Docstrings Python

Commenter c'est aussi documenter !

Adel Daouzli

LOL (LyonOpenLab) - Viveris Technologies
Mél : daouzli@gmail.com
Web : http://www.daouzli.com/blog/pyconfr-fr.html

25/10/2014

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

Commentaires

Commenter...

```
//C++
/* C, Java, Javascript,... */


#Python/sh
...
```

- description / compréhension / collaboration
- annotations / tags (__AUTHOR__, __DATE__, FIXME, TODO)
- autogénération de documentation

Autogénération Documentation (Doxygen/Javadoc)

- pionniers (fin du XXe siècle)
- commentaires annotés / taggés -> documentation
  - Doxygen:

    parse tags spécifiques (`\brief, \param, \retval,...`) C/C++ (d'autres dont Python) Exemple:

```
/** \brief Brève description .
*
* Description plus détaillée.
* \param p1 un paramètre
* \return le resultat de la fonction
* \retval -1 échec
*/
int fonction (int p1);
```

  - Javadoc tool: parse tags (`@param, @return, @rtype,...`) Exemple:

```
/** Description de la fonction .
*
* @param p1: un paramètre
* @type p1: int
* @return: le resultat de la fonction
* @rtype: int
*/
int fonction (int p1);
```

Python Docstrings (syntaxe, help, __doc__, doctest)

- « documentation string »
- 2 délimiteurs possible (' ' ', " " ")
- chaine de caractère multiligne
- description : début fonction / 1ers args de classe (__init__)
- doctest -> petits tests

```
"""
>>> a = 2
>>> b = 5
>>> a + b
7
"""
```

Python Docstrings (syntaxe, help, __doc__, doctest)

- introspection
- récupérer la docstring de n'importe quel élément Python via l'attribut __doc__
- iPython permet d'obtenir la docstring ajoutant ? à un élément
- fonction help() fournit documentation interactive comme un *man* (Pydoc)

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

PEPs: recommandations globales

- PEP 8: docstring == chaine de caractères, cf PEP257
  - docstrings pour tout module, classe, fonction et méthode publique. (méthodes non-publiques peuvent se contenter de commentaires)
  - important """ de fin de multi-lignes doit être seul sur la ligne.
  - docstring d'une ligne: """ sur la même ligne

- PEP 257: description haut niveau
  - tout modules devrait avoir une docstring
  - fonctions/classes exportées, méthodes publiques (__init__())
  - __init__.py d'un paquetage
  - « attribute docstring »: assignation docstring début module, classe, __init__
  - préfixes de chaine r (raw) et u (unicode)

```
r"""ceci \n est échapé"""
```

  - description triplecotée même sur une ligne (délimiteurs sur la même ligne)
  - description IMPERATIVE ("return this")
  - ne pas répéter signature de fonction/méthode (introspection)
  - multiligne: brève description, ligne vide, description détaillée
  - multiligne: ligne vide avant/après (sauf fonctions/méthodes)
  - multiligne: indentation précédant 1 délimiteur retirée aux lignes suivantes
  - *"override"* pour remplace méthode, *"extend"* pour étend méthode de classe mère

PEPs: Autres recommandations

D'autres PEP parlent aussi de docstrings:

- PEP 256 - «Docstring Processing System Framework»
  Mais la PEP commence par:
  Rejection Notice
  This proposal seems to have run out of steam.
- PEP 258 - «Docutils Design Specification»
  De même la PEP commence par:
  Rejection Notice
  While this may serve as an interesting design document for the now-independent docutils, it is no longer slated for inclusion in the standard library.
- PEP 287 - «reStructuredText Docstring Format»
  Il est recommandé d'utiliser le reStructuredText comme format pour les Docstrings Python.
  Le reST fait partie du projet Docutils.

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

Javadoc / Epytext

- 1er formalisme Python
- hérité du format Javadoc
- interpréteur: Epydoc
- tend à disparaître

Exemple:

```
"""
This is a Epytext style description .
@param param1: this is a first param
@type param1: int
@param param2: this is a second param
@type param2: str
@return: this is a description of what is returned
@rtype: bool
@raise keyError: raises an exception
"""
```

reStructuredText

- tags reST pour Python
- format de docstring le plus populaire.
- très répandu.
- préconisé par la PEP 287.
- utilise des tags de docstrings très ressemblant avec la syntaxe d'Epytext (et donc Javadoc).
- interpréteur: Sphinx.
- utilisé au-delà des docstrings Python (CMake, wikis, blogs, Github, Trac, Project Gutenberg...)

