Experiences with automatic grading

Frank Mulder

Context

- Intermediate Python Programming (350 students, minor)
- Software Quality and Testing (450 students, 1st year BSc)
- Similar setup for both courses:
  - One programming exam at the end of the course (week 10)
  - All exam assignments automatically graded
  - Throughout the course: similar assignments to practise with
  - “It works”

Why

- Scalability: similar effort for 10, 100 or 1000 students
  - Takes less time/money in total.
- Consistency: the grade does not depend on the person who graded, the time of day, etc.
- Transparency: it is very clear how the grade was determined
- Long-term benefit: old exams serve as practice material with automatic feedback
- Enjoyment: it’s more fun to write automated tests than to grade hundreds of submissions manually

Drawbacks

- Partially correct solutions lead to different scores than when grading manually:
  - A solution could look similar to a correct solution, but still lead to very few points (e.g. returning the wrong value, or writing ‘<‘ instead of ‘>’)
  - Syntax errors lead to 0 points
  - -> But I do give an indication of the score, so students can fix the problem during the exam
  - -> **Communication** is essential
- Some learning objectives cannot easily be assessed automatically. E.g. making trade-offs.
- Mork work to set up
- Workload shifts earlier (could also be an advantage)

Demo: Intermediate Python Programming

- Calculate score based on how many unit tests pass

Demo: Software Quality and Testing

- Calculate score based on how many bugs are caught (“meta-tests” / mutation)

Experiences

- Communication is essential
  - (and practice)
- Writing good automated tests (for this purpose) is difficult.
  - Not easy to delegate (to TAs for instance)
  - Although writing meta-tests (“bugs”) for testing assignments seems to be easier
- When grading programming assignments
  - Be aware of functions that return booleans, (empty) sets, etc.
  - Reduce dependencies between functions in the design
  - Base the tests on the description
  - Think of partially correct solutions
- When grading testing assignments, be aware of:
  - Base the meta-tests on the description (but also on the code under test)
  - Think of realistic bugs (use bad student solutions from a programming course 😅)

Audience discussion

- Do you have experience with automatic grading? Do you do things differently?
- Do you think you could introduce automatic grading in your course (if you’re not doing so right now)?