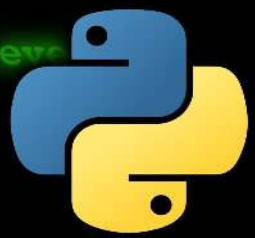
Full Circle

LA RIVISTA INDIPENDENTE PER LA COMUNITÀ LINUX UBUNTU

EDIZIONE SPECIALE SERIE PROGRAMMAZIONE

EDIZIONE SPECIALE
SERIE PROGRAMMAZIONE



python™

PROGRAMMARE IN PYTHON
VOLUME 9

Cos'è Full Circle

Full Circle è una rivista gratuita e indipendente, dedicata alla famiglia Ubuntu dei sistemi operativi Linux. Ogni mese pubblica utili articoli tecnici e articoli inviati dai lettori.

Full Circle ha anche un podcast di supporto, il Full Circle Podcast, con gli stessi argomenti della rivista e altre interessanti notizie.

Si prega di notare che questa edizione speciale viene fornita senza alcuna garanzia: né chi ha contribuito né la rivista Full Circle hanno alcuna responsabilità circa perdite di dati o danni che possano derivare ai computer o alle apparecchiature dei lettori dall'applicazione di quanto pubblicato.

Full Circle

LA RIVISTA INDIPENDENTE PER LA COMUNITÀ LINUX UBUNTU

Ecco a voi un altro 'Speciale monotematico'

Come richiesto dai nostri lettori, stiamo assemblando in edizioni dedicate alcuni degli articoli pubblicati in serie.

Per ora, questa è un'onesta ristampa della serie '**Programmare in Python' parti 49-53**, pubblicata nei numeri 79-84, permettendo all'impareggiabile professore di Python Gregg Walters il nr.83 come tempo libero per buona condotta.

Vi preghiamo di tenere conto della data di pubblicazione: le versioni attuali di hardware e software potrebbero essere diverse rispetto ad allora. Controllate il vostro hardware e il vostro software prima di provare quanto descritto nelle guide di queste edizioni speciali. Potreste avere versioni più recenti del software installato o disponibile nei repository delle vostre distribuzioni.

Buon divertimento!

Come contattarci

Sito web:
http://www.fullcirclemagazine.org/

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

IRC:#fullcirclemagazine su chat.freenode.net

Gruppo editoriale

Capo redattore: Ronnie Tucker (aka: RonnieTucker)
ronnie@fullcirclemagazine.org

Webmaster: Rob Kerfia (aka: admin / linuxgeekery-admin@fullcirclemagazine.org

Modifiche e Correzioni
Mike Kennedy, Lucas Westermann, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred

Si ringrazia la Canonical e i tanti gruppi di traduzione nel mondo.

Programmare In Python - Parte 49

Questa settimana, mentre stavo lavorando, una persona molto saggia dal nome Michael W. ha suggerito che dovrei considerare cosa accade con i numeri a virgola mobile e le uguaglianze.

Prendiamo per esempio un semplice calcolo: 1.1 + 2.2

La risposta, dite, è 3.3! Ogni bambino che va scuola e che si è occupato di frazioni lo sa. Bene, chiedetelo al vostro computer. Se avviate la Shell interattiva di Python e al prompt digitate

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

potreste essere sorpresi che la Shell risponda

```
"False"
```

COSA?!!?!?

Ora, confusi, digitate al prompt:

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

E la Shell vi restituisce:

```
3.3000000000000003
```

Guardate increduli lo schermo e pensate subito "Devo aver digitato qualcosa di sbagliato". Poi realizzate che non lo avete fatto. Quindi digitate:

```
>>>2.2+3.3

5.5
```

Ora siete ancora più confusi e pensate fra voi "Ok. C'è un bug o un qualche tipo di Easter egg malato". No, non è né un bug né un Easter egg. È vero. Sebbene lo sapessi molto tempo fa, era però scivolato nelle ragnatele nascoste degli oscuri recessi della mia vecchia mente, quindi ho dovuto riportarlo a galla. Quello a cui stiamo assistendo è la gioia dei numeri binari a virgola mobile.

Tuti noi sappiamo che 1/3 è uguale a 0.3333333333333333... per sempre, ma prendete, per esempio, la frazione 1/10 . Tutti sanno che la frazione 1/10 è uguale a 0.1, giusto? Se usate la finestra interattiva voi potete vederlo:

```
>>>1/10

0
```

Oh, giusto. Dobbiamo avere almeno uno dei valori a virgola mobile per mostrare ogni punto decimale poiché un intero fratto un intero restituisce un intero. Quindi proviamo di nuovo.

```
>>>1/10.0

0.1
```

Ok. Siamo tornati alla realtà. No, non proprio. Python sta semplicemente mostrandoci una versione arrotondata della risposta. Quindi, come vediamo la risposta "vera"? Possiamo usare la libreria decimal per vedere cosa accade realmente.

```
>>> from decimal import *

>>> Decimal(1/10.0)

Decimal('0.1000000000000000055511
51231257827021181583404541015625')
```

WOW. Allora proviamo la nostra formula iniziale e vediamo cosa mostrerebbe:

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

Decimal('3.3000000000000002664535259100375697016716003417968
75')
```

Sembra proprio essere sempre peggio. Allora cosa sta realmente accadendo?

Questo è chiamato Errore di Rappresentazione ed esiste in quasi tutti i moderni linguaggi di programmazione (Python, C, C++, Java e perfino Fortran e altri) e su quasi tutti i moderni computer. Questo perché queste macchine usano il calcolo per la virgola mobile IEEE-754 che (su molte macchine e piattoforme OS) si associa a un numero a doppia precisione IEEE-754. Tale numero a doppia precisione ha una precisione di 53 bit. Quindi, il nostro 0.1, quando rappresentato in tale modo, cambia in:

```
0.0001100110011001100110011001100110
011001100110011001100110011010
```

Ciò è molto vicino a 1, ma non abbastanza da evitare problemi.

Quindi, cosa facciamo al riguardo? Bene, la risposta veloce è che possiamo probabilmente conviverci per il 90% delle cose che dobbiamo fare fuori da

qui nel mondo reale, usando il metodo round() (arrotonda - N.d.T.). Sebbene dobbiamo decidere il numero di cifre decimali che ci occorrono per arrivare alla precisione voluta, per la maggior parte sarà una soluzione accettabile.

Onestamente non ricordo se abbiamo parlato del metodo round(), quindi lo tratterò brevemente. La sintassi è molto semplice:

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

dove v è il valore che vogliamo arrotondare e d è il numero (massimo) di decimali che vogliamo avere dopo il punto. Secondo la documentazione di Python, "I valori vengono arrotondati al più vicino multiplo di 10 della potenza n, a cui viene attribuito segno negativo: se due multipli sono ugualmente vicini, l'arrotondamento viene fatto partendo da 0". Detto tutto ciò, se il numero è 1.4144 e lo arrotondiamo a 3 cifre decimali, il valore restituito sarà 1.414. Se il numero è 1.4145, sarebbe restituito come 1.415.

Per esempio, usiamo il valore di pi (Pi greco - N.d.T.) che viene fornito dalla libreria math (dovete importare la libreria math prima di poterlo fare, tra l'altro).

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

Ora, se vogliamo arrotondare tale valore solo al quinto posto decimale, dovremmo usare:

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

Questo è il valore "standard" di pi che quasi tutti conosciamo bene. Ottimo. Comunque, se impostiamo il numero di posti di decimali a 4, guardate cosa succede.

