Full Circle

LE MAGAZINE INDÉPENDANT DE LA COMMUNAUTÉ UBUNTU LINUX

ÉDITION SPÉCIALE SÉRIE PROGRAMMATION

ÉDITION SPÉCIALE
SÉRIE PROGRAMMATION

PROGRAMMER EN PYTHON

Volume neuf

Parties 49 à 53

Au sujet du Full Circle

Le Full Circle est un magazine gratuit, libre et indépendant, consacré à toutes les versions d'Ubuntu, qui fait partie des systèmes d'exploitation Linux. Chaque mois, nous publions des tutoriels, que nous espérons utiles, et des articles proposés par des lecteurs. Le Podcast, un complément du Full Circle, parle du magazine même, mais aussi de tout ce qui peut vous intéresser dans ce domaine.

Clause de non-responsabilité :

Cette édition spéciale vous est fournie sans aucune garantie ; les auteurs et le magazine Full Circle déclinent toute responsabilité pour des pertes ou dommages éventuels si des lecteurs choisissent d'en appliquer le contenu à leurs ordinateur et matériel ou à ceux des autres.

Spécial Full Circle Magazine

Full Circle

LE MAGAZINE INDÉPENDANT DE LA COMMUNAUTÉ UBUNTU LINUX

Bienvenue dans une nouvelle édition spéciale consacrée à un seul sujet !

En réponse aux requêtes des lecteurs, nous avons réuni le contenu de certains articles consacrés à la programmation en python.

Pour l'instant, il s'agit d'une réédition directe de la série **Programmer en Python**, **parties 49 à 53**, des numéros 79 à 84 (en exceptant le n° 83), par l'incomparable professeur en Python Greg Walters.

Veuillez considérer l'origine de la publication ; les versions actuelles du matériel et des logiciels peuvent différer de ceux que nous présentons, ainsi vérifiez bien votre matériel et la version de vos logiciels avant d'émuler les tutoriels de cette édition spéciale. Vous pouvez installer des versions de logiciels plus récentes ou disponibles dans les dépôts de votre distribution.

Amusez-vous !

Nos coordonnées

SiteWeb :
http://www.fullcirclemagazine.org/

Forums :
http://ubuntuforums.org/forumdisplay.php?f=270

IRC : #fullcirclemagazine on chat.freenode.net

Équipe éditoriale :
Rédacteur en chef : Ronnie Tucker (pseudo : RonnieTucker)
ronnie@fullcirclemagazine.org
Webmaster : Rob Kerfia (pseudo : admin / linuxgeekery-
admin@fullcirclemagazine.org

Nos remerciements vont à Canonical ainsi qu'aux nombreuses équipes de traduction à travers le monde.

TUTORIEL Programmer en Python - Partie 49

Alors que j'étais au boulot cette semaine, une personne très sage du nom de Michael W. m'a suggéré d'examiner ce qui se passe avec des nombres à virgule flottante et l'égalité.

Prenez par exemple un simple calcul : 1,1 + 2,2

La réponse, direz-vous, est 3,3 ! Toute enfant qui a manipulé des fractions à l'école le sait. Eh bien, dites ça à votre ordinateur. Si vous démarrez le Shell Interactif Python et saisissez à l'invite :

```
(1.1+2.2) == 3.3,
```

vous pourriez être surpris qu'il réponde :

```
« False » (faux).
```

QUOI ?!?!?

Maintenant, confus, vous tapez à l'invite :

```
>>>1.1+2.2
```

Et la réponse est :

```
3.3000000000000003
```

Vous regardez fixement l'écran, incrédule, et pensez d'abord : « je dois avoir saisi quelque chose de travers ». Ensuite, vous vous rendez compte que non. Et vous tapez :

```
>>>2.2+3.3

5.5
```

Maintenant, vous êtes encore plus confus et vous vous dites : « D'accord. Il s'agit soit d'un bug soit d'une sorte d'œuf de Pâques [Ndt : blague informatique] ». Non, ce n'est ni un bug ni un œuf de Pâques. C'est réel. Même si je connaissais ce phénomène il y a très longtemps, il avait glissé dans les profondeurs et les recoins les plus sombres de mon vieux cerveau et il a fallu que je le fasse remonter. Ce que nous voyons là est la joie des nombres binaires à virgule flottante.

Nous savons tous que ⅓ vaut 0,33333333333333333… à l'infini, mais prenons par exemple la fraction 1/10. Tout le monde sait que 1/10 est égal à 0,1, non ? Si vous utilisez le shell interactif, vous pouvez voir que :

```
>>>1/10

0
```

Oh, c'est vrai. Nous devons avoir au moins l'une des valeurs en virgule flottante pour voir les décimales car entier/entier retourne un entier. Donc, nous essayons de nouveau.

```
>>>1/10.0

0.1
```

Bon. La réalité est de retour. Non, pas vraiment. Python vous montre simplement une version arrondie de la réponse. Alors, comment voyons-nous la « vraie » réponse ? Nous pouvons utiliser la bibliothèque décimale pour voir ce qui se passe réellement.

```
>>> from decimal import *
>>> Decimal(1/10.0)

Decimal('0.100000000000000055511151231257827021181583404541015625')
```

WOW. Essayons donc notre formule originale et voyons ce que cela affiche :

```
>>> Decimal(1.1+2.2)

Decimal('3.300000000000000266453525910037569701671600341796875')
```

Cela semble être de pire en pire. Alors qu'est-ce qui se passe réellement ?

C'est ce qu'on appelle une Erreur de représentation, elle existe dans presque tous les langages de programmation moderne (Python, C, C++, Java, Fortran et d'autres) et sur presque tous les ordinateurs modernes. C'est parce que ces machines utilisent l'arithmétique en virgule flottante IEEE-754 qui (sur la plupart des machines et des systèmes d'exploitation) correspond à un nombre en double précision IEEE-754. Ce nombre en double précision a une précision de 53 bits. Ainsi, notre 0,1, quand il est représenté en 53-bit double précision, se transforme en :

```
0.0001100110011001100110011001100110011001100110011010
```

C'est proche de 0,1 mais pas assez proche pour éviter les problèmes.

Alors, que faisons-nous avec ça ? Eh bien, la réponse rapide est que vous pouvez probablement vivre avec dans 90% des cas que nous avons à traiter dans le monde réel – en

utilisant la méthode round(). Vous devrez décider du nombre de décimales dont vous avez besoin dans votre monde pour avoir la précision dont vous avez besoin, mais la plupart du temps ce sera une solution acceptable.

Je ne me souviens pas vraiment si nous avons vu la méthode round, donc je vais la décrire brièvement. La syntaxe est très simple :

```
round(v,d)
```

où v est la valeur que vous souhaitez arrondir et d est le nombre de décimales (maximum) que vous voulez après la virgule. Selon la documentation Python, « Les valeurs sont arrondies au plus proche multiple de 10 à la puissance moins n, si deux multiples sont à égale distance, l'arrondi se fait en s'écartant du 0 ». Tout cela étant dit, si le nombre est 1,4144, et que nous arrondissons à 3 décimales, la valeur retournée sera 1,414. Si le nombre est 1,4145, il serait arrondi à 1,415.

Par exemple, utilisons la valeur de pi qui provient de la bibliothèque mathématique. (Vous devez importer la bibliothèque « math » avant de pouvoir le faire, d'ailleurs.)

```
>>> math.pi
3.141592653589793
```

Maintenant, si nous voulons arrondir cette valeur à 5 décimales, on peut utiliser :

```
>>>  round(math.pi,5)
3.14159
```

C'est la valeur « standard » de pi que presque tout le monde connaît par cœur. C'est très bien. Cependant, si nous fixons le nombre de décimales à renvoyer à 4, regardez ce qui se passe.

```
>>> round(math.pi,4)
3,1416
```

Tout cela fonctionne parfaitement, jusqu'à ce que vous ayez une valeur comme 2,675 et essayez de l'arrondir à 2 décimales. L'hypothèse (car on est exactement à mi-chemin entre 2,67 et 2,68), c'est que la valeur retournée sera 2,68. Essayez-le.

```
>>>  round(2.675,2)
2.67
```

Cela pourrait poser un problème. Et on revient à la question initiale dont nous parlions. Avec la conversion en un nombre binaire à virgule flottante de 53 bits de long, le nombre devient :

```
2,67499999999999822365316059
974953522189331054687 5
```

qui est arrondi à 2,67.

L'essentiel ici est qu'en essayant de comparer les nombres à virgule flottante, il faut être conscient que certaines choses ne se traduisent pas bien.

