Doctests en fitxers separats

Author: Sebastià Vila-Marta
Date: 25-abril-2011

Introducció

Quan volem usar doctests de forma intensiva, escriure'ls dins dels docstrings dels blocs Python és inconvenient com a mínim per tres raons:

1. Inflen els fitxers font de Python fent la seva comprensió i el treball amb el codi més molest i procliu a fer errors.
2. Si generem docuemntació automàticament a partir dels docstrings la definició de les classes, mètodes, funcions, etc. s'omple de tests extensos que emmascaren la documentació.
3. Si el test és extensiu, és interessant comentar aspectes relacionats amb el propi test. Aquests comentaris és interessant mantenir-los junts amb el test però separats de la documentació del propi element.

Doctests en fitxers separats

Una solució al problema esboçat a la introduccó consisteix a escriure els doctests en fitxers separats. A tal efecte només cal crear un fitxer de text associat a cada mòdul o conjunt de moduls que volem testejar i, escrivint en format ReST, anar descrivint del doctest corresponent acompanyat dels comentaris que es creguin oportuns.

Per exemple, en el cas del del mòdul *word.py* del projecte de curs, podríem crear un fitxer de nom *test_word.rst* i, en format ReST anar escrivint el test que ens interessa. Fixeu-vos que és corrent batejar els fitxer de test prefixant el nom de mòdul amb *test_*.

En aquest cas, el fitxer *test_word.rst* podria ser similar al següent (noteu que a l'exemple només en surt una part i no el test complet):

```
==================
Test del mòdul word
==================


Com `BitVector` és una classe abstracta, els tests els realitzem
únicament sobre les classes derivades.

Classe `Byte`
=============

Importem les eines

.. code-block:: python

   >>> from word import Byte, Word

Comprovem els constructors i el mètode de representació
simultàniament. Fixem-nos en com actua la màscara sobre el
mètode d'entrada:
```

```
.. code-block:: python

   >>> print Byte(1)
   01
   >>> print Byte(15)
   0F
   >>> print Byte(403)
   93
   >>> print Byte(-1)
   FF

Comprovem ara les operacions aritmètiques sobre Bytes. Fem èmfasi en
la suma byte-enter, que també hauria de funcionar correctament:

.. code-block:: python

   >>> Byte(12) + Byte(2)
   0E
   >>> Byte(12) + 2
   0E
   >>> Byte(12) - Byte(2)
   0A
   >>> Byte(12) - 2
   0A
   >>> Byte(403) - 3
   90

Les operacions bit a bit entre bytes també hauríen de fer la feina que
toca:

.. code-block:: python

   >>> Byte(3) | Byte(4)
   07
   >>> Byte(7) | Byte(0b0010)
   07
   >>> Byte(7) & Byte(0b0010)
   02
   >>> Byte(0xff) & Byte(0b1011)
   0B
   >>> Byte(0xff) ^ Byte(0b1011)
   F4
   >>> Byte(0xff) ^ Byte(0b1011)
   F4
   >>> ~Byte(0xf0)
   0F
   >>> ~Byte(0b101010)
   D5
   >>> Byte(1) << 4
   10
   >>> Byte(0xfe) << 3
   F0
   >>> Byte(0xff) >> 4
   0F
   >>> Byte(0xff) >> 4
   0F
```

Aquests fitxers que contenen tests també poden ser usats per l'eina *nose* si se li indica convenientment. Assumiu que en el directori *src* hi teniu tant els fonts Python com els tests, llavors podeu demanar a nose que busqui i executi els fitxers de test de la següent forma:

```
$ nosetests --with-doctest --doctest-extension=rst .
```

Mireu-vos a la man page de *nosetests* el significat d'aquestes opcions i recordeu que sempre podeu configurar *nosetests* a través del fitxer *.noserc* de forma que no s'hagin d'escriure cada vegada les opcions necessàries.

Integració en la documentació

Deixar els tests en fitxers separats té l'inconvenient que no apareixen en la documentació generada (per Sphinx). Com els tests són una bon font d'exemples d'ús dels mòduls és una llàstima perdre aquesta part de la documentació. La solució consisteix a incloure directament els fitxers de test en el lloc de la documentació que ens sembli escaient. A tal efecte podem usar la directiva *include* de ReST. Per exemple, si volem incloure els tests del mòdul *word* al final de la documentació del mateix mòdul, podem retocar el fitxer font de Sphinx en que, usant autodoc, es genera aquesta documentació reescriure'l així:

```
.. automodule:: word
   :members:

.. include:: ../../src/test_word.rst
```

Voilà! ara tenim el millor dels dos mons.