Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

Programozói sorozat – Különkiadás

Programozói sorozat Különkiadás

Programozzunk Pythonban

9. Kötet

A Full Circle magazin különkiadása

Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

Üdvözöllek egy újabb, „egyetlen témáról szóló különkiadásban”

Válaszul az olvasók igényeire, néhány sorozatként megírt cikk tartalmát összegyűjtjük dedikált kiadásokba.

Most ez a **„Programozzunk Pythonban” 49–53. részének** az újabb kiadása (a magazin 79–84. számaiból), semmi extra, csak a tények.

Kérlek, ne feledkezz meg az eredeti kiadási dátumról. A hardver és szoftver jelenlegi verziói eltérhetnek az akkor közöltektől, így ellenőrizd a hardvered és szoftvered verzióit, mielőtt megpróbálod emulálni/utánozni a különkiadásokban lévő ismertetőket. Előfordulhat, hogy a szoftver későbbi verziói vannak meg neked, vagy érhetők el a kiadásod tárolóiban.

Jó szórakozást!

Programozzunk Pythonban 49. rész

A héten egy nagyon bölcs ember – név szerint Michael W. - azt javasolta, hogy nézzek már rá, hogy mi történik a lebegőpontos számokkal és azok egyenlőségével.

Vegyünk például egy egyszerű sima példát: 1.1 + 2.2

Azt gondolnád, hogy a válasz 3.3! Bármely iskolás gyerek, aki látott már tizedes törteket tudná. Nos, mondd a számítógépednek is. Ha elindítod a Python interaktív konzolját (Interactive Shell), és beírod, hogy:

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

akkor nagyon meg fogsz lepődni az eredményen:

```
„False”
```

Micsoda!?

Most összezavarodtál, ugye? Most írd be a következőt:

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

És a konzolon a válasz:

```
3.3000000000000003
```

Bámulsz a képernyőre és azon mélázol hitetlenül, hogy biztosan félreütöttél valamit. Aztán rájössz, hogy mégsem. Úgyhogy folytatod:

```
>>>2.2+3.3

5.5
```

Most még jobban összezavarodsz, és azon gondolkozol, hogy ez most valami szoftver hiba, vagy valami átverés. Nem, ez nem hiba és nem is átverés. Ez így van. Bár ismertem ezt már régről, de mégis az agytekervények szürke bugyrában nagyon rejtve maradt. Úgyhogy elő kellett bányásznom onnan. Mert tulajdonképpen, amit itt látunk az a lebegőpontos számábrázolás nagyszerűsége.

Mindannyian tudjuk, hogy 1/3 egyenlő 0.3333333333333333...-al, ahol végtelenségig írhatod a 3-ast. De vegyük például az 1/10-et. Mindenki tudja, hogy 1/10 egyenlő 0.1-el. Igaz? Ha használod az interaktív konzolt, akkor magad is láthatod:

```
>>>1/10

0
```

Ó, persze! Tartalmaznia kell legalább egy lebegő-pontos számot, hogy tizedes számot kapjunk, elvégre integer eredményt kapunk. Úgyhogy újra próbáljuk:

```
>>>1/10.0

0.1
```

Ok. Akkor kezdünk visszatérni a valósághoz. Vagy mégsem? Python egyszerűen a válasznak egy kerekített eredményét adja vissza. Hogyan láthatjuk tehát a „valódi" választ? Használhatjuk például a decimális könyvtárat, hogy lássuk mi történik.

```
>>> from decimal import *

>>> Decimal(1/10.0)

Decimal('0.1000000000000000055511151231257827021181583404541015625')
```

Hoppá! Akkor próbáljuk az eredeti képletünket, és nézzük, hogy mit mutat:

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

Decimal('3.300000000000002664535259100375697016716003417968675')
```

Úgy tűnik, hogy ez csak egyre rosszabb lesz. Mi történik valójában?

Ezt nevezzük Reprezentációs hibának (Representation Error), ami majdnem minden modern programozási nyelvben megvan (Python, C, C++, Java sőt még Fortran-ban és a többiben is), és majdnem minden modern számítógépen. Ez azért van, mert ezek a gépek az IEEE-754 lebegőpontos számítást használják (amelyek a legtöbb gépen és operációs rendszeren vannak), amely az IEEE-754 dupla-precizíós számábrázolásnak felel meg. Ennek a számábrázolásnak a pontossága 53 bit. Tehát a 0.1 értékünk, amikor 53 bites dupla-precizíós rendszerben ábrázoljuk, akkor

