Automated Detection of coding idioms and flaws in student code

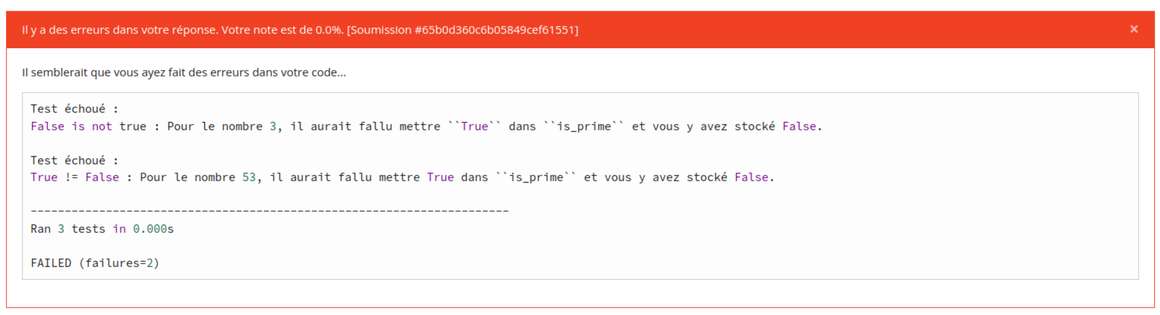
Julien Lienard

UCLouvain

Promotors: Kim Mens, Siegfried Nijssen

Context

- First programming course
- Utilization of autograders in education
- More than 600 students
- Most autograders provide feedback on code execution



Research Question

Can we create a tool to generate tests, that are easy to read and maintain, from flaws discovered in student code?

3 steps:

1. Discover flaws → Done manually(or through mining)
2. Manually express flaws in dedicated pattern language → Current focus
3. Automatically generate queries to detect such flaws → Future work

Student Validation

We created two very similar exercises:

1. Approximate π using the Gregory-Leibniz series
2. Approximate $\ln 2$ using a harmonic series

Advanced feedback, based on Pyttern patterns that were discovered in former student submissions, given only in the first exercise
Verify knowledge transfer to the second one (did students learn from the feedback received)

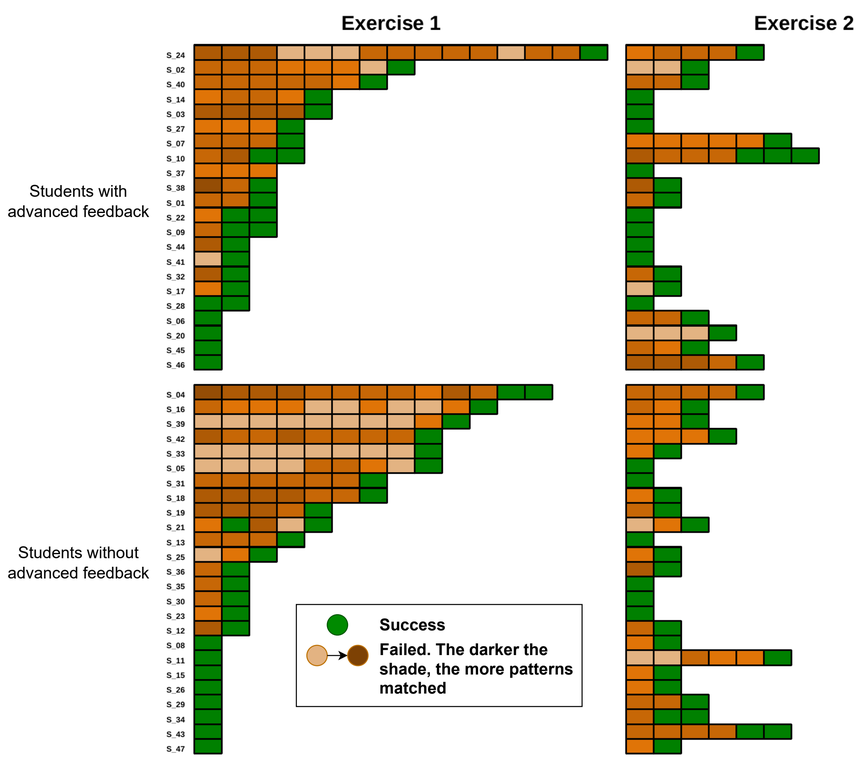


Fig. 1: Result of first validation on student

Exercise	Pattern matching	Mean Submission
Excercise 1	With feedback	3.68
	Without feedback	4.4
Excersise 2	With feedback	2.54
	Without feedback	2.48

Fig. 2: Mean number of submissions in both exercises

Current Status

- Analyzed students' code to discover typical coding flaws [1]
- Generate tests to detect such flaws based on parse tree analysis [2]
- Such tests are hard to maintain, read and adapt

Pyttern, a Code Query language

Fully written in Python
Uses '?' wildcards for matching
Implemented using AST traversal

Wildcard	Description
?	Any single node
?:	A node with a body
?[T1, T2, ...]	A node of Type T1 or T2 ...
?{n,m}	Between n and m of any node
?*	Any number of nodes
?*	Any number of nested bodies

```
def approx_pi(n):
    x = 0
    for i in range(0, n+1):
        x += ((-1)**i)/(2*i+1)
    x *= 4
    return x
```

```
def ?(n):
    ?var = 0
    ?:
        ?var *= 4
```

```
def ?(?*):
    ?acc = 0
    ?[While, For]:
        ?:*
            ?acc += ?
    return ?acc
```

```
def facteurs(n):
    prime = [2, 3, 5, 7, 11, 13]
    fac = 0
    for i in prime:
        if n % i == 0:
            fac += 1
    return fac
```

```
def ?(n):
    ? = [?[Constant]{3,}]
```

Teacher Validation

Proposed validation of Pyttern language with teachers

- We select a subset of students' submissions containing known flaws
- We split our group of validation participants (= teachers) in 3
- Each group use a different technique to try and detect flaws in the students' code
  1. Pyttern (our tool)
  2. Regex
  3. Python AST
- Time limit to detect a maximum number of flaws
- Each participant (= teacher) is then asked to write advanced student feedback based on their assigned technique

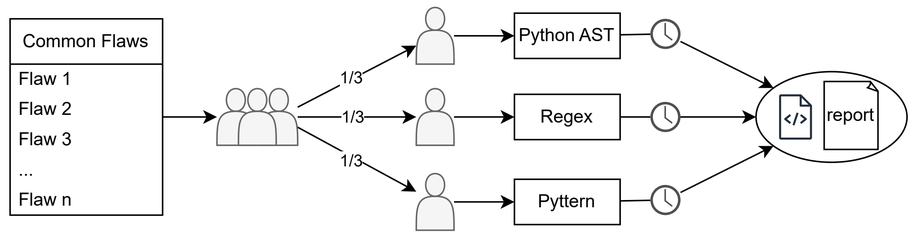


Fig. 3: Teacher validation methodology

- We will analyze all the patterns they create and the feedback to extract:
  - precision
  - recall
  - global feeling of the tool
  - number of flaws detected

[1] Kim Mens, Siegfried Nijssen, and Hoang-Son Pham. "The good, the bad, and the ugly: mining for patterns in student source code". In: *Proceedings of the 3rd International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*. 2021, pp. 1–8.

[2] Julien Lienard, Kim Mens, and Siegfried Nijssen. "Extracting Unit Tests from Patterns Mined in Student Code to Provide Improved Feedback in Autograders". In: 2023.