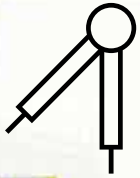
Patrice **CLEMENT**

Python et Raspberry Pi

Apprenez à développer sur votre nano-ordinateur

2e édition

Fichiers complémentaires à télécharger



LA FABRIQUE

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **LF2RASPYT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
Raspberry Pi 3, premier contact

Chapitre 2
Python : bases et concepts avancés

Chapitre 3
Administration du Raspberry Pi en Python

Chapitre 4
Le Raspberry Pi en console avec urwid

Chapitre 5
Programmation d'interfaces graphiques avec tkinter

Chapitre 6
À l'assaut du Web avec le Raspberry Pi

Chapitre 7
Multimédia et audio sur le Raspberry Pi

Chapitre 8
Persistance de données sur le Raspberry Pi

Chapitre 9
Documenter et tester ses scripts en Python

Chapitre 10
Raspberry Pi et GPIO

Chapitre 9
Documenter et tester ses scripts en Python

1. Introduction

L'écriture de tests unitaires est désormais incontournable dans l'élaboration d'un programme informatique. Dans ce domaine, Python est livré avec des modules de choix qui répondent aux attentes des développeurs les plus ambitieux. Python offre aussi la possibilité d'inspecter son code interactivement avec le REPL, et de vérifier instantanément le contenu d'un objet, son type et les méthodes qu'il offre. Pour assister le développeur dans les tâches liées à la documentation, à l'analyse de performance et à la résolution de problèmes en rapport avec le code, la gamme des modules proposés par le langage est assez vaste. Par exemple, lorsque la taille d'un projet devient critique, l'usage de tests unitaires permet d'implémenter plus rapidement de nouvelles fonctionnalités, et de détecter les régressions de code dès le début de l'implémentation. Ce qui fait ainsi gagner du temps et de la productivité. La recherche de performance va aider le développeur à identifier les fonctions gourmandes en exécution de celles qui le sont moins, afin de procéder à de la refactorisation et/ou de la réécriture de code. Dans le cas de l'écriture de scripts destinés au Raspberry Pi où les ressources sont restreintes, auditer et benchmarker son code peut améliorer le temps d'exécution d'un programme. Encore une fois, même si ce livre n'a pas pour but d'enseigner la conduite de projet, sachez qu'il est important de connaître les outils que propose Python dans ce domaine afin de mieux apprécier l'écosystème dans son ensemble.

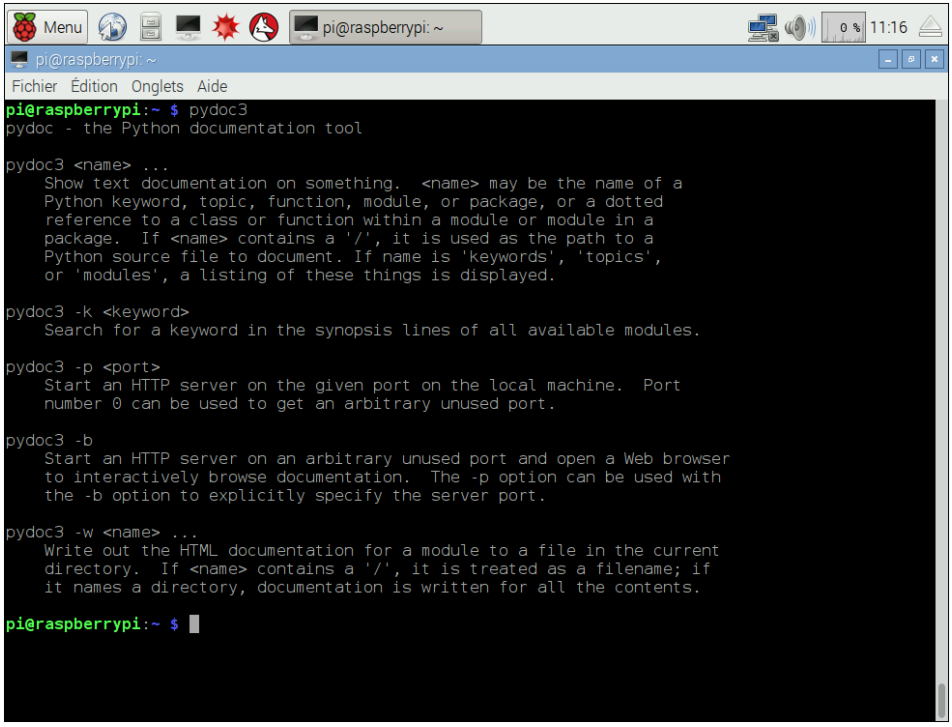
2. Consulter de la documentation avec pydoc3