```
>>> round(math.pi,4)
3.1416
```

Tutto ciò sembra buono finché non lo eseguiamo su un valore come 2.675 e tentiamo di arrotondarlo a 2 cifre decimali. L'ipotesi (poiché è esattamente a metà tra 2.67 e 2.68) è che il valore restituito sarà 2.68. Proviamo.

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

Questo potrebbe causare un problema. Si torna alla questione iniziale di cui abbiamo parlato. L'effettiva conversione in un numero binario a virgola mobile di 53 bit. Il numero diventa:

```
2.674999999999999822365316059974953522189331054687 5
```

che viene quindi arrotondato a 2.67.

La morale qui è che quando si cerca di confrontare numeri a virgola mobile, dobbiamo essere consapevoli che alcune cose semplicemente non si convertono bene.

Ci vediamo la prossima volta!

Greg Walters è il proprietario della RainyDay Solutions, LLC, una società di consulenza in Aurora, Colorado e programma dal 1972. Ama cucinare, fare escursioni, la musica e passare il tempo con la sua famiglia. Il suo sito web è www.thedesignatedgeek.net.

Il Podcast Ubuntu copre tutte le ultime notizie e novità che si presentano agli utenti di Ubuntu Linux e ai fan del software libero in generale. La rassegna è rivolta tanto all'utente più fresco quanto al programmatore più esperto. Le nostre discussioni riguardano lo sviluppo di Ubuntu ma non sono eccessivamente tecniche. Siamo abbastanza fortunati da avere qualche gradito ospite nello show a passarci novità di prima mano sugli ultimi eccitanti sviluppi a cui stanno lavorando, in modo comprensibile a tutti! Parliamo inoltre della comunità Ubuntu e di cosa le interessa.

Lo show è offerto dai membri della comunità Ubuntu Linux del Regno Unito. Ed essendo coperto dal Codice di condotta di Ubuntu è adatto a tutti.

Lo show è trasmesso live ogni due settimane il martedì sera (ora inglese) ed è disponibile per il download il giorno seguente.

podcast.ubuntu-uk.org

Programmare in Python - Parte 50

Questo mese, ho pensato di parlarvi di un paio di funzioni meno conosciute, maketrans e translate.

Inizieremo con il metodo translate. Il metodo translate restituisce una copia di una stringa con tutti i caratteri, della tavola di traduzione, sostituiti, o con i caratteri, del parametro opzionale replaced, rimossi dalla stringa. Ecco qui la sintassi.

```
s =
str.translate(table[,deletechar
acters])
```

Prima di entrare nella parte table del metodo, diamo uno sguardo alla parte delete. Diciamo che abbiamo la stringa "The time has come". Vogliamo cancellare tutte le vocali (per qualche strana ragione) da tale stringa. Possiamo codificarlo in questo modo:

```
astr = "The time has come"

astr.translate(None, 'aeiou')
```

restituirà:

```
"Th tm hs cm"
```

Notare che abbiamo incluso None come tavola di traduzione. Sebbene questa parte sia fantastica, c'è di meglio. C'è una funzione chiamata maketrans. Prende come parametri una stringa in input e una in output e restituisce una tavola che è usata come primo parametro nel metodo translate. Qui (in alto a destra) c'è un esempio molto semplice.

Restituisce:

```
"Th2 t3m2 h1s c4m2"
```

Diamo un'occhiata a cosa fa. Assegniamo a intable una stringa di vocali, come fatto prima. A outtable vengono assegnati i numeri 1,2,3,4,5 come stringa. Quando facciamo la chiamata a maketrans, la nostra effettiva trantable è come la seguente (mostrata sotto. La "\x" significa che è un carattere esadecimale):

```
intable = 'aeiou'
outtable = '12345'
trantable = maketrans(intable,outtable)
astr = "The time has come"
astr.translate(trantable)
```

Se la guardiamo attentamente, vedremo che le vocali in minuscolo sono sostituite con i numeri che abbiamo specificato:

```
1bcd2fgh3jklmn4pqrst5vwxyz
```

Se guardiamo ancor più da vicino, vedremo che in realtà ci sono 256 voci che iniziano con "\x00" e finiscono con "\xff". Quindi la tavola contiene la totalità dei 256 insiemi di caratteri ascii. Quindi, quando il metodo translate prende la tavola, itera (o passa attraverso) ciascun carattere, prendendo il valore del carattere in esadecimale e poi trova tale valore nella tavola di traduzione e lo sostituisce nella stringa di output. La rappresentazione in esadecimale della nostra originale stringa astr ('The time has come') è mostrata di sotto.

Così ora dovrebbe avere senso.

Ora lo scopo dell'intera faccenda. Pensiamo ai tempi della scuola quando studiavamo Giulio Cesare. Ogni volta che voleva mandare un messaggio confidenziale, avrebbe utilizzato un

```
\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'
```

cifrario che spostava tutte le lettere dell'alfabeto di tre caratteri a destra. Quindi, usando l'odierno alfabeto inglese:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz
```

diventa:

```
DEFGHIJKLMNOPQRSTUVWXYZabcdef
ghijklmnopqrstuvwxyzABC
```

Sebbene ciò sembra molto semplice con gli standard odierni, quando ero uno scolaretto lo usavamo sempre per mandare messaggi agli altri. Usavamo un indice diverso nella stringa per avviare la stringa di cifratura, la logica dietro era la stessa.

Nessuno sa quanto ciò fosse realmente efficace per il buon vecchio Giulio. Si potrebbe pensare che se qualcuno intercettava il messaggio, avrebbe pensato che era in qualche lingua straniera. Possiamo solo ipotizzare.

Possiamo usare facilmente i metodi translate e maketrans come funzioni di supporto per permetterci di divertirci con esse. Diciamo che vogliamo fare un semplice programma che ci permette di inserire una stringa di "testo semplice" e di ottenere una stringa crittografata usando lo stesso esatto metodo usato da Cesare. Per amore della semplicità, useremo solo caratteri maiuscoli (mostrati in alto a destra).

Ogni cosa nel suddetto codice è praticamente quello che abbiamo trattato sopra o nei recenti articoli su Python, ma lo riesaminerò velocemente.

Le prime due righe sono le stringhe in ingresso e in uscita. Abbiamo solo spostato i caratteri e girato intorno per creare la stringa in uscita. Le successive due righe creano una tavola per cifrare e una per decifrare. La riga 5 chiede all'utente di inserire una stringa da codificare. Codifichiamo poi tale stringa (EncString) nella riga successiva. Per decifrarla, usiamo semplicemente il metodo translate sulla stringa codificata per ottenere nuovamente il testo semplice. Infine stampiamo entrambe le stringhe. Ecco il prodotto del programma.

