

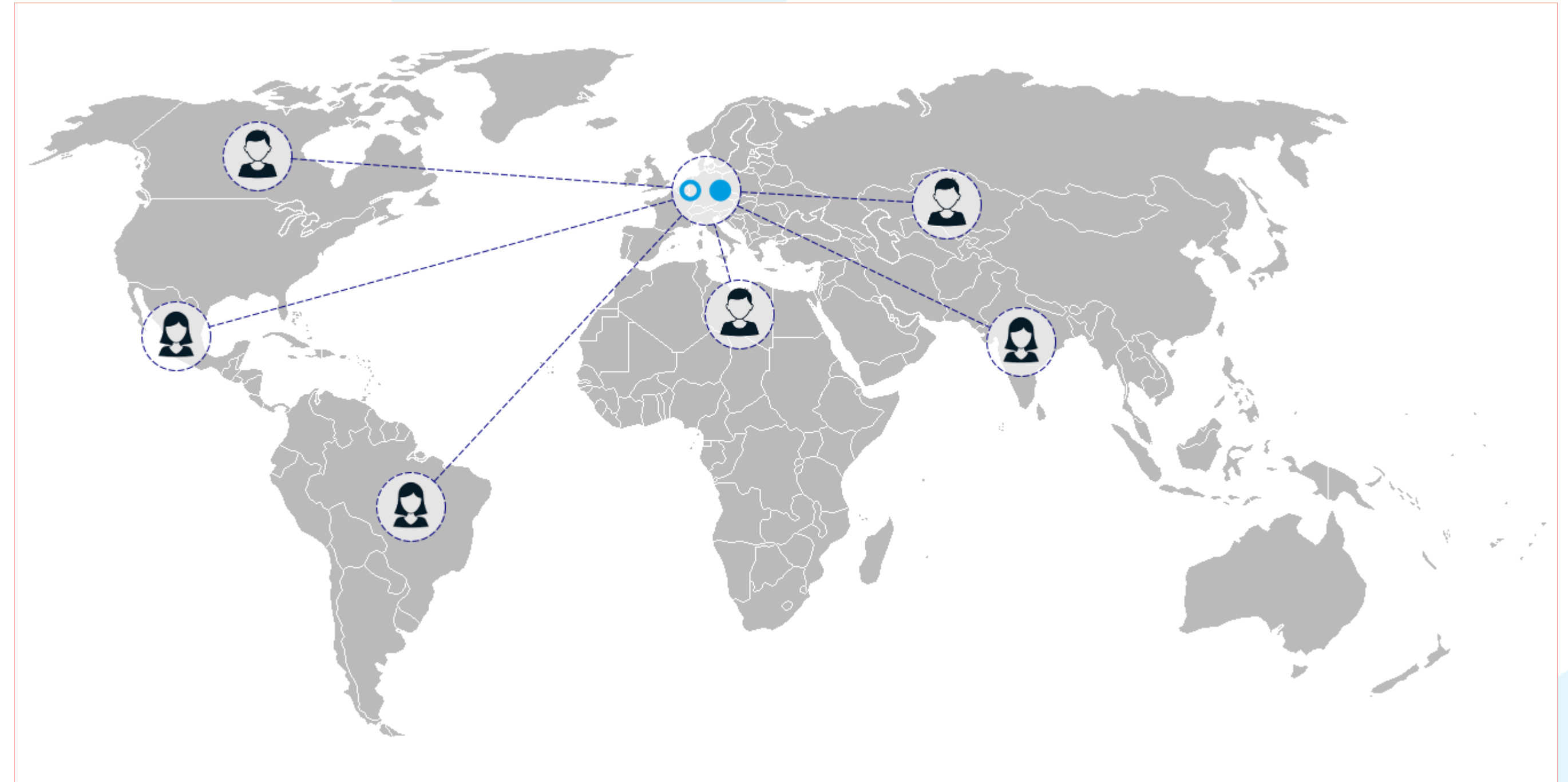
e2xgrader: An Add-on for Improved Grading and Teaching with Jupyter Notebooks at Scale

Tim Metzler
Mohammad Wasil
Prof. Dr. Paul G. Plöger

10.05.2023

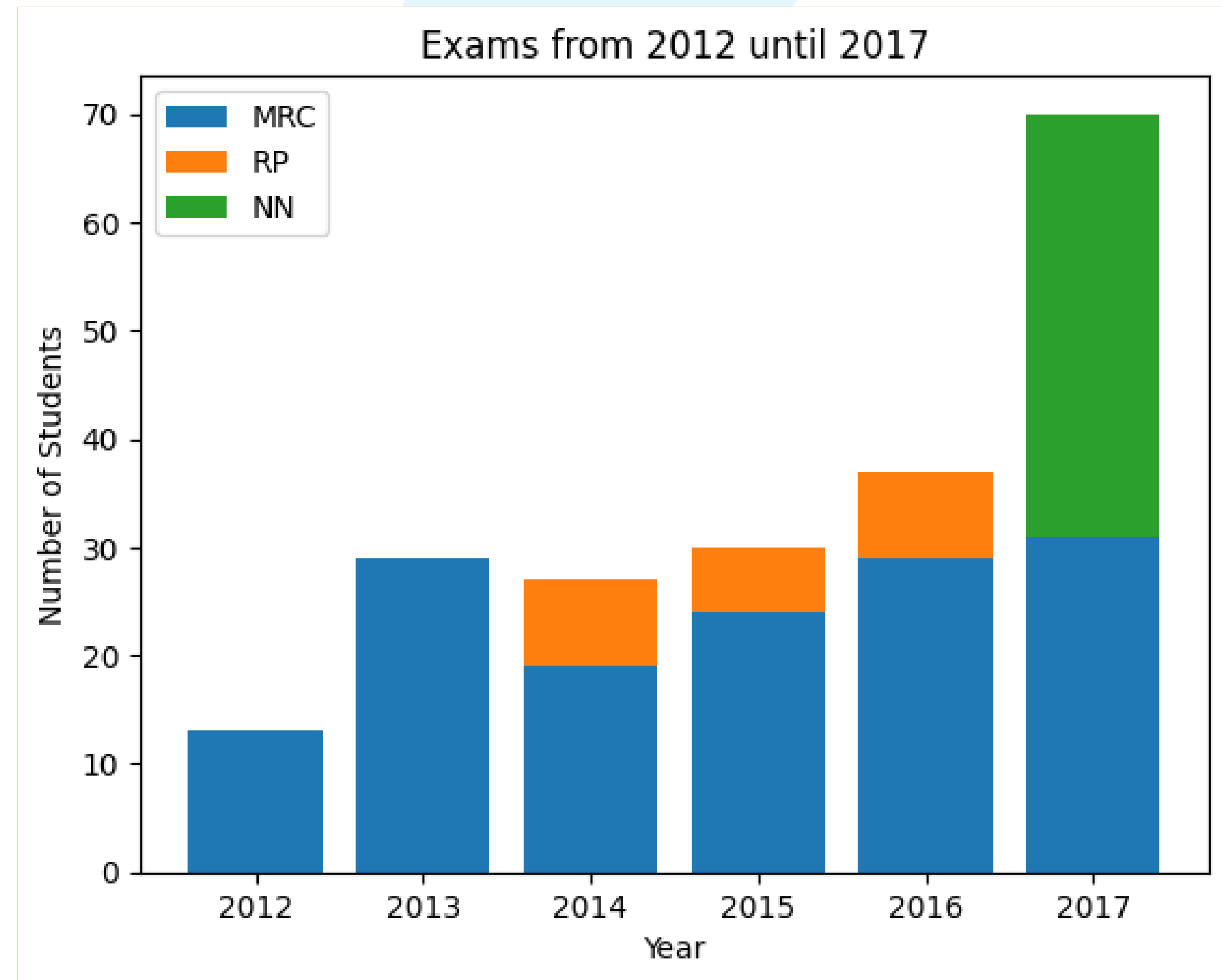
Who are we?