Rendez-vous la prochaine fois !

Greg Walters est propriétaire de Rainy-Day Solutions LLC, une société de consultants à Aurora au Colorado, et programme depuis 1972. Il aime faire la cuisine, marcher, la musique et passer du temps avec sa famille. Son site web est www.thedesignatedgeek.net.

ubuntu uk podcast

Le Podcast Ubuntu couvre toutes les dernières nouvelles et les problèmes auxquels sont confrontés les utilisateurs de Linux Ubuntu et les fans du logiciel libre en général. La séance s'adresse aussi bien au nouvel utilisateur qu'au plus ancien codeur. Nos discussions portent sur le développement d'Ubuntu, mais ne sont pas trop techniques. Nous avons la chance d'avoir quelques supers invités, qui viennent nous parler directement des derniers développements passionnants sur lesquels ils travaillent, de telle façon que nous pouvons tous comprendre ! Nous parlons aussi de la communauté Ubuntu et de son actualité.

Le podcast est présenté par des membres de la communauté Ubuntu Linux du Royaume-Uni. Il est couvert par le Code de Conduite Ubuntu et est donc adapté à tous.

L'émission est diffusée en direct un mardi soir sur deux (heure anglaise) et est disponible au téléchargement le jour suivant.

podcast.ubuntu-uk.org

Ce mois-ci, je vais parler de deux fonctions un peu moins connues : maketrans et translate.

Nous allons commencer par la méthode translate. La méthode translate retourne une copie d'une chaîne, avec tous les caractères de la table translate remplacés (traduits) ou bien avec des caractères supprimés de la chaîne, s'ils sont contenus dans « deletecharacters » (optionnel). Voici la syntaxe à utiliser :

```
s =
str.translate(table[,deletech
aracters])
```

Avant de passer à la partie « table » de cette méthode, regardons la partie suppression. Disons que vous avez la chaîne «The time has come » et que vous voulez supprimer toutes les voyelles (pour une raison certes étrange, mais pourquoi pas) de cette chaîne. Vous pouvez coder comme ceci :

```
astr = “The time has come”

astr.translate(None,’aeiou’)
```

cela retourne :

```
“Th tm hs cm”
```

Notez que nous avons inclus « None » comme table de traduction. Bien que cette méthode soit déjà assez sympa, il y a mieux avec une autre méthode appelée « maketrans ». Cette méthode utilise une chaîne d'entrée et une chaîne de sortie en tant que paramètres et renvoie une table qui sera utilisée en tant que premier paramètre de la méthode « translate ». Vous trouverez, en haut à droite de cette page, un exemple très simple.

Cela retourne :

```
“Th2 t3m2 h1s c4m2”
```

Regardons comment ça marche… Nous attribuons à « intable » une chaîne de voyelles comme auparavant, « outtable » utilise les numéros 1,2,3,4,5 comme une chaîne. Lorsque nous faisons appel à « maketrans », notre « trantable » réelle est la suivante (illustrée ci-dessous « \x » signifie qu'il est de type char hexadécimal) :

```
intable = ‘aeiou’
outtable = ‘12345’
trantable = maketrans(intable,outtable)
astr = “The time has come”
astr.translate(trantable)
```

Si vous regardez attentivement, vous vous apercevrez que les voyelles minuscules sont remplacées par les chiffres que nous avons spécifiés :

```
1bcd2fgh3jklmn4pqrst5vwxyz
```

Si vous regardez encore de plus près, vous verrez qu'il y a effectivement 256 entrées, en commençant par « \x00 » et en se terminant par « \xff ». Ainsi, le tableau contient l'ensemble des 256 codes possibles du jeu de caractères ascii. Ainsi, lorsque la méthode « translate » parcourt chaque caractère du tableau, elle obtient la valeur des caractères en hexadécimal, puis recherche la valeur dans le tableau « translate » et la remplace dans la chaîne de sortie. La représentation hexadécimale de notre chaîne « astr » originale (« The time has come ») est représentée ci-dessous.

Maintenant, vous devez commencer à comprendre.

Alors, à quoi tout cela peut-il servir ? Faites un retour en arrière et pensez à la période où vous avez étudié le personnage de Jules César à l'école… Lorsque César voulait envoyer un message à caractère confidentiel, il utilisait un chiffrement qui consistait à décaler, de trois caractères vers la droite, les lettres de l'alphabet. Ainsi, en utilisant l'alphabet latin d'aujourd'hui :

```
\x54\x68\x65\x20\x74\x69\x6d\x65\x20\x68\x61\x73\x20\x63\x6f\x6d\x65
  T    h    e         t    i    m    e         h    a    s         c    o    m    e
```

```
'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\
x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./0123456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`1bcd2fgh3jklmn4pqrst5vwxyz{|}~\x7f\x80\x81\x82\x83
\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97
\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab
\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf
\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3
\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7
\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb
\xfc\xfd\xfe\xff'
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz
```

cela se transformait en :

```
DEFGHIJKLMNOPQRSTUVWXYZabcdef
ghijklmnopqrstuvwxyzABC
```

Bien que cela semble très simple par rapport aux normes de chiffrement d'aujourd'hui, quand j'étais petit, nous utilisions également cette méthode de chiffrement pour envoyer des messages entre nous. Nous utilisions un indice (de décalage) différent pour démarrer la chaîne de chiffrement à un autre endroit mais la logique restait la même.

Personne ne sait si cette méthode fut un succès pour ce bon vieux Jules César mais on peut penser que si quelqu'un interceptait un tel message à l'époque, il aurait certainement cru qu'il s'agissait d'une langue étrangère ! Enfin, c'est ce que l'on peut supposer…

Nous pouvons facilement utiliser la méthode « translate » et la fonction « maketrans » pour nous amuser un peu avec ça. Partons du principe que nous voulons faire un programme simple qui nous permet d'entrer une chaîne de « texte en clair » et d'obtenir en retour une chaîne cryptée en utilisant la même méthode de décalage à droite que celle utilisée par César. Par souci de simplicité, nous allons uniquement utiliser des caractères majuscules (voir en haut et à droite de cette page).

Tout ce que vous trouverez dans le code à droite est à peu près ce que nous avons vu au début de cet article ou dans les précédents articles sur Python, mais je vais expliquer rapidement…

Les deux premières lignes représentent les chaînes d'entrée et de sortie. Nous avons juste déplacé les caractères vers la droite pour créer la chaîne de sortie. Les deux lignes suivantes représentent les fonctions d'encodage et de décodage. La ligne 5 invite l'utilisateur à saisir une chaîne à encoder. On encode alors cette chaîne (EncString) dans la ligne suivante. Pour décoder, nous utilisons simplement la méthode « translate » sur la chaîne codée pour obtenir le texte en clair. Enfin, nous affichons les deux chaînes. Voici ce qu'affiche le programme :

```
Enter the plaintext string ->
THE TIME HAS COME
Encoded string is -
WKH WLPH KDV FRPH
Decoded string is -
THE TIME HAS COME
```

Voilà, c'est comme à l'école ! Mais nous allons étoffer un peu notre code pour le rendre un peu plus utilisable. Le code est pratiquement le même à quelques exceptions près. Tout d'abord, nous avons ajouté une espace à la fin de la chaîne intab et entre le « Z » et le « A » dans la chaîne outtab. Cela permet de mieux cacher les mots d'origine dans notre chaîne cryptée. L'autre changement concerne l'endroit où nous demandons si l'utilisateur veut coder ou décoder la chaîne. Enfin , nous avons ajouté une instruction « if » pour contrôler ce que nous affichons (voir en bas).

```
from string import maketrans
#----------------------
intab  = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
outtab = "DEFGHIJKLMNOPQRSTUVWXYZABC"
EncTrantab = maketrans(intab,outtab) #Encode
DecTrantab = maketrans(outtab,intab) #Decode
instring = raw_input("Enter the plaintext string -> ")
EncString = instring.translate(EncTrantab)
DecString = EncString.translate(DecTrantab)
print("Encoded string is - %s" % EncString)
print("Decoded string is - %s" % DecString)
```

La sortie du programme est :

```
Encode or Decode (E or D) -> E
Enter the string -> THE TIME
HAS COME
Encoded string is -
WKHCWLPHCKDVCFRPH
```

Et pour tester le côté « décodage » de la chose :

```
Encode or Decode (E or D) -> D
Enter the string ->
WKHCWLPHCKDVCFRPH
Decoded string is - THE TIME
HAS COME
```

Eh bien, j'espère que vous commencez à avoir des idées sur la façon d'utiliser ces nouvelles informations dans votre propre code.

