MINTFIT HAMBURG
Eine Initiative der Hamburger Hochschulen

CodeRunner

– automatisch ausgewertete Programmieraufgaben –
Ein Pilotprojekt von MINTFIT

Louis Kobras (Universtät Hamburg), Marcus Soll (Projekt MINTFIT)

November 2019

Was ist CodeRunner und was kann ich damit machen?

CodeRunner[1] ist ein Plugin für das eLearning-System *Moodle*[2], welches von der MIN-Fakultät betrieben wird[3]. Es ermöglicht, im Rahmen von Kursen auf der Moodle-Plattform Programmieraufgaben zu stellen, die automatisiert ausgewertet werden. Dies reduziert den relativen Betreuungsaufwand für eine Programmierausbildung. Andererseits fördert es die Studierenden über direktes automatisiertes Feedback in der Reflexion zum Lernerfolg, wie möglicherweise bereits aus Self-Assessments (vgl. OLAT-Tests o.Ä.) bekannt.

Wie kann ich CodeRunner einsetzen?

CodeRunner-Quizzes können im Moodle zu jedem Organisationsblock hinzugefügt werden. Über das Plugin wird die Aktivität "Quiz" um einen Fragetyp "CodeRunner" erweitert, welcher dann mit den bekannten anderen Fragetypen in einem Quiz gemischt werden kann. CodeRunner kann wahlweise semesterbegleitend, in Blockveranstaltungen bzw. Crashkursen oder sogar für Prüfungen eingesetzt werden.

Semesterbegleitend. Wie im Pilotprojekt in der Grundlehrveranstaltung "Algorithmen und Datenstrukturen" (InfB-AD, mehrere hundert Teilnehmer) erfolgreich gezeigt, können CodeRunner Quizzes semesterbegleitend eingesetzt werden, um bestimmte Inhalte zu vertiefen oder das Kern-Curriculum des Kurses um weitere Inhalte zu ergänzen. Dabei wurde synchron zu den Hausaufgabenzetteln im Moodle je ein CodeRunner-Quiz zu den Sprachen Java und Python3 angeboten, welche eine praktische Perspektive auf die in der Vorlesung behandelten Algorithmen geboten haben.

1 https://moodle.org/plugins/qtype_coderunner
2 https://moodle.de/
3 https://lernen.min.uni-hamburg.de/login/index.php

Blockveranstaltungen. In Blockveranstaltungen oder Crashkursen können die CodeRunner-Quizzes genutzt werden, um mit relativ wenig ad-hoc-Betreuungsaufwand den Studierenden einen umfassenden Einstieg in relevante Programmierkonzepte oder die Syntax der verwendeten Sprache zu geben.

Prüfungen. Über das MINTFIT-Projekt[4] können mit Standardsoftware ausgestattete Laptops bezogen werden, die für e-Klausuren verwendet werden können[5]. Darüber ließen sich Prüfungen zu Grundlagen der Programmierung realisieren, die einen praktischen Anteil umfassen oder sogar gänzlich praktisch sind, anstatt auf Lückentexte und händischen Quellcode angewiesen zu sein.

Welche Schwierigkeiten können auf mich zukommen?

Erstellen der Tests. Um die Technologie sinnvoll integrieren zu können, muss das didaktische Konzept ein wenig angepasst werden. Ebenso müssen die CodeRunner Tests neu konzipiert und gerade in den ersten paar Durchgängen vermutlich noch ein paar mal überarbeitet werden. Ebenso ist extensives Testen der Aufgaben nötig, und die (erstmalige) Vorbereitung der Aufgaben stellt einen nicht zu unterschätzenden Aufwand dar.

Kommunikation der Kriterien. Wie bei allen neuen Lehrkonzepten muss sehr stark darauf geachtet werden, dass die Studierenden die Anforderungen genau verstehen und nicht wegen eines Missverständnisses das Kursziel nicht erreichen. Dies ist natürlich insbesondere ein Problem, wenn Studierende am Kurs teilnehmen, deren Muttersprache nicht die Kurssprache ist.

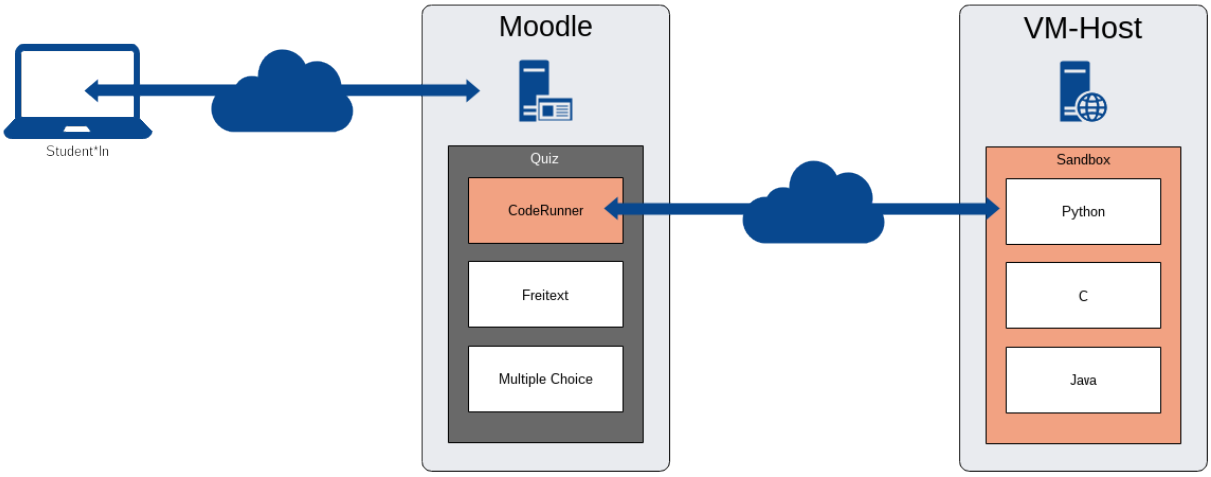
Ansprechperson. Erfahrungen haben gezeigt, dass es für die Studierenden von Vorteil ist, eine zentrale Ansprechperson zu haben, an die sie sich mit organisatorischen und Verständnisfragen für die Programmieraufgaben wenden können. Auch in der 4. Vorlesungswoche kann es noch vorkommen, dass jemand den Unterschied zwischen "Vorabprüfung" und "Prüfung" nicht verstanden hat.