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

È proprio come tornare a scuola. Ma arricchiamolo un po' per renderlo leggermente più usabile. Il codice è pressoché lo stesso con qualche eccezione. Primo, abbiamo aggiunto uno spazio alla fine della stringa intab e fra la "Z" e la "A" nella stringa outtab. Ciò aiuta a mantenere le parole reali dall'essere troppo ovvie nella stringa codificata. Il cambiamento successivo è dove chiediamo se l'utente vuole codificare o decodificare la stringa. Infine aggiungiamo un'istruzione if per controllare quello che stampiamo (mostrato in basso a destra).

```
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)
```

L'output del programma è:

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

E per provare la parte della decodifica:

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

Bene, spero che inizierete ad avere delle idee su come usare queste nuove informazioni nel vostro codice.

```
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)
```

Programmare in Python - Parte 51

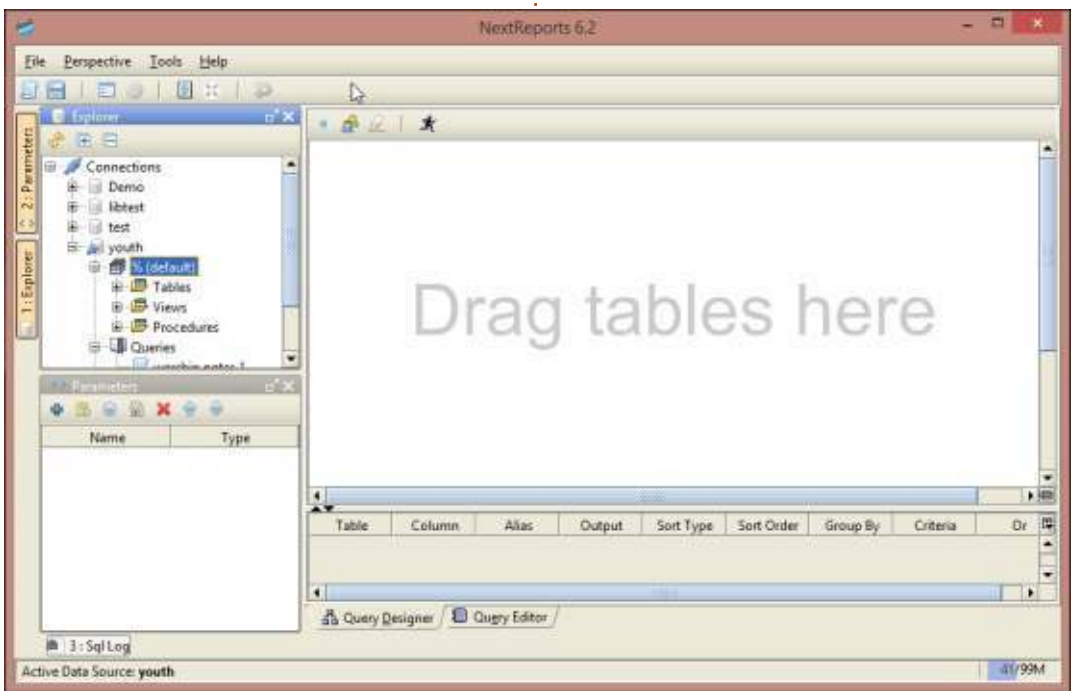
Scritto da Greg Walters

Questo mese discuterò di un prodotto che è nuovo per me, ma apparentemente esiste da un certo numero di anni. Si chiama NextReports della Advantage Software Factory e potete ottenerlo gratuitamente presso http://www.next-reports.com. E non è tutto, è anche a codice aperto e funziona sotto Windows e Linux!

Prima che inizi a raccontarvi del prodotto, lasciatemi sul mio podio a sfogarmi per un momento o due. Per un lungo periodo ho lavorato su database e report. Una delle cose con cui ho avuto problemi è che mentre erano disponibili soluzioni gratuite per database, come SQLite o MySql, c'era veramente poca disponibilità di strumenti di reportistica gratuiti. La maggior parte delle volte qualsiasi report doveva essere fatto con strumenti molto costosi, o lo sviluppatore doveva costruirselo con le sue mani. Alcuni strumenti erano disponibili, ma erano carenti. Quando si arrivava ai grafici, non c'era altra scelta che usare strumenti costosi. Credetemi, ho cercato per anni buoni strumenti gratuiti di reportistica e non sono sicuro di come possa aver trascurato questo pacchetto per così tanti anni (la versione 2.1 è stata rilasciata nel marzo 2009 e adesso sono arrivati alla versione 6.3). Ma ora che l'ho trovato sono assolutamente gasato.

Ora che sono sceso dal podio, posso cominciare a cantare le sue lodi. E' un insieme di tre parti: uno strumento per progettare i report, un motore di report e un report server. Tutto ciò su cui ho avuto la possibilità di giocare è stato lo strumento per progettare i report, ma se tale strumento è un qualche indicatore della potenza, facilità e flessibilità del resto della suite, questa cosa è vincente.

Questo mese ci concentreremo sullo strumento per la progettazione. A causa di alcuni vincoli sul mio tempo, sto lavorando su una macchina Windows, ma ogni cosa che mostro che può essere fatta con Linux (quindi per favore perdonatemi in anticipo).



Una delle prima cose che dovreste sapere è che supporta database quali Oracle, MySql, SQLite, MSSQL e altri. Ogni cosa è basata su query ed è veramente una bella cosa che solo le query di tipo SELECT siano permesse. Questo significa che niente nel database può essere cambiato per sbaglio. Potrete inserire le vostre query o utilizzare uno strumento di progettazione visuale.

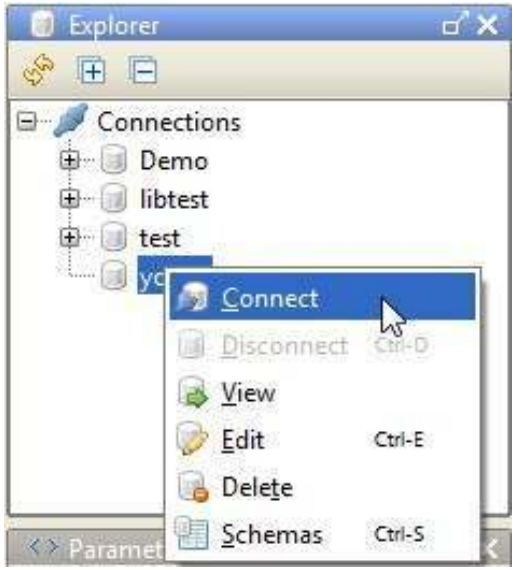
La schermata mostra quanto è bella la UI. Le cose sono abbastanza intuitive e non richiederà molto l'essere produttivi. Diamo una occhiata ai passi per andare avanti.

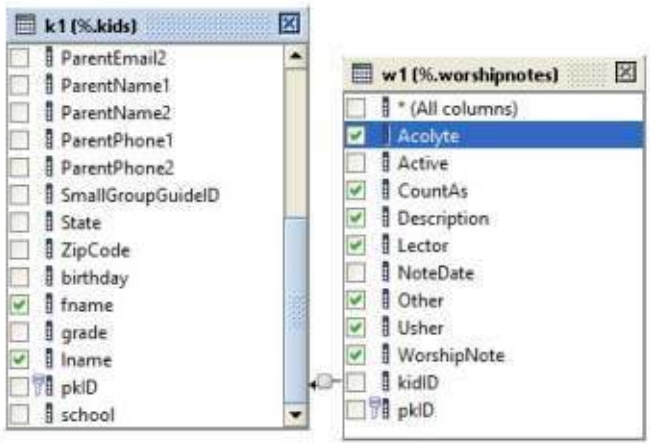
Cominciamo con File > New > Data Source. Diamo quindi alla nostra sorgente dati il nome che vogliamo.

