

Teaching programming at scale

Tue Herlau, tuhe@dtu.dk



DTU Compute

Department of Applied Mathematics and Computer Science

This talk

- A practical talk – hopefully things you can act on
- A combination of several things:
 - What is happening in the new campus-wide python programming course
 - Practical ideas on using programming (exercises, projects, exams)

Teaching experience

- 02450: Introduction to machine learning and data mining
 - My first experience teaching
 - Went from approx. 100 students/year to 1100 students/year
 - Mainly an administrative challenge
- 02465: Introduction to reinforcement learning and control theory
 - Core course in bachelor of artificial intelligence (KID)
 - Small course (100 students)
 - Challenge: Diverging programming competences/motivation
- 02002: Python Programming
 - Will run from fall 2023
 - Pretty much all students at DTU will take this course

Summary

Unitgrade: Autograding solution used in 02465:

- Shallow extension of unittests
- Student experience
- Requires less work to set up than codejudge
 - Vedranas exam from introduction to programming
 - Alcestes exam from the CPP course and exercise 6

CMU Autolab: Platform used at CMU and many other universities

- Functions like a container for tests
- Fully text-configured

• <https://gitlab.compute.dtu.dk/tuhe/unitgrade>

• Videos: <https://youtu.be/jC9AzZA5FcQ>

• <https://autolabproject.com/>

Unitgrade: what got me started

Online autograders

- They do many things
 - Exercise descriptions
 - Administrative tasks (student onboarding, grading, exports)
 - Calendar-tasks (due-date, handling delays)
 - Assignment handin
 - Plagiarism checking
 - An IDE
 - A CI/CD system (upload code and run)
 - Various forms of automation/ideas about organizing exercises
 - A test system

My goals

User experience: **Best** allow students to solve their programming problems

Administrative ease: Make the course **software**, **tests**, and **written material** easy to write and maintain

User experience

- Which test system best allows **students** to fix problems in **their** homework?
- Which test system best allow **me** to fix problems in **my code**?
- **Web-based tests**
 - No debugger
 - Local/remote code (or use the bad online IDE)
 - Increased run-time (can tests be run in isolation?)
 - Blackboxing (environment, packages, file locations, how code is called)
 - Drivers
 - Preprocessors
 - Postprocessors
 - Hacks (cat, fix-floats, etc.)
 - Adopt conventions from the tool (is reading from stdin optimal?)
- **Unittests:**
 - Best-practice for 25 years
 - Use your favorite IDE: Debugger + autocomplete + inline documentation
 - Test are runnable in isolation
 - Transparency; everything is python
 - Everything can be unittested
 - Student learns relevant skills (unittesting)

Example: Intro to python exam

Setup: `pip install unitgrade`

Example: Writing your own test

Casestudy: Introduction to python (Vedrana)

Codejudge

- 5 problems
- 11 tests per problem
- 60 files
- 282 lines of code

Unitgrade

- 5 problems
- 11 tests per problem
- 3 files
- 127 lines of code

Bonus: My version contains a handout stub for students to work with.

Casestudy: Introduction to python (grading)

Casestudy: Problem set 6 (Alceste)

Class representing fractions

Casestudy: Problem set 6 (Alceste)

- Problem set 6, create a Fraction class with addition

Codejudge

- 1 problem
- 8 files (.in, .ans)
- 2 tests
- 12 lines of code (only specifies input & output; contains more code in LaTeX)

Unitgrade

- 1 problems
- 2 files
- 6 tests
- 86 lines of code

Bonus: Contains more specific tests, generates handout files, contains a solution, and allows automatic checks of implementation for later refactoring.

Features: **Automation**

Casestudy: Exam in C++ Course 2021

- Exam set for 2021
- 4 problems

Codejudge

- 72 files (!)
- 16 .yaml files
- 18 .ans files
- 552 lines of code (excluding .ans files)

Unitgrade

- 6 files (4 problems + 1 test + 1 deploy)
- 116 lines of code
- More tests

- Classic autograder: Students, classes, assignments.
- Sign-up can be handled using student email (confirmation emails)

- An assignment is a single .tar file:
- tl;dr:
 - The container will copy your autograder files the students into the same directory
 - Then it calls `autograde_Makefile`
 - You return json with the scores to stdout
- Easy to call e.g. a unittest script
- Tests can be run locally
- Many examples online

- Unitgrade allows one-line deployment to an autolab .tar file for upload