Classroom 50: Ein Klassenzimmer einrichten

Von einem laufenden Classroom 50 bis zu Noten in Moodle — für ein **beliebiges** Klassenzimmer. Reihenfolge einhalten; jeder Schritt ist einzeln prüfbar.

Platzhalter	Bedeutung	Beispiel
<ORG>	GitHub-Organisation der Klasse	`m320-ix25`
<CLASSROOM>	Kurzname des Klassenzimmers = Ordner im Config-Repo	`m320-ix25`
<SLUG>	Slug einer Aufgabe	`m320-lu04-a4-objektkommunikation`
<TAG>	gepinnte pygrader50-Version	`v2.0.0`

<ORG> und <CLASSROOM> sind oft gleich, müssen es aber nicht sein — der Kurzname steht in `<CLASSROOM>/classroom.json`.

0. Voraussetzungen

Ein `classroom50`-Config-Repo unter <ORG> mit laufenden Workflows (`publish-pages`, `collect-scores`), erzeugt von `gh teacher classroom add`.

```
gh extension install foundation50/gh-teacher
gh auth refresh -h github.com -s admin:org,read:org,repo,workflow
gh teacher classroom list <ORG>
```

Ohne `admin:org` bricht **jeder** `gh teacher`-Aufruf mit einer Scope-Meldung ab. Der Refresh öffnet den Browser und braucht ein interaktives Terminal.

1. Engine veröffentlichen

Nur nötig, wenn pygrader50 selbst weiterentwickelt wird — für ein neues Klassenzimmer pinnt man einen bestehenden Tag und überspringt diesen Schritt.

```
git tag v2.0.0 && git push origin main --tags
```

Der Tag ist das, was die Klassenzimmer pinnen. Ohne Tag kein reproduzierbares Semester: `main` würde sich unter laufenden Bewertungen verändern.

Prüfen, dass der Pin installierbar ist — sonst scheitert er erst im ersten Studi-Lauf:

```
python3 -m venv /tmp/pin && /tmp/pin/bin/pip install \
  "pygrader50 @ git+https://github.com/BZZ-Commons/pygrader50@<TAG>"
```

2. Default-Autograder setzen

Pro Klassenzimmer einmal:

```
gh teacher autograder set-default <ORG> <CLASSROOM> --from
bootstrap/autograder.py
```

Das legt `<CLASSROOM>/autograder.py` im Config-Repo ab; `publish-pages` stellt die Datei auf die Pages-Site, wo `runner.py` sie bei jeder Abgabe holt.

Die gepinnte Version steht in `bootstrap/autograder.py`:

```
VERSION = 'v2.0.0'
```

Ein Upgrade heisst später: Tag hochziehen, Zeile ändern, `set-default` erneut ausführen — pro Klassenzimmer, das mitziehen soll. Klassen können bewusst auf unterschiedlichen Versionen bleiben.

Vorher prüfen, was überschrieben wird:

```
gh teacher autograder show <ORG> <CLASSROOM>
gh teacher autograder list <ORG> <CLASSROOM>
```

Nachher prüfen, dass Pages die Datei ausliefert:

```
curl -sS -o /dev/null -w '%{http_code}\n' \
  https://<ORG>.github.io/classroom50/<CLASSROOM>/autograder.py
```

Das Klassen-Segment im Pfad ist Pflicht. `…/classroom50/autograder.py` **ohne** `<CLASSROOM>/` liefert 404 — eine beliebte Fehlspur beim Prüfen. Der Deploy braucht nach dem Push rund 30 Sekunden.

Dann in einem Test-Repo etwas pushen: Das Release muss eine echte Punktzahl zeigen (z.B. `0/7`) statt `0/0`. Im Job-Log steht, welche Konfigurationsdateien gefunden wurden.

3. Aufgaben-Konfiguration ablegen

Die drei Dateien können bleiben, wo sie sind (`.github/autograding/` im Studi-Repo) — dann ist nichts zu tun. Manipulationssicher wird es erst im Config-Repo, weil Lernende ihr eigenes `.github/` bearbeiten können.

Aufbau, Vorrangregeln und Beispiele: Wie bewertet wird.

```
classroom50/
└── <CLASSROOM>/
    └── autograders/
        └── <SLUG>/
```

```
            ├── unittests.json
            ├── lint.json
            └── pylintrc
```

Prüfen: Punkte in der Bundle-`unittests.json` ändern, pushen, neu bewerten lassen — die neue Maximalpunktzahl muss im Release stehen.

4. Template-Repos migrieren

Nur bei Klassen, die von GitHub Classroom kommen — vollständig beschrieben unter Migration von GitHub Classroom. Kurzfassung:

```
scripts/remove-legacy-classroom-yml.sh <ORG> <CLASSROOM>            # Trockenlauf
scripts/remove-legacy-classroom-yml.sh <ORG> <CLASSROOM> --apply    # löschen
```

Danach den alten Moodle-Token **löschen und rotieren**.

5. Moodle-Übertrag einrichten

5.1 Workflow einbauen

```
cp classroom50/moodle-sync.yaml <config-repo>/.github/workflows/moodle-sync.yaml
```

Eigener Dateiname, keine bestehende Datei anfassen: `gh teacher` überschreibt die mitgelieferten Skeleton-Workflows bei einem Refresh.

Die Datei ist klassenzimmer-neutral. Bleibt die Eingabe `classroom` leer, wird jeder Ordner mit einer `scores.json` übertragen — ein Config-Repo mit mehreren Klassen braucht keine zweite Kopie. Anzupassen ist nur der gepinnte Tag, falls er von `bootstrap/autograder.py` abweicht.

5.2 Zugangsdaten, Moodle-Seite, erster Lauf

Siehe Noten nach Moodle übertragen. Kurz:

1. Secret `MOODLE_TOKEN` und Variable `MOODLE_URL` im **classroom50-Repo** setzen.
2. In Moodle je Aufgabe eine Aktivität *External Assignment* mit dem Slug als Namen.
3. **Actions → Moodle Sync → Run workflow** mit `dry_run`, Ausgabe prüfen.
4. Dasselbe ohne `dry_run`. Danach liegt `<CLASSROOM>/moodle-state.json` im Repo.

6. Betriebsfallen

Vier Verhaltensweisen der Classroom-50-Seite, die man einmal wissen muss — `scores.json` wird nie aufgeräumt, eingesammelt wird nach Teams statt nach Roster, das Roster heisst `roster.csv`, und der Service-Token braucht Administration-Rechte. Ausgeführt unter Klassenzimmer und Aufgaben verwalten.

Verantwortlichkeiten

Was	Wo	Warum dort
Bewertungs-Engine, Moodle-Übertrag	`BZZ-Commons/pygrader50`	eine Quelle, versioniert, für alle Klassenzimmer
Default-Autograder, Aufgaben-Bundles, Moodle-Token	`<ORG>/classroom50`	von der Lehrperson kontrolliert, für Lernende nicht schreibbar
Startcode, Tests, Musterlösung	Template-Repo	gehört zur Aufgabe
Abgabe	Studi-Repo	gehört den Lernenden