Tout d'abord, l'outil le plus courant qui permet de chercher dans la documentation des modules porte le nom de `pydoc3`. L'utilisation de `pydoc3` intervient lorsque l'on souhaite afficher la documentation d'un module, d'une classe ou d'un mot-clé. Par exemple, la documentation de tous les mots-clés passés en revue dans les chapitres précédents (`with`, `def`, `lambda`, etc.) peut être consultée avec `pydoc3`. Point important à noter avant d'aller plus loin : la majorité de la documentation installée avec le système par défaut de Python est écrite dans la langue anglaise.

Cet outil s'utilise exclusivement en ligne de commande. Ouvrez une console en cliquant sur **Menu - Accessoires - LXTerminal**, comme expliqué au chapitre Raspberry Pi 3, premier contact. Tapez ensuite la commande :

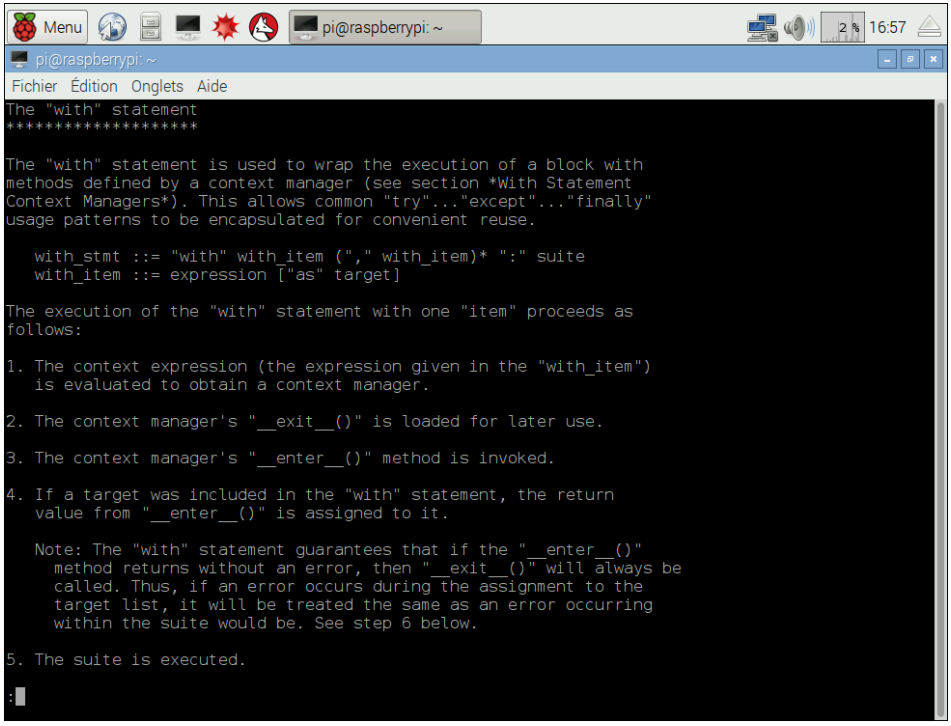
```
pi@raspberrypi:~ $ pydoc3
```

Ce qui devrait afficher le résultat suivant :



Cela n'est rien d'autre que l'aide de `pydoc3`. Attention cependant car chaque version de Python installée sur le système est livrée avec sa propre version de `pydoc`. Comme expliqué au chapitre Raspberry Pi 3, premier contact, différentes versions de l'interpréteur Python doivent cohabiter sur le Raspberry Pi. Il en est de même pour `pydoc` qui est livré dans ses versions 2.7 et 3.4. Par défaut, le binaire `pydoc` pointe vers `/usr/bin/pydoc2.7`. Pensez donc à toujours utiliser `pydoc3` pour lire la documentation en rapport avec la version 3 de Python.