```
0.0001100110011001100110011001100110011001100110011010
```

lesz belőle.

Ez közelít a 0.1-hez, de nincs elég közel, hogy elkerüljük a problémákat.

Mit kezdünk akkor ezzel? Nos a legegyszerűbb válasz, hogy az esetek 90%-ban tudunk ezzel együtt élni, amikor az életben ezt a round() metódust használjuk. Egyrészt el kell döntened, hogy hány tizedes pontosságig számolsz, ami elegendő számodra a megfelelő pontossághoz... de legtöbb esetben ez a kerülő megoldás elfogadható.

Őszintén nem emlékszem, hogy valaha átvettük volna a kerekítés (round) metódusát, ezért most megemlítem. A szintakszis nagyon egyszerű:

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

ahol v az érték, amit kerekíteni akarsz, és a d a szám, ahány tizedes értéket akarsz az egész szám mögött. A Python dokumentáció szerint „az értékek kerekítése a számok 10 hatványának -n számjegyét jelentik; ha két szám hatvány kitevője közel egyenlő, akkor a kerekítés értéke 0". Mindezek után tehát, ha a szám 1.4144 és 3 tizedes hellyel kerekítjük, akkor a kapott érték 1.414. Ha a számunk 1.4145, akkor a kapott érték 1.415.

Például vegyük a Pi értékét a matematikai könyvtárból. (Ehhez a math könyvtárat kell importálnod, mielőtt ezt használnád)

```
>>> math.pi

3.141592653589793
```

Na már most, ha ezt kerekíteni szeretnéd 5 tizedes helyre, akkor így használod:

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

3.14159
```

Ez a „mindennapokban használt" Pi érték, amit mindenki álmából felkeltve is tud. Nagyszerű. Bár ha beállítjuk, hogy a szám tizedes helye legyen 4, nézd mit kapunk:

```
>>> round (math-pi,4)

3.1416
```

Ez mindaddig jól hangzik, míg bele nem futunk egy olyanba, mint 2.675 és megpróbáljuk 2 tizedes pontossággal kerekíteni. A feltételezés (mivel fél úton van 2.67 és 2.68 között), hogy a kapott érték 2.68 lesz. Tégy egy próbát.

```
>>round(2.675,2)

2.67
```

Ez problémát okozhat. Ez visszavezet minket a kezdeti problémánkhoz, amiről eddig beszéltünk. A konkrét 53 bites bináris lebegőpontos szám konvertálásánál, a szám átalakul ezzé:

```
2.674999999999999822365316059
974953522189331054687 5
```

ami viszont kerekítve eredményezi a 2.67-et.

A lényege az egésznek, hogy lebegőpontos számok összehasonlításánál, ne felejtsük el, hogy a kapott érték nem magától értetődő.

Viszlát következő alkalommal!

Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta programozik. Szeret főzni, túrázni, szereti a zenét és idejét a családjával tölteni. Honlapja: www.thedesignatedgeek.net.

ubuntu uk podcast

Az Ubuntu Podcast lefedi a legfrissebb híreket és kiadásokat amik általában érdekelhetik az Ubuntu Linux felhasználókat és a szabadszoftver rajongókat. A műsor felkelti a legújabb felhasználók és a legöregebb fejlesztők érdeklődését is. A beszélgetésekben szó van az Ubuntu fejlesztéséről, de nem túlzottan technikai. Szerencsések vagyunk, hogy gyakran vannak vendégeink, így első kézből értesülünk a legújabb fejlesztésekről, ráadásul olyan módon ahogyan mindenki megérti! Beszélünk továbbá az Ubuntu közösségről is, és a benne zajló dolgokról is.

A műsort a nagy-britanniai Ubuntu közösség tagjai szerkesztik. Mivel az Ubuntu viselkedési kódexnek megfelelően készítik, bárki meghallgathatja.

A műsor minden második hét keddjén élőben hallgatható (brit idő szerint), másnap pedig letölthető.

podcast.ubuntu-uk.org

Ebben a hónapban arra gondoltam, hogy néhány kevésbé ismert függvényről, maketransról és a translate-ről fogok írni.A translate (fordítás) metódussal kezdünk. Ez a karakterláncnak egy olyan másolatát adja vissza, amelyből a cserére kijelölt karaktereken már elvégeztük a műveletet, vagy a deletechars opcionális paraméterként megadottakat eltávolítottuk. Íme a szintaxis:

```
s =
str.translate(table[,delete
characters])
```