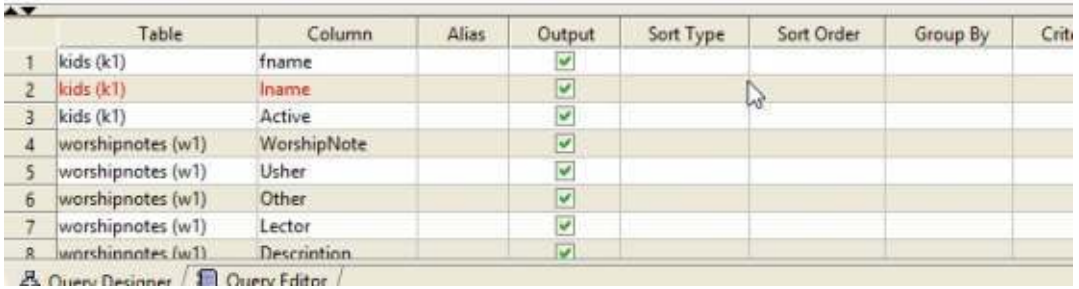
Ora diciamo a NextReports il tipo di database, dal menù a tendina denominato "Type". Possiamo saltare la sezione Driver e andare alla sezione URL. Qui metteremo il percorso al database. Se stiamo usando, per esempio, un database SQLite, questo sarà riempito per noi: "jdbc:sqlite:<percorso-dbfile>". Sostituiamo <percorso-dbfile> con il

indice ^

percorso del database. Altri tipi di database hanno tipologie di informazioni simili, già popolate per aiutarci. Successivamente premiamo il pulsante "Test" per essere sicuri di poterci connettere. Se tutto va bene, premiamo "Save" e vedremo aggiungere la nostra connessione all'albero delle connessioni. La prossima cosa di cui abbiamo bisogno è di fare la connessione al database appena aggiunto. Ora facciamo clic con il tasto destro sul database e poi su 'Connect'.







Una volta connessi, vedremo che abbiamo quattro possibili scelte. La "%" sono la tabelle del database. Le tre successive servono per creare nuove query, report e grafici. Abbastanza semplice. Premiamo sul segno "+" a sinistra di "%", che ci aprirà la vista della tabelle del database. Ora avremo nell'albero: Tables, Views e Procedures. Premiamo di nuovo il segno "+" vicino a "Tables". Questo ci mostrerà tutte le nostre tabelle. Adesso, se vogliamo usare lo strumento per la progettazione visuale delle query, basta trascinare nella finestra di progettazione a destra la tabella/e con cui vogliamo avere a che fare.

Una volta che abbiamo qui tutte le tabelle, possiamo iniziare a fare delle connessioni tra le tabelle.

Nell'esempio qui, ho due tabelle, una con le informazioni relative ai ragazzi di una classe di conferma e l'altra con delle voci per le note di merito prese. La tabella con le note di merito non ha il nome del ragazzo dentro di essa, ma solo un id che punta alla tabella delle informazioni dei ragazzi. Ho trascinato e rilasciato per fare in modo di collegare il campo kidID e la pkID della tabella dei ragazzi. Quindi ho selezionato ogni campo che volevo nell'insieme risultante. In questo caso, il nome e cognome del ragazzo e un flag attivo (o non cancellabile) nella tabella dei ragazzi e molti campi della tabella delle note. La griglia sotto mostra

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		☑				

indice ^

ciascun campo dal quale proviene la tabella e altre informazioni.

Come potete vedere, possiamo impostare criteri come "Active = 1", scegliere di visualizzare o meno un campo e configurare tipo e grado di ordinamento. Una volta soddisfatti di ciò, possiamo fare clic sulla tabella sotto e vedere la nostra effettiva query SQL.

```
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
```

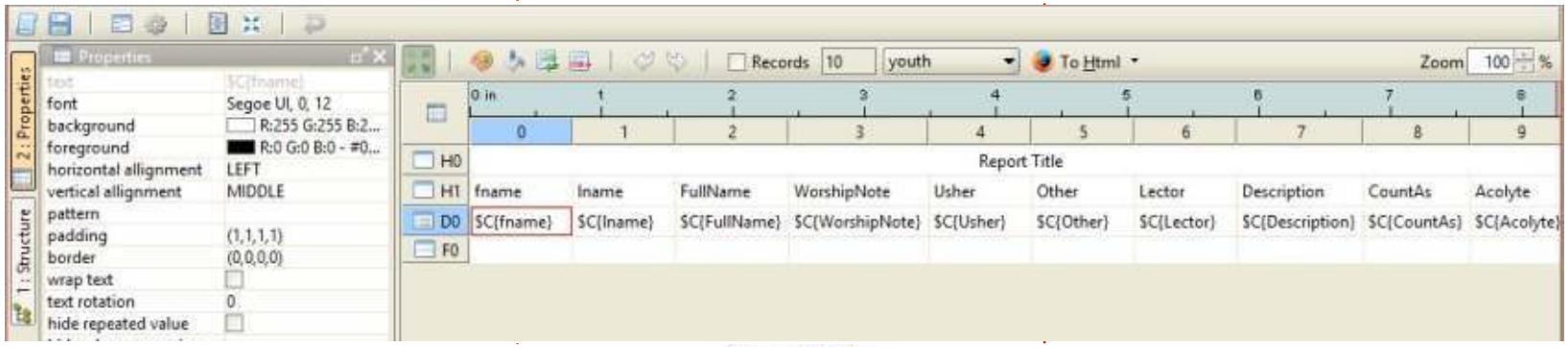
Per provare la query basta premere sull'icona "uomo che corre", ottenendo (sperando di aver fatto tutto correttamente) il risultato della query in una griglia al di sotto dell'editor. Se volete potete aggiungere manualmente alcune righe. Per esempio, voglio unire il nome e il cognome del ragazzo (fname e lname) in un nome completo. Possiamo farlo mettendo una linea dopo "k1.lname" come questa:

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

I caratteri "||" sono i caratteri per la concatenazione così avremo i due campi, con uno spazio in mezzo, in un campo denominato "FullName". Non dimenticate la virgola alla fine. Una volta che abbiamo la nostra query fatta nel modo in cui vogliamo, premiamo sul bottone "Save" per salvarla. Ci verrà chiesto in quale modo vogliamo chiamarla.

Successivamente, premiamo l'elemento Query nell'albero e poi facciamo clic con il pulsante destro sulla query appena creata. Selezioniamo "New Report From Query..". Lo strumento di progettazione delle query va via e viene sostituito dallo strumento di progettazione dei report.

Sulla sinistra c'è la finestra con le proprietà di ogni singolo campo o dell'intero report. Sulla destra c'è lo strumento di progettazione dei report stesso. Notare che è simile a un foglio elettronico. Ciascuna riga è considerata una "banda" e contiene informazioni per quella riga di report. Nel caso di questo esempio, abbiamo quattro righe, due righe di intestazione, una riga di dettaglio e una riga di piè di pagina. Si possono aggiungere o togliere righe a proprio piacimento. Questo modo non è molto a "schema libero" come altri strumenti, ma crea un report molto

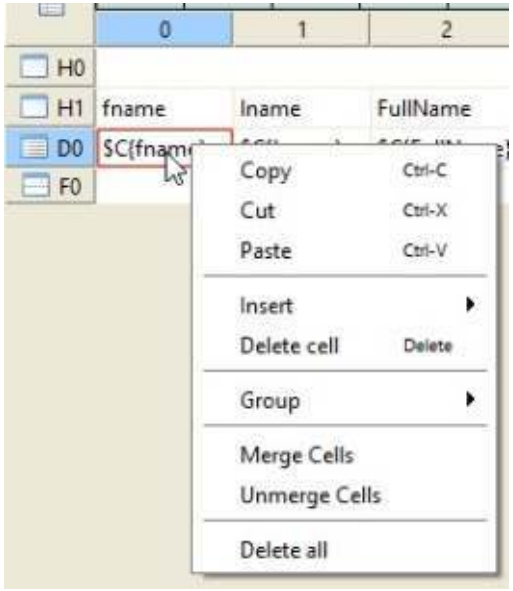


Report Title

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

bello e pulito.

Le due righe di intestazioni contengono il titolo del nostro report e le intestazioni delle colonne. La riga di dettaglio ha ciascun campo che stiamo riportando e sulla riga del piè di pagina vi è il piè di pagina. Diamo una occhiata a come si presenta il report predefinito. Premere sul pulsante in cima alla barra degli strumenti segnato "To Html" per vedere il report (ho cancellato i cognomi dei ragazzi, non è un problema del generatore).



