

Patrice CLEMENT

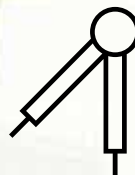
# Python et Raspberry Pi

Apprenez à développer  
sur votre nano-ordinateur

2<sup>e</sup> 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

1. Introduction au Raspberry Pi .....	7
2. Démarrage de Raspbian .....	11
2.1 Historique .....	11
3. Comprendre l'écosystème Python : quelle version utiliser ? .....	16
4. Installer des bibliothèques Python .....	17
4.1 La méthode aptitude .....	18
4.2 La méthode pip .....	19
4.3 pip ou aptitude ? .....	21
5. IDLE : l'éditeur de code en Python pour Python .....	22
6. Conclusion .....	27

### Chapitre 2

## Python : bases et concepts avancés

1. Hello World .....	29
2. Les types de base : int, float, str et bool .....	30
2.1 L'entier : int .....	30
2.2 Le flottant : float .....	31
2.3 La chaîne de caractères : str .....	31
2.4 Le booléen .....	35

## 2 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

3. Les structures de données : list, dict, tuple .....	35
3.1 La liste : list.....	35
3.2 Les tuples .....	36
3.3 La table de hachage ou dictionnaire : dict.....	37
3.4 L'ensemble : set.....	38
4. Les instructions, conditions et boucles .....	38
4.1 La condition if .....	39
4.2 La condition else .....	39
4.3 La boucle for .....	40
4.4 L'instruction break .....	40
4.5 L'instruction continue .....	41
4.6 La boucle while.....	41
5. Les opérateurs.....	42
5.1 Opérateurs arithmétiques .....	42
5.2 Opérateurs logiques.....	45
5.3 Opérateur d'appartenance.....	45
5.4 Opérateur d'affectation.....	46
6. Les classes : définition avec le mot-clé class .....	47
6.1 Premiers pas .....	48
6.2 Exposer les attributs d'une classe.....	49
6.3 Composition de classes.....	50
6.4 Réutiliser du code .....	51
7. Les fonctions : les mots-clés def et lambda .....	53
7.1 Définir une fonction .....	53
7.2 La fonction anonyme .....	56
8. La syntaxe en compréhension .....	57
9. Itérateur et générateur : les mots-clés iter et yield .....	60
10. La gestion des exceptions avec les mots-clés try, except, raise et finally .....	64
11. L'import des modules avec le mot-clé import .....	68
12. La gestion de contexte avec les mots-clés with et as .....	70
13. Conclusion.....	72

## Chapitre 3

**Administration du Raspberry Pi en Python**

1. Introduction .....	73
2. Naviguer dans le système de fichiers avec les modules os et pwd .....	74
2.1 Manipuler et interroger le système de fichiers .....	77
2.2 Explorer le système de fichiers du Raspberry Pi .....	78
3. Interagir avec l'interpréteur Python via le module sys .....	81
4. Lancer des commandes shell avec le module subprocess .....	83
5. Chercher des fichiers avec le module glob .....	85
6. Comparer des fichiers ou répertoires avec le module filecmp .....	87
7. Capturer des signaux UNIX avec le module signal .....	89
8. Écriture de scripts avec le module argparse .....	92
9. Conclusion .....	95

## Chapitre 4

**Le Raspberry Pi en console avec urwid**

1. Introduction .....	97
2. urwid, les fondamentaux .....	97
3. Projet #1 : une horloge en console .....	100
4. Projet #2 : un navigateur de fichiers en console .....	102
5. Conclusion .....	106

## Chapitre 5

**Programmation d'interfaces graphiques avec tkinter**

1. Les fondamentaux .....	107
2. Projet #1 : Hello world avec tkinter .....	110
3. Projet #2 : une visionneuse d'images .....	117
4. Projet #3 : un éditeur de texte .....	125
5. Conclusion .....	136

## 4 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

### Chapitre 6

## À l'assaut du Web avec le Raspberry Pi

1. Webscraping facile avec les modules urllib et HTMLParser .....	137
2. Développer un serveur HTTP avec le module http.server .....	142
3. Exécuter des scripts avec le module cgi .....	148
4. Envoyer des e-mails avec le module smtplib .....	155
5. Écrire une API légère avec Flask .....	158
6. Conclusion .....	166