Rendez-vous la prochaine fois !

```
from string import maketrans

#Be sure to include the space character in the strings
intab  = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "
outtab = "DEFGHIJKLMNOPQRSTUVWXYZ ABC"
EncTrantab = maketrans(intab,outtab) #Encode
DecTrantab = maketrans(outtab,intab) #Decode

which = raw_input("Encode or Decode (E or D) -> ")
instring = raw_input("Enter the string -> ")
EncString = instring.translate(EncTrantab)
DecString = instring.translate(DecTrantab)

if which == "E":
    print("Encoded string is - %s" % EncString)
else:
    print("Decoded string is - %s" % DecString)
```

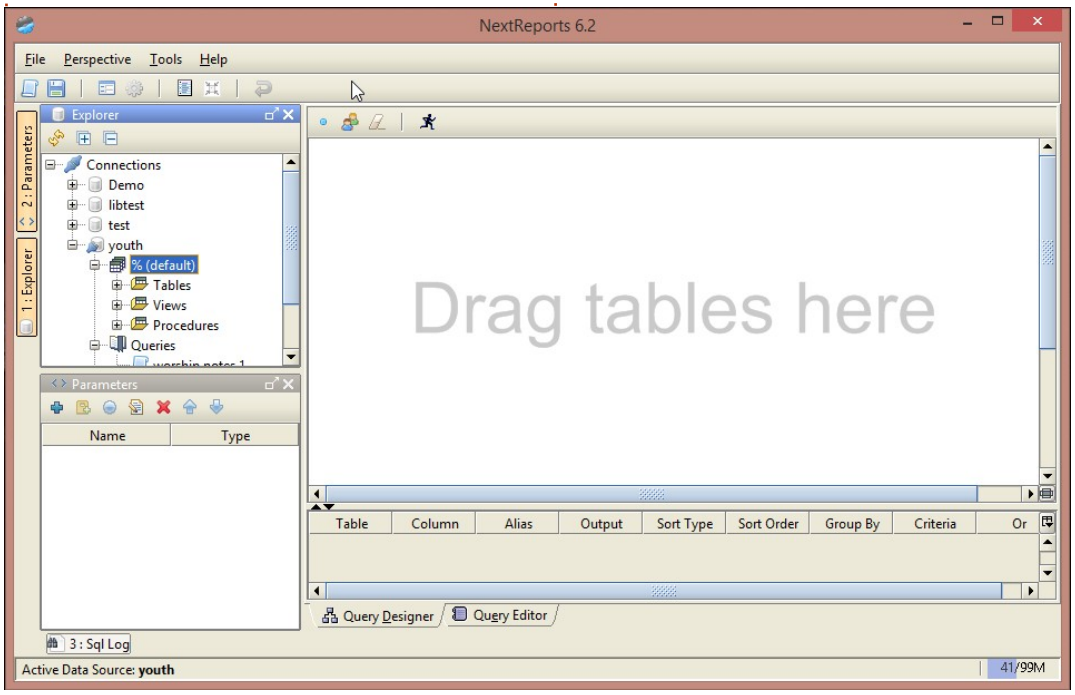
Ce mois-ci je vais vous parler d'un outil que je ne connaissais pas, mais qui existe apparemment depuis un certain nombre d'années. Il s'agit de NextReports de Advantage Software Factory, et vous pouvez l'obtenir gratuitement sur http://www.next-reports.com/. En plus, il est Open Source et il fonctionne sous Windows et Linux !

Avant de commencer, laissez-moi vider un peu mon sac pendant une minute ou deux. Je travaille avec des bases de données et des rapports depuis longtemps. Une des choses qui m'a le plus ennuyé, c'est que, bien qu'il existe des solutions de base de données libres, comme SQLite et MySQL, il y a vraiment très peu d'outils de génération de rapports gratuits. Le plus souvent, les rapports devaient être réalisés à l'aide de logiciels très coûteux, ou alors le développeur devait se les fabriquer. Certains outils étaient disponibles, mais insatisfaisants. Et concernant les graphiques, vous n'aviez souvent pas d'autre choix que d'utiliser des produits chers. Croyez-moi, cela fait des années que je cherche de bons outils de reporting libres, et je ne sais vraiment pas comment j'ai pu manquer celui-ci pendant si longtemps (la version 2.1 est sortie en mars 2009 et ils en sont actuellement à la version 6.3). Mais maintenant que je l'ai trouvé, j'en suis absolument dingue.

Maintenant que j'ai dit ce que je tenais à dire, je peux commencer à vous vanter ses qualités. C'est une suite en trois parties, un concepteur de rapports, un moteur de rapport et un serveur de rapports. J'ai seulement eu l'occasion de jouer avec le concepteur de rapports, mais si ce concepteur donne un indice de la puissance, la facilité et la flexibilité du reste de la suite, alors on tient là le gros lot.

Ce mois-ci, nous allons nous concentrer sur le concepteur. En raison de certaines contraintes de temps, je travaille sur une machine Windows, mais tout ce que je montre peut être fait sous Linux (donc s'il vous plaît ne m'en tenez pas rigueur).

Une des premières choses que vous devez savoir, c'est que NextReports prend en charge les bases de données comme Oracle, MySql, SQLite, MSSQL et autres. Tout est basé sur les requêtes et, c'est une très bonne chose, seules les requêtes de type SELECT sont autorisées. Cela signifie que rien dans la base de données source ne peut être modifié par accident. Vous pouvez entrer vos propres requêtes ou utiliser un concepteur visuel.

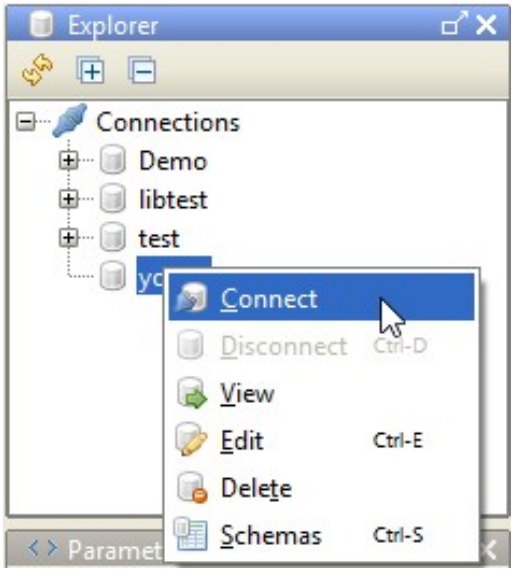


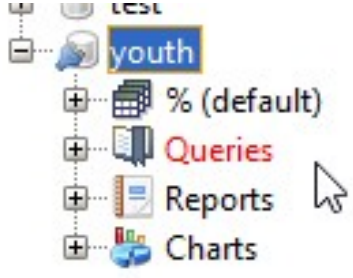
La capture d'écran montre à quel point l'interface est agréable. Les choses sont assez intuitives et être productif ne vous prendra pas longtemps. Jetons un regard sur les étapes à réaliser.

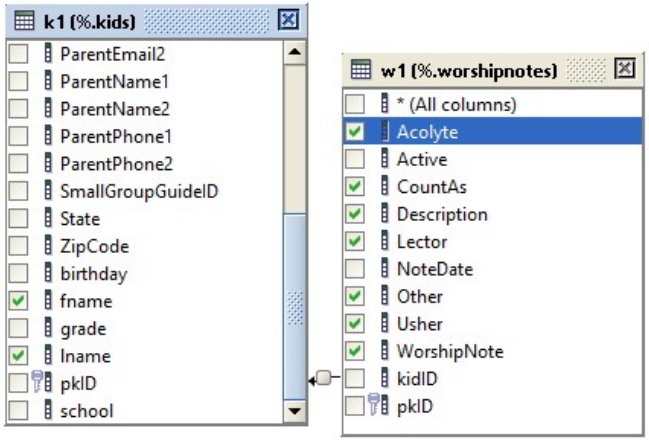
Commencez par Fichier | Nouveau | Source de données. Ensuite, nommez votre source comme vous le souhaitez.