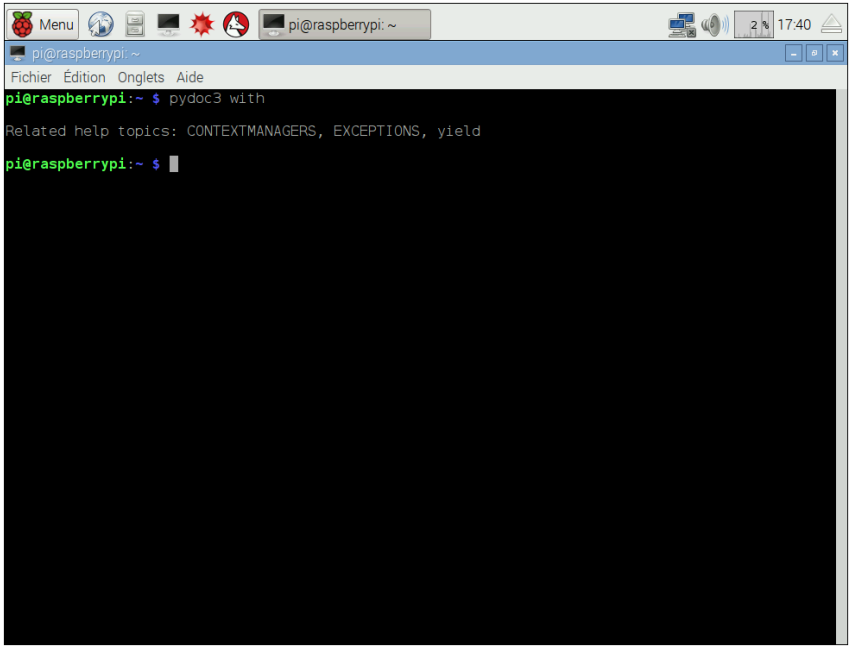
`pydoc3` permet donc de chercher et d'afficher la documentation de nombreux *topics* ou sujets. Le sujet recherché correspond au terme passé en paramètre de la commande. Dans cet exemple, le terme `with` est recherché et affiche la documentation qui lui est associée :



Par défaut, la lecture de la documentation repose sur l'utilisation d'un programme nommé `less`, qui est le pager par défaut de la plupart des distributions Linux actuelles. La navigation dans `less` peut être difficile pour les néophytes qui débutent avec la ligne de commande. Voici un tableau rassemblant les commandes de base :

Défiler vers le haut d'un caractère	[Flèche en haut]
Défiler vers le bas d'un caractère	[Flèche en bas]
Défiler vers le haut d'une page	[Page Up]
Défiler vers le bas d'une page	[Page Down]
Rechercher un terme	Barre oblique (slash) / suivi du terme
Prochaine occurrence du terme	Lettre en minuscule n
Précédente occurrence du terme	Lettre en majuscule N
Ferme (quitter) la documentation	Lettre en minuscule q ou en majuscule Q

Naviguer dans `less` est relativement facile lorsque ces quelques raccourcis sont assimilés. Une fois `less` fermé, `pydoc3` suggère des sujets annexes qui seraient susceptibles d'intéresser le développeur :