### Chapitre 7

## Multimédia et audio sur le Raspberry Pi

1. Dessiner avec Pillow .....	167
1.1 Créer et manipuler des images .....	167
1.2 Dessiner des figures géométriques .....	173
2. Contrôler les entrées et sorties audio avec pyalsaudio .....	175
3. Projet #1 : un enregistreur/lecteur audio .....	182
4. Conclusion .....	187

### Chapitre 8

## Persistance de données sur le Raspberry Pi

1. Introduction .....	189
2. Sérialisation et désérialisation avec les modules pickle et shelve .....	189
3. Traiter des fichiers CSV avec le module csv .....	193
3.1 Création et lecture d'un fichier CSV .....	193
3.2 Créer son propre dialecte CSV .....	195
4. Manipuler des données XML avec le module xml.etree.ElementTree .....	196
4.1 Créer et sérialiser un fichier XML .....	197
4.2 Interroger un fichier XML .....	198
4.3 Ajouter et supprimer des nœuds .....	199
5. Travailler avec le format d'échange de données JSON via le module json .....	201

6. Gestion d'une base de données SQL légère avec le module sqlite3.....	203
7. Conclusion.....	208

## Chapitre 9

### Documenter et tester ses scripts en Python

1. Introduction.....	209
2. Consulter de la documentation avec pydoc3.....	210
3. Documenter et tester son code en une seule fois avec le module doctest.....	217
4. Écriture de tests unitaires avec le module unittest.....	221
5. Benchmarker son code avec le module timeit.....	226
6. Déboguer ses programmes avec le module pdb.....	230
6.1 Déboguer pas à pas.....	231
6.2 Déboguer à un endroit précis du programme.....	234
6.3 Procéder à l'autopsie de son programme.....	234
7. Conclusion.....	235

## Chapitre 10

### Raspberry Pi et GPIO

1. Les GPIO, comment ça marche ?.....	237
2. Connecter un écran LCD 16x2 au Raspberry Pi.....	239
3. Projet #1 : communiquer avec l'écran LCD.....	245
4. Projet #2 : créer un tube FIFO dédié à l'écran LCD.....	248
5. Projet #3 : écrire des messages depuis une interface en ligne de commande.....	253
6. Projet #4 : piloter l'écran LCD depuis une interface graphique tkinter.....	256
7. Conclusion.....	259

Index.....	261
------------	-----



## 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.

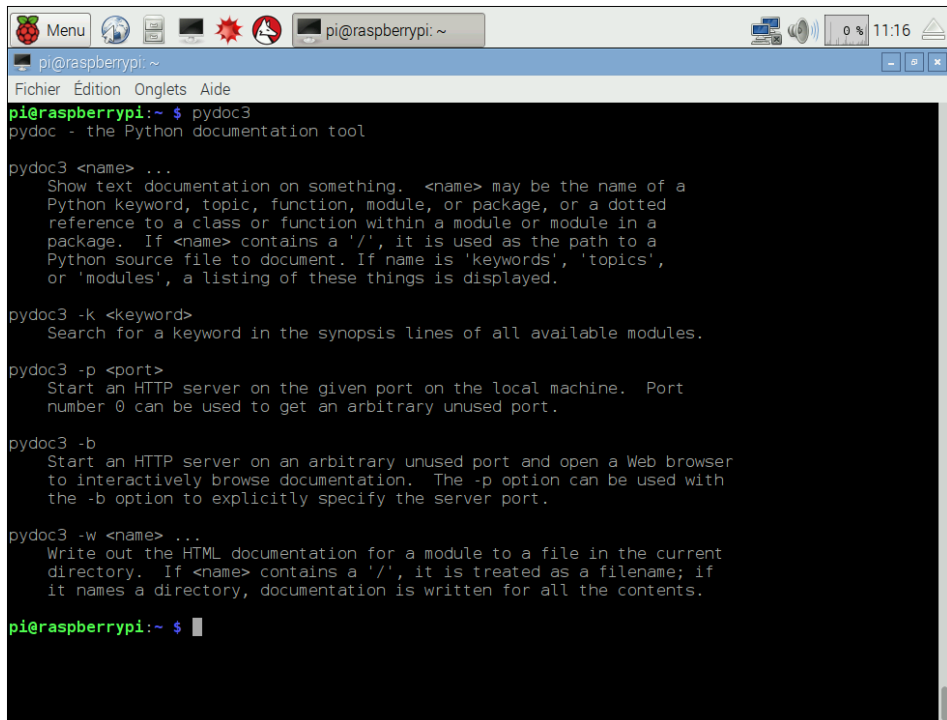
## 2. Consulter de la documentation avec pydoc3