Maintenant, indiquez à NextReports le type de base de données à l'aide de la liste déroulante « Type : ». Vous pouvez passer la section Pilote et aller à la section URL pour indiquer le chemin de la base de données. Si vous utilisez par exemple une base de données SQLite, ce sera pré-rempli pour vous : « jdbc:sqlite:<chemin-

fichier-base> ». Remplacez <chemin-fichier-base> par le chemin d'accès à votre base de données. D'autres types de bases de données possèdent des informations similaires déjà pré-remplies pour vous aider. Ensuite, cliquez sur le bouton « Test » pour vous assurer que vous pouvez vous connecter. Si tout se passe correctement, cliquez sur « Enregistrer » et vous verrez une entrée supplémentaire dans l'arbre des connexions. La prochaine étape est de créer une







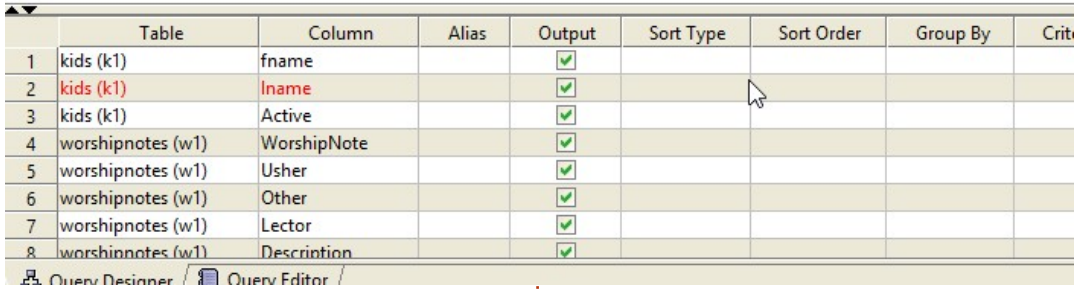


	Table	Column	Alias	Output	Sort Type	Sort Order	Group By	Crit
1	kids (k1)	fname		☑				
2	kids (k1)	lname		☑				
3	kids (k1)	Active		☑				
4	worshipnotes (w1)	WorshipNote		☑				
5	worshipnotes (w1)	Usher		☑				
6	worshipnotes (w1)	Other		☑				
7	worshipnotes (w1)	Lector		☑				
8	worshipnotes (w1)	Description		☑				

Query Designer | Query Editor

connexion à la base de données que vous venez d'ajouter. Maintenant, faites un clic droit sur la base de données, puis cliquez sur Connecter.

Une fois connecté, vous verrez que vous avez quatre choix possibles. Le « % » est pour les tables de bases de données. Les trois choix suivants permettent de créer de nouvelles requêtes, rapports et graphiques… plutôt simple. Maintenant, cliquez sur le signe « + » à gauche de « % » pour ouvrir l'affichage des tables de votre base de données. Vous voyez alors les Tables, les Vues et les Procédures dans l'arbre. À nouveau, cliquez sur le signe « + » à côté de « Tables », cela affichera toutes vos tables. Maintenant, si vous voulez utiliser le concepteur visuel de requêtes, il suffit de glisser la(les) table(s) que vous voulez traiter sur le canevas du concepteur à droite.

Une fois que toutes vos tables s'y trouvent, vous pouvez commencer à créer des liens entre les tables.

Dans mon exemple, j'ai deux tables, l'une avec des informations sur des enfants dans une classe de catéchisme et l'autre avec des entrées pour les notes de culte. La table des notes de culte ne contient pas le nom de l'enfant, juste un identifiant qui pointe vers la table d'informations sur l'enfant. J'ai fait un glisser-déposer pour créer le lien entre le champ kidID et le champ pkID de la table des enfants. Ensuite, j'ai choisi chaque champ que je voulais voir dans le jeu de résultats.

Table	Column	Alias	Output	Sort Type	Sort Order	Group By	Criteria
kids (k1)	fname		☑	Ascending	2		
kids (k1)	lname		☑	Ascending	1		
kids (k1)	Active		☐				= 1
worshipnotes (w1)	WorshipNote		☑				
worshipnotes (w1)	Usher		☑				
worshipnotes (w1)	Other		☑				
worshipnotes (w1)	Lector		☑				

Dans ce cas, le prénom et le nom de l'enfant et un drapeau actif (ou non supprimé) dans la table des enfants et plusieurs champs de la table des notes. La grille ci-dessous montre chacun des champs, de quelle table il provient, et d'autres informations.

```
SELECT
    kl.fname,
    kl.lname,
    wl.WorshipNote,
    wl.Usher,
    wl.Other,
    wl.Lector,
    wl.Description,
    wl.CountAs,
    wl.Acolyte
FROM
    kids kl,
    worshipnotes wl
WHERE
    wl.kidID = kl.pkID AND
    kl.Active = 1
ORDER BY
    kl.lname,
    kl.fname
```

Comme vous le constatez, nous pouvons indiquer des critères comme « actif = 1 », choisir d'afficher un champ ou pas, et définir le type de tri et l'ordre de tri. Une fois le résultat voulu obtenu, vous pouvez cliquer sur l'onglet en-dessous et voir votre requête SQL réelle.

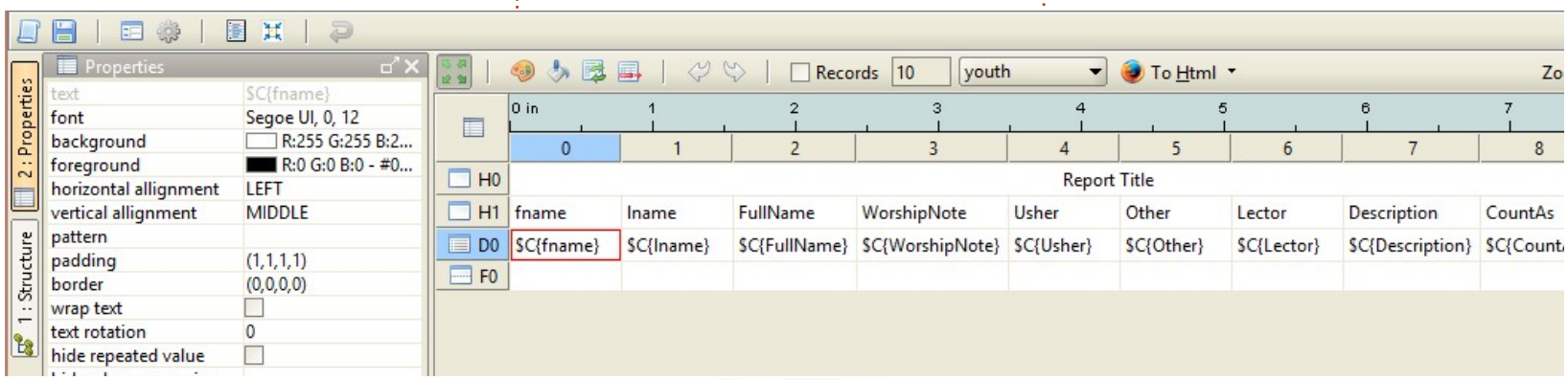
Pour tester votre requête, il suffit de cliquer sur l'icône représentant « l'homme qui court » et vous obtiendrez (si vous avez tout fait correctement) les résultats de la requête dans une grille sous l'éditeur. Si vous voulez ajouter des lignes manuellement, vous pouvez le faire. Par exemple, je veux combiner les prénoms et noms des enfants (fname et lname) dans un nom complet. Nous pouvons le faire en ajoutant une ligne après la ligne « k1.lname, » comme ceci :

```
k1.fname || " " || k1.lname
as FullName,
```

Les caractères « || » sont des caractères de concaténation et nous obtiendrons les deux champs avec un espace entre eux dans un champ nommé « FullName ». N'oubliez pas la virgule à la fin. Une fois que votre requête vous convient, cliquez sur le bouton Enregistrer pour enregistrer la requête ; vous devrez indiquer son nom.

Ensuite, cliquez sur Requête dans l'arborescence et faites un clic droit sur la requête que vous venez de créer. Sélectionnez « Nouveau rapport à partir de la requête ». Le concepteur de requêtes disparaît, remplacé par le concepteur de rapport.

