py.test and the py lib

holger.krekel@merlinux.de

merlinux

Testing and Python

- Automatically testing Python programs is a honking great idea, let’s do more of it!
- Unfortunately, Python projects have rather disparate test tools and “runners”
- pythoneers deserve better than the classical unittest.py

What is py.test

- py.test is an external project independent tool which
  - automatically and iteratively collects tests from directories, files, modules and classes
  - has extensive debugging aid
  - interacts with project-specific configuration to alter the collection and test execution process, including adding new command line options
  - is flexible enough to allow running PyPy's tests on a different virtual machine while still providing nice tracebacks and introspection on failing tests

py.test features (1)

- assert with **assert** statement
- automatic collection of tests on all levels
- testing starts immediately
- generative tests: yielding more user-defined tests
- specify different python versions / executables
- no interference with cmdline utilities

py.test features (2)

- debug with the **print** statement
- order of running tests as they appear in the files
- useful tracebacks, e.g. recursion detection
- Manages test state across modules, classes and methods
- it has documentation

py.test: asserting the obvious

The idea of “py.test” and the py lib in general is “no API is the best API”. For testing this e.g. means reusing the assert statement.

```
def f():
    return 23

def test_f():
    assert f() == 42
```

```
def test_f():
E       assert f() == 42
~       assert 23 == 42
        +  where 23 = f()
```

managing test state

setup and teardown resources at various levels

```
def setup_module(mod):             def teardown_module(mod):
   mod.testfile = ...                 mod.testfile.close()

class TestSomething:
   def setup_class(cls):           def teardown_class(cls):
        cls.resouce = ...             cls.resouce.finalize()

   def setup_method(self, method): def teardown_method(self, method):
        self.permethod = ...          self.permethod.done()
```

generative tests

the easiest “Non-API” way to extend the collection process is with generators which allow to produce more tests on-the-fly:

```
def func(arg1, arg2):
    assert arg1 == arg2 * 2

def test_more():
    for x,y in ((1,2), (2,3), (2,4)):
        yield func, x, y
```

it’s nice when it fails

- py.test offers a number of helpful debugging features
    - very useful tracebacks
    - isolating print statements per test
    - showing locals
    - dropping into pdb on failures
    - session: rerunning continously failing tests only
    - **--nomagic --nocapture** may be your friend :-)

interactive example

py.test py/documentation/example/pytest/failure_demo.py

py.test near-future plans

py.test is to become a unified python testing tool

- refining and documenting project config

- doctests!

- gui/html reporting

- distributing tests across platforms

- releasing it ...

on to the py lib

- **py.test** is part and makes use of of **the py lib**:
  - **py.path**: local and subversion filesystem objects
  - **py.execnet**: ad-hoc distribution of programs
  - **py.magic:** provides e.g. **greenlets**
  - **py.code**: nicifying python introspection
  - **py.xml**: providing simple xml/html object generation
- runs on python 2.2 onwards

goals of the py lib

- provide a high level and integrated standard set of services and methods useful for development
- **first step**: project independent and flexible testing tool
- experiment with improving python library development, e.g. via explicitly exporting names
- repackaging and extending python's standard library, e.g. offering a unified Path object for both local and remote access

py lib highlight: ad-hoc distributing programs

- **py.execnet** provides a simple mechanism to execute arbitrary code in remote locations
- communication between local and remote sites happen through channels
- It uses a **radically different idiom than traditional Remote Method Invocation** (i.e. CORBA/Pyro/ XMLRPC, SOAP ...)
- ----> Interactive SSH Example

future (and thanks for) fish

- Finalize design of py.test & include doctests
- refine consistency and actually release the damn thing
- expose remote Path-Over-Ssh objects
- try improve windows-interactions

thanks especially to Ian Bicking, Patrick K.O'Brien, Matthew Scott, Martijn Faassen, Florian Bauer, Laura Creighton, Grig Gheorghiu, Jan Balster, Christian Tismer for help and contributions

py lib info

the py lib is driven by Holger Krekel and Armin Rigo

http://codespeak.net/py

py-dev@codespeak.net

feedback and co-developers welcome!

all code released under the MIT license