Mielőtt a metódus table részéig érnénk, nézzük meg jobban a delete (törlés) részt. Használjuk az alábbi karakterláncot: „The time has come”. Valamilyen furcsa oknál fogja szeretnénk eltávolítani belőle a magánhangzókat. Ezt az alábbi módon tehetjük meg:

```
astr = "The time has come"

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

Az eredmény:

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

Vegyük észre a translate táblában a **None**-t. Ez így már egész jó, de van tovább is. Létezik egy **maketrans** nevű metódus, amely a ki- és bemeneti karakterláncot paraméterként kezeli és visszatérési értéke az átmenetet biztosító tábla. Lássunk erre egy egyszerű példát: jobbra, fent.

Az eredmény:

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

Nézzük meg mi történik. Egy, magánhangzókat tartalmazó karakterlánchoz hozzárendeljük az **intable**-t, az **outtable**-höz pedig az 1,2,3,4,5 számokat (szintén mint karakterlánc). A **maketrans** hívásával a **trantable** valójában így fog kinézni (a „\x” annyit jelöl, hogy hexadecimális karakterről van szó):

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

Ha jól megnézzük látható, hogy a kisbetűs magánhangzók lettek kicserélve számokká, ahogy azt mi korábban megadtuk:

```
1bcd2fgh3jklmn4pqrst5vwxyz
```

Még jobban megnézve azt tapasztalhatjuk, hogy valójában 256 bejegyzésünk van, az első a „\x00”, az utolsó pedig a „\xff”. A tábla tehát tartalmazza mind a 256 lehetséges ascii karaktert. Amikor a translate metódus megkapja a táblát, végigsétál (iterál) az összes karakteren, megnézi, hogy Hex-ben mi az értéke, majd ezt az értéket lecseréli a fordítótábla által meghatározott értékre, ez lesz a kimeneti karakterláncunk. Az eredeti szövegünk („The time has come”) Hex reprezentációja lent látható.

Remélem így már érthető.

Jöjjön most a dolog lényege, mire jó ez? Emlékezzünk vissza Julius Caesar-ra. A bizalmas információkat tartalmazó leveleit titkosította, mégpedig úgy, hogy az ábécé betűit három karakterrel eltolta jobbra.

```
\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'
```

Használjuk most ezt a módszert az angol ábécére:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabc
defghijklmnopqrstuvwxyz
```

A titkosítás után:

```
DEFGHIJKLMNOPQRSTUVWXYZabcdef
ghijklmnopqrstuvwxyzABC
```

Mai szemmel nézve ez ugyan nagyon egyszerűnek tűnik, emlékszem, hogy gyerekként állandóan ezt a módszert használtuk arra, hogy egymásnak üzenetet küldjünk. Különböző eltolási indexeket használtunk persze, de a dolog lényegén ez semmit sem változtatott.

Senki sem tudja már megmondani, hogy ez a módszer mennyire volt hatékony vagy éppen használható a jó öreg Julius számára. Aki esetleg elfogta az üzenetet, talán azt gondolhatta, hogy valamilyen idegen nyelven van az üzenet.

Mindenesetre a **translate** metódust és a **maketrans** segédfüggvényt felhasználva semmi sem tart minket vissza egy kis mókától. Játszunk hát Caesarosat, írjunk egy olyan programot, amely három karakterrel eltolja az általunk beírt szöveget. Az egyszerűség kedvéért csak az angol abécé nagy betűit fogjuk használni.

Tulajdonképpen minden, ami a kódban szerepel, már előkerült vagy a mai, vagy valamelyik korábbi írásomban, de azért gyorsan nézzük át.

Az első két sor a be- és kimeneti karakterlánc. Ahogy megbeszéltük, az egészet eltoljuk hárommal. A következő két sor a kódoláshoz és a dekódoláshoz szükséges táblákat adja meg. Az ötödik sor kéri meg a felhasználót a szöveg bevitelére. Ezt a szöveget a következő sorban titkosítva eltároljuk (**EncString**). A dekódoláshoz egyszerűen csak felhasználjuk a translate metódust a kódolt karakterláncon és visszakapjuk az eredeti szöveget. Végül mindkét karakterláncot kiíratjuk a képernyőre. Programunk eredménye valahogy így fest:

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

Akárcsak régen, az iskolában. No de gyúrjunk még rajta egy kicsit, tegyük használhatóbbá a programunkat. Adjuk hozzá a szóköz karaktert is a táblánkhoz (az A és Z közé). Így kicsit nehezebb lesz megfejteni, hogy a szavak hol kezdődnek és hol végződnek. Továbbá, kérdezzük meg a felhasználót, hogy titkosítani, vagy dekódolni szeretné-e a beírt szöveget. Végül szúrjunk be egy if utasítást is és csak azt írjuk ki a képernyőre, amire szükség van.

```
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)
```