- Hochschule Bonn-Rhein-Sieg
Department of Computer Science
Master of Autonomous Systems (robotics,
machine learning, AI)
- Tim Metzler
- Mohammad Wasil
- Dr. Paul G. Plöger

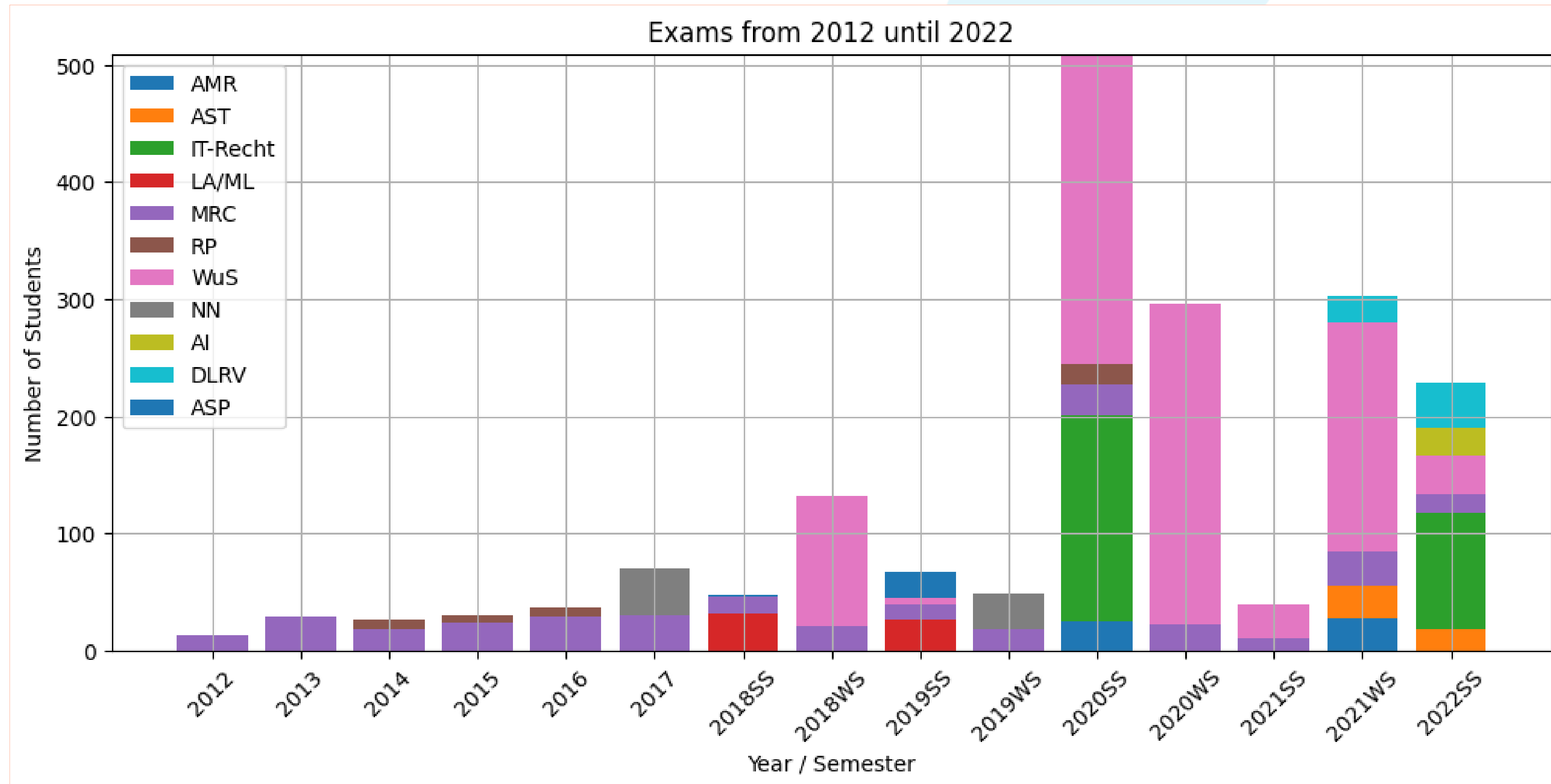


Scaling Up Exams and Assignments with Jupyter Notebooks

- Started using Jupyter Notebooks for exams and assignments in 2012
- Students solved assignments on their own machine
- Exams: Initially used laptops with USB sticks for distribution
- Challenges in scaling up:
 - Manual grading process
 - Difficulty tracking changes made by students
 - Version conflicts