Per un report praticamente senza lavoro è veramente bello. Ma rendiamolo un pochino più accattivante. Creiamo un gruppo che mette insieme tutti i dati di ciascun ragazzo sotto il suo nome.

Fare clic con il pulsante destro sulla prima colonna della riga dei dati. Selezionare Group e quindi Add.

Si aprirà una nuova finestra in cui ci verrà chiesto per quali campi vogliamo creare il gruppo. In questo caso ho selezionato FullName e ho quindi premuto il pulsante OK. Ora abbiamo un'interruzione di raggruppamento. Possiamo anche liberarci dei tre campi (fname, lname e Fullname) nella sezione di dettaglio, poiché visualizzeremo il nome nella banda del gruppo. Premiamo semplicemente il pulsante destro su essi e selezioniamo "Delete Cell". Ora possiamo ridimensionare le tre celle vuote sulla sinistra in modo da rendere meno evidente la differenza.

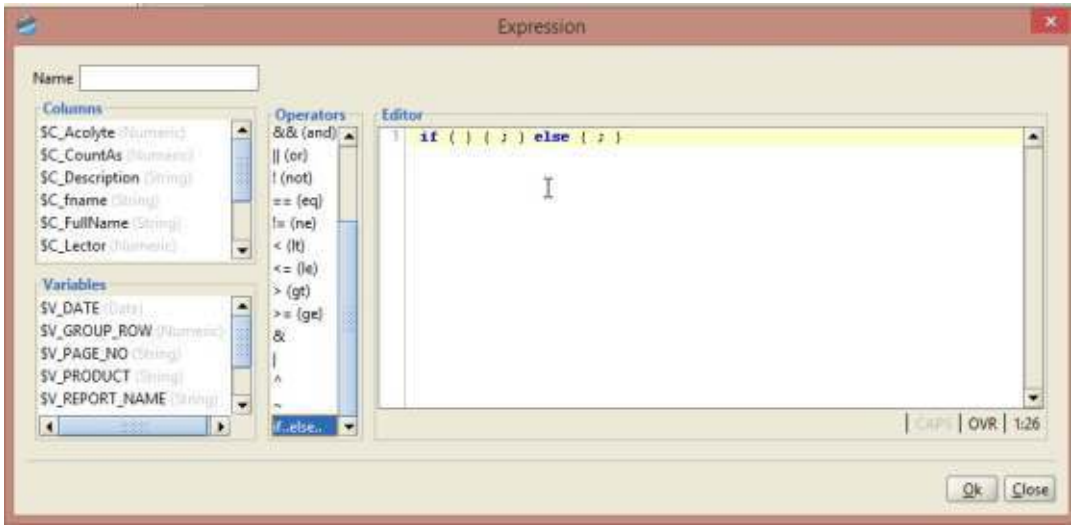
	0	0	1	0	praise song	1	0
Garrett							
	1	0	0	0		1	1
	1	0	0	0		1	0
	1	0	0	0		1	0
	1	0	0	0		1	0
	0	0	0	0		1	1
Trevor							
	1	0	0	0		1	0
	1	0	0	0		1	1
	1	0	0	0		1	0
	0	0	0	0		1	1
	0	0	0	0		1	1
Zachary							

Dando una rapida occhiata a come appare il report ora, si vede che le informazioni per ciascun ragazzo sono tutte gradevolmente raggruppate insieme.

Questo è più bello, ma lasciatemi fare qualcosa di divertente. Tutte gli 1 e gli O stanno per si e no. Ciò è abbastanza noioso per un report, così lasciatemi aggiungere un'ulteriore istruzione avanzata per ciascuno di questi campi che mostrerà un riquadro con una spunta per Sì (o 1) e un riquadro vuoto per No (o 0). E' veramente facile da fare, ma fa sembrare il report come se ci aveste speso dei giorni interi sopra. Usando il carattere WingDings, i due caratteri che vogliamo sono 0x6F(0168) per il riquadro vuoto e 0xFE(0254) per il riquadro selezionato.

Prima di andare avanti, una cosa che Windows fa meglio di Linux (che ho trovato) è l'uso di ALT+NumPad per l'inserimento di caratteri speciali. Linux non permette questa funzionalità. C'è un soluzione simile che utilizza CTRL+SHIFT+U e il codice unicode del carattere voluto. Comunque non funziona su tutte le macchine. Il modo più semplice che ho trovato su Linux è di aprire la mappa dei caratteri, usare la funzione cerca per trovare il carattere unicode voluto, fare doppio clic su di esso per copiarlo nel riquadro "Testo da copiare:", quindi premere "Copia" per incollarlo nel documento. I caratteri unicode per le caselle sono 2610 (riquadro vuoto) e 2611 (riquadro selezionato), usando il carattere WingDings 2. Sono sicuro che ci sono molti altri modi più facili per affrontare questa situazione, ma mi manca il tempo (assicurarsi di avere Common selezionato nell'elenco degli script).

Cominceremo con il campo WorshipNotes. Nella riga di dettaglio, facciamo clic con il tasto destro sul campo in cui vogliamo operare. In questo caso è marcato $C{WorshipNote}. Scegliamo Insert e poi Expression. Un'altra cosa meravigliosa che NextReports ci da è



l'abilità di fare quasi ogni cosa con il minimo di digitazione possibile. Guardate al centro della finestra dove dice "Operators". Fate un doppio clic su "if..else" e riempirà per voi l'editor con ciò, come un modello, in modo tale che voi non possiate commettere errori.

Ora, vogliamo mettere il campo WorshipNotes nelle parentesi dell'editor. Basta fare clic tra le due parentesi per posizionare il cursore e poi fare un doppio clic sul campo da posizionarci. BAM! È riempito per noi. Ora facciamo clic dopo il campo nome nell'editor e poi doppio click sull'operatore "==(eq)". Quindi aggiungiamo un "1", in modo tale che nella riga dell'editor si legga

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

Abbiamo quasi finito con la nostra espressione. Il primo insieme di parentesi tonde definisce che cosa fare quando l'espressione è vera e il secondo invece che cosa fare quando l'espressione è falsa. In questo caso, useremo CharMap (in windows, anche Linux ne ha una, per esempio gucharmap se state usando Gnome) per copiare i caratteri nel nostro editor di stringhe. O, sotto windows, possiamo tener premuto il tasto {Alt} e premere 0168 per il riquadro vuoto e 0254 per il riquadro selezionato. Così ora la nostra espressione è (almeno in windows):

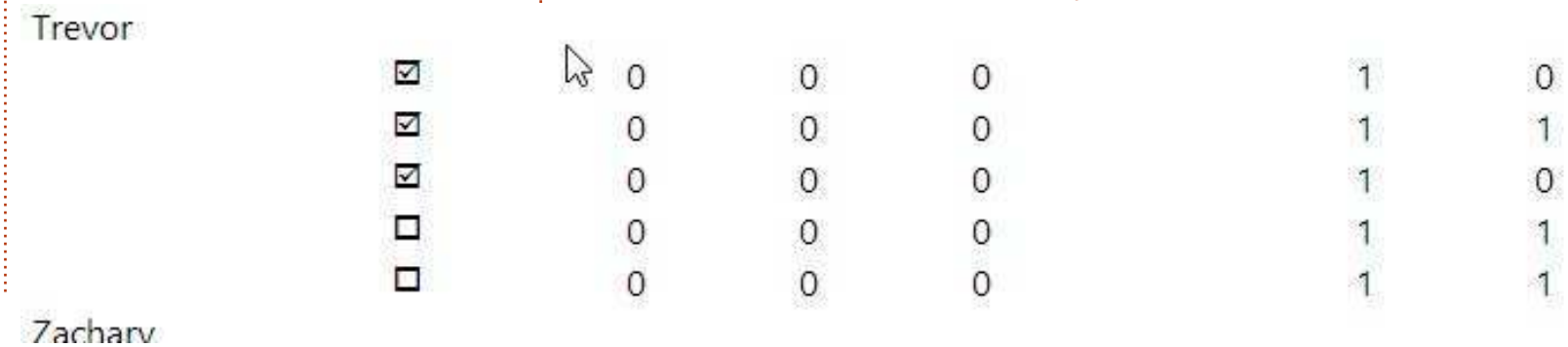
```
if ( $C_WorshipNote == 1 ) {
"þ"; } else { "o"; }
```

Diamo un nome all'espressione (io ho usato WNotes) e salviamola. Sotto Proprietà, per questo campo, selezioniamo il carattere (WingDings è ciò che ho usato qui) e assomiglierà a questo.

