nbgrader has expanded to support a number of other features, including the ability to handle multiple courses on the same JupyterHub instance; the option to either include or hide automatically graded tests; customizable late penalties; and support for importing assignment files downloaded from a Learning Management System (LMS).

JOSE
The Journal of Open Source Education

Statement of Need

The use of computational methods has become increasingly widespread in fields outside of computer science (Wing, 2008). As these disciplines begin to require computational tools, undergraduate curricula also begin to include topics in programming and computer science. However, perhaps because students are focused on the discipline that is the object of their study—and programming is likely a secondary interest—it has been shown that teaching computer science can be more effective when courses include interdisciplinary motivations (Cortina, 2007; Forte & Guzdial, 2005; Guzdial & Forte, 2005). One approach for teaching programming in a way that facilitates exploration with interdisciplinary questions is to teach students computational concepts in an interactive environment where it is possible to quickly write, test, and tweak small units of code. Many such environments exist, including Mathematica (Wolfram Research, Inc., 2018), Maple (Maplesoft, a division of Waterloo Maple Inc., n.d.), MATLAB (The MathWorks, Inc., 2018), Sage (Stein & others, 2018) and IPython (Pérez & Granger, 2007). However, these are often focused on programming in a single language, and lack an efficient system for distributing, collecting, and evaluating student work.

In recent years, the IPython project introduced the *Jupyter Notebook* (Kluyver et al., 2016), an interface that is conducive to interactive and literate computing, where programmers can interleave prose with code, equations, figures, and other media. Jupyter notebooks were originally developed for programming in the Python programming language, but multiple languages are now supported using the same infrastructure. The Jupyter Notebook is ideal for educators because it allows them to create assignments which include instructions along with code or Markdown cells, in which students can provide solutions to exercises. Students can, for example, write code both to compute and visualize a particular result. Because the notebook is interactive, students can iterate on a coding problem without needing to switch back and forth between a command line and a text editor, and they can rapidly see results alongside the code which produced them.

Instructors in many fields have begun using the Jupyter Notebook as a teaching platform. The notebook has appeared in over 100 classes (Castaño, 2018a, 2018b; J. B. Hamrick & Team, 2016) on subjects including geology, mathematics, mechanical engineering, data science, chemical engineering, and bioinformatics. Software Carpentry, an organization that aims to teach graduate students basic computational skills, has also adopted the notebook for some of its lessons (Wilson, 2014).

Despite its appearance in many classrooms, yet before the existence of nbgrader, the notebook was rarely used on a large scale for *graded* assignments. Instead, it was often used either for ungraded in-class exercises, or in classes small enough that notebooks could be graded by hand (sometimes even by printing them out on paper and grading them like a traditional assignment). This is because there are several challenges to using the notebook for graded assignments at scale. First, for large classes, it is not feasible for an instructor to manually grade the code that students write: there must be a way to automatically grade the assignments. However, a Jupyter Notebook is not a typical script that can be run and may contain multiple parts of a problem within the same notebook; thus, automatically grading a notebook is less straightforward than it is for a traditional script. Second, for many courses, programming is a means to an end: understanding concepts in a specific domain. Specifically, instructors may want students to provide both code and written free-responses interpreting the results of that code. Instructors thus need to be able to rely on automatic grading for the coding parts of an assignment, but also be able to manually grade the written responses in the surrounding context of the student's code. Third, the process of distributing assignments to students and later collecting them can be tedious, even more so with Jupyter Notebooks because there is a separate interface for accessing them beyond the standard system file browser. This often leads to confusion on the part of students about how to open notebooks after downloading

them, and where to find the notebooks in order to submit them.

nbgrader streamlines the repetitive tasks found in course management and grading, and its flexibility allows greater communication between instructor and student. nbgrader has moreover enabled instructors to use notebook-based assignments in classes with hundreds of students, which was previously not possible to do without excessive human effort. Overall, nbgrader does—and with further development will continue to—improve the learning experience for both instructors and students, enabling them to focus on content and building understanding.

References

- nbgrader source code: https://github.com/jupyter/nbgrader
- nbgrader documentation: http://nbgrader.readthedocs.io/en/stable/

Castaño, E. L. (2018a, May). Jupyter map. doi:10.5281/zenodo.1245087

Castaño, E. L. (2018b, May). Jupyter usage in institutions with coordinates. doi:10.5281/zenodo.1244833

Cortina, T. J. (2007). An introduction to computer science for non-majors using principles of computation. *ACM SIGCSE Bulletin*, *39*(1), 218. doi:10.1145/1227504.1227387

Forte, A., & Guzdial, M. (2005). Motivation and non-majors in computer science: Identifying discrete audiences for introductory courses. *IEEE Transactions on Education*, *48*(2), 248–253.

Guzdial, M., & Forte, A. (2005). Design process for a non-majors computing course. *ACM SIGCSE Bulletin*, *37*(1), 361. doi:10.1145/1047124.1047468

Hamrick, J. B., & Team, J. D. (2016, May). 2016 jupyter education survey. doi:10.5281/zenodo.51701

Jupyter Development Team. (2018). *JupyterHub*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87–90).

Maplesoft, a division of Waterloo Maple Inc. (n.d.). *Maple*. Waterloo, Ontario.

Pérez, F., & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science and Engineering*, *9*(3), 21–29. doi:10.1109/MCSE.2007.53

Stein, W., & others. (2018). *Sage Mathematics Software*. The Sage Development Team.

The MathWorks, Inc. (2018). *MATLAB*. Natick, MA.

Wilson, G. (2014). Software Carpentry: lessons learned. *F1000Research*, *3*, 62. doi:10.12688/f1000research.3-62.v1

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717–3725.

Wolfram Research, Inc. (2018). *Mathematica*. Champaign, IL.