Classroom 50: CLI-Referenz

Zwei Kommandos aus pygrader50: eines bewertet, eines überträgt. Dazu ein Migrationsskript.

python -m pygrader50 — bewerten

Wird vom Classroom-50-Runner im Studi-Checkout aufgerufen. Keine Argumente — alles kommt aus der Umgebung.

Umgebung

Variable	Bedeutung	Pflicht
`CLASSROOM`	Kurzname des Klassenzimmers	ja
`ASSIGNMENT`	Slug der Aufgabe	ja
`SUBMISSION_TAG`	`submit/<Zeitstempel>-<Kurz-SHA>`	ja
`OWNER` / `USERNAME`	GitHub-Login der besitzenden Person	ersatzweise `GITHUB_ACTOR`
`ASSIGNMENT_TYPE`	`individual` oder `group`	Vorgabe `individual`
`COMMIT_URL`, `RELEASE_URL`, `REVIEW_URL`	Links fürs Payload	ersatzweise aus `GITHUB_SERVER_URL` / `GITHUB_REPOSITORY` / `GITHUB_SHA`
`RUNNER_TEMP`	Wurzel für das entpackte Bundle	ohne die Variable entfällt die Bundle-Suche
`PYGRADER50_CONFIG_DIR`	zusätzlicher Config-Ort, nur lokal	nein

Ausgabe

Datei	Inhalt
`./result.json`	`classroom50/result/v1`, vom Runner ans Release gehängt
`./release-body.md`	Feedback-Tabellen, Release-Text und Job-Summary

Exit-Codes

Code	Bedeutung
`0`	Bewertung abgeschlossen — **auch** wenn alle Tests scheitern
`1`	Infrastrukturfehler: Umgebung fehlt, Konfiguration kaputt, Absturz. Der Runner zeichnet die Abgabe als `error` auf

Eine fehlende Bewertungs-Konfiguration ist **kein** Fehler: 0/0 mit Warnung im Log, Exit 0.

Lokal ausprobieren

```
cd /pfad/zum/studi-repo
CLASSROOM=m320-ix25 \
```

```
ASSIGNMENT=m320-lu04-a4-objektkommunikation \
SUBMISSION_TAG=submit/2026-08-13T12-00-00Z-abc1234 \
OWNER=anna \
python -m pygrader50
cat release-body.md
```

python -m pygrader50.moodle — übertragen

Liest `scores.json` aus dem Config-Repo und schickt die Punkte an Moodle. Je Kombination aus Aufgabe und Person geht die **neueste** Abgabe raus.

```
python -m pygrader50.moodle SCORES [Optionen]
```

Argumente und Optionen

Option	Wirkung
SCORES	Pfad zu <CLASSROOM>/scores.json (Pflicht)
–assignment SLUG	nur diese Aufgabe
–user LOGIN	nur diesen GitHub-Login
–state PFAD	Zustandsfile; ohne Angabe wird jedes Mal alles übertragen
–force	auch Unverändertes erneut senden
–dry-run	nur anzeigen, nichts senden; funktioniert ohne Zugangsdaten
–no-feedback	ohne Feedback-Text (spart einen API-Aufruf pro Abgabe)

Der bei –user angegebene Login muss im Moodle-Kurs eingeschrieben sein. Der eigene Lehrer-Account ist es meist nicht — Moodle antwortet dann `No matching assignment found`, obwohl die Aktivität existiert.

Umgebung

Variable	Bedeutung
MOODLE_URL	Basis-URL der Moodle-Instanz, z.B. https://moodle.bzz.ch
MOODLE_TOKEN	Webservice-Token
MOODLE_FUNCTION	Vorgabe mod_externalassignment_update_grade
GH_TOKEN / GITHUB_TOKEN	optional, liest die Release-Bodies für den Feedback-Text

Ohne –dry-run sind MOODLE_URL und MOODLE_TOKEN Pflicht.

Zustandsfile

```
{
  "schema": "pygrader50/moodle-state/v1",
  "entries": {
    "m323-lu01-a02-imperativer-ggt/graphics80": {
      "submission": "submit/2026-08-13T08-41-09Z-35bdcb2",
      "score": 5,
```

```
      "max-score": 7
    }
  }
}
```

Übersprungen wird nur, wenn Abgabe **und** Punktzahl identisch sind — eine Nachbewertung derselben Abgabe geht also erneut raus. Nur erfolgreiche Übertragungen werden vermerkt; gescheiterte versucht der nächste Lauf wieder.

Exit-Codes

Code	Bedeutung
`0`	alles übertragen oder übersprungen
`1`	mindestens eine Übertragung scheiterte, oder Zugangsdaten/Datei fehlen

Ein Fehler bricht den Lauf **nicht** ab: die übrigen Abgaben gehen trotzdem raus.

