Plugin

pytest-doctest-custom

v1.0.0

Hackeando doctests com o plugin (ementa):
https://gist.github.com/danilobellini/b76a36c4fcc946ecb1d6cb92987f30d3

Plugin:
https://github.com/danilobellini/pytest-doctest-custom

PyPI:
https://pypi.python.org/pypi/pytest-doctest-custom

Testes – Por quê?

"[...], perhaps the most important benefit of unit tests is not that it ensures that your code works - it's that it ensures that it stays working, that is, that a future change doesn't cause a regression."

https://www.wiki.wxpython.org/Unit%20Testing%20with%20wxPython

doctest

documentation + tests
standard library

docstrings

- Documentação inserida no próprio código
- Acessada pela função help() do Python e pelo "?" no IPython
- Armazenada no atributo "__doc__"
- Pode constar em:
  - Funções
  - Métodos
  - Classes
  - Módulos/pacotes

docstring

```
def snake2ucamel(value):
    """Converte snake_case em UpperCamelCase."""
    return "".join(el.capitalize()
                   for el in value.split("_"))
```

doctest

- Documentação pode conter exemplos
  - Por que não testar os exemplos?
- Consistência
- Standard library e pronto para usar:

```
python -m doctest nome_arquivo
```

- Procura por exemplos como no REPL do Python:

```
>>> comando
... continuação
resultado
```

- Compara strings de representação, não valores
  - Dependente da ordenação (não serve para sets/dicts)

```
$ python -m doctest snake2ucamel.py -v
Trying:
    snake2ucamel("teste")
Expecting:
    'Teste'
ok
Trying:
    snake2ucamel("um_nome_muito_longo")
Expecting:
    'UmNomeMuitoLongo'
ok
1 items had no tests:
    snake2ucamel
1 items passed all tests:
   2 tests in snake2ucamel.snake2ucamel
2 tests in 2 items.
2 passed and 0 failed.
Test passed.
```

```python
def snake2ucamel(value):
    """
    Converte snake_case em UpperCamelCase.

    Exemplo:

    >>> snake2ucamel("teste")
    'Teste'
    >>> snake2ucamel("um_nome_muito_longo")
    'UmNomeMuitoLongo'
    """
    return "".join(el.capitalize() for el in value.split("_"))
```

py.test

e

tox

py.test

- Instalação:

```
pip install pytest
```

- Uso sem argumentos:

```
py.test
```

ou

```
python -m pytest
```

- Coleta automática pelo nome:
  - Funções/métodos: test*
  - Classes: Test*
  - Arquivos: test_*.py
- Uso do statement "assert"

```python
def snake2ucamel(value):
    """Converte snake_case em UpperCamelCase."""
    return "".join(el.capitalize() for el in value.split("_"))

def test_empty():
    assert snake2ucamel("") == ""

def test_no_under():
    assert snake2ucamel("maxsize") == "Maxsize"
    assert snake2ucamel("abigname") == "Abigname"

def test_has_under():
    assert snake2ucamel("max_size") == "MaxSize"
    assert snake2ucamel("a_big_name") == "ABigName"
```

Plugin pytest-doctest-custom – 2016-08-13 – GruPy-SP @ SciELO

Danilo de Jesus da Silva Bellini | @danilobellini | facebook.com/djsbellini | github.com/danilobellini

```python
def snake2ucamel(value):
    """Converte snake_case em UpperCamelCase."""
    return "".join(el.capitalize() for el in value.split("_"))

def test_empty():
    assert snake2ucamel("") == ""

def test_no_under():
    assert snake2ucamel("maxsize") == "Maxsize"
    assert snake2ucamel("abigname") == "Abigname"

def test_has_under():
    assert snake2ucamel("max_size") == "MaxSize"
    assert snake2ucamel("a_big_name") == "ABigName"
```

```
$ py.test
========= test session starts ==========
platform linux -- Python 3.5.2, pytest-
2.9.2, py-1.4.31, pluggy-0.3.1
rootdir:
/home/danilo/Desktop/Grupy_2016-08-
13/Exemplos/pytest, inifile:
plugins: doctest-custom-1.1.0.dev0, cov-
2.3.1, timeout-1.0.0
collected 3 items

test_snake2ucamel.py ...

======= 3 passed in 0.01 seconds =======
```

