Designing Loops with Invariants
by example:
Normalizing Python indentation

Our coding standards (http://www.cs.uoregon.edu/Classes/12F/cis210/handouts/styleguide.php) say

Indentation
Your Python code should use four spaces for each level of indentation. Don’t use tabs, and especially don’t mix tabs and spaces. Your text editor may have an option to use the tab key on the keyboard to insert the right number of spaces.

We may also write an indentation adjuster as a programming assignment or in-class exercise.

Expanding tabs is easy enough:

```python
import sys
def infile():
    fname = sys.argv[1]
f = open(fname)
return f
def main():
    src = infile()
    for line in src:
        line = line.rstrip()  # Take away pesky newline
        line = line.expandtabs()  # Change tabs to spaces
        print(line)
main()
```

Normalizing indentation

```
Original:
| one |
| two |
| two |

Normalized:
| one |
| two |
| three |
| four |

Determine indentation level of each line, insert proper number of spaces
Preliminary: How many spaces is the input line indented?

```python
import sys
def infile():
    fname = sys.argv[1]
f = open(fname)
return f
def count_indent(txt):
    ### What goes here? ###
def main():
    src = infile()
    for line in src:
        line = line.rstrip().expandtabs()
        src_indent = count_indent(line)
        print(src_indent, line)
main()
```

Terminology: State

Something we keep track of (e.g., indentation) especially through iterations of a loop
An abstraction of one or more variables

Ex: “The state we keep in this loop is how many spaces we have counted at the beginning of the string”

In discrete math (and later in CS) you will study “state machines”; same underlying concept

Strategy

I want to normalize indentation.
What state do I need to keep as I loop through the lines?

```
one
  two
  two
  three
  four
  three
one
```

My first cut (wrong)

I kept two state variables:
Current source indentation
Current indent level

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>one</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(1)</td>
<td>two</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>(2)</td>
<td>two</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>(3)</td>
<td>two</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(2)</td>
<td>three</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(1)</td>
<td>three</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>one</td>
<td></td>
</tr>
</tbody>
</table>

Seemed reasonable ... simple ... but it wasn’t enough. Why?
**Harder to reindent**

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I was not keeping enough state in the loop. I didn’t need just the current input indentation ... what did I need to handle multiple de-indent correctly?

**Time to re-think**

Not such a simple loop. I needed an invariant:

An explicit property of state maintained each time through the loop.

Threw away my loop code and re-wrote it, reasoning about the invariant for each case.

**An invariant on indentation state**

indents is a list of all source indentation up to the current line

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,1,17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,1,17,18</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Initializing the invariant**

indents is a list of all source indentation up to the current line

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,1,17</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0,1,17,18</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two trees are shown, each representing a list of integers and their corresponding indentation levels. The list is structured as follows:

- **First Tree:**
  - [0]
  - [0, 1]
  - [0, 1, 17]
  - [0, 1, 17, 18]
  - [0, 1]
  - [0, 1, 17, 18]
  - [0, 1]
  - [0]

- **Second Tree:**
  - [0]
  - [0, 1]
  - [0, 1, 17]
  - [0, 1, 17, 18]
  - [0, 1]
  - [0, 1, 17, 18]
  - [0, 1]
  - [0]
Maintaining the invariant

indents is a list of all source indentation up to the current line

| [0]   | one   |
| [0, 1]| two   |
| [0, 1]| two   |
| [0, 1, 17]| three |
| [0, 1, 17, 18]| four |
| [0, 1]| two   |
| [0]   | one   |

Now we can write code

Initialize the invariant before the loop
Loop body is broken into cases: for each condition we might encounter, maintain the invariant