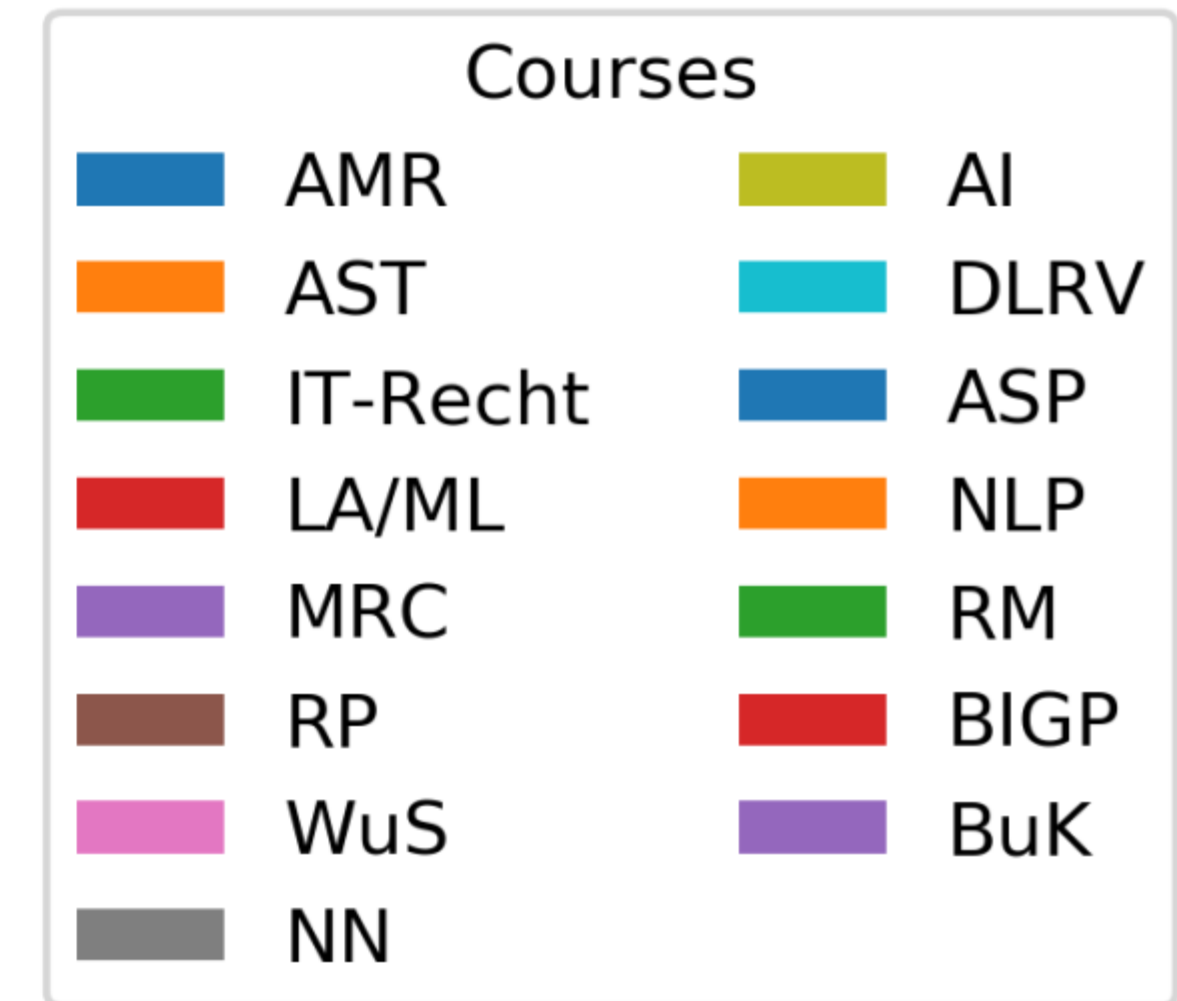
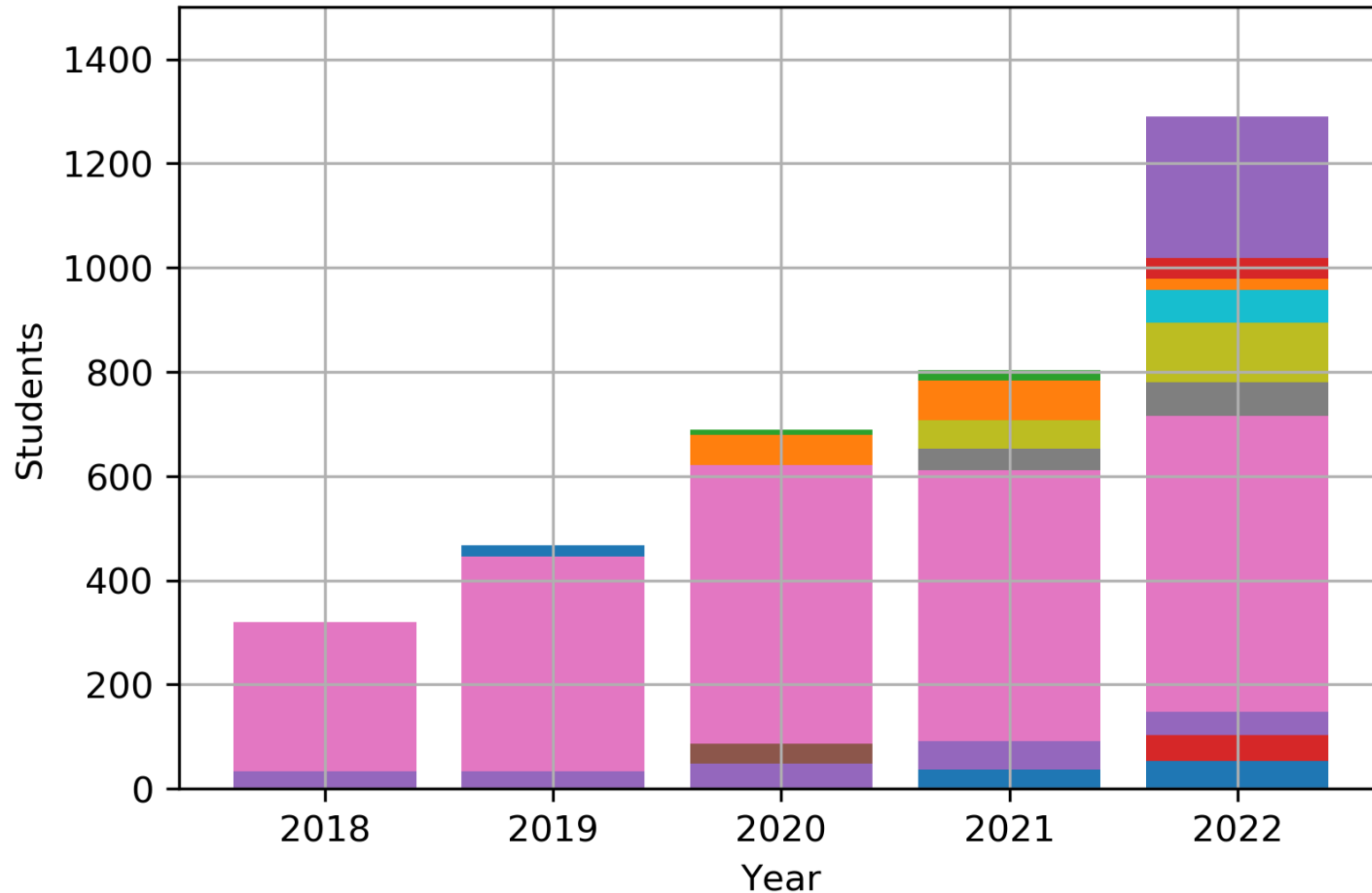