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 :



```

pi@raspberrypi:~ $ pydoc3
pydoc - the Python documentation tool

pydoc3 <name> ...
  Show text documentation on something. <name> may be the name of a
  Python keyword, topic, function, module, or package, or a dotted
  reference to a class or function within a module or module in a
  package. If <name> contains a '/', it is used as the path to a
  Python source file to document. If name is 'keywords', 'topics',
  or 'modules', a listing of these things is displayed.

pydoc3 -k <keyword>
  Search for a keyword in the synopsis lines of all available modules.

pydoc3 -p <port>
  Start an HTTP server on the given port on the local machine. Port
  number 0 can be used to get an arbitrary unused port.

pydoc3 -b
  Start an HTTP server on an arbitrary unused port and open a Web browser
  to interactively browse documentation. The -p option can be used with
  the -b option to explicitly specify the server port.

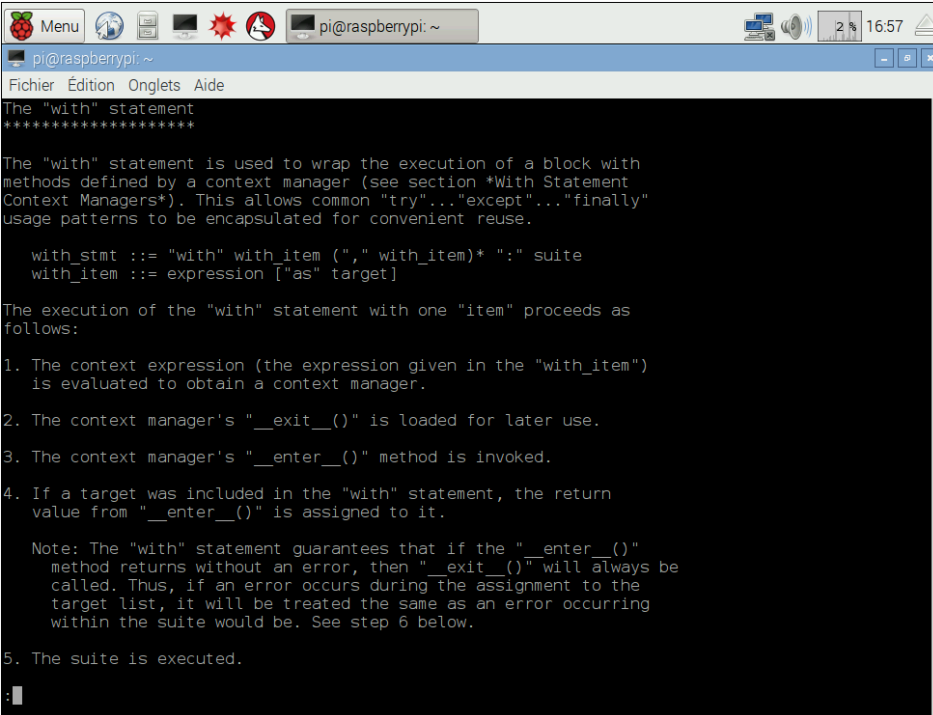
pydoc3 -w <name> ...
  Write out the HTML documentation for a module to a file in the current
  directory. If <name> contains a '/', it is treated as a filename; if
  it names a directory, documentation is written for all the contents.

pi@raspberrypi:~ $
  