A program kimenete:

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

Teszteljük a dekódolás részét is:

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

Nos, remélem tetszett a dolog és hasznodra fog válni a ma elsajátított tudás. Találkozzunk legközelebb is!

```
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)
```

Hogyanok

Írta: Greg Walters

Programozzunk Pythonban – 51. rész

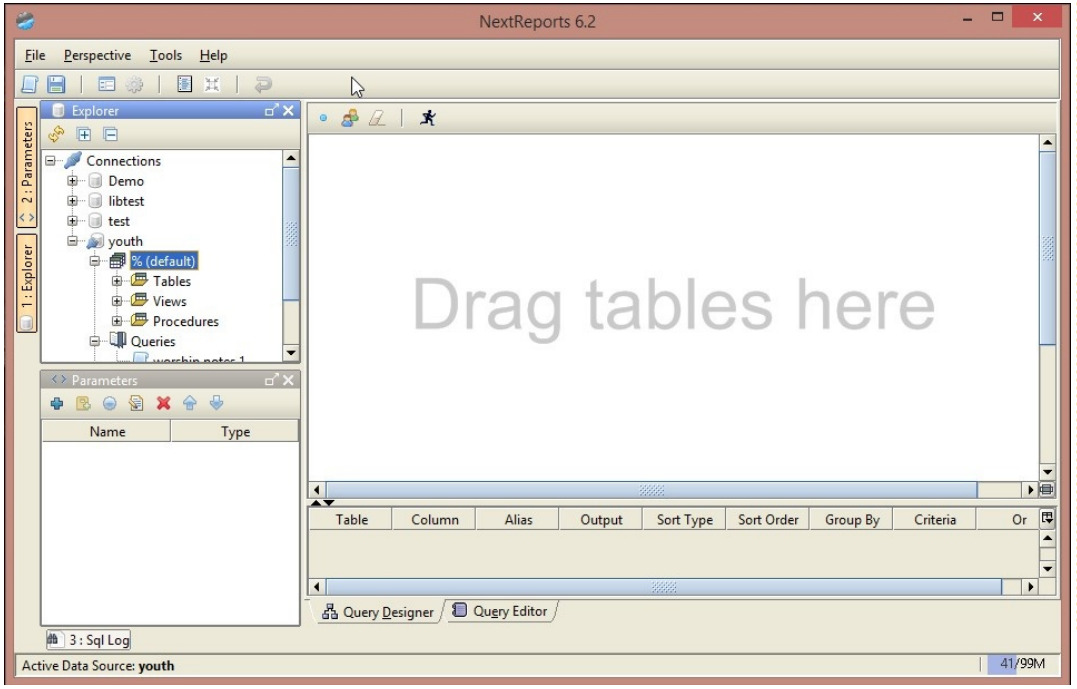
E hónapban egy programról szeretnék beszélni, mely számomra ugyan új, de már évek óta létezik. Az Advantage Software Company által fejlesztett NextReportsról lesz szó, amit megszerezhetsz ingyen innen: http://www.next-reports.com/. Ráadásul nyílt forrású és fut Windows és Linux alatt is!

Mielőtt a programról kezdenék beszélni, had álljak fel a hordómra és szónokoljak egy kicsit. Sokáig adatbázisokkal és riportokkal foglalkoztam. Az egyik dolog, amivel gondom akadt, hogy léteznek ingyenes adatbázisok – mint az SQLite és MySql –, de kevés köztük az olyan, melyhez van ingyenes riportkészítő eszköz. A riportok elkészítéséhez legtöbbször nagyon drága szoftverek szükségesek vagy a fejlesztőnek kell egyet csinálnia. Volt ugyan néhány eszköz, de mind hiányos. Amikor például grafikont kellett rajzolni, nem maradt más választás, csak a drága cucc. Hidd el, én évekig kutattam jó, ingyenes riportkészítő eszköz után, és nem értem, hogy nem vettem észre ezt a programot olyan sokáig (a 2.1.-es verzió 2009 márciusában jött ki, az aktuális változat a 6.3.). De rátalálva kihasználom amennyire lehet.

Most, hogy leszálltam a hordóról, elkezdhetek dicshimnuszokat zengeni a programról. Három részből áll, van egy riport tervező, egy riport motor és egy riport szerver. Csak a tervezőt volt lehetőségem kipróbálni, de ha ez is olyan hatékony, egyszerű és rugalmas, mint a többi rész, akkor ez egy nyerő alkalmazás.