Sur la gauche se trouve la fenêtre des propriétés pour un champ donné ou l'ensemble du rapport. Sur la droite se trouve le concepteur de rapport lui-même. Remarquez que cela ressemble à une feuille de calcul. Chaque ligne est considérée comme un « bandeau » et contient des informations pour cette ligne de rapport. Dans cet exemple, nous avons quatre lignes : deux lignes d'en-tête, une ligne de détail et une ligne de bas de page. Vous pouvez ajouter ou supprimer des lignes à volonté. Cette méthode n'est pas aussi souple que celle

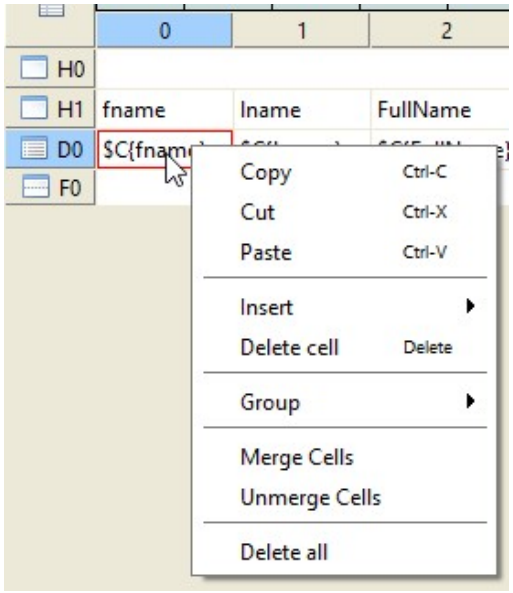


Report Title

fname	lname	FullName	WorshipNote	Usher	Other	Lector	Description	CountAs	Acolyte
Michael			1	0	0	0		1	0
Michael			1	0	0	0		1	1
Michael			1	0	0	0		1	0
Michael			1	0	0	0		1	1
Michael			1	0	0	0		1	0
Michael			1	0	0	0		1	0
Michael			1	0	0	0		2	0
Michael			1	0	0	0		2	0

d'autres concepteurs de rapports, mais fournit néanmoins un rapport très agréable et propre.

Les deux rangées d'en-tête contiennent le titre et les en-têtes de colonnes du rapport. La ligne de détail contient chaque champ pour lequel nous ferons un rapport et il reste le pied de page. Jetons un coup d'œil à la présentation par défaut du rapport. Cliquez sur le bouton en haut de la barre marquée « Version Html » pour voir le rapport. (J'ai flouté le nom de famille des enfants, ce n'est pas un problème dans le générateur.)



Pour un rapport presque sans effort, c'est vraiment pas mal. Mais améliorons-le un peu. Créons un groupe qui place toutes les données concernant un enfant sous le nom de l'enfant.

Faites un clic droit sur la première colonne de la ligne de données. Sélectionnez Groupe, puis Ajouter.

Vous obtiendrez une nouvelle fenêtre pour indiquer sur quels champs vous voulez créer le groupe. Dans ce cas, je sélectionne FullName puis je clique sur le bouton OK. Maintenant, nous avons un regroupement. Nous pouvons également nous débarrasser des trois champs (fname, lname et FullName) dans la section Détails, puisque nous afficherons le nom dans le bandeau de groupe. Un simple clic droit sur ces champs et un clic sur « Supprimer cellule » suffit. Maintenant, vous pouvez redimensionner les trois cellules vides sur la gauche pour rendre l'espace moins évident.

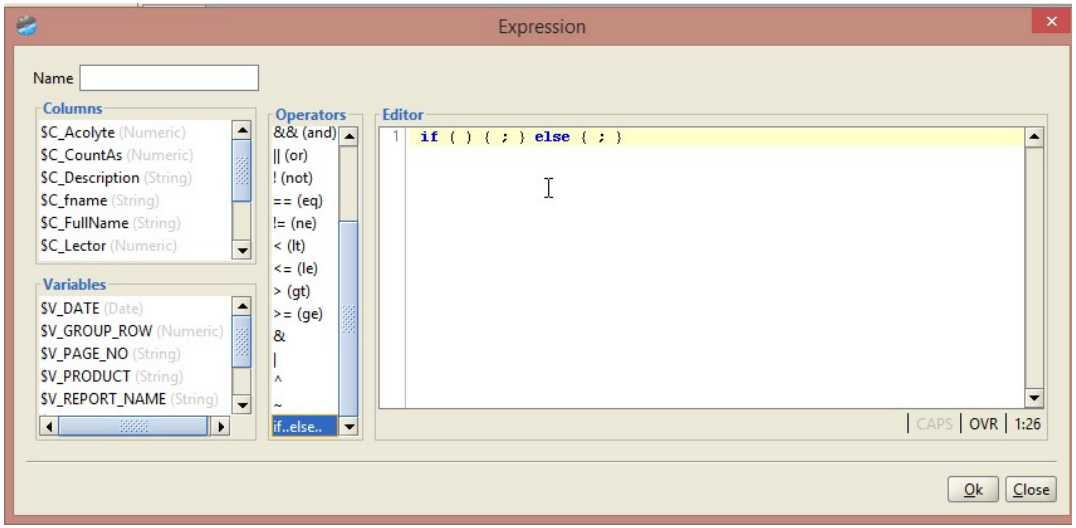
		0	0	1	0	praise song	1	0
Garrett							
		1	0	0	0		1	1
		1	0	0	0		1	0
		1	0	0	0		1	0
		1	0	0	0		1	0
		0	0	0	0		1	1
Trevor							
		1	0	0	0		1	0
		1	0	0	0		1	1
		1	0	0	0		1	0
		0	0	0	0		1	1
		0	0	0	0		1	1
Zachary							

Jetez un coup d'œil au nouvel aspect du rapport et vous verrez que l'information pour chaque enfant est joliment regroupée.

C'est plus agréable, mais maintenant faisons quelque chose d'amusant. Tous les 1 et 0 signifient oui et non. Puisque c'est plutôt ennuyeux pour un rapport, nous allons ajouter une instruction conditionnelle avancée pour chacun de ces champs afin d'afficher une case ; elle sera cochée Oui (ou 1) et vide pour Non (ou 0). C'est vraiment facile à faire et on pourrait croire que vous avez passé des jours à créer votre rapport. En utilisant la police Wingdings de Windows, les deux caractères dont nous avons besoin sont 0x6F (0168) pour une case vide et 0xFE (0254) pour une case cochée.

Avant de continuer, la seule chose que Windows fait mieux que Linux (selon moi) est la possibilité d'utiliser Alt+pavé numérique pour saisir des caractères spéciaux. Linux ne le permet pas. Il y a un contournement avec Ctrl+Maj+U puis la valeur Unicode pour le caractère souhaité. Toutefois, cela ne fonctionne pas sur toutes les machines. La meilleure façon que j'ai trouvé pour faire cela sous Linux est d'ouvrir la table des caractères, utiliser la fonction de recherche pour trouver le caractère Unicode que vous voulez, puis de double-cliquer sur le caractère pour le copier dans la zone « Caractères à copier : », puis cliquer sur « Copier » et, enfin, le coller dans votre document. Les caractères Unicode correspondants sont 2610 (case vide) et 2611 (case cochée) en utilisant la police WingDings 2. Je suis sûr qu'il y a beaucoup d'autres façons plus faciles de le faire, mais je manque de temps. (Assurez-vous que « Commun » est sélectionné dans la liste Script.)

Nous allons commencer par le champ NoteCulte ; sur la ligne Détails, faites un clic-droit sur le champ concerné. Dans ce cas, il est marqué $C{NoteCulte}. Choisissez Insérer, puis Expression. Une autre chose merveilleuse que nous fournit NextReports est la possibilité de faire à peu près



tout avec très peu de saisie. Regardez au centre de la fenêtre Opérateurs. Double-cliquez sur la sélection « if..else.. », et il la saisira comme modèle dans l'éditeur pour vous éviter les erreurs.

Maintenant, nous voulons mettre le champ NoteCulte dans les parenthèses de l'éditeur. Il suffit de cliquer entre les deux parenthèses pour placer le curseur, puis de double-cliquer sur le champ que vous voulez insérer. BAM ! Il est rempli automatiquement. Maintenant, cliquez après le nom du champ dans l'éditeur, puis double-cliquez sur l'opérateur « == (eq) ». Puis ajoutez un « 1 » pour qu'on obtienne ceci :

```
if ( $C_WorshipNote == 1 ) {
; } else { ; }
```