Scaling Up Exams and Assignments with Jupyter Notebooks



Students enrolled in courses with Jupyter Notebook assignments

Statistics from 2018 to WS2022 on assignment server





Scaling Up to Large Classes: Challenges and Solutions



From courses with 20 students to courses with 500 students with nbgrader

- Started using nbgrader in 2017 locally and 2018 on a JupyterHub
- Problems were amplified:
 - Students deleted, copied or reordered cells
 - Students put their answers in the wrong cells
 - Many submissions required manual fixing to make them autogradable



Pros and Cons of nbgrader for Grading Jupyter Notebooks

- Benefits:
 - Frontend for manual grading
 - Autograding
 - Feedback for students
 - Adds structure via different cell types for solutions, tests and descriptions
 - Exchange for releasing, collecting and submitting assignments
- Limitations:
 - Appears as a normal notebook to students, leading to potential issues with accidental **deletion, reordering, or copying of cells**
 - Graders do usually not create proper ids for cells and may **forget** to switch cell types
 - Customizing nbgrader's grading process can be complex, making it challenging to meet the unique needs of different courses and instructors.



e2xgrader

- Add-on for nbgrader
- Customizable components
- Extensions to restrict students from changing the structure of the notebook
- New cell types
- Customizable autograders
- Authoring component for creating notebooks from small units using templates
- Can switch between teacher mode, student mode and student exam mode

e^2x

New Cell Types

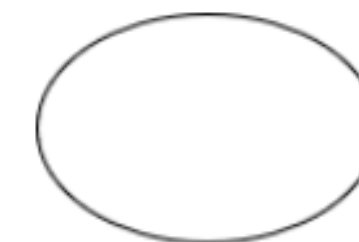
- Custom Markdown Cells:
- Single choice
- Multiple choice
- Interactive diagrams via diagrams.net
- Answers are stored in the metadata of the cell. The cells render function is replaced by a custom renderer.



What is the square root of 64?

- 2
- 4
- 6
- 8

UML Use Case Diagram A)