py.test

- Suporte a plugins
  + plugins internos
- Inclui plugins prontos
- Fixtures!
- Testes parametrizados
- FLOSS
  - Eu que fiz o plugin interno do py.test isolar/contar corretamente os doctests
- Não te obriga a usar assertNomesForaDaPEP8

http://pytest.org

tox

- Gerenciador de virtualenvs
  - Automação
  - Testes em vários ambientes
  - Diferentes versões do Python e/ou de algum requisito
- Configurável
  - tox.ini

```
[tox]
envlist = py{36,35,34,27}
skipsdist = True

[testenv]
deps = pytest
commands = py.test {posargs}
```

```
$ tox
[...]
py36 runtests: commands[0] | py.test
[...]
py35 runtests: commands[0] | py.test
[...]
py34 runtests: commands[0] | py.test
[...]
py27 runtests: commands[0] | py.test
[...]
_____________ summary
_____________
  py36: commands succeeded
  py35: commands succeeded
  py34: commands succeeded
  py27: commands succeeded
  congratulations :)
```

Misturando tudo!

tox.ini

```
[tox]
envlist = py{36,35,34,27}
skipsdist = True

[testenv]
deps = pytest
commands = py.test {posargs}

[pytest]
addopts = --doctest-modules
```

doctests com o py.test apenas editando o tox.ini

```
$ tox
[...]
py36 runtests: commands[0] | py.test
[… detalhado abaixo ...]
py35 runtests: commands[0] | py.test
[...]
py34 runtests: commands[0] | py.test
[...]
py27 runtests: commands[0] | py.test
[...]
_____________ summary
_____________
  py36: commands succeeded
  py35: commands succeeded
  py34: commands succeeded
  py27: commands succeeded
  congratulations :)
```

snake2ucamel.py

```
def snake2ucamel(value):
    """
    Converte snake_case em UpperCamelCase.

    Exemplo:

    >>> snake2ucamel("teste")
    'Teste'
    >>> snake2ucamel("um_nome_muito_longo")
    'UmNomeMuitoLongo'
    """
    return "".join(el.capitalize() for el in value.split("_"))
```

test_snake2ucamel.py

```
from snake2ucamel import snake2ucamel

def test_empty():
    assert snake2ucamel("") == ""

def test_no_under():
    assert snake2ucamel("maxsize") == "Maxsize"
    assert snake2ucamel("abigname") == "Abigname"

def test_has_under():
    assert snake2ucamel("max_size") == "MaxSize"
    assert snake2ucamel("a_big_name") == "ABigName"
```

```
py36 runtests: commands[0] | py.test
========= test session starts ==========
platform linux -- Python 3.6.0a3+,
pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir:
/home/danilo/Desktop/Grupy_2016-08-
13/Exemplos/ambos, inifile: tox.ini
collected 4 items

snake2ucamel.py .
test_snake2ucamel.py ...

======= 4 passed in 0.03 seconds =======
```

Representando objetos como strings

pprint (stdlib)
e
IPython.lib.pretty

Representação em strings

- Representação
  - repr
  - ascii
  - atributo __repr__
  - atributo __str__
  - pprint.pformat
- Impressão / escrita em stream
  - print
  - pprint.pprint
  - sys.stdout.write

- Representação mais legível
  - Quebra de linha (list, dict, etc.)
  - Ordenação (set, dict, etc.)
- Tratamento de recursão
  - e.g. lista que contém a si própria

```
def impressão(objeto): # Roughly
    stream.write(representação(objeto))
```

pprint
Pretty Printer

- Standard Library
- Não é extensível
- Fallback: repr
- Internamente separado em 2
  - Resultado em uma linha
  - Resultado em múltiplas linhas
- API inclui:
  - pprint.PrettyPrinter
  - pprint.pformat
  - pprint.pprint
- A classe PrettyPrinter permite personalização, mas se associa ao sys.stdout durante a construção

IPython.lib.pretty

- Fork do “pretty” do Armin Ronacher
- Extensível: procura um método “_repr_pretty_”
  - Mas possui funções prontas para os tipos básicos e coleções do Python
- Subpacote do IPython

```
pip install ipython
```

