6.009: Fundamentals of Programming

Lecture 7: Encapsulation

[Slides, code on course website]

*Suggestion:* download lecture7.py from Calendar page, open your laptop, and work on the lecture problems as we do them!
Using Classes to Encapsulate Data

Encapsulate: store data using some convenient internal representation; provide interface functions for accessing data in useful ways.

Pythonic interpretation: store data using instance variables of an object; provide methods for accessing data in useful ways.

Example: database for course assignments

<table>
<thead>
<tr>
<th>Name</th>
<th>SID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>2</td>
</tr>
<tr>
<td>Charlie</td>
<td>3</td>
</tr>
</tbody>
</table>

Pset info:

<table>
<thead>
<tr>
<th>PID</th>
<th>Max pts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>PID</th>
<th>Pts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>
Version 1: straightforward approach

- Create classes for the various objects
  - Student: SID, Name
  - Pset: PID, max points
  - Grade: SID, PID, points earned
  - DB: lists of Students, Psets, Grades

- Write DB methods for various data operations
  - Adding new students, psets, grades
  - Look-up using IDs

- Issues
  - Searching through lists is sloooow.
Version 2: create indices

• Use a dict for fast access
  – D[key] time is independent of dictionary size
  – List search time is proportional to length of list
  – In database parlance: create an index
    • Replace slow search with fast lookup

• Issues
  – Not interested in grades per se; always access using SID or PID
Version 3: grade info per student & pset

• Get rid of global grade list
  – For each Student use a dict: pid -> Grade object
  – For each Pset use use a dict: sid -> Grade object
  – But only create on Grade object for each grade!

• Add appropriate methods for students and psets

• Issues
  – How to reduce cost of an expensive query?
Version 4: avoid duplicate work

- Precompute some or all of the needed data
- Cache (save) results of expensive operations and don’t recompute unless necessary
  - When to *invalidate* the cache, i.e., when to throw away saved data?

- Issues
  - Need to support special access methods
    - e.g. create a gradesheet for a pset that’s sorted by student name
  - Fix #1: sort, but cache result
    - But what happens if cache is frequently invalidated?
  - Fix #2: linked lists, kept in desired order
Version 5: linked lists

item1 \rightarrow item2 \rightarrow item3

previous \rightarrow current

Item2.5

None