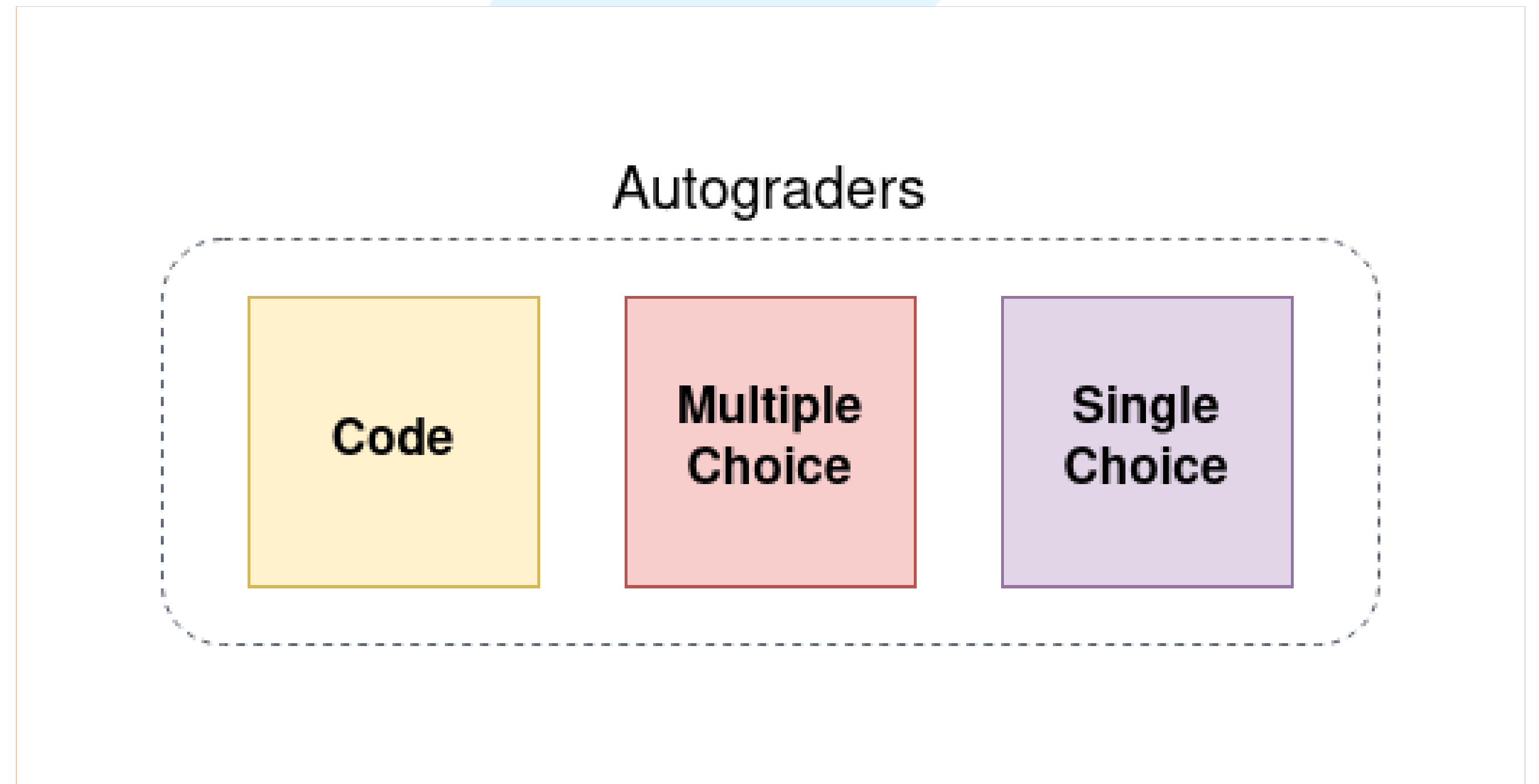


Actor

Edit Diagram

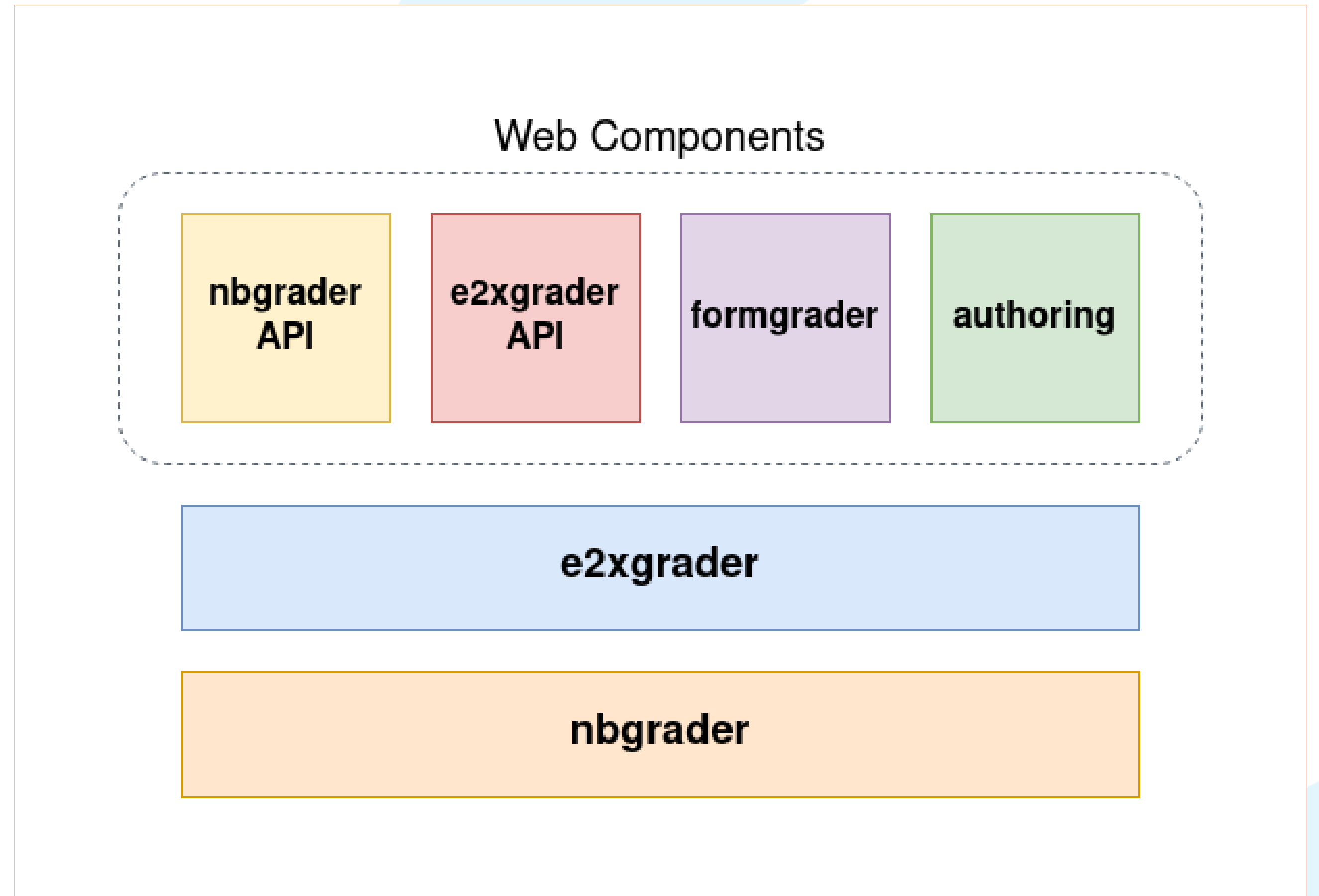
Custom Autograders

- e2xgrader allows for custom autograders to be added via the `nbgrader_config.py` file
- Multiple choice and single choice cell graders have already been implemented in e2xgrader
- The code grader has also been adapted to fit e2xgrader's architecture
- The appropriate grader is automatically chosen based on the cell type
- If no custom grader is available, e2xgrader falls back to the nbgrader autograder
- This allows for greater flexibility and customization in grading student assignments.