- Consistente entre diferentes versões e interpretadores do Python
- A API inclui:
  - IPython.lib.pretty.pretty
  - IPython.lib.pretty.pprint

Unindo mundos:

pytest-doctest-custom

Pytest-doctest-custom

- Plugin para o py.test
- Ativado/controlado através de um único argumento no py.test:

  `--doctest-repr=MODULO:OBJETO`
- Selecione a função que quiser para representar a saída de doctests, e.g.:

  `py.test --doctest-modules --doctest-repr=IPython.lib.pretty:pretty`
- Tanto funções de representação como impressão podem ser usadas
- É o fim dos problemas com a ordenação ao fazer o seu doctest

```
$ py.test --doctest-modules
========================== test session starts ==========================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/danilo/Desktop/Grupy_2016-08-13/Exemplos/setdict,
inifile:
plugins: doctest-custom-1.1.0.dev0, cov-2.3.1, timeout-1.0.0
collected 1 items

test_setdict.py F

================================ FAILURES ================================
_____________________ [doctest] test_setdict _____________________
002 >>> set("qwertyuiop")
Expected:
    {'e', 'i', 'o', 'p', 'q', 'r', 't', 'u', 'w', 'y'}
Got:
    {'r', 'e', 'q', 'w', 'y', 't', 'p', 'u', 'o', 'i'}

/home/danilo/Desktop/Grupy_2016-08-
13/Exemplos/setdict/test_setdict.py:2: DocTestFailure
======================== 1 failed in 0.02 seconds ========================
```

```
"""
>>> set("qwertyuiop")
{'e', 'i', 'o', 'p', 'q', 'r', 't', 'u', 'w', 'y'}
>>> d = {}
>>> d["Uma primeira chave beeeem longa... "] = 1
>>> d["Outra primeira chave beeeem longa!"] = 2
>>> d
{'Outra primeira chave beeeem longa!': 2,
 'Uma primeira chave beeeem longa... ': 1}
"""
```

```
$ py.test --doctest-modules
========================= test session
platform linux -- Python 3.5.2, pytest
rootdir: /home/danilo/Desktop/Grupy_20
inifile:
plugins: doctest-custom-1.1.0.dev0, co
collected 1 items

test_setdict.py F

============================== FAILURES ==============================
_____________________ [doctest] test_setdict _____________________
002 >>> set("qwertyuiop")
Expected:
    {'e', 'i', 'o', 'p', 'q', 'r', 't', 'u', 'w', 'y'}
Got:
    {'r', 'e', 'q', 'w', 'y', 't', 'p', 'u', 'o', 'i'}

/home/danilo/Desktop/Grupy_2016-08-
13/Exemplos/setdict/test_setdict.py:2: DocTestFailure
====================== 1 failed in 0.02 seconds ======================
```

```
$ py.test --doctest-modules --doctest-repr=IPython.lib.pretty:pretty
========================= test session starts ==========================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/danilo/Desktop/Grupy_2016-08-13/Exemplos/setdict,
inifile:
plugins: doctest-custom-1.1.0.dev0, cov-2.3.1, timeout-1.0.0
collected 1 items

test_setdict.py .

======================== 1 passed in 0.01 seconds ======================
```

pytest-doctest-custom dependências, testes, cobertura

- Requer py.test 2.1+
- Compatível com os interpretadores
  - CPython 2.6, 2.7
  - CPython 3.2, 3.3., 3.4, 3.5, 3.6 (dev)
  - PyPy 2.x, 4.x, 5.x
  - PyPy3 2.x, 5.x
  - Jython 2.7 (py.test 2.2.4+)
- Testado em todas as versões de py.test em cada um dos interpretadores
  - 337 ambientes no tox.ini
  - 84 jobs no Travis CI (py.test 2.8.5+)
- Cobertura de código de 100% no módulo do plugin

