Exceptions, Assertions

6.100 LECTURE 20

SPRING 2026

Announcements

- Pset 5 checkoff due next Tue 4/21
  - CPW this weekend, next Monday is holiday
- Pset 6 due next Fri 4/24

Debugging

Social debugging

- Find someone to talk to
  - explain what you're trying to do
  - explain the bug you see
  - explain why you're stuck
- If no one is available, talk to yourself
- If you don't like talking to yourself, talk to a rubber duck

Make the code run

- lec_pre_1
  - no returns needed in **`edit_score()`** and **`add_score()`**
- lec_pre_2
  - indexing into score lists should use assignment number minus 1
- lec_pre_3
  - use constant variables to avoid misspelling str labels
- lec_pre_4
  - use the correct user/kerb strings (harder to protect against)

Fix results that don't make sense

- lec_pre_5
  - scores and weights need to have equal length
- lec_pre_6
  - valid scores to add/edit should be between 0 and 10
- lec_pre_7
  - also check valid assignment numbers, valid user strings
  - wrap checks in a **`check_valid()`** helper function
- lec_pre_8
  - avoid aliasing when copying records
  - avoid exposing dict structure to user, properly manage in a **`copy_record()`** function

Exceptions

Raise exceptions and catch/handle them

- lec_1
  - rather than just returning **`False`** from **`check_valid()`**, would like to signal when something isn't right
  - raise **`ValueError()`** for bad user/kerb
  - causes program to crash: execution stops and unwinds all the frames, including the global frame
- lec_2
  - catch and handle the error in **`review_scores()`**
  - **`try`** block enables exception catching
  - **`except`** block catches/"absorbs" the **`ValueError`** and runs it body
  - "fail gracefully" when trying to edit score for nonexistent user **`"vibegirl"`**

Convert boolean returns to raised exceptions

- lec_3
  - **`copy_record()`** calls **`check_valid()`**, can take advantage of errors raised by **`check_valid()`**
  - want **`check_valid()`** on **`old_user`** to not fail
  - but want **`check_valid()`** on **`new_user`** to actually fail
  - if it doesn't, use an **`else`** block to raise a new **`ValueError()`**
  - in **`review_scores()`**, second attempt to **`copy_record()`** will fail
- lec_4
  - **`check_valid()`** can also raise **`ValueError()`**s for invalid assignment numbers or scores
  - in **`review_scores()`**, attempts to edit Tony's (**`"richboi"`**) scores in will fail and be caught

Use exceptions raised by built-in operations

- lec_5
  - Python's built-in operations already take advantage of exceptions
  - can catch **`KeyError`** to detect nonexistent user in scorebook dict
  - "ask for forgiveness" vs "ask for permission" philosophy
- lec_6 and lec_7
  - catch **`IndexError`** to detect bad assignment number
  - catch **`TypeError`** to detect non-numeric scores
  - sometimes the extra complexity of exception handling might not be worth it
- lec_8
  - clean up **`review_scores()`** by moving **`try-except`**s to within operations **`get_score()`**, **`add_score()`**, **`edit_score()`**

Assertions

Check program safety throughout

- lec_9
  - **`weighted_average()`** could raise an exception if provided scores and weights with inconsistent lengths
  - but it's such a "low-level" function that there's no good reason it should ever be passed bad input
  - **`assert`** statements do nothing if condition is **`True`**, but raise an **`AssertionError`** if **`False`**
  - assertions are meant to verify that program state along the way still makes sense
  - Python uses the exception mechanism for when assertions fail, but you should never catch an **`AssertionError`**

Using exceptions vs assertions

- lec_10
  - what if **`weighted_average()`** receives empty scores/weights?
  - this could be interpreted as no submitted assignments yet, and not a bad input
  - thus, should return a default value (**`0`**) rather than raising **`AssertionError`**
  - good opportunity to catch **`ZeroDivisionError`**
- lec_11
  - another **`assert`** opportunity: make sure **`category`** string actually obeys the **`PSETS/EXAMS`** convention we set for ourselves

Next time

- **Today's** takeaway
  - **exceptions** are a mechanism to signal failure
    - they cause the stack frames to unwind until a **`try-except`** block catches and handles it
  - many ways to use them, the additional complexity may not always be worth it
  - be clear on their rules of the mechanism
  - use **assertions** liberally throughout your code to catch bugs early
- No class next **Monday**
- Next **Wedesday**
  - extend classes with **inheritance**
  - **Pset 7** will exercise