Ci sono i nostri bei piccoli riquadri. Fare lo stesso per gli altri campi è semplicissimo.

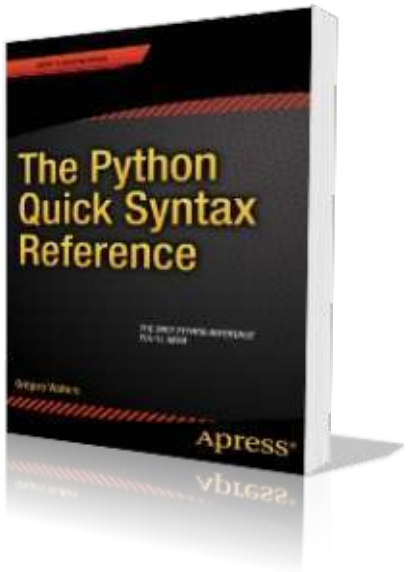
Mi ha richiesto solo tre ore circa giocare con il pacchetto per arrivare a questo punto e altro ancora. Posso sinceramente dire che ho molto altro da imparare ma questo è per un altro giorno. Si possono usare dei modelli per colorare i propri report, si possono aggiungere immagine e molto altro.

La prossima volta vi parlerò di come si possono inserire questi report in un programma Python. Fino ad allora divertitevi a giocare con questo meraviglioso programma libero.



Greg Walters è il proprietario della RainyDay Solutions, LLC, una società di consulenza in Aurora, Colorado e programma dal 1972. Ama cucinare, fare escursioni, ascoltare musica e passare il tempo con la sua famiglia. Il suo sito web è www.thedesignatedgeek.net.

Programmare In Python - Parte 52



Prima di iniziare con l'effettivo argomento di Python di questo mese, permettetemi di spostare l'attenzione per un minuto su un altro argomento. Alla fine di dicembre inizio gennaio è stato pubblicato da Apress il mio primo libro su Python. Si intitola "The Python Quick Syntax Reference" ed è disponibile in vari posti. Potete trovarlo presso il sito di Apress (http://www.apress.com/9781430264781), di Springer.com (http://www.springer.com/computer/book/978-1-4302-6478-1) e di Amazon (http://www.amazon.com/The-Python-Quick-Syntax-Reference/dp/1430264780) così come su altri siti. È, come il titolo suggerisce, un riferimento alla sintassi che aiuterà quelli di noi che programmano in altri linguaggi, così come in Python, a ricordare come funzionano certi comandi e i loro requisiti necessari. Per favore aiutate un povero e vecchio programmatore a vivere comprando il libro, se potete.

Ora passiamo a cose più grandi e migliori.

Quando stavo lavorando al mio ultimo libro per Apress, ho riscoperto un comando SQL di cui non ho trattato quando stavamo lavorando con i database SQL molto tempo fa, perciò ho pensato di condividere l'informazione con voi. Si tratta del comando CREATE TABLE AS SELECT, che ci permette di estrapolare una query da una tabella (o da tabelle di unione) e di crearne una al volo. La sintassi generale è:

```
CREATE TABLE [IF NOT EXISTS]
{Nuovo nome tabella} AS SELECT
{query}
```

La parte tra parentesi quadre (IF NOT EXISTS) è assolutamente opzionale, creerà la tabella solo se non esiste già. La parte tra parentesi graffe, tuttavia, non lo è. La prima è il nome della nuova tabella e la seconda è la query che si vuole usare per estrarre i dati e creare la nuova tabella.

Si presuppone di avere un database con tabelle multiple al suo interno. Una di queste tabelle è chiamata "study" che contiene i dati ricevuti dalle operazioni. Ci sono sei campi che sono mostrati di seguito.

Uno degli insiemi di dati che avremmo bisogno di produrre da questi dati grezzi è un raggruppamento del totale dei pacchetti e il numero dei giorni, nell'arco dello studio, in cui la quantità di pacchetti è entrata. Supponendo che i giorni sono feriali (dal lunedì al venerdì) e che non siano una festività, poiché le festività hanno un minor numero di pacchetti. La nostra query è mostrata sotto.

Ciò ci fornisce quindi i dati che

```
SELECT pkgs, Count(DOW) as CountOfDOW FROM study
   WHERE (Holiday <> 1)
      AND DayName in ("Lunedì","Martedì","Mercoledì","Giovedì","Venerdì")
   GROUP BY pkgs
```

```
pkID - Integer, Primary Key, AutoIncrement
DOM -  Integer - Giorni del mese (1-31)
DOW -  Integer - Giorni della settimana (1-7 (Lunedì = 1, Martedì = 2, etc))
pkgs - Integer - Numero di pacchetti ricevuti questo giorno
DayName - TEXT - "Lunedì","Martedì", ecc
Holiday - Integer 0 o 1 (Questo giorno è festivo o no?) 1 significa SI
```

dovrebbero somigliare a questo:

```
pkgs
CountOfDow
31                              1
32                              2
33                              1
...
48                              3
```

Quindi i dati ci mostrano che durante lo studio di 65 giorni, solo un giorno della settimana ha avuto 31 pacchetti, ma 3 giorni settimanali ne hanno avuti 48 e così via. Interrogazioni similari possono essere create per includere festività e fine settimana.

Anziché avere i dati semplicemente come un insieme ricevuto dalla query, potremmo voler fare ulteriori analisi sugli stessi, per cui vogliamo porli in una tabella. Nel seguente esempio, mostrato a destra, abbiamo quindi creato una tabella chiamata "weekdays" usando la stessa interrogazione visualizzata precedentemente.

Ora, ogni volta che avremo bisogno dei dati per tale insieme di risultati nei giorni feriali, possiamo semplicemente lanciare l'interrogazione nella tabella weekdays.

Una volta che sappiamo cosa ci serve e abbiamo provato le query, allora si può cominciare col nostro codice. Supponendo di avere già creato e caricato la tabella study, possiamo utilizzare Python per creare poi la nostra tabella nel database principale. Giusto per informazione, sto usando la libreria APSW di SQLite per far funzionare il database.

```
CREATE TABLE IF NOT EXISTS weekdays AS
   SELECT pkgs, Count(DOW) as CountOfDOW FROM study
   WHERE (Holiday <> 1)
      AND DayName in ("Lunedì","Martedì","Mercoledì","Giovedì","Venerdì")
   GROUP BY pkgs
```

Dobbiamo, ovviamente, aprire una connessione (a destra) e creare un cursor al database SQLite. Abbiamo trattato di ciò su degli articoli passati.

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

Ora abbiamo bisogno di creare la routine, mostrata sotto, che creerà effettivamente la tabella con il ritorno del dataset derivato dall'interrogazione, quindi alterarla ed eseguire qualche calcolo.