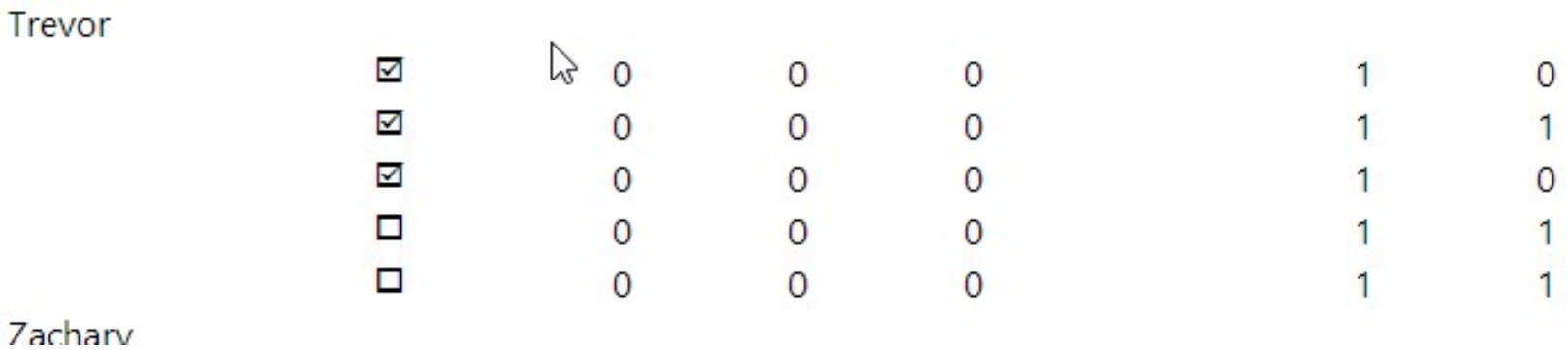
Nous avons presque terminé notre expression. La première série d'accolades définit ce qu'il faut faire si l'expression est vraie et la seconde indique quoi faire si c'est faux. Dans ce cas, nous allons utiliser le CharMap (dans Windows, Linux en a un aussi, par exemple gucharmap si vous utilisez Gnome) pour copier les caractères spéciaux dans l'éditeur ou bien, sous Windows, vous pouvez maintenir la touche Alt et appuyez sur 0168 pour la case vide et sur 0254 pour la case cochée. Maintenant, notre expression est (au moins sous Windows) :

```
 if ( $C_WorshipNote == 1 ) {
"þ"; } else { "o"; }
```

Nommez l'expression (j'ai utilisé NotesR) et enregistrez-la. Sous les propriétés de ce champ-là, sélectionnez la police (ici j'ai utilisé WingDings) et voici à quoi cela va ressembler.

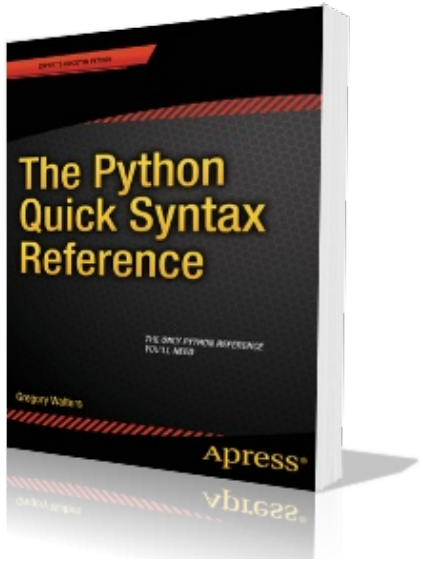
Voici nos jolies petites cases. Faire de même pour les autres champs est tout aussi simple.

Il ne m'a fallu qu'environ 3 heures de prise en main avec cet outil pour arriver à ce stade, voire un peu plus loin. Je peux vraiment dire que j'ai encore beaucoup de choses à apprendre, mais ce sera pour un autre jour. Vous pouvez utiliser des modèles pour colorer votre rapport, vous pouvez ajouter des images, et bien plus encore.



La prochaine fois, je parlerai de la façon dont nous pourrions améliorer l'intégration de ces rapports dans un programme Python. Jusque-là, **amusez-vous bien** avec ce formidable logiciel GRATUIT.

Greg Walters est propriétaire de RainyDay Solutions LLC, une société de consultants à Aurora au Colorado, et programme depuis 1972. Il aime faire la cuisine, marcher, la musique et passer du temps avec sa famille. Son site web est www.thedesignatedgeek.net.



Avant de commencer l'article python de ce mois-ci, permettez-moi une courte page d'auto-publicité. Fin décembre et début janvier, mon premier livre sur Python a été publié aux éditions Apress. Intitulé *The Python Quick Syntax Reference*, il est disponible dans un certain nombre d'endroits. Vous pouvez le trouver sur les sites Apress (http://www.apress.com/9781430264781), Springer.com (http://www.springer.com/computer/book/978-1-4302-6478-1) et Amazon (http://www.amazon.com/The-Python-Quick-Syntax-Reference/dp/1430264780), ainsi que d'autres. Il s'agit, comme le titre l'indique, d'un ouvrage de référence sur la syntaxe qui aidera ceux d'entre nous qui programment aussi dans d'autres langages que Python à se rappeler le fonctionnement d'une certaine commande et les prérequis pour cette commande. S'il vous plaît, aidez un ancien programmeur à gagner sa vie en achetant le livre, si vous le pouvez.

Passons maintenant à des choses plus importantes et passionnantes.

Alors que je travaillais sur mon dernier livre pour Apress, j'ai redécouvert une commande SQL dont je n'avais pas parlé lorsque nous travaillions avec des bases de données SQL, il y a longtemps, et j'ai pensé que je partagerais l'information avec vous. Il s'agit de la commande CREATE TABLE AS SELECT, qui nous permet d'exécuter une requête sur une table (ou sur des tables jointes) et de créer une autre table à la volée. La syntaxe générale est :

```
CREATE TABLE [IF NOT EXISTS]
{Nouveau nom de la table} AS
SELECT {requête}
```

La partie entre crochets (IF NOT EXISTS) est totalement optionnelle et créera la table seulement si elle n'existe pas déjà. Les parties entre accolades, par contre, ne le sont pas. La première est le nouveau nom de la table et la seconde est la requête que vous souhaitez utiliser pour extraire des données et créer la nouvelle table.

Supposons que nous ayons une base de données contenant plusieurs tables. L'une des tables est nommée « étude » et contient les données de réception de colis. Il y a six champs, présentés ci-dessous.

L'un des ensembles de données que nous devrons produire à partir de ces données brutes est un regroupement du nombre de colis et le nombre de jours au cours de l'étude pendant lesquels ces nombres de colis sont arrivés, en supposant que les jours sont les jours de semaine (lundi au vendredi) et que le jour n'est pas un jour férié, car pendant les vacances il y a moins de colis que d'habitude. Notre requête est indiquée ci-dessous.

Elle nous fournit des données qui ressemblent à ceci (page suivante) :

```
SELECT pkgs, Count(DOW) as NombreDeJoursDeSemaine FROM etude
 WHERE (Holiday <> 1)
    AND DayName in ("lundi","mardi","mercredi","jeudi","vendredi")
 GROUP BY pkgs
```

```
pkID - Integer, Primary Key, AutoIncrement
DOM - Integer - Jour du mois (1-31)
DOW - Integer - Jour de la semaine (1-7 (dimanche = 1, lundi = 2, etc.))
pkgs - Integer - Nombre de colis reçus ce jour
DayName - TEXT - "dimanche","lundi", etc
Holiday - Integer 0 ou 1 (Est-ce que ce jour est considéré comme congé ou pas) 1 pour oui
```

```
colis       NbDeJoursDeSemaine
31                1
32                2
33                1
...
48                3
```

Ainsi, les données montrent que lors de l'étude de 65 jours, un seul jour de la semaine avait 31 paquets, mais 3 jours de la semaine avaient 48 paquets et ainsi de suite. Des requêtes similaires pourraient être créées pour couvrir les vacances et les week-ends.

On reçoit simplement un ensemble de données retourné par la requête, mais nous pourrions avoir envie de faire une analyse plus approfondie des données, donc nous voulons mettre les données résultant de la requête dans une table. C'est pourquoi nous aimerions créer une table à partir de la requête. Ainsi, dans l'exemple suivant, ci-dessus à droite, nous créons une table nommée « JoursDeSemaine » en utilisant la même requête que nous avons montrée ci-dessus.

Maintenant, quand nous aurons besoin de cet ensemble de données des jours de semaine, nous pourrons simplement exécuter une requête sur la table « JoursDeSemaine ».

Une fois que nous savons ce que nous voulons, et que nous avons testé la requête, nous pouvons commencer notre code. En supposant que nous avons déjà la table « étude » créée et remplie, nous pouvons utiliser Python pour créer notre nouvelle table dans la base de données principale. Pour votre information, j'utilise la bibliothèque APSW SQLite pour faire le travail de base de données.

```
CREATE TABLE IF NOT EXISTS JoursDeSemaine AS
  SELECT pkgs, Count(DOW) as NbDeJoursDeSemaine FROM etude
  WHERE (Holiday <> 1)
     AND DayName in ("lundi","mardi","mercredi","jeudi","vendredi")
  GROUP BY pkgs
```

Nous devons, bien sûr, ouvrir une connexion (à droite) et créer un curseur pour la base de données SQLite. Nous avons vu ceci dans de nombreux articles précédents.

```
def OpenDB():
    global connection
    global curseur
    connection = apsw.Connection("labpackagestudy.db3")
    curseur = connection.cursor()
```

Maintenant, nous devons créer la routine qui crée la table avec l'ensemble de données retourné par la requête, ci-dessous, puis la modifier et exécuter quelques calculs.

