Code analysis for python

Pyflakes, Pychecker & Pylint

Neil Muller

March 21, 2009

Why Bother?

- The aim -> improve code quality
  - trivially a good idea
- Detect obvious errors & warn of dangerous constructs
- Static analysis has limitations, especially when dealing with flexible languages like python
  - Remains a useful additional tool

Other tools

- rats (Rough Auding Tool for Security) - checks for possibly insecure code constructs. Very limited python support
- django-lint - wrapper around pylint for django projects. Includes several additional django specific checks
- owasp-python-static-analysis - static input validation for python web apps. Still very much under development
- Starkiller - type inference analysis. Intended to be a part of a more complete static analysis project, but seems to have died.
- clonedigger - detects duplicated blocks of code

Pyflakes

- Pure static analysis of the parse tree
- Checks only for logic errors, no style checks
- Comparatively limited range of errors detected
- No support for modules, will check only files on command line or all files in directories on the command line
- Very fast - cheap enough to be setup as a simple pre-commit check or called from an editor

Pychecker

- Actually imports the file/module - somewhat different semantics from the other checkers
- More through set of checks than pyflakes
- Development has been slow - pretty much dead during 2006 & 2007, but picked up in 2008.
- Will recursively check imports (can be annoying for the standard library, so various checks can be disabled)
- Can be imported into the module space, so checks also done at runtime
- Quite configurable - limit checks via command line, config file, or using _pychecker_ hints in the program

Pylint

- Purely static analysis of the parse tree
- Adds lots of style checks
- Very configurable - can specify naming conventions, etc.
- Can be controlled by commandline options, configfile and comments in the file
- Nice report formatting options
  - "-f colorized" rocks
  - Total score, with configurable metrics
  - Caches previous results - useful for spotting changes that need closer attention
- Warns of duplicated code, although is across files - doesn't detect copy and paste within a file.
  - Test is also very simplistic, and easily fooled by minor style changes

Customisation

- Pyflakes - no configuration options
- Pychecker:
  - Enable/ Disable checks (several checks default to off)
  - Supports lists of acceptable names for various checks (dummy variables, etc)
  - Warning can be enabled/disabled locally using__pychecker__ variable, but note scoping effects.
- Pylint: Most flexible of the three
  - allows specifying regexes for naming conventions,
  - Warnings can be selectively enabled/disabled by infile comments (although repeated enables & disables is a bit buggy at times)
  - Plugin support

Extending Pylint

- Can add additional checks using plugins
- Two types of plugin - RawChecker & ASTNGChecker
- RawChecker - Fed lines from the file directory
  - Good for various formatting checks (mixing spaces & tabs, etc).
- ASTNGChecker
  - Pylint uses an extended version of the default python AST
  - Stores a bunch of extra state in the AST for use in checkers
- Things to note
  - Message ID's must be unique (used as dictionary keys)
  - Config file sections are assumed to be per-checker. Extended/accessing existing config sections fiddly
  - Checkers initialised only once per run, not per file.