Come potete vedere, vogliamo creare un secondo cursor, per non rischiare che il primo cursor abbia dati che dobbiamo conservare. Useremo ciò nella parte finale del codice. Eliminiamo poi la tabella, se esiste, ed eseguiamo la query sulla tabella "study".

Ora creeremo altre tre colonne (mostrato sotto) all'interno della tabella weekdays, che chiameremo "probability", "lower" e "upper". Faremo ciò utilizzando il comando SQL "ALTER TABLE".

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

```
def DoWeekDays():
    # Create a second cursor for updating the new table
     cursor2 = connection.cursor()
     q1 = "DROP TABLE IF EXISTS weekdays"
     cursor.execute(q1)
     query = '''CREATE TABLE IF NOT EXISTS weekdays AS SELECT pkgs,
               Count(DOW) as CountOfDOW FROM study WHERE (Holiday <> 1)
               AND DayName in
               ("Lunedì","Martedì","Mercoledì","Giovedì","Venerdì")
               GROUP BY pkgs'''
     cursor.execute(query)
```

Il prossimo passaggio (in alto a destra) sarà sommare i dati nel campo CountOfDOW.

C'è solo un record restituito ma lo faremo comunque con il ciclo for. Ricordando la discussione precedente, il campo "CountOfDow" contiene il numero dei giorni, durante lo studio, in cui la quantità di pacchetti è entrata. Ciò fornisce un valore contenente la somma delle voci in "CountOfDow". Solo così abbiamo un riferimento mentre andiamo avanti, il numero che ho ottenuto da tutti i miei dati fittizi è 44.

```
upquery =  "SELECT * FROM
settimana"

c1 =  cursor.execute (upquery)
```

Qui abbiamo operato un'interrogazione "SELECT all" così qualsiasi record nel database è nel cursor "c1". Esamineremo ogni riga del dataset, trascinando i dati di pkgs (riga[0]) e di CountOfDow dentro le variabili.

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

Ora creeremo nel database una probabilità per ogni conteggio giornaliero di pacchetti e calcoleremo un valore massimo e minimo che verranno usati in un altro processo più avanti. Da notare che abbiamo controllato se la variabile LastUpper contiene "0". Se lo contiene, la impostiamo al valore di probabilità, altrimenti la impostiamo al valore più basso di probabilità.

Finalmente abbiamo usato un'istruzione SQL di aggiornamento per mettere i nuovi valori elaborati nel database.

Quello che abbiamo ottenuto alla fine è un conteggio dei pacchetti (pkgs), un conteggio dei giorni nei quali vi è stato il conteggio dei pacchetti entranti, una probabilità che ciò si verifichi all'interno dell'intero studio (31 pacchetti in un giorno su un totale di 44 - giorni feriali in questo studio di 60 e più giorni - abbiamo un probabilità dello 0.02).

Se aggiungiamo tutti i valori di probabilità alla tabella il valore dovrebbe essere 1.0.

I valori massimo e minimo riflettono quindi un numero in virgola mobile compreso tra 0 e 1 che rispecchierà la possibilità di qualsiasi numero casuale all'interno di questo intervallo che ci fornirà un numero

```
sumquery = "SELECT Sum(CountOfDOW) as Sm FROM weekdays"
tmp = cursor.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 DoWeekDays
#====================================
```

casuale di pacchetti. Questo numero può essere utilizzato per fare analisi statistiche su questi dati. Un esempio tratto dalla realtà di tutti i giorni potrebbe essere la predizione del numero di macchine che potrebbero arrivare all'autolavaggio basandosi sull'osservazione dei dati su campo. Se vuoi capire di più, puoi dare un'occhiata a http://www.algebra.com/algebra/homework/Probability-and-statistics/Probability-and-statistics.faq.question.309110.html per vedere un esempio di ciò. Tutto quello che abbiamo fatto (la parte più difficile) è generato facilmente con Python.

Il codice per le due routines che abbiamo presentato in questa sessione si trova presso: http://pastebin.com/kMc9EXes

Alla prossima.

Greg Walters è il proprietario di RainyDay Solution, LLC, una compagnia di consultazioni in Aurora, Colorado e programma dal 1972. Gli piace cucinare, fare escursioni, la musica e passare il tempo con la propria famiglia. Il suo sito è www.thedesignatedgeek.net.

Programmare In Python - Parte 53

Questo mese ho pensato di creare una routine che crea una chiave di licenza da un indirizzo email. Conosciamo tutti le ragioni per avere una chiave di licenza e se mai fosse necessario avere una sporca e veloce routine per crearla, è possibile utilizzare questa. Ricordate, Python è un linguaggio di scripting, quindi il codice sorgente è sempre leggibile. Ci sono modi per evitarlo; ne discuteremo in un altro articolo. Diamo uno sguardo alla "grossolana" logica dietro il codice prima di tuffarcisi dentro.

Innanzitutto chiederemo un indirizzo email e poi lo spezzeremo in due parti, la parte locale (quella antecedente il carattere "@") e la parte del dominio (quella dopo il carattere "@"). Ci sono regole ben precise per la validità dell'indirizzo email e può diventare molto complicato. Per i nostri scopi, useremo solo alcune delle regole e solo sulla parte locale. Potete fare una ricerca sul web per l'effettivo insieme di regole. Nel nostro codice, verificheremo solo:

- caratteri minuscoli
- caratteri maiuscoli
- numeri tra 0 e 9
- i caratteri speciali (! # $% & '* + - / = ^ _ `? {|} ~).
- saranno consentiti i punti, ma non ripetuti uno accanto all'altro (..., etc)

Una volta che abbiamo convalidato l'email, creeremo quindi un "carattere di checksum", basato sul valore ASCII di ciascun carattere dell'intero indirizzo email per poi dividerlo per il numero totale di caratteri. Ad esempio, usiamo il fittizio indirizzo fredjones@someplace.com. Se lo scorriamo, è possibile ottenere il valore ASCII di ogni carattere utilizzando la funzione ord(). Sommando ciascuno valore ASCII si ottiene 1670, che dividiamo poi per la lunghezza dell'indirizzo email (23) ottenendo 72. Ricordatevi che stiamo usando la divisione intera, quindi il nostro risultato sarà un numero intero.

Ora che abbiamo il nostro valore di checksum, sottraiamoci 68 (ascii 'D'), per creare un offset. Lo useremo per codificare ciascun carattere della email. Giusto per rendere le cose un po' più difficili da decodificare, inseriamo la lunghezza (con offset)

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

come posizione 2 del carattere e il checksum come posizione 4.

Quindi, per l'indirizzo fredjones@someplace.com, otteniamo la chiave di licenza:

```
j [vHihnsriwDwsqitpegi2gsq
```

Iniziamo con il codice. Dal momento che questo è il 53° articolo della serie, non sarò molto dettagliato da qui in avanti.

Prima le nostre importazioni.

```
import sys
```

Ora (come mostrato in alto a destra) creeremo una stringa che includerà tutti i nostri caratteri "ammessi" per la funzione IsValidEmail. L'ho divisa in 3 stringhe in modo che si adatti bene alla rivista. Le combiniamo nella routine IsValidEmail. Impostiamo inoltre la variabile globale 'offset' a 0. Questo sarà il valore da aggiunge (in seguito) a ogni carattere quando creiamo la stringa codificata.