Comme vous pouvez le voir, nous créons un deuxième curseur pour ne pas risquer que le premier curseur contienne des données que nous devons conserver. Nous l'utiliserons dans la dernière partie du code. Nous supprimons la table si elle existe et exécutons notre requête sur la table « étude ».

Maintenant, nous créons trois colonnes de plus (ci-dessous) dans la table des jours de semaine, nommées « probabilité », « inférieur » et « supérieur ». Nous faisons cela en utilisant la commande SQL « ALTER TABLE ».

```
addcolquery = 'ALTER TABLE JoursDeSemaine ADD COLUMN probability REAL'
cursor.execute(addcolquery)
addcolquery = 'ALTER TABLE JoursDeSemaine ADD COLUMN lower REAL'
cursor.execute(addcolquery)
addcolquery = 'ALTER TABLE JoursDeSemaine ADD COLUMN upper REAL'
cursor.execute(addcolquery)
```

```
def TraiterJoursDeSemaine():
  # on cree un second curseur pour mettre a jour la nouvelle table
    curseur2 = connection.cursor()
    q1 = "DROP TABLE IF EXISTS JoursDeSemaine"
    curseur.execute(q1)
    query = '''CREATE TABLE IF NOT EXISTS JoursDeSemaine AS SELECT pkgs,
    Count(DOW) as NombreDeJoursDeSemaine FROM etude WHERE (Holiday <> 1)
    AND DayName in
    ("lundi","mardi","mercredi","jeudi","vendredi")
    GROUP BY pkgs'''
    curseur.execute(query)
```

L'étape suivante (en haut à droite) sera d'additionner les données dans le champ NombreDeJoursDeSemaine.

Il n'y a qu'un seul enregistrement retourné, mais nous faisons quand même une boucle for. Rappelez-vous de ce qui précède que le champ « NombreDeJoursDeSemaine » contient le nombre de jours au cours de l'étude où un nombre déterminé de paquets est arrivé. Cela nous donne une valeur qui contient la somme de toutes les entrées de « NombreDeJoursDeSemaine ». Juste pour que vous ayez une référence pendant que nous progressons : le nombre que j'ai obtenu avec mes données factices est 44.

```
 upquery = "SELECT * FROM
JoursDeSemaine"

c1 = cursor.execute(upquery)
```

Ici, nous avons fait une requête « SELECT all », aussi chaque enregistrement de la table de données est dans le curseur « c1 ». Nous allons parcourir chaque ligne de l'ensemble de données, en extrayant les données pkgs (row[0]) et NombreDeJoursDeSemaine (row[1]) dans des variables.

```
LastUpper = .0
for row in c1:
  cod = row[1]
  pkg = row[0]
```

Maintenant, nous allons créer une probabilité pour chaque compte quotidien de paquets dans la base de données et calculer une valeur supérieure et inférieure qui seront utilisées dans un autre processus plus tard. Notez que nous vérifions pour voir si la variable LastUpper contient « .0 ». Si c'est le cas, nous la réglons à la valeur de probabilité, sinon nous la réglons à la plus faible valeur plus la valeur de probabilité.

Enfin, nous utilisons l'instruction SQL « UPDATE » pour mettre les nouvelles valeurs calculées dans la base de données.

Nous nous retrouvons avec un nombre de paquet (pkgs), le décompte du nombre de jours où ce nombre de paquets est arrivé, une probabilité que cela se produise dans l'ensemble de l'étude (31 colis sur une journée sur un total de 44 (jours de semaine dans cette étude de plus de 60 jours), auront une probabilité de 0,02).

Si l'on additionne toutes les valeurs de probabilité de la table, on devrait trouver 1.

Les valeurs supérieures et inférieures reflètent alors un nombre compris entre 0 et 1 qui mime la possibilité qu'un nombre aléatoire soit dans cette plage et qui va nous donner un nombre aléatoire de paquets. Ce nombre peut alors être utilisé pour une analyse statistique de ces données. Un exemple du « monde réel normal » serait de prévoir le nombre de voitures qui arrivent à un centre de lavage auto sur la base d'observations effectuées sur le terrain. Si vous voulez mieux comprendre, vous pouvez consulter http://www.algebra.com/algebra/homework/Probability-and-statistics/Probability-and-statistics.faq.question.309110.html pour voir un exemple de cela. Tout ce que nous avons fait est de le générer (le plus dur) facilement avec Python.

```
sumquery = "SELECT Sum(NombreDeJoursDeSemaine)
as Sm FROM JoursDeSemaine"
      tmp = curseur.execute(sumquery)
      for t in tmp:
      DaySum = t[0]
```

```
       prob = cod / float(DaySum)
       if LastUpper != .0:
          lower = LastUpper
          LastUpper = (lower + prob)
       else:
          lower = .0
          LastUpper = prob
```

```
      nquery = 'UPDATE weekdays SET probability = %f, \
           lower = %f, upper = %f WHERE pkgs = %d' \
           % (prob,lower,LastUpper,pkg)
      u = cursor2.execute(nquery) #
#====================================
#     End of TraiterJoursDeSemaine
#====================================
```

Le code pour les deux routines que nous avons présentées cette fois-ci est ici :
http://pastebin.com/kMc9EXes
et en français :
http://pastebin.com/7EF7epVG

À la prochaine fois.

Greg Walters est propriétaire de Rainy-Day Solutions LLC, une société de consultants à Aurora au Colorado, et programme depuis 1972. Il aime faire la cuisine, marcher, la musique et passer du temps avec sa famille. Son site web est www.thedesignatedgeek.net.

Ce mois-ci, j'ai pensé créer une routine qui fabrique une clé de licence à partir d'une adresse de courriel. Nous avons tous une raison pour créer une clé de licence, et si vous avez besoin d'avoir rapidement quelques routines vite écrites, vous pouvez utiliser ceci. Rappelez-vous, Python est un langage de script, donc la source est toujours lisible. Il y a des façons de contourner cela ; nous les aborderons dans un autre article. Jetons un coup d'œil à la logique « brute » sous-jacente, avant de nous plonger réellement dans le code.

Tout d'abord, nous allons demander une adresse de courriel, puis la diviser en deux parties, la partie locale (avant le caractère « @ ») et le nom de domaine (après le caractère « @ »). Il existe des règles très précises pour valider une adresse de courriel, et cela devient vite très compliqué. Nous nous contenterons de quelques-unes de ces règles et uniquement sur la partie locale. Vous pouvez chercher l'ensemble des règles sur le web. Dans notre code, nous allons seulement regarder :

- minuscules
- majuscules
- nombres compris entre 0 et 9
- les caractères spéciaux (!#$%&'*+-/=?^_`{|}~.)
- les points sont autorisés, mais ne peuvent pas être répétés côte à côte (…, etc.)

Une fois l'adresse validée, nous allons créer un « caractère somme de contrôle » qui est basé sur la valeur ASCII de chaque caractère de l'adresse complète, puis le diviser par le nombre de caractères de l'adresse. Par exemple, prenons l'adresse factice pierredupont@quelquepart.fr. En parcourant l'adresse, nous pouvons obtenir la valeur ASCII de chaque caractère en utilisant la fonction ord(). En additionnant chacune des valeurs ASCII, on obtient une somme de 2048, que l'on divise par la longueur de l'adresse (27) ; et nous obtenons 75. Nous utilisons la division entière ici, de sorte que notre résultat soit un entier.

Maintenant que nous avons notre somme de contrôle, on en soustrait 68 (ascii 'D') pour créer un décalage. Nous utilisons ce décalage pour encoder chaque caractère de l'adresse. Pour rendre les choses un peu plus difficiles à décoder, nous mettons la longueur (avec décalage) comme caractère en deuxième position et le caractère somme de contrôle en quatrième position.

```
localvalid1 = "abcdefghijklmnopqrstuvwxyz"
localvalid2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890"
localvalid3 = "!#$%&'*+-/=?^_`{|}~."
decalage = 0
```

Donc, pour l'adresse pierredupont@quelquepart.fr nous obtenons cette clé de licence :

```
w_pKlyylk|wvu{Gx|lsx|lwhy{5my
```

Commençons à écrire le code. Puisque c'est le 53e article de la série, je vais commencer à être un peu moins explicite à partir de maintenant.