Beispiele

```
# Trockenlauf über alles, ohne Zugangsdaten
python -m pygrader50.moodle <CLASSROOM>/scores.json --dry-run --no-feedback

# Eine einzelne Person nachtragen
python -m pygrader50.moodle <CLASSROOM>/scores.json --user anna

# Eine Aufgabe nach einer Nachbewertung komplett neu schicken
python -m pygrader50.moodle <CLASSROOM>/scores.json --assignment <SLUG> --
force

# Wie der Nachtlauf
python -m pygrader50.moodle <CLASSROOM>/scores.json \
  --state <CLASSROOM>/moodle-state.json
```

Ohne lokalen Klon des Config-Repos reicht die Datei allein:

```
gh api repos/<ORG>/classroom50/contents/<CLASSROOM>/scores.json \
   -H 'Accept: application/vnd.github.raw' > scores.json
GH_TOKEN=$(gh auth token) python -m pygrader50.moodle scores.json --dry-run
```

Von Hand auslösen

```
gh workflow run moodle-sync.yaml --repo <ORG>/classroom50 \
  -f classroom=<CLASSROOM> -f dry_run=true
```

Oder im Config-Repo unter **Actions → Moodle Sync → Run workflow**. Bleibt das Klassenzimmer leer, laufen alle Ordner mit einer `scores.json`.

scripts/remove-legacy-classroom-yml.sh — migrieren

Entfernt den alten GitHub-Classroom-Workflow aus den Template-Repos einer migrierten Klasse und aus den bereits angenommenen Studi-Repos.

```
scripts/remove-legacy-classroom-yml.sh <ORG> <CLASSROOM> [--apply]
```

Aufruf	Wirkung
ohne `–apply`	Trockenlauf, listet jedes Ziel als `would rm` oder `absent`
mit `–apply`	löscht `.github/workflows/classroom.yml`, ein Commit pro Repo

Die Zielliste kommt aus dem `template`-Block von `<CLASSROOM>/assignments.json`, **nicht** aus dem Repo-Listing der Template-Organisation. Wiederholbar; fehlende Dateien sind `absent`, kein Fehler. Exit 1, sobald ein Repo scheitert.

Hintergrund und die nötige Token-Rotation: Migration.

scripts/sync-template-pins.py — Pins aktualisieren

Hebt die Werkzeug-Versionen in den Template-Repos einer Klasse und legt `.python-version` an.

```
scripts/sync-template-pins.py <ORG> <CLASSROOM> [--apply]
```

Was	Verhalten
`pylint`, `pytest`	werden überall gesetzt, fehlende Zeilen angehängt
`httpx`, `pytest-asyncio`	nur dort gehoben, wo sie schon stehen
alles andere	bleibt Zeile für Zeile erhalten, inkl. Kommentaren
`.python-version`	wird auf die konfigurierte Version gesetzt

Die Versionen stehen als Block am Kopf des Skripts und werden einmal pro Semester angefasst. Wiederholbar: ein aktuelles Template meldet `ok`.

`pytest-asyncio` muss mit `pytest` mitziehen — 0.23.8 und pytest 9 lassen sich nicht gemeinsam auflösen. Pakete, welche die Lernenden im Rahmen der Aufgabe selbst eintragen sollen (Flask in den lu06-Aufgaben), gehören **nicht** in die Pin-Liste.

scripts/publish-wiki.py — diese Seiten aktualisieren

Spiegelt den Ordner `wiki/` aus dem Repo hierher. Pfad = Seiten-ID: `wiki/howto/git/classroom50/start.txt` wird zu `howto:git:classroom50:start`.

```
scripts/publish-wiki.py [--apply] [--summary "Text"]
```

Geschrieben wird nur, was abweicht — ein erneuter Lauf meldet alles als `gleich` und hinterlässt keine leeren Versionen. Zugangsdaten kommen aus der Umgebung: `DOKUWIKI_TOKEN`, sonst

`DOKUWIKI_USER` und `DOKUWIKI_PASSWORD`.

Ein untauglicher API-Token wird **nicht abgelehnt, sondern ignoriert** — die Aufrufe laufen dann als Gast weiter und scheitern erst beim Schreiben mit einer 401, die wie ein falsches Passwort aussieht. Das Skript prüft deshalb vor dem Schreiben per `core.whoAmI`, als wer es angemeldet ist, gibt das aus und fällt bei totem Token auf die Anmeldung per `core.login` zurück.

gh teacher — Klassenzimmer verwalten

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

gh teacher classroom list <ORG>
gh teacher autograder show <ORG> <CLASSROOM>
gh teacher autograder list <ORG> <CLASSROOM>
gh teacher autograder set-default <ORG> <CLASSROOM> --from
bootstrap/autograder.py
gh teacher rotate-service-token <ORG>
```

Vollständige Referenz: Wiki von foundation50/classroom50.