Designing Types

- **Type**: set of values and the operations on them
  - `int`: set; `ops`: +, -, *, /, ...
  - `Time`: times of day; `ops`: time span, before/after, ...
  - `Worker`: all possible workers; `ops`: hire, pay, promote, ...
  - `Rectangle`: set of axis-aligned rectangles in 2D; `ops`: contains, intersect, ...
- To define a class, think of a **real type** you want to make
  - Python gives you the tools, but does not do it for you
  - Physically, any object can take on any value
  - Discipline is required to get what you want

From first day of class!

Planning out a Class

- **Class Invariant**: States what attributes are present and what values they can have.
  A statement that will always be true of any Time instance.

- **Method Specification**: States what the method does.
  Gives preconditions stating what is assumed true of the arguments.

Implementing an Initializer

- **Self initialization**
  - This true to start
  - You put code here
  - This should be true at the end

Implementing a Method

- **Self increment**
  - This is true to start
  - What we are supposed to accomplish
  - This is also true to start
  - You put code here
  - This should be true at the end

Role of Invariants and Preconditions

- They both serve two purposes
  - Help you think through your plans in a disciplined way
  - Communicate to the user how they are allowed to use the class
- Provide the interface of the class
  - Interface b/w two programmers
  - Interface b/w parts of an app
- Important concept for making large software systems
  - …who might well be you!

**Invariants & Preconditions**

<table>
<thead>
<tr>
<th>Invariant</th>
<th>Precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>True of any Time instance.</td>
<td>States what attributes are present and what values they can have.</td>
</tr>
<tr>
<td>A statement that will always be true of any Time instance.</td>
<td>Gives preconditions stating what is assumed true of the arguments.</td>
</tr>
</tbody>
</table>

**Assert Statements**

- `assert <boolean>`: # Creates error if <boolean> false
- `assert <boolean>, <string>`: # As above, but displays <String>

- **Way to force an error**
  - Why would you do this?
- **Enforce preconditions!**
  - Put precondition as assert.
  - If violate precondition, the program crashes
- **Provided code in A3 uses asserts heavily**

---

10/5/14
**Data Encapsulation**

- **Idea:** Force the user to only use methods
- **Do not allow direct access of attributes**

### Setter Method

- Used to change an attribute
- Replaces all assignment statements to the attribute
- **Bad:**
  - $>>> t\text{.min} = 55$
- **Good:**
  - $>>> t\text{.testMin}(55)$

### Getter Method

- Used to access an attribute
- Replaces all usage of attribute in an expression
- **Bad:**
  - $>>> h = 60*\text{t.min}$
- **Good:**
  - $>>> h = 60*\text{t.testMin}()$