Using Software Development Tools in a Classroom

Roman Yasinovskyy, D. Sc. [*]

Luther College

Abstract

In this paper I describe a way to set up a classroom environment using the common tools used in software development with the focus on Python. I cover version control system *Git*, source formatter *black*, and the unit testing framework *pytest*. Proper use of these tools helps students preserve history of code changes, eliminates subjective style preferences from the code review process, and provides feedback before a student submits their assignment.

Instructors can use *Git* to collect assignments rather than rely on their school's LMS and review code in their editor of choice faster. They can also use the course repository to publish assignments and class notes, as well as distribute minor fixes and clarifications. I recommend using private repositories for individual students, thus protecting their code from infringements and preserving academic integrity of the class.

black is not the only one code formatter available for Python and the recommendations for its use should apply to *yapf* and *autopep8*. Using an automated code formatter allows the instructor to have consistent and readable code style in lecture notes and assignments.

pytest allows the instructor to specify the expected outcomes and check if a submission meets those without the need to run a program manually. It is also possible to set time limits on individual tests and flag inefficient solutions in addition to those causing runtime errors or producing wrong answers.

The proposed setup works on Linux, macOS, and Windows without major modifications. I recommend publishing a standard *requirements.txt* for students to install and using Python's built-in module *venv* to isolate a class from the rest of the system packages. This approach helps eliminate (or at least greatly reduce) the common "works on my machine" problem by locking both Python version and installed packages.

The proposed approach helps bridge the gap between academia and industry, promotes consistency of students' code, and decreases grading time. Using testing framework like *pytest* allows students to test their program multiple times before submitting it and reduces the amount of uncertainty in the grading process.

[*]Special thanks to Karina Hoff and David Oniani