Generator Pipelines

co-routines FTW!

John Hanley

12th November 2024

agenda

- a micro rant
- co-routine pipelines

Comments lie!

> The code works, seems it is returning valid values

It doesn't. There are two crucial errors in the routine, and you can see them both in this assignment:

```
# Make a copy of i-th column vector from matrix m
cvt = m[i]
```

That's not a copy, and it's not a column vector. It's a read-write view to a row vector, and among other effects this ends up producing an `m` that is fully broadcast-assigned to `v`, which was not your intent.

We now have to guess which is the truth and which is a lie — the comments or the code.

SLOC

SLOC

source lines of code

PHP comments

- `# foo`

C++ comments

- `//` single line
- `/*` multi `\n` line `*/`

separation of concerns

We would like to be able to swap in one or another module, based on current filetype, similar to these shell pipelines:

```
$ cat example1.php | filter_php_comments.sh |
    filter_blank_lines.sh | wc -l


$ cat example2.C   | filter_cxx_comments.sh |
    filter_blank_lines.sh | wc -l
```

CLOC

- https://github.com/AlDanial/cloc
- `cloc` counts lines of code: blank lines, comment lines, and physical lines of source code, in diverse programming languages.
- We choose to define the `cloc` counts as the "correct" counts.

three counts

- a line is "blank" or "text"
  - a text line starting with comment is a "comment" line
  - else a text line is a "code" line

CLOC

Example output:

```
$  cloc /tmp/repos/llama.cpp/examples/jeopardy/graph.py
       1 text file.
       1 unique file.
       0 files ignored.


github.com/AlDanial/cloc v 2.02 ...
-------------------------------------------------
Language      files     blank     comment     code
-------------------------------------------------
Python        1         8         0           50
-------------------------------------------------
```

Demo

src/count/tests/sloc_test.py:

- SlocTest

src/count/sloc.py:

- LineCounter
- BashLineCounter
- PythonLineCounter

recurring themes

- separation of concerns
- write down the spec
- rapid edit-test cycle
- preserve interactive testing as an artifact
- scope it down