Tout d'abord les importations.

```
import sys
```

Maintenant (ci-dessus), nous allons créer une chaîne qui inclura tous nos caractères « autorisés » pour la fonction AdresseValide. Je l'ai découpée en 3 parties pour qu'elle s'intègre parfaitement dans le magazine. Nous les combinons dans la routine AdresseValide. Nous réglons également une variable globale « decalage » à 0. Ce sera la valeur que nous ajouterons (plus tard) à chaque caractère lorsque nous créerons la chaîne codée.

Maintenant, voici notre première fonction. C'est (ci-dessous) la routine AdresseValide. Essentiellement, nous passons l'adresse dans la variable s, et un drapeau optionnel de débogage. Nous utilisons le drapeau de débo-

```
def AdresseValide(s,debug=0):
    adresse = s
    pos = adresse.rfind("@")
    local = adresse[:pos]
    domaine = adresse[pos+1:]
    if debug == 1:
          print local
          print domaine
    valide = False
    localvalid = localvalid1 + localvalid2 + localvalid3
```

gage, comme nous l'avons fait dans le passé, pour fournir des instructions d'affichage, afin de voir comment les choses se passent. En général, nous passerons la valeur 1 comme second paramètre si nous voulons afficher la progression.

D'abord, nous affectons l'adresse reçue à la variable « adresse » et cherchons le caractère « @ » qui sépare la partie locale du domaine. Puis, nous affectons la partie locale de l'adresse à (je pense que c'est approprié) « local », et la partie de domaine à « domaine ». Nous réglons ensuite le drapeau booléen « valide » à False et enfin créons la chaîne « localvalid » avec les 3 chaînes plus courtes dont nous avons parlé plus haut.

Ensuite (en haut à droite) nous comparons tout simplement chaque caractère dans la partie locale de l'adresse à la liste de caractères autorisés à l'aide du mot-clé « in ». Si n'importe quel caractère échoue au test, nous sortons de la boucle, en réglons l'option « valide » à False.

Enfin, nous cherchons s'il y a des points qui se suivent. Nous utilisons la routine string.find qui trouvera tout ce qui ressemble à « .. » ou « … » et ainsi de suite. Étant un programmeur paresseux, j'ai utilisé un seul contrôle de « double point » qui fonctionne pour tout le reste.

```
    r = adresse.find("..")
if r > -1:
    valide = False
```

La dernière chose que fait la routine est de retourner la valeur de l'indicateur « valide ».

```
valide return
```

La routine suivante (en bas à droite) est la routine de CheckSum qui est assez courte. Nous parcourons chaque caractère de l'adresse et créons la somme des valeurs ASCII de chacun en s'aidant de la fonction ord intégrée qui convertit en nombres. Comme je l'ai dit plus tôt, nous divisons cette somme par la longueur de l'adresse. Nous retournons la somme de contrôle et le caractère correspondant.

Maintenant, la routine EncodeCle. Elle paraît simple, mais elle nécessite une certaine concentration donc faites bien attention ! La variable decalage est mise à l'état « global », pour qu'on puisse la modifier dans la fonction et l'utiliser ensuite dans d'autres fonctions. Nous réglons ensuite la variable decalage à la somme de contrôle moins 68. Pour l'exemple présenté au

```
# Verifie la partie locale
for compteur in range(0,len(local)):
      if local[compteur] in localvalid:
          if debug == 1:
             print local[compteur],ord(local[compteur]),"True"
          valide = True
      else:
          if debug == 1:
             print local[compteur],ord(local[compteur]),"False"
          valide = False
          break
```

```
def CheckSum(s,debug = 0):
    somme = 0
    adresse = s.upper()
    for compteur in range(0,len(adresse)):
        if debug == 1:
            print adresse[compteur],ord(adresse[compteur])
        somme += ord(adresse[compteur])
    cs = somme/len(adresse)
    if debug == 1:
        print('somme = %d' % somme)
        print('ChkSum = %d' % cs)
        print('ChkSum = %s' % chr(cs))
```

début de l'article, cela ferait 75-68 donc 7. Nous modifions ensuite chaque caractère de l'adresse en ajoutant le décalage à sa valeur ascii. Pour le « p » de « pierredupont », cela fait 112 + 7 soit 119 ce qui équivaut à « w ». En utilisant la variable « compteur », nous construisons la chaîne « NouvelleAdresse » caractère par caractère. Remarquez dans le code que nous allons de 0 à la longueur de l'adresse, donc le caractère 0 est « p », le caractère 1 est « i » et ainsi de suite. Maintenant vient la partie qui pourrait en perdre quelques-uns parmi vous. Lorsque compteur vaut 1 (« i »), nous insérons le caractère correspondant à la longueur de l'adresse + 68 puis le caractère « décalé », ce qui fait pour notre exemple w_p. La prochaine fois que nous passerons dans la boucle, compteur sera égal à 2, mais nous avons déjà 3 caractères dans l'adresse. C'est là que nous voulons insérer le caractère somme de contrôle (« K ») puis le troisième caractère « décalé ». De là, nous ajoutons simplement chaque caractère « dé-

calé » à la chaîne, et lorsque la boucle est terminée, nous retournons la clé (en haut à droite).

La routine DecodeCle (en bas à droite) renverse simplement le processus utilisé dans la routine EncodeCle. Une chose à remarquer ici, c'est que dans la première déclaration « if debug » de cette fonction, j'ai utilisé « != 0 » plutôt que « == 1 », tout simplement pour vous rappeler que les deux sont interchangeables.

La fonction FaisLe (ci-dessous) demande une adresse de courrier électronique en utilisant « raw_input », puis appelle les fonctions afin de créer la clé de licence.

Enfin, nous appelons la routine FaisLe.

```
if __name__ == "__main__":

    FaisLe()
```

Bon, bien sûr le résultat n'est pas super-crypté et, si quelqu'un voulait y passer pas mal de temps, il pourrait comprendre assez facilement comment nous avons créé la clé. Cependant, cela devrait vous donner un bon point de départ pour que vous puissiez simplement modifier le code pour le rendre beaucoup plus difficile à casser. Vous pourriez, par exemple, utiliser un nombre aléatoire plutôt que le « D » (68). Si vous faites cela, indiquez une « graine » (« seed ») dans le code pour qu'il génère toujours le même nombre aléatoire. Vous pouvez aussi aller un peu plus loin et placer la valeur de decalage quelque part dans la clé de licence, par exemple le dernier caractère, pour pouvoir l'utiliser comme decalage de décryptage.

Comme toujours, la source complète est disponible à http://pastebin.com/ipFm77XJ. En attendant la prochaine fois, **amusez-vous bien.**

```
def FaisLe():
    adresse = raw_input("Merci de saisir une adresse de courriel -> ")
    estOK = AdresseValide(adresse,0)
    if estOK == True:
            csum,csumchr = CheckSum(adresse)
            ke = EncodeCle(adresse,csum,0)
            print("Cle de licence     = %s" % ke)
            print("Adresse originale = %s" % DecodeCle(ke,0))
```

Greg Walters est propriétaire de RainyDay Solutions LLC, une société de consultants à Aurora au Colorado, et programme depuis 1972. Il aime faire la cuisine, marcher, la musique et passer du temps avec sa famille. Son site web est www.thedesignatedgeek.net.

```
def EncodeCle(s, csum, debug = 0):
    global decalage
    adresse = s
    decalage = csum - 68
    if debug == 1:
        print("decalage is %d" % decalage)
    NouvelleAdresse = ""
    for compteur in range(0,len(adresse)):
        car = ord(adresse[compteur]) + decalage
        if compteur == 1:
            NouvelleAdresse = NouvelleAdresse +
(chr(len(adresse)+68)) + chr(car)
        elif compteur == 2:
            NouvelleAdresse = NouvelleAdresse + chr(csum) +
chr(car)
        else:
            NouvelleAdresse = NouvelleAdresse + chr(car)
    if debug == 1:
```

```
def DecodeCle(s,debug = 0):
    global decalage
    adr = ""
    for compteur in range(0,len(s)):
        if debug != 0:
            print compteur,s[compteur],ord(s[compteur])-
decalage,chr(ord(s[compteur])-decalage)
        if compteur == 0:
            adr = adr + chr(ord(s[compteur])-decalage)
        elif compteur == 1:
            adrlen = ord(s[compteur])-decalage
        elif compteur == 3:
            csumchr=s[compteur]
        else:
            adr = adr + chr(ord(s[compteur])-decalage)
        if debug == 1:
```