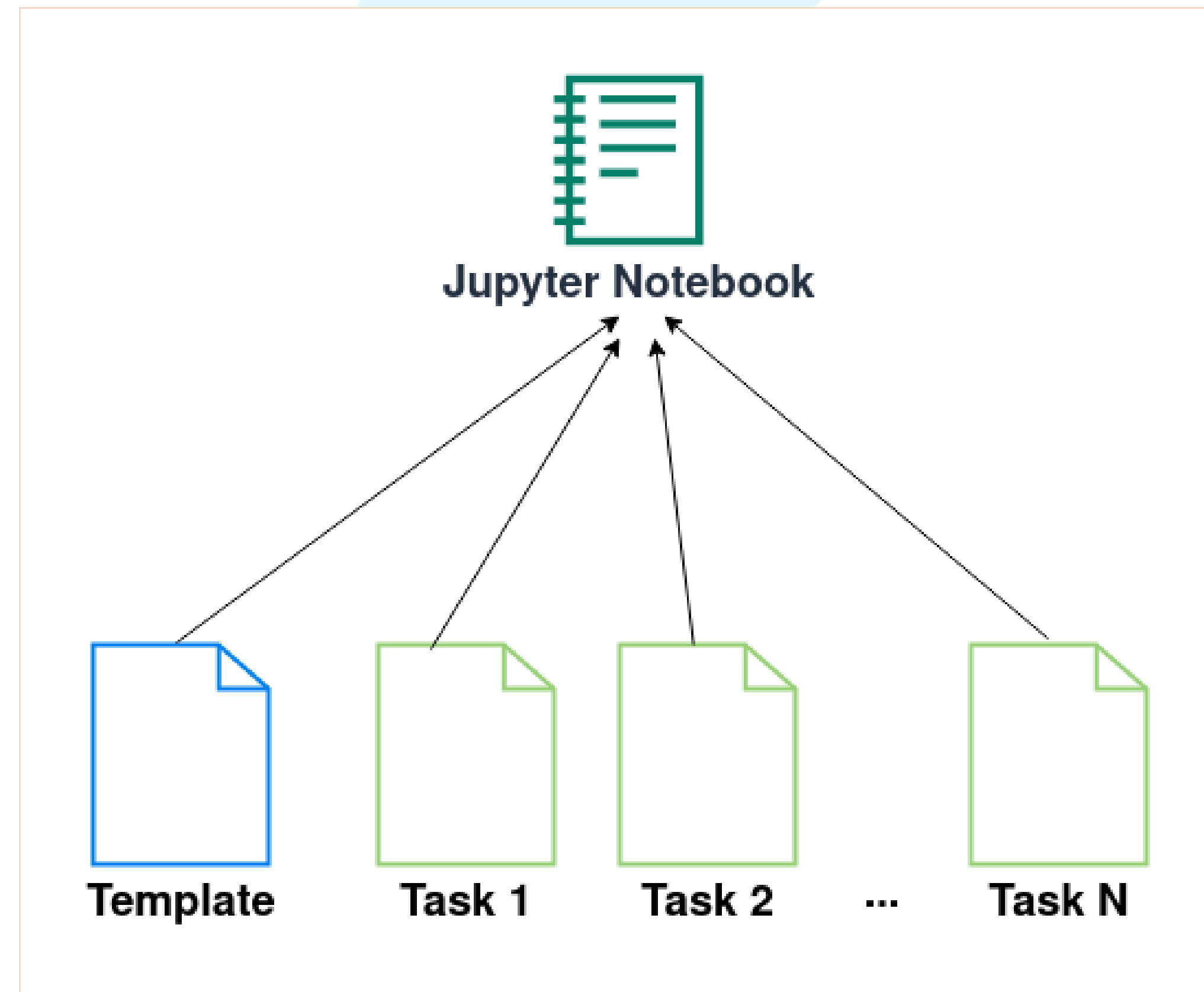
Customizable nbgrader with e2xgrader's Component-Based Architecture

- Different users have different requirements for nbgrader
- With e2xgrader's Component-Based Architecture, users can write custom plugins to change specific parts of nbgrader
- This approach avoids the need to maintain a fork of nbgrader
- Users can customize nbgrader to their needs without having to understand the entire nbgrader codebase



Authoring component

- Create assignment notebooks from collections of tasks
- Use a template to define the structure of the notebook (header, footer, standard imports)
- Version control for tasks to track changes
- Custom question templates
- Automatic id for questions

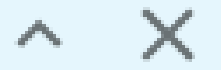




Task Pools

ⓘ Instructions

Task pools are collections of tasks about the same topic. A task consists of a collection of related questions.

[EXPORT](#)

Name	# Tasks	Version Control	
MC_Markov	8	Yes	
MC_ZV_Stetige_Verteilungen	10	Yes	
MC_Kombinatorik	5	Yes	
MC_Descriptive_Statistik	14	Yes	
MC_Bedingte_Wahrscheinlichkeiten	5	Yes	
Testen	1	Yes	
LinReg	1	Yes	
MC_Testen	9	Yes	
MC_Basic_und_FehlerBias	6	Yes	
MC_ZV_Diskrete_Verteilungen	11	Yes	

Rows per page: 10 ▾ 1-10 of 12 < >

[ADD TASK POOL](#)



ASSIGNMENTS

TASKS

TEMPLATES

Task Pools > Testen

Instructions

A task is a single Jupyter notebook consisting of a task with several questions (i.e. Task 1.1, Task 1.2, Task 1.3).

EXPORT

Search...

Name	# Questions	# Total Points	Version Control
t_Test	5	12	modified  


Rows per page: 10 1-1 of 1


ADD TASK

ASSIGNMENTS TASKS TEMPLATES

Task Pools > Test1 > Squares

File Edit View Insert Cell Kernel Widgets Help Trusted Kernel ○

e²x  Add Question Add Files Manage Tags ▶ Run ■ ↺ ▶▶ ⬆ ⬇ Total points: 0

Code (Autograded) 
Code (Manual)
Diagram
Freetext
Multiple Choice
Single Choice
Upload Files
Reset Namespace Cell

ID: Squares_Header Read-only

Square

Insert Question - Code (Autograded)

Name:	<input type="text" value="Squares_A"/>
Points:	<input type="text" value="5"/>

OK Cancel



File Edit View Insert Cell Kernel Widgets Help Trusted Kernel

e²x Add Question Add Files Manage Tags Run Total points: 5

🔒 ID: Squares_Header Read-only

Squares

🔒 ID: Squares_A_description0 Read-only

Question

Please write your question here!

In []: ID: Squares_A Autograded answer

```
### BEGIN SOLUTION
# Answer

# Please write your code answer here!

### END SOLUTION
```

In []: 🔒 Points: 5 ID: test0_Squares_A Autograder tests

```
### BEGIN HIDDEN TESTS
# Test

# Please write your code test here!

### END HIDDEN TESTS
```

Grading component

- Add graders for new cell types
- Grading using a pen by drawing on the student solutions
- Per task grading view for easier grading
- Export student grades to csv



Squares A) [10 points]

Please implement the function square which takes a number and returns a square.

