Ashish Mahabal
California Institute of Technology

Best Programming Practices - IV

Sponsored by the Jet Propulsion Laboratory and by the Keck Institute for Space Studies, California Institute of Technology

Metaprogramming

- Configure
- Abstraction in code, details in metadata
  - Decode design
  - docstrings

```
"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""
```

Portfolio building

- learn general tools, invest in different ones
  - plain text
    - easier to test (config files, for instance)
  - Shells
    - find, sed, awk, grep, locate
    - .tcshrc, .Xdefaults
  - learn different (types of) languages
  - Editor
    - if you know emacs, learn just a little bit of vi (or sublime)
    - Configurable, extensible, programmable (cheat sheet)
      - syntax highlighting
      - auto completion
      - auto indentation
      - Boilerplates
      - built-in help

Text manipulation

perl and ruby are very powerful

- Code generators
  - make files, config files, shell scripts., …
- Active code generator:
  - Skyalert (transient astronomy streams)
    - new transient
    - obtain distributed archival data
    - incorporate it
    - if certain conditions met,
      - run other programs
      - or raise alerts
      - drive other telescopes
      - and obtain feedback

- Languages/tools/OSes/editors
  - 99 bottles of beer
  - Programming shootout
  - Project Euler
    - Python
    - Perl
    - J
    - Haskell

Memoization example

```
factorial_memo = {}
def factorial(k):
    if k < 2: return 1
    if not k in factorial_memo:
        factorial_memo[k] = k * factorial(k-1)
    return factorial_memo[k]
```

Exercise

- Write a program to count number of ways to split an amount using coins of denominations 1,5,10,25
- For numbers 1 through 100, sum those for which the answer to the first question is an odd number
- Is it odd or even?

Exercise

- Duplicate as much as possible the following using only Unix commands: http://lifehacker.com/5898720/a-better-strategy-for-hangman

Number of letters	Optimal calling order
1	AI
2	AOEIUMBH
3	AEOIUYHBCK
4	AEOIUYSBF
5	SEAOIUYH
6	EAIOUSY
7	EIAOUS
8	EIAOU
9	EIAOU

What are the lessons?

- Chain as weak as its weakest link
- Comment! For others and for yourself
- Tests!
- Orthogonality
- Don’t duplicate
- Designing by contract
- Know the features

Law 1: Every program can be optimized to be smaller.

Law 2: There's always one more bug.

Corollary: Every program can be reduced to a one-line bug.

Follow the Best Practices, and have fun coding