Exemple:

```
"""
This is a reST style.
:param param1: this is a first param
:type param1: int
:param param2 str: this is a second param
:returns: this is a description of what is returned
:rtype: bool
:raises keyError: raises an exception
"""
```

Google

Le format Google pour les docstrings:

- supporté par Google.
- assez répandu.
- interpréteur: plugin pour Sphinx
- regroupement d'éléments (section paramètres, section valeurs de retour,...).
- ressources: Guide de style de Google: `http://google-styleguide.googlecode.com/svn/trunk/pyguide.html#Comments`
  Exemples: `http://sphinxcontrib-napoleon.readthedocs.org/en/latest/example_google.html`

Exemple:

```
"""
This is a groups style docs.
Args:
    param1 (int): this is the first param
    param2 (str): this is a second param
Returns:
    bool: This is a description of what is returned
Raises:
    KeyError: raises an exception
"""
```

Numpydoc

Le format Numpydoc pour les docstrings:

- très répandu dans le monde scientifique (mais pas uniquement)
- structure proche de Google (regroupement par sections).
- syntaxe également basée sur reST.
- interpréteur: plugins pour Sphinx
- ressources: guide sur le Github du projet Numpy:
  `https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt`

Numpydoc

Exemple:

```
"""
My numpydoc description of a kind
of very exhautive numpydoc format docstring .

Parameters
----------
first : array_like
    the 1st param name 'first'
second : {'value', 'other'}, optional
    the 2nd param, by default 'value'
Returns
-------
string
    a value in a string
Raises
------
KeyError
    when a key error
OtherError
    when an other error
"""
```

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

Pyment

- licence GPL3
- créer des docstrings (squelettes)
- convertir des docstrings
- formats acceptés: reST, google, numpydoc, javadoc
- parse module (pas d'import)
- génère des patchs
- pas de dépendance
- Python 2.7/3+ (2.6 avec argparser)
- sources: `https://github.com/dadadel/pyment\` Tuto: `http://daouzli.com/blog/pyment.html`

Installation

```
git clone https://github.com/dadadel/pyment.git
virtualenv env
source env/bin
pip install -e pyment
# ou:
cd pyment && python setup.py install
pyment -h  # pour afficher l'aide
```

Utilisation

Format de sortie par défaut: reStructuredText

```
pyment fichier.py
# génère le patche *fichier.py.patch*
patch -p1 < fichier.py.patch
# docstrings créées/converties au format reStructuredText
```

ou un module :

```
pyment mon/dossier
# produit dans le dossier actuel les patches des fichiers trouvés
```

Options

- `-h/--help:` afficher le message d'aide (liste des options)
- `-v/--version:` afficher la version de Pyment
- `-c` FICHIER: récupère les options depuis un ficheir de configuration
- `-o/--output` STYLE: reST, google, numpydoc or javadoc (défaut reST)
- `-q/--quotes:` type de délimiteur ' ' ' ou " " " (défaut " " ")
- `-f/--first-line` BOOL: commentaire doit commencer à la ligne suivant le délimiteur
- `-t/--convert` BOOL: convertit seulement les docstrings existantes (n'en génère pas)
- `-d/--init2class` BOOL: si la classe n'a pas de docstring, y palcer celle de __init__().
- `-p/--ignore-private` BOOL: ne pas générer de docstring pour les classes débutant par un double underscore (__)

Exemples

```
pyment -q "'''" example.py
# génère format reST avec des ''' au lieu de """
pyment -o google example.py
# génère format Google
pyment -o numpydoc -t example.py
# génère format Numpydoc uniquement pour les docstrings existantes
```

```
# Patch generated by Pyment v0.2.0
--- a/example.py
+++ b/example.py
@@ -1,21 +1,23 @@
 def func(prm1=True, prm2='val'):
-    '''Description of groups style (Googledoc).
+    """Description of groups style (Googledoc).

-    Params:
-        prm1 - descr of prm1 with True default value.
-        prm2 - descr of prm2
+    :param prm1: descr of prm1 with True default value
+    :param prm2: descr of prm2 (Default value = 'val')
+    :returns: some value
+    :raises keyError: raises key exception
+    :raises TypeError: raises type exception

-    Returns:
-        some value
-
-    Raises:
-        keyError: raises key exception
-        TypeError: raises type exception
-
-    '''
+    """
     pass

 class A:
+    """ """
     def method(self, prm1, prm2=None):
+        """
```