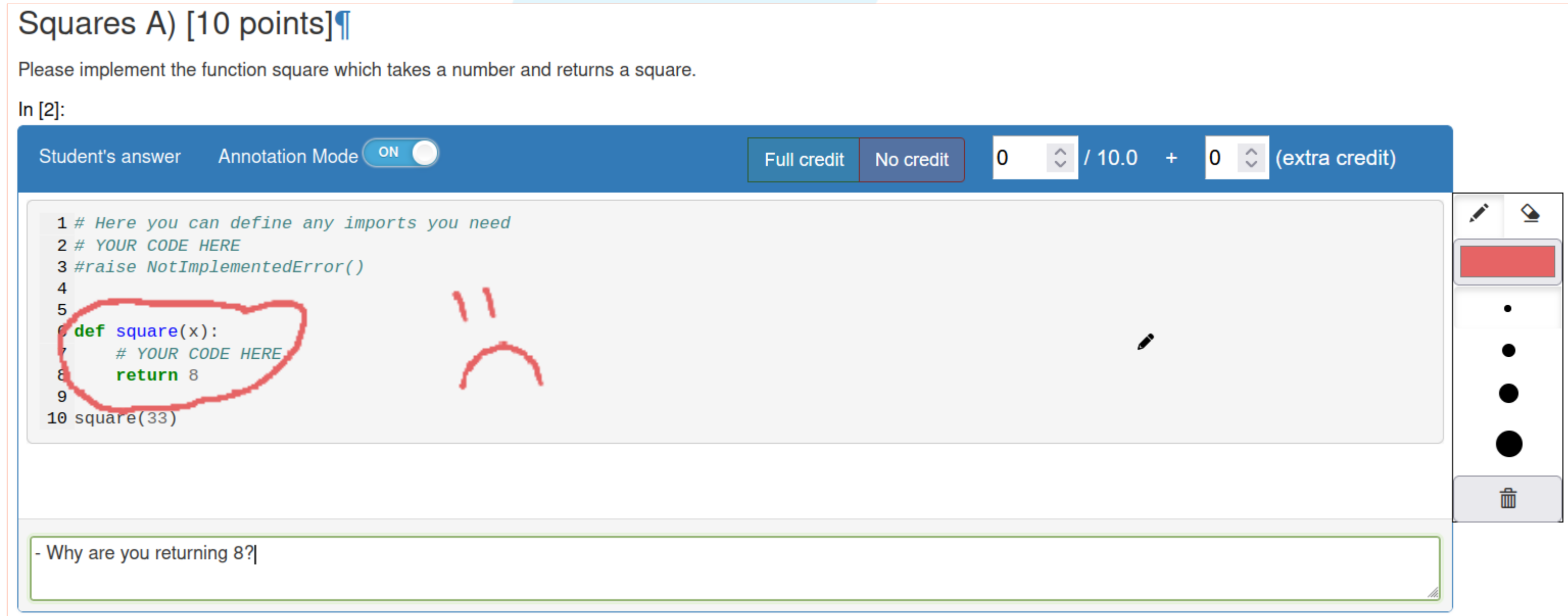
In [2]:

Student's answer Annotation Mode

Full credit No credit 0 / 10.0 + 0 (extra credit)

```
1 # Here you can define any imports you need
2 # YOUR CODE HERE
3 #raise NotImplementedError()
4
5
6 def square(x):
7     # YOUR CODE HERE
8     return 8
9
10 square(33)
```

- Why are you returning 8?




Student Exam Mode

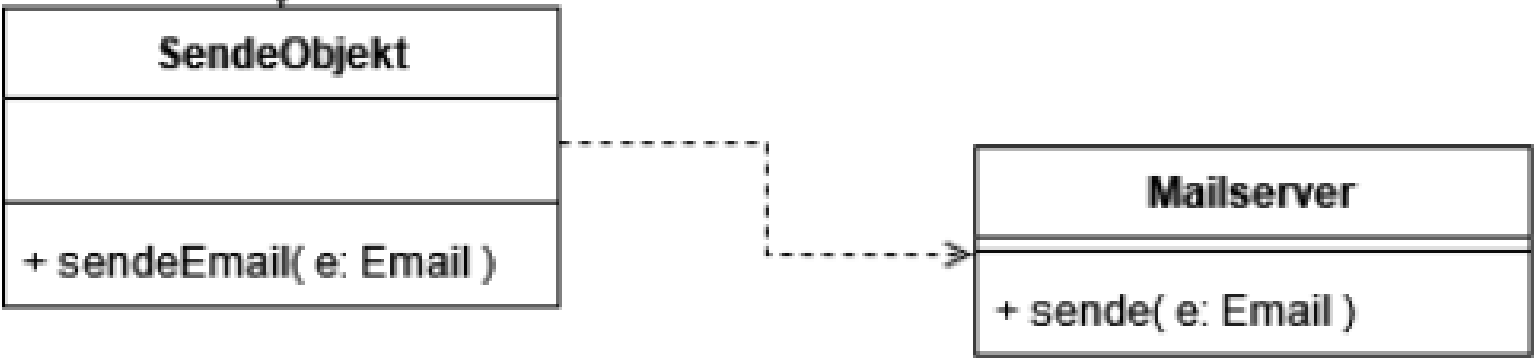
- Students can't delete notebooks via the frontend
- Students can't modify the structure of a notebook
- Add toolbar for submitting within the notebook

jupyterhub SE1-Klausur Last Checkpoint: 02/28/2023 (autosaved)
Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help
Trusted

e²x


▶ Run
■
↺
▶▶
Java Syntax ○
Advanced Submit ↗



```

classDiagram
    class SendeObjekt {
        + sendEmail( e: Email )
    }
    class Mailserver {
        + send( e: Email )
    }
    SendeObjekt ..> Mailserver
    
```

Edit Diagram

Anwendung von Design Patterns C) [9 Punkte]