Ora la prima funzione. Questa (in basso) è la routine IsValidEmail. Fondamentalmente passiamo alla variabile s l'indirizzo email e un flag opzionale di debug. Usiamo il flag di debug, come abbiamo fatto in passato, per fornire alcune

```
def IsValidEmail(s,debug=0):
    email = s
    pos = email.rfind("@")
    local = email[:pos]
    domain = email[pos+1:]
    if debug == 1:
        print local
        print domain
    isgood = False
    localvalid = localvalid1 + localvalid2 + localvalid3
```

dichiarazioni di stampa per controllare come stanno andando le cose. Di solito passeremmo semplicemente un 1 come secondo parametro, se volessimo monitorare l'andamento.

Prima passiamo l'indirizzo email convalidato alla variabile 'email' e cerchiamo il carattere '@' che separa la parte email locale da quella di dominio. Assegniamo quindi la parte locale a 'local' (credo sia appropriato) e la parte di dominio a 'domain'. Impostiamo poi il flag booleano 'isgood' a False e infine creiamo la stringa 'localvalid' con le 3 stringhe più corte impostate in precedenza.

Poi (in alto a destra) confrontiamo semplicemente ciascun carattere della porzione locale della email con l'elenco dei caratteri validi, utilizzando la parola chiave in. Se qualsiasi carattere nella porzione locale dell'email fallisce il test, usciamo dal ciclo for, impostando il flag 'isgood' a False.

Infine, cerchiamo qualsiasi gruppo di caratteri punto contigui. Usiamo la funzione string.find, che abbinerà tutto ciò che è '..' o '...' e così via. Essendo un programmatore pigro, ho usato un solo controllo "doppio punto" che funziona anche con più punti.

```
    r = email.find ("..")
    if r> -1:
        isgood = False
```

L'ultima cosa che facciamo nella funzione è restituire il valore del flag 'isgood'.

```
    return isgood
```

La routine successiva (in basso a destra) è CheckSum, che è piuttosto breve. Scorriamo ogni carattere nella email e creiamo una somma parziale del valore ascii di ciascuno, utilizzando la nativa conversione per tipo 'ord'. Come ho detto prima, prendiamo quella somma e la dividiamo per la lunghezza dell'indirizzo email. Restituiamo il valore del checksum e il carattere rappresentato da tale checksum.

Ora la routine EncodeKey. Benché sembri semplice, richiede una certa concentrazione quindi fate attenzione! Impostiamo la variabile Offset allo stato global, in modo da poterla cambiare all'interno della funzione e poterla quindi utilizzare in altre funzioni. Impostiamo quindi la variabile offset con il valore di checksum meno 68. Come nell'esempio presentato all'inizio dell'articolo, sarebbe 72-68, che è uguale a 4. Scorriamo poi ogni

```
    # Check Local Part
    for cntr in range(0,len(local)):
        if local[cntr] in localvalid:
            if debug == 1:
                print local[cntr],ord(local[cntr]),"True"
            isgood = True
        else:
            if debug == 1:
                print local[cntr],ord(local[cntr]),"False"
            isgood = False
            break
```

```
def CheckSum(s,debug = 0):
    sum = 0
    email = s.upper()
    for cntr in range(0,len(email)):
        if debug == 1:
            print email[cntr],ord(email[cntr])
        sum += ord(email[cntr])
    cs = sum/len(email)
    if debug == 1:
        print('Sum = %d' % sum)
        print('ChkSum = %d' % cs)
        print('ChkSum = %s' % chr(cs))
    return cs,chr(cs)
```

carattere dell'indirizzo email aggiungendo l'offset al valore ascii di quel carattere. Per la 'f' in 'fredjones', sarebbe 102 + 4 o 106 che equivale a 'i'. Usando la variabile contatore 'cntr' determiniamo quindi cosa aggiungere alla stringa 'NewEmail', che costruiamo carattere per carattere. Notare che nel codice si passa da 0 alla lunghezza dell'email, quindi il carattere 0 è 'f', il carattere 1 è 'r' e così via. Ora arriva la parte che potrebbe confondere alcuni di voi. Se cntr ha il valore 1 ('r'), inseriamo il carattere per la lunghezza dell'email + 68 e poi il carattere offset, che, usando il nostro esempio, sarebbe IYT. La prossima volta che eseguiamo il ciclo, cntr sarà uguale a 2, ma abbiamo già 3 caratteri nell'email. Qui è dove vogliamo inserire il carattere di checksum ('F') e poi il terzo carattere offset. Da lì, aggiungiamo semplicemente ciascun carattere di offset alla stringa e, quando il ciclo è finito, restituiamo la chiave (in alto a destra).

La funzione DecodeKey (in basso a destra) inverte sostanzialmente il processo che abbiamo usato nella routine EncodeKey. Una cosa che si può notare qui è che nella prima dichiarazione 'if debug' di questa funzione, ho usato '! = 0' piuttosto che '== 1', per ricordare semplicemente che i due possono essere intercambiabili.

La funzione DoIt (sotto) chiede un indirizzo email usando 'raw_input', poi chiama le funzioni al fine di creare la chiave di licenza.

Infine, chiamiamo la routine DoIt.

```
if __name__ == "__main__":
    DoIt ()
```

Ora, ovviamente, il risultato non è super-criptato e se qualcuno impiegasse una discreta quantità di tempo, potrebbe scoprire abbastanza facilmente cosa abbiamo usato per creare la chiave. Tuttavia, ciò dovrebbe darvi abbastanza spunti da poter modificare semplicemente il codice per renderlo molto più difficile da bucare. Si potrebbe, ad esempio, utilizzare un numero casuale, piuttosto che la 'D' (68). Se lo fate, impostate un seme nel codice in modo da generare sempre lo stesso numero casuale. Si potrebbe anche andare un po' più a fondo e mettere il valore di offset da qualche parte nel codice di licenza, magari l'ultimo carattere, in modo da utilizzarlo come offset di decrittazione.

Come sempre, il sorgente completo è disponibile all'indirizzo http://pastebin.com/MH9nVTNK. Fino alla prossima volta, divertitevi.

Greg Walters è il proprietario della RainyDay Solutions, LLC, una società di consulenza in Aurora, Colorado e programma dal 1972. Ama cucinare, fare escursioni, la musica e passare il tempo con la sua famiglia. Il suo sito web è www.thedesignatedgeek.net.

```
def DoIt():
    email = raw_input("Please enter email address -> ")
    isok = IsValidEmail(email,0)
    if isok == True:
        csum,csumchr = CheckSum(email)
        ke = EncodeKey(email,csum,0)
        print("License Key    = %s" % ke)
        print("Original email = %s" % DecodeKey(ke,0))
```

```
def EncodeKey(s, csum, debug = 0):
    global Offset
    email = s
    Offset = csum - 68
    if debug == 1:
        print("Offset is %d" % Offset)
    NewEmail = ""
    for cntr in range(0,len(email)):
        ch = ord(email[cntr]) + Offset
        if cntr == 1:
            NewEmail = NewEmail + (chr(len(email)+68)) +
chr(ch)
        elif cntr == 2:
            NewEmail = NewEmail + chr(csum) + chr(ch)
        else:
            NewEmail = NewEmail + chr(ch)
    if debug == 1:
        print cntr, NewEmail
    return NewEmail
```

```
def DecodeKey(s,debug = 0):
    global Offset
    eml = ""
    for cntr in range(0,len(s)):
        if debug != 0:
            print cntr,s[cntr],ord(s[cntr])-
Offset,chr(ord(s[cntr])-Offset)
        if cntr == 0:
            eml = eml + chr(ord(s[cntr])-Offset)
        elif cntr == 1:
            emllen = ord(s[cntr])-Offset
        elif cntr == 3:
            csumchr=s[cntr]
        else:
            eml = eml + chr(ord(s[cntr])-Offset)
    if debug == 1:
        print eml
    return eml
```