Fehler in der Aufgabe. Dies impliziert auch, dass gerade in der ersten Iteration der neuen Aufgaben eine Abgabe eines Studierenden, die faktisch richtig, aber nicht bepunktet ist, der Fehler in der Konfiguration der Aufgabe oder im Verständnis des Systems gelegen haben kann. Es schadet nicht, beim Troubleshooting auch mal in die Einstellungen des Quizzes zu sehen, wenn ein Studierender das Scheinkriterium zu verfehlen droht.

Fehler im System. Studierende vermuten Fehler im System, wenn ein logischer Fehler in ihrer Abgabge vorliegt.

4 https://www.mintfit.hamburg/
5 https://www.mintfit.hamburg/ueber-mintfit/mintfit-e-assessment/

Wie sieht das System aus?



(Grafik von Marcus Soll)

CodeRunner erweitert die "Quiz"-Funktion von Moodle um einen Fragetyp namens "CodeRunner". Dieser stellt einen Wrapper dar für Template-Klassen, Test Cases und die eigentliche Aufgabenstellung.
Schickt ein Studierender eine Antwort ab, übermittelt das CodeRunner-Plugin diese an eine Sandbox, die in einer VM liegt. Dort wird die Antwort berechnet, indem der Quellcode ausgeführt und die Test Cases angewandt werden. Die Ergebnisse werden an CodeRunner zurückgeschickt, welches sie für den Studierenden darstellt und für den Lehrenden die Bepunktung errechnet.

Wen kann ich wegen der Nutzung des Plugins ansprechen?

Für das Setup der Fragen im Kontext von InfB-AD im WiSe 2019/2020 waren Louis Kobras <kobras@informatik.uni-hamburg.de> und Marcus Soll <2soll@informatik.uni-hamburg.de> verantwortlich. Die organisatorische und technische Unterstützung von Seiten des MIN-Dekanats kam von Marcus Soll.

Worauf muss ich bei der Konfiguration vom CodeRunner-Plugin achten?

CodeRunner ist auf dem MIN-Moodle aufgesetzt und konfiguriert und mit einer Sandbox-VM im Informatik-Rechenzentrum verbunden; das System ist lauffähig und live im Einsatz. Es kann direkt in Kurse eingebettet werden.

Grafiken im Anhang:

- Beispielhafte Aufgabenstellung
- Antwortfeld
- Rückmeldung an die Studierenden

Gefördert im Rahmen des BWFG-Projektes MINFIT [https://www.mintfit.hamburg/] als Pilotprojekt für digitale Klausuren. Laufzeit: 1.1.2019 - 31.12.2020. Fachliche Leitung: Prof. Dr. Chris Biemann[https://www.inf.uni-hamburg.de/en/inst/ab/lt/people/chris-biemann.html], biemann@informatik.uni-hamburg.de. Umsetzung: Marcus Soll[https://www.min.uni-hamburg.de/ueber-die-fakultaet/mitarbeiterverzeichnis/soll-marcus.html] 2soll@informatik.uni-hamburg.de

In der Vorlesung haben Sie den Algorithmus von Eiklid zum Berechnen des Größten Gemeinsamen Teilers kennen gelernt. Implementieren Sie diesen als Java-Methode in einer der Varianten.

BSP [Algorithmus]: GGT (Euklid, 300 v.Chr.)

ggt(a,b): 1. a = b*q + r mit r < b (ganzzahlige Division)
2. falls r=0: output b
3. a <= b; b <= r;
4. gehe zu Schritt 1.

Aufgabenstellung.
Der Text kann komplett frei gewählt werden.
Einbindung von Bildern u.ä. möglich

Iterative Variante:

```
GGT(a, b)      // Annahme: a > b
  while  a mod b > 0
    r = a mod b
    a = b; b = r
  return b
```

Rekursive Variante:

```
GGT(a, b)      // Annahme: a > b
  r = a mod b
  if r > 0   return GGT(b, r)
  else       return b
```

Zum Beispiel:

Test	Result
`print(GGT(12,18))`	6
`print(GGT(24,7))`	1

Beispiele für Testfälle.
Es kann konfiguriert werden, welche / wie viele hier angezeigt werden.
Daneben kann es beliebig viele Testfälle geben, die erst bei der Bewertung / gar nicht angezeigt werden

```
1  def GGT(a, b):
2      while a%b != 0:
3          r = a % b
4          a = b
5          b = r
6      return b
```

Eingabe der Lösung durch Studierende.

Beliebig oft testen ohne Bewertung auf einem kleinen Teil der Testfälle. Kein Abzug.

Testen mit Bewertung auf allen Testfällen. Abzug bei Fehler konfigurierbar.

Vorabprüfung | Prüfen

	Test	Erwartet	Erhalten	
✓	`print(GGT(12,18))`	6	6	✓
✓	`print(GGT(24,7))`	1	1	✓
✓	`print(GGT(105,55))`	5	5	✓

Ergebnisse der einzelnen Testfälle. Konfigurierbar, in wie weit sie gezeigt werden.

Alle Tests bestanden! ✓

Gesamtergebnis