```



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 :



```

pi@raspberrypi: ~
Fichier Édition Onglets Aide
The "with" statement
*****

The "with" statement is used to wrap the execution of a block with
methods defined by a context manager (see section *With Statement
Context Managers*). This allows common "try"... "except"... "finally"
usage patterns to be encapsulated for convenient reuse.

    with_stmt ::= "with" with_item ("," with_item)* ":" suite
    with_item ::= expression ["as" target]

The execution of the "with" statement with one "item" proceeds as
follows:

1. The context expression (the expression given in the "with_item")
   is evaluated to obtain a context manager.
2. The context manager's "__exit__()" is loaded for later use.
3. The context manager's "__enter__()" method is invoked.
4. If a target was included in the "with" statement, the return
   value from "__enter__()" is assigned to it.

Note: The "with" statement guarantees that if the "__enter__()"
method returns without an error, then "__exit__()" will always be
called. Thus, if an error occurs during the assignment to the
target list, it will be treated the same as an error occurring
within the suite would be. See step 6 below.

5. The suite is executed.

:

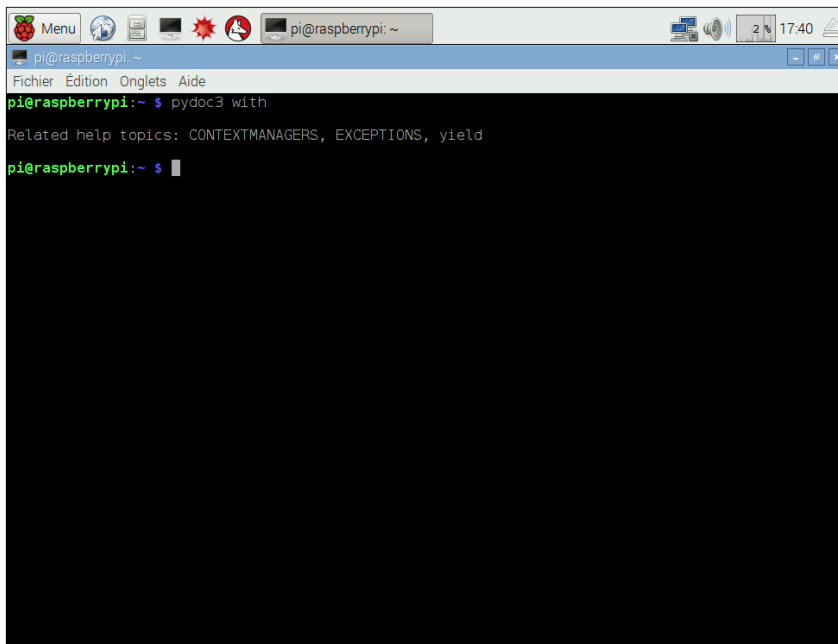
```

## 212 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

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 :



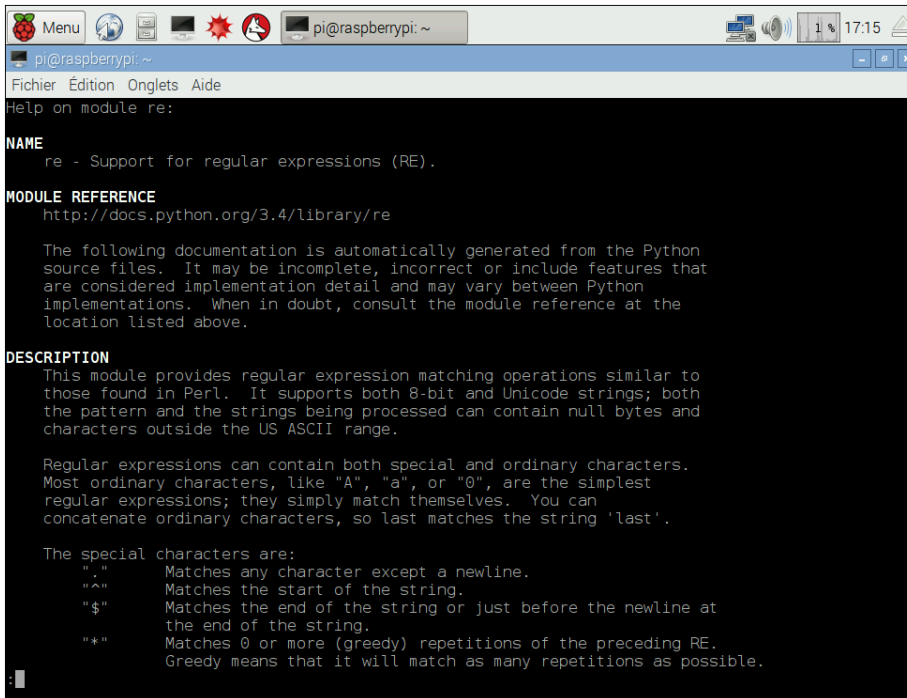
```
pi@raspberrypi: ~
Fichier Edition Onglets Aide
pi@raspberrypi:~ $ pydoc3 with
Related help topics: CONTEXTMANAGERS, EXCEPTIONS, yield
pi@raspberrypi:~ $
```

