Towards Practical Programming Exercises and Automated Assessment in Massive Open Online Courses

Thomas Staubitz, Hauke Klement, Jan Renz, Ralf Teusner, Christoph Meinel
Hasso-Plattner-Institute
openHPI – Short Introduction

Course Production

Programming Exercises
Thomas Staubitz, Research Associate, HPI
openHPI – Short Introduction

Learning cycle consisting of weekly released videos, self tests, and assignments, plus a final exam
➡️ Record of Achievement

Programming Exercises
Thomas Staubitz, Research Associate, HPI
Add 17 + 4

Write a code snippet that prints the sum of 17 + 4 to the screen

Results
1 test files have been executed.

Test File 1 (r111.py)
- Passed Tests: 1 out of 1
- Score: 1.00 out of 1
- Feedback: Well done!
- Program Output: Show

Score: 1.00 / 1.0
Why do we need it?

- Introductory programming courses attract large amounts of participants from all ages (Java für Einsteiger – 14,000, PythonJunior2014 – 10,000, PythonJunior2015 – 8000 Learners from 10 – 80 y.)

- Study/Survey by European Commission related to web skills:
  - IT professionals consider MOOCs the best way to learn such abilities.
  - Learners value practical experience over theoretical content

- Radical differences in the way different concepts are grasped by students
  - Learning to program does not only involve acquiring complex knowledge but also practical skills
  - Gaining programming expertise requires rigorous practice (Berges et al, Feldman and Zelenski, Robins and Rountree, Vivahainen et al, Douce et al)
CodeOcean – How it all began

- Literature review on practical programming tasks and automated assessment
- Analysis of current practices in MOOCs – Pros and cons of different scenarios
- Map existing tools and components
- History of automated assessment tools

→ Decision making if, why, and what should we develop
## Scenarios for Practical Exercises (1/3)

<table>
<thead>
<tr>
<th>S1:</th>
<th><strong>Editor and Execution Environment</strong></th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Local (native or VM)</td>
<td>In Platform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S2:</th>
<th><strong>Editor and Execution Environment</strong></th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Third party online tool</td>
<td>Depending on tool (e.g. STEAP)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S3:</th>
<th><strong>Editor and Execution Environment</strong></th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In Platform</td>
<td>In Platform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S4:</th>
<th><strong>Editor and Execution Environment</strong></th>
<th>Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In Platform/Browser</td>
<td>Browser</td>
</tr>
</tbody>
</table>
## Scenarios for Practical Exercises (2/3)

<table>
<thead>
<tr>
<th>S1:</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Scalability</td>
<td>• Heterogeneity (Support)</td>
</tr>
<tr>
<td></td>
<td>• Allows to use existing</td>
<td>• Sparse data (final result only)</td>
</tr>
<tr>
<td></td>
<td>specialized or favorite</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tools</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S2:</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Homogeneity</td>
<td>• Dependency on third party</td>
</tr>
<tr>
<td></td>
<td>• Installation free</td>
<td>• Limited possibilities</td>
</tr>
<tr>
<td></td>
<td>• Low Barrier</td>
<td>• Sparse data</td>
</tr>
<tr>
<td></td>
<td>• Scalability outsourced</td>
<td></td>
</tr>
</tbody>
</table>
### Scenarios for Practical Exercises (3/3)

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>-</th>
</tr>
</thead>
</table>
| **S3:** | • Homogeneity  
  • Installation free  
  • Low Barrier  
  • Rich data  
  • Complete control  
  • Wide range of supported languages | • Scalability issues  
  • Costs  
  • Server  
  • Development  
  • Security issues |
|   |   |   |
| **S4:** | • Resource efficient | • Cheating possibilities  
  • Limited set of languages |
## Component Landscape

| Educational online programming tools | • Codewars, CodingBat, CodeHunt  
| • … |
| Online development tools | • CodePen, jsFiddle, repl.it  
| • … |
| Web based IDEs | • Cloud9, Codio, Nitrous.IO  
| • … |
| SaaS/PaaS | • Web-based IDE (e.g. Cloud9)  
| • Code hosting, issue tracking (e.g. GitHub)  
| • Execution platform (e.g. Heroku) |
Automated Assessment (1/2)

- **Categories**
  - Dynamic: Code is executed, results are analyzed
  - Static: Code is analyzed

- **What is assessed?**
  - Functionality, completeness, performance
  - Code quality and style
  - GUIs
  - Testing skills

- **How is assessed?**
  - I/O based: predefined input values → verification of expected output
    - Challenge: Output format might differ from expected while still producing the correct result (e.g. Sonnabend vs. Samstag, whitespace, spelling mistakes)
  - Based on testing frameworks: Unit testing, Acceptance testing
Automated Assessment (2/2)

- Automated assessment ...
  - ... increases the need for carefully designed assignments as poor design cannot be compensated by assessment
    - less time for grading
    - more time for designing
  - ... the larger the size of the course, the better the ROI
  - ... inevitable for MOOCs (Alternative: Peer Assessment)
Conclusion

- Practical programming tasks are essential for programming courses
- In the context of MOOCs automated assessment is key
- Comparing the benefits and drawbacks of the scenarios that have been introduced in the beginning:
  - We need a flexible solution that is able to handle things differently depending on a course’s main target group
    - Beginners benefit more from a browser based environment
    - Advanced users will prefer familiar tools
- The landscape of existing programming languages is wide spread:
  - Providing a new programming environment for each course is not desirable
  - We need a solution that supports multiple programming languages
Future Work

- Introduction of our own tool: CodeOcean – A tool for browser based code execution and automated assessment
  - Implemented
  - Open Source (https://github.com/openHPI/codeocean)
  - Employed in three courses on openHPI
- In depth evaluation of the results of these courses, particularly analysis of the students’ code submissions and feedback
- In depth comparison of the tools that have been employed in various MOOCs and the experiences that have been made.
- ...
Questions?

Thomas.Staubitz@hpi.de

Thomas Staubitz
Research Associate
Hasso Plattner Institute