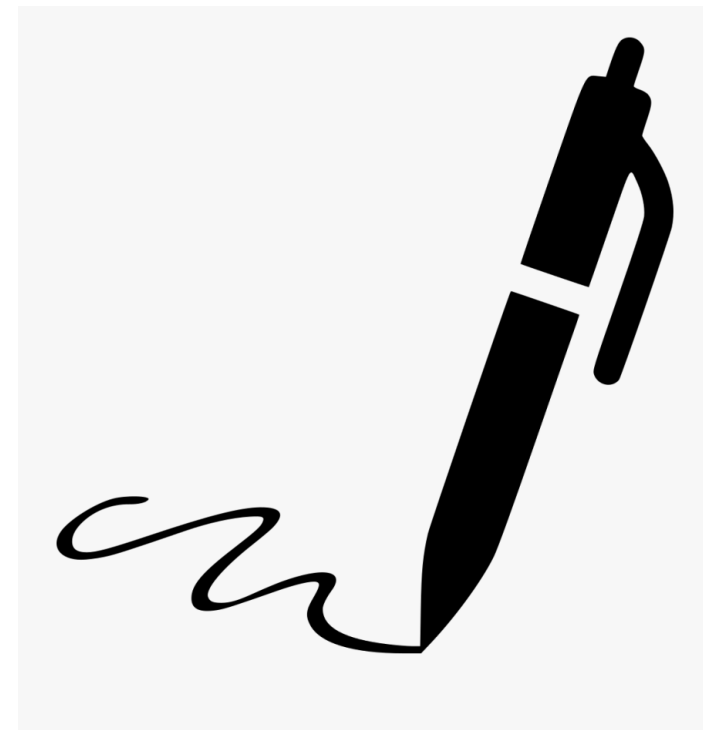
Geben Sie den Java Source Code der Klasse SendeObjekt an. Es muss der Rumpf der Klasse, die Attribute, die Methoden und die Implementierungen der Methoden angegeben werden.

In []: Run

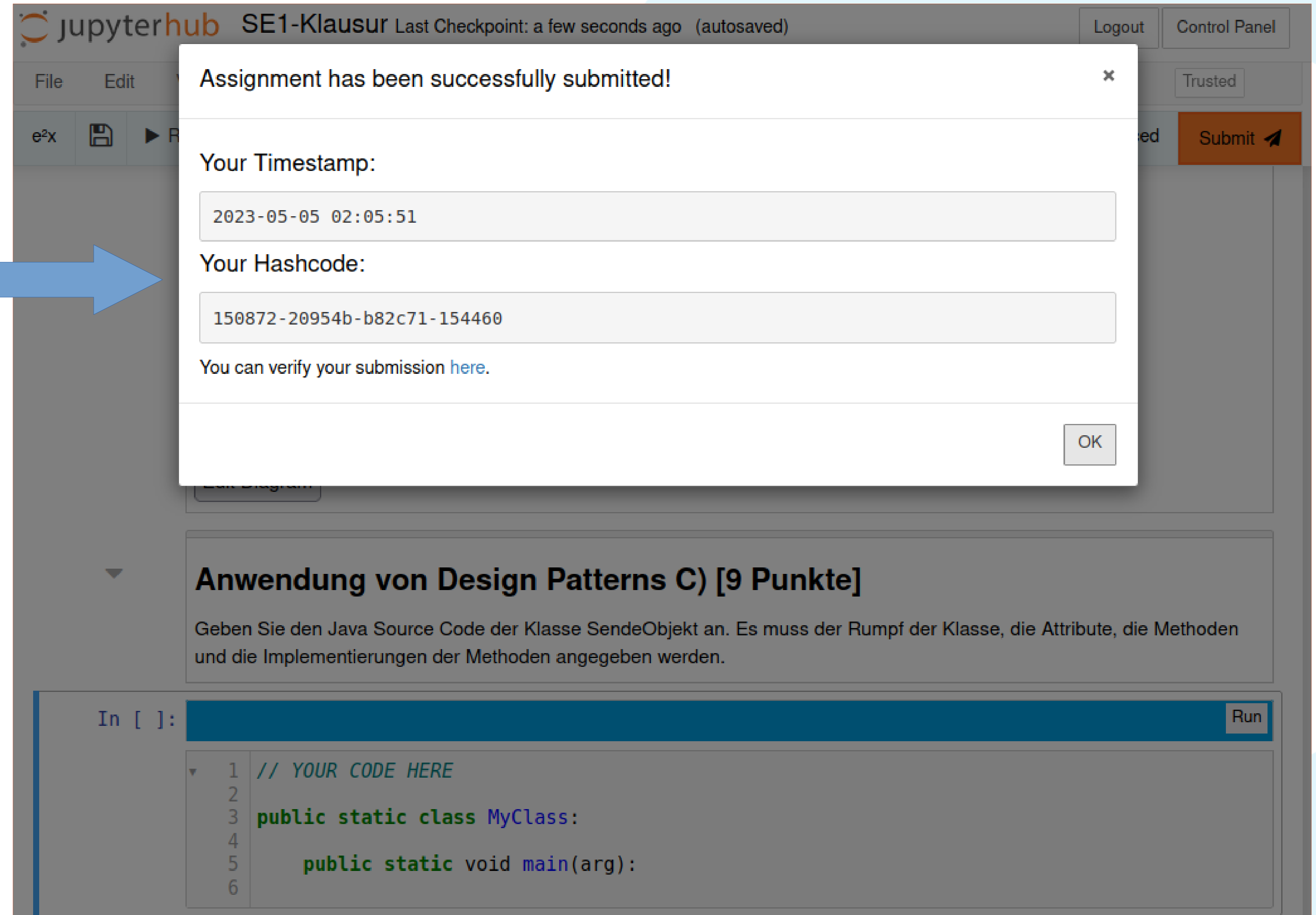
```

1 // YOUR CODE HERE
2
3 public static class MyClass:
4     public static void main(arg):
5
6
    
```

Student Exam Mode



- Hashcode to guarantee the assignment is not changed during the grading process
- Students can look at a html copy of the submitted notebook



The screenshot shows a JupyterHub interface for a course titled "SE1-Klausur". A modal dialog box is displayed in the center, titled "Assignment has been successfully submitted!". The dialog contains the following information:

- Your Timestamp:** 2023-05-05 02:05:51
- Your Hashcode:** 150872-20954b-b82c71-154460
- A link: "You can verify your submission [here](#)."
- An "OK" button at the bottom right.

Below the dialog, the JupyterHub interface shows a code editor with the following code:

```
In [ ]:
1 // YOUR CODE HERE
2
3 public static class MyClass:
4     public static void main(arg):
5
6
```

The code editor also shows a "Run" button on the right side.



Summary

- New Cell Types
- Component-based Architecture
- Customizable Autograders



Thank you for your attention!

CONTACT

Tim Metzler

Mohammad Wasil

Dr. Paul G. Plöger

Mail: e2x@inf.h-brs.de
github.com/Digiklausur