pytest-doctest-custom stream proxy

- Proxies no
  - pytest_doctest_custom.stdout_proxy
  - pytest_doctest_custom.stderr_proxy
- Caso sua função seja de impressão em um stream que é fixado no instante da criação, use um desses proxies
- Pode-se usar no conftest.py no mesmo diretório do projeto, por exemplo:

```
# conftest.py
from IPython.lib.pretty import pretty
def doctest_pretty(value):
    return pretty(value, max_width=5)
```

```
"""
>>> set("qwertyuiop")
{'e',
 'i',
 'o',
 'p',
 'q',
 'r',
 't',
 'u',
 'w',
 'y'}
"""
```

```
$ py.test --doctest-modules --doctest-repr=conftest:doctest_pretty
============================= test session starts ==============================
platform linux -- Python 3.5.2, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /home/danilo/Desktop/Grupy_2016-08-13-2016-08-13/Exemplos/conftest,
inifile:
plugins: doctest-custom-1.1.0.dev0, cov-2.3.1, timeout-1.0.0
collected 1 items

test_set.py .

=========================== 1 passed in 0.01 seconds ===========================
```

pytest-doctest-custom internalidades

- doctest.DocTestRunner.run
  - Modifica sys.stdout e sys.stderr
  - Troca sys.displayhook pelo dunder
- doctest.DocTestRunner._DocTestRunner__run
  - Utiliza sys.displayhook para a impressão dos resultados dos doctests
- O plugin altera o método run para temporariamente trocar o sys.__displayhook__ por uma função personalizada de impressão

CPython 2.x, 3.x
PyPy 2.x, 4.x, 5.x
PyPy3 2.x, 5.x
Jython 2.7

Diferentes interpretadores representam objetos da mesma forma?

IPython

- Python 2.6/3.2: usar o IPython 1.2.1
  - Verificado na instalação
- Até o IPython 5.0.0
  - Trata todo dict do PyPy como types.DictProxyType (bug)
  - Formatação de types.DictProxyType completamente diferente das demais
  - Não formata types.MappingProxyType
  - Já corrigi isso para o IPython 5.1 (PR aceito)
- Pretty printer extremamente consistente em todas as versões do CPython/PyPy (a menos do dict)
- Incompatível com Jython

pprint

- set()
  - `set()    # CPython 2.6, 3.x; PyPy3; Jython 2.7`
  - `set([]) # CPython 2.7; PyPy`
- set(range(5))
  - `{0, 1, 2, 3, 4}       # CPython 3.x; PyPy3`
  - `set([0, 1, 2, 3, 4]) # CPython 2.x; PyPy ; Jython 2.7`
- Ordenação
  - CPython 2.6+: sets sempre ordenados
  - CPython 3.2+: sets ordenados apenas se a representação não couber em uma linha
- Módulo collections
  - Rotinas de formatação inseridas no pprint somente no Python 3.5
  - Mesmo problema que ocorre com sets (até o momento)

PyPy

- Nem tudo tem como ser representado adequadamente em versões antigas do PyPy
  - e.g. “super(…).__thisclass__”
- dict is types.DictProxyType
  - Isso fazia o IPython perder a rotina de formatação de dicts
  - Instâncias dict_proxy somente via ctypes (o tipo não possui construtor), e essa classe inexiste no Python 3
  - Python 3.3+ (PyPy3 5.2.0-alpha1) possui types.MappingProxyType (com construtor)
- As rotinas do pretty printer do IPython for a preparadas para ter o mesmo comportamento no CPython 2.7, CPython 3.3+ e PyPy 5.3+

Outros assuntos

Sympy

- CAS (Computer Algebra System) em Python
- Possui a própria função “pprint”:
  - `>>> from sympy import pprint, exp, pi`
  - `>>> pprint(exp(1) + pi)`
  - $e + \pi$
- Com o plugin, os exemplos ficariam mais diretos na documentação (usando --doctest-repr=sympy.pprint e admitindo exp e pi já importados no módulo com o teste):
  - `>>> exp(1) + pi`
  - $e + \pi$

Contexto

- O plugin foi criado para facilitar os testes do projeto PyScanPrev, atualmente testado apenas com doctests
  - https://github.com/danilobellini/pyscanprev
- Uma ideia similar surgiu durante os sprints da Python Brasil [9] (Brasília-DF, 2013): um plugin para modificar o “>>>” e trocar pelo “In [#]” do IPython, em que o # é um número e a saída possui um prefixo “Out [#]” correspondente. Essa ideia de mudar o PS1 foi abandonada, mas a ideia de um plugin que possibilitasse doctests ao estilo do IPython não.

Fim

Danilo de Jesus da Silva Bellini | @danilobellini | facebook.com/djsbellini | github.com/danilobellini