E hónapban a tervezőre fogunk összpontosítani. Kevés az időm és egy windowsos gépen dolgozom, de amit mutatok, azt mind meg lehet csinálni Linuxon is (előre is elnézést kérek emiatt).

Elsőként el kell mondanom, hogy Oracle, MySql, SQLite, MSSQL és más adatbázisokat is támogat. Minden lekérdezéseken alapul, és az egy jó dolog, hogy csak SELECT típusú lekérdezések megengedettek. Ez azt jelenti, hogy semmi nem változtatható meg véletlenül az adatbázisban. Beírhatod a saját lekérdezéseidet vagy használhatod a vizuális tervezőt.



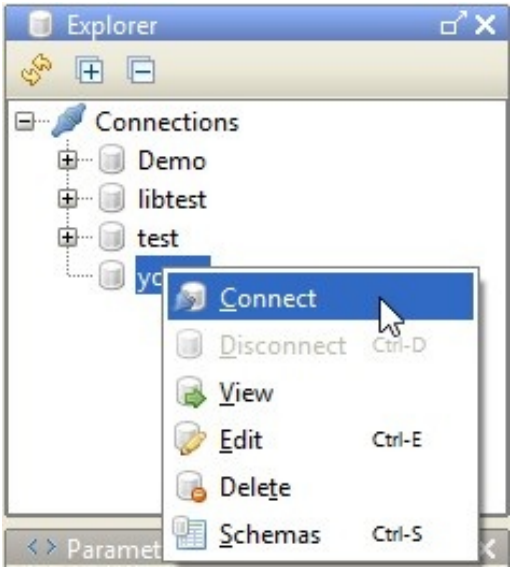
A képernyőképen látható, milyen szép is az UI. Minden nagyon intuitív és nem tart sokáig megtanulni hatékonyan használni. Először vessünk egy pillantást a folyamat lépéseire!

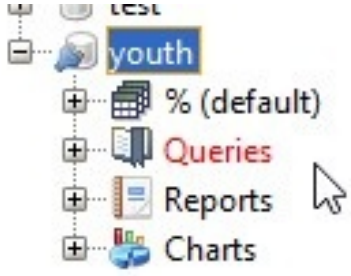
Kezdd a File | New | Data Source-nál, aztán adj egy tetszőleges nevet a forrásnak.

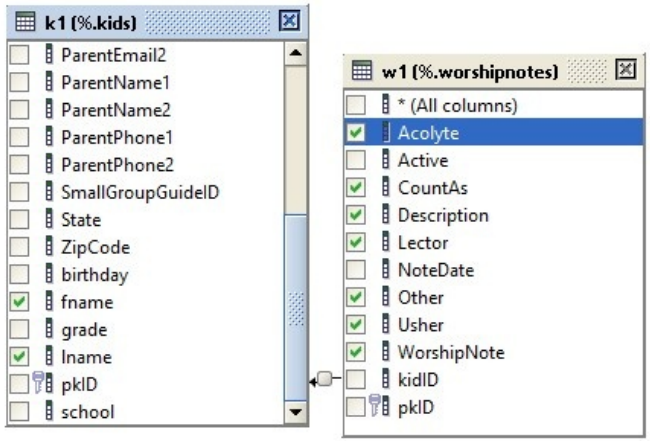
A Type legördülő menüben add meg az adatbázis típusát. A Driver részt átugorhatod, így az URL:-hez jutsz. Itt kell megadnod az adatbázis elérési útvonalát. Ha például SQLite adatbázist használsz, az alapértelmezett értéke ennek a mezőnek „jdbc:sqlite:<dbfile-path>”. A <dbfile-path> helyére írd az adatbázis elérési útvonalát. Más adatbázisokhoz hasonló előre kitöltött mezőket kapsz. Ezután kattints a Test gombra a kapcsolat ellenőrzéséhez. Ha minden jó, menj a Save-re, és a Connections fában megjelenik egy új elem. A következő szükséges lépés a kapcsolódás az újonnan létrehozott adatbázishoz. Jobb kattintás az adatbázisra és válaszd a Connect-et.

Ha a kapcsolódás sikeres, négy lehetőségből választhatsz. A „%” az

adatbázis tábláit jelenti. A másik három az új lekérdezések, riportok és diagramok létrehozása. Elég egyszerű. Most menj a „%” jel bal oldalán a „+”-ra, mely megnyitja az adatbázis táblák megjelenítőjét. Most a fában látható a Tables, a Views