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` :



```

pi@raspberrypi: ~
Fichier  Edition  Onglets  Aide
Help on module re:

NAME
  re - Support for regular expressions (RE).

MODULE REFERENCE
  http://docs.python.org/3.4/library/re

  The following documentation is automatically generated from the Python
  source files. It may be incomplete, incorrect or include features that
  are considered implementation detail and may vary between Python
  implementations. When in doubt, consult the module reference at the
  location listed above.

DESCRIPTION
  This module provides regular expression matching operations similar to
  those found in Perl. It supports both 8-bit and Unicode strings; both
  the pattern and the strings being processed can contain null bytes and
  characters outside the US ASCII range.

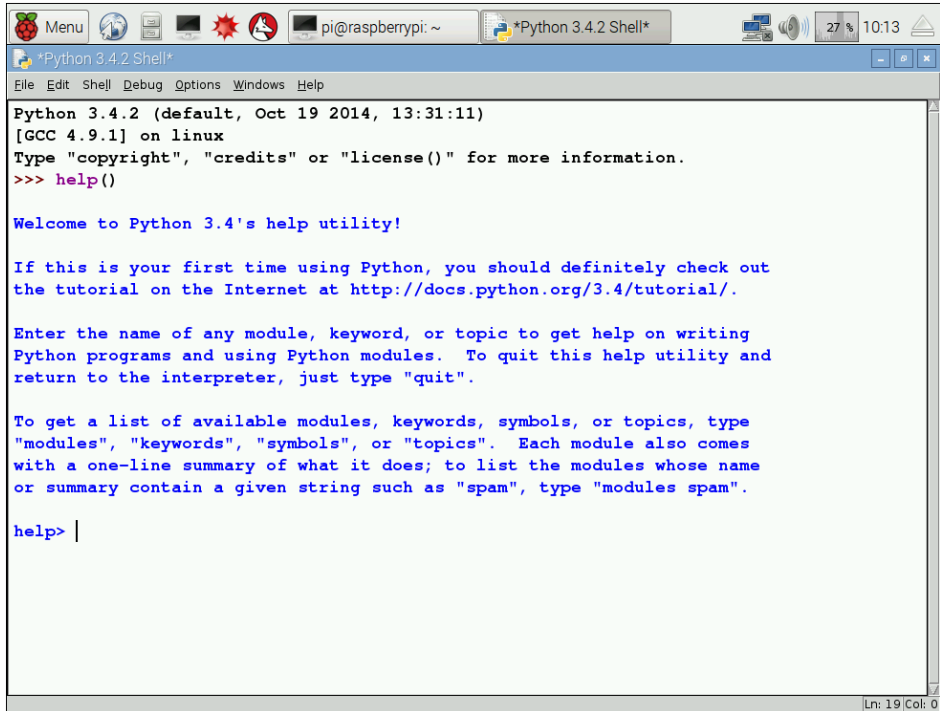
  Regular expressions can contain both special and ordinary characters.
  Most ordinary characters, like "A", "a", or "0", are the simplest
  regular expressions; they simply match themselves. You can
  concatenate ordinary characters, so last matches the string 'last'.

  The special characters are:
  "." Matches any character except a newline.
  "^" Matches the start of the string.
  "$" Matches the end of the string or just before the newline at
  the end of the string.
  "*" Matches 0 or more (greedy) repetitions of the preceding RE.
  Greedy means that it will match as many repetitions as possible.
  
```

`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.

## 214 Python et Raspberry Pi - Apprenez à développer sur votre nano-ordinateur

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



```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> help()

Welcome to Python 3.4's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> |
```

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)
```