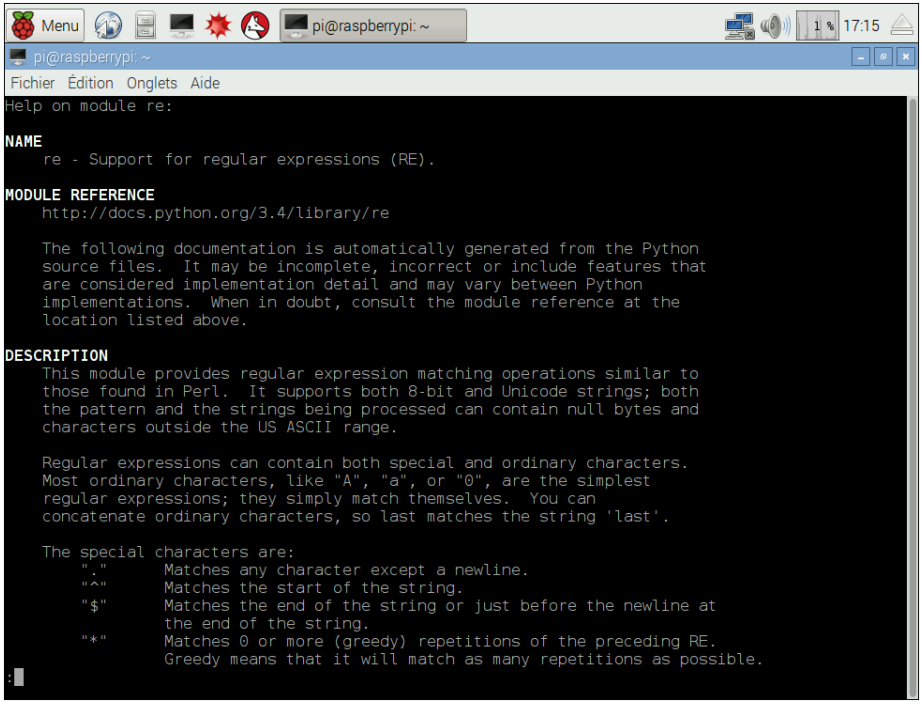


En plus de chercher de la documentation pour pratiquement n'importe quel sujet touchant au langage, `pydoc3` recherche aussi la documentation associée aux modules de la bibliothèque standard ainsi que les modules installés avec `pip3`, si l'auteur du module en question a évidemment pris le soin de l'écrire.

Pour y parvenir, il suffit de passer en paramètre de `pydoc3` le nom du module dont vous souhaitez consulter la documentation :

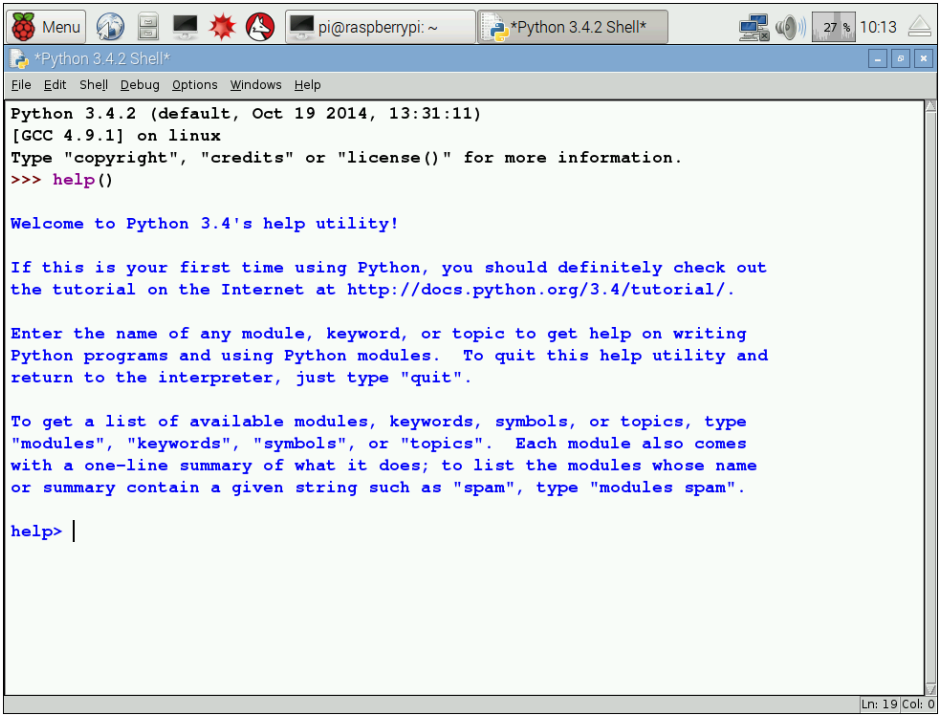
```
pi@raspberrypi:~ $ pydoc3 re
```

Ce qui a pour effet d'afficher la documentation associée au module `re` :



`pydoc3` est aussi disponible depuis le REPL d'IDLE grâce à la fonction `help()`. En effet, cette fonction fait indirectement appel à `pydoc3` pour la recherche de documentation. `help()` s'utilise de deux manières : la première consiste à appeler la fonction une fois dans le REPL en ne donnant aucun paramètre. L'utilisateur bascule alors dans une seconde console interactive spécialisée dans la recherche de documentation.

Tapez un terme et laissez la console effectuer la recherche afin de présenter l'aide associée au terme :



La deuxième consiste à passer en paramètre un objet. `help()` va alors analyser l'objet en question et afficher la documentation associée. Dans ce cas de figure, vous ne pouvez pas passer autre chose que des objets déjà instanciés ou importés dans le contexte courant.

Vous pouvez aussi afficher la documentation d'une fonction en particulier provenant d'un module :

```
>>> help(re)
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
NameError: name 're' is not defined
>>> import re
>>> help(re)
(affiche la documentation du module re)
>>> s = str
>>> help(s)
(affiche la documentation de la classe str)
>>> help(s.replace)
(affiche la documentation de la fonction replace de la classe str)
```