Exercise recommender for novice programmers

A knowledge-based recommender system for students of an introductory programming course

Ingrid Næss Johansen | Colin Nordgård | Oda Inanna Stene

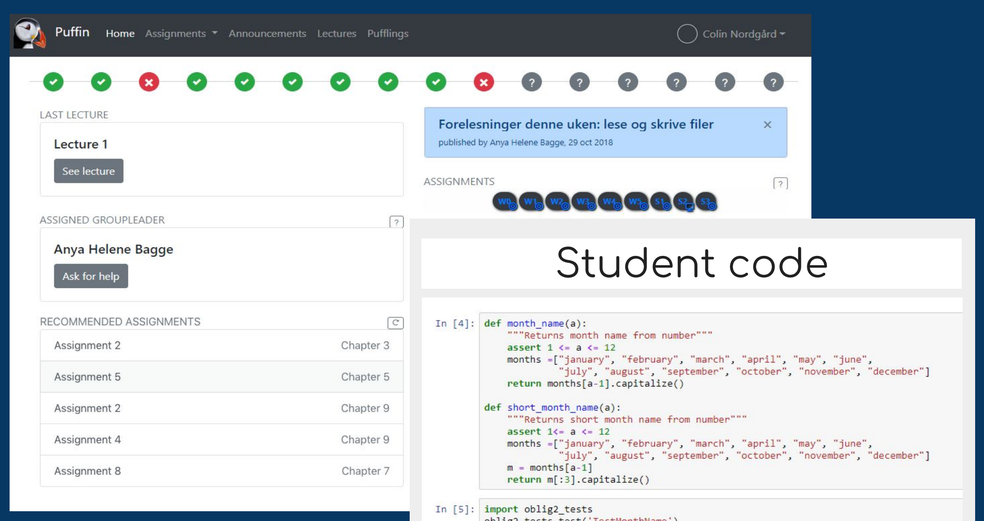
The Hypothesis

Personalized coding-exercise recommendations will result in better grades and a higher completion ratio of the first computer science course at university level.

The Experiment

Implement a recommender, divide the students into three groups and give them recommendations based on their own code, lecture code and code from mandatory assignments.

Gamification



Student code

```
In [4]: def month_name(a):
            """Returns month name from number"""
            assert 1 <= a <= 12
            months =["january", "february", "march", "april", "may", "june",
                     "july", "august", "september", "october", "november", "december"]
            return months[a-1].capitalize()

        def short_month_name(a):
            """Returns short month name from number"""
            assert 1<= a <= 12
            months =["january", "february", "march", "april", "may", "june",
                     "july", "august", "september", "october", "november", "december"]
            m = months[a-1]
            return m[:3].capitalize()

In [5]: import oblig2_tests
        oblig2_tests.test('TestMonthName')

Loaded oblig2_tests

...analysing function month_name():
Lookin' good so far !

...analysing function short_month_name():
- Style hint: You might consider using slightly longer and more descriptive variable names

s
----------------------------------------------------------------------
Ran 1 test in 0.010s

OK (skipped=1)

"Nice!"

Badges for this code:
```

Subject groups

Group 1:
random exercises
Group 2:
exercises based on used concepts
Group 3:
exercises based on used concepts and concepts from lectures and mandatory assignments

Lecture code

```
from csv import DictReader
with open("studenter.csv") as infile:
    reader = DictReader(infile)
    for student in reader:
        #if student['srettliste'] == 'BAMN-DSIK':
        print("Hei, " + student['fornavn'] + "!")
    print(reader.fieldnames)
```

First exercise

```
def hello():
    return 'Hello world!'

assert hello() == 'Hello world!'
```

The recommender

Concept parser

Exercises

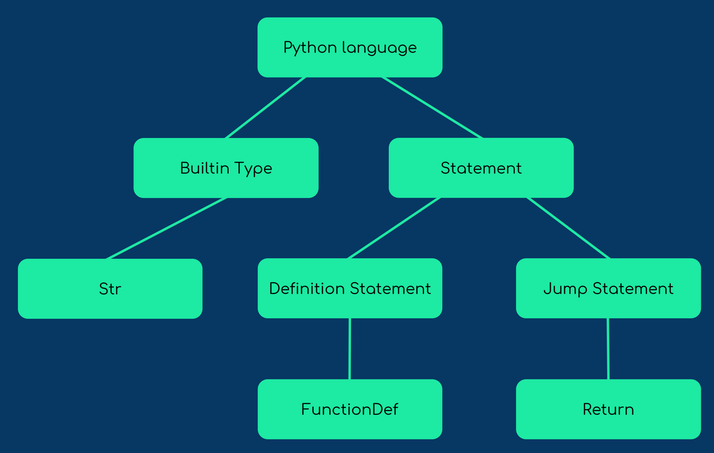
Exercise 5

if	2
else	1
Str	3

Status: unlocked

Concepts

Course Model	Student Model	
if	Str	1
else	functionDef	1
functionDef	Return	1
Str		
Return		



The Results

- Design experiment
- Implement recommender
- Collect data
- Analyze results

Data collection

- Minute-by-minute Git history
- Time spent programming
- Results of automatically tested exercises
- Results of manually graded exercises
- Error messages
- Attendance
- Study program
- Questionnaires
- Other subjects this semester
- Typical statistical variables
- Exam scores

Future work

- Recommend learning material
- Machine learning on collected data
- Crawl the data to find an ideal way through the course
- Like or dislike an exercise
- Active learning
- More advanced programming courses
- Other courses and interdisciplinary learning

UNIVERSITY OF BERGEN
Department of Informatics

References
Mordechai Ben-Ari 2001. Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching 20*, 1 (2001), 45-73.
Roya Hosseini and Peter Brusilovsky. 2013. JavaParser: A fine-grain concept indexing tool for Java problems. In *CEUR Workshop Proceedings*, Vol. 1009. University of Pittsburgh, 60-63.
John K Tarus, Zehndong Niu, and Ghulam Mustafa. 2018. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review 50*, 1 (2018), 21-48.