Conversion vers reStructuredText

Avant

```
def func(prm1=True, prm2='val'):
    '''Description of groups style (Googledoc).

    Params:
        prm1 - descr of prm1 with True as default.
        prm2 - descr of prm2

    Returns:
        some value

    Raises:
        keyError: raises key exception
        TypeError: raises type exception

    '''
    pass

class A:
    def method(self, prm1, prm2=None):
        pass
```

Après

```
def func(prm1=True, prm2='val'):
    """Description of groups style (Googledoc).

    :param prm1: descr of prm1 with True as default.
    :param prm2: descr of prm2 (Default value = 'val')
    :returns: some value
    :raises keyError: raises key exception
    :raises TypeError: raises type exception

    """
    pass

class A:
    """ """
    def method(self, prm1, prm2=None):
        """

        :param prm1:
        :param prm2:  (Default value = None)

        """
        pass
```

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

Pydoc

- module intégré à Python (>= 2.1)
- fonction help() de Python
- indépendant de Python, ligne de commande
- documentation interactive (équivalent à help())

```
pydoc os # démarre une aide interactive pour le module os
```

- serveur HTTP

```
pydoc -p 8080 # démarre serveur HTTP port 8080 de la machine local
```

- génère de la documentation HTML

```
pydoc -w toto # génère une documentation HTML
```

Epydoc/HappyDoc

- Epydoc importe module
- HappyDoc: parse module
- développé en Python
- générer une documentation pour modules Python
- 1ère version publiée en 2002
- dernière version en 2008 : 3.0.1
- site officiel: `http://epydoc.sourceforge.net` .
- supporte le Epytext mais peut aussi gérer reST et Javadoc
- peut être utilisé en ligne de commande
- possède une interface graphique (old school :p)
- peut générer des graphs avec Graphviz
- peut générer du HTML ainsi que du PDF (via LaTeX)

Sphinx & reStructuredText

- licence BSD
- générer de belles documentations
- créé pour générer la documentation de Python
- parse du Python (C/C++, et d'autres langages dans le futur)
- traiter des documentations au-delà d'APIs
- générer du HTML, LaTeX (et donc PDF), ePub, Texinfo, des man, du texte
- coloration syntaxique
- nombreuses extensions
- interprête reStructuredText
- ressources : `http://sphinx-doc.org`

Sphinx & reStructuredText

Installation:

- depuis vos dépôts (sudo apt-get install python-sphinx)
- télécharger depuis pypi (`https://pypi.python.org/pypi/Sphinx`)
- easy_install -U Sphinx
- pip install Sphinx

Utilisation:

- non triviale
- scripts de préparation
- plusieurs méthodes d'utilisation (interactif avec `sphinx-quickstart`)

Exemple:
Voici comment convertir en HTML avec `sphinx-apidoc` le module dans le dossier **src/** :

```
mkdir docs
sphinx-apidoc -F -o docs src/
# il faut décommenter vers le début du fichier docs/conf.py la ligne ressemblant à:
# sys.path.insert(0, os.path.abspath('/home/user/src/projet/src/'))
make html
```

Votre doumentation est générée dans `docs/_build/html` (index.html).

Sphinx Napoleon/Numpydoc

- Napoleon

  L'extension Sphinx Napoleon: docstrings au format Google et Numpydoc
  `https://pypi.python.org/pypi/sphinxcontrib-napoleon`

- Numpydoc

  Numpydoc se base sur plusieurs extensions Sphinx pour gérer le format Numpydoc:
  `https://pypi.python.org/pypi/numpydoc`

1 Introduction

2 Recommandations

3 Formats de docstrings

4 Pyment

5 Outils génération de doc

6 Questions ?

Fin

Merci pour votre attention...
`http://www.daouzli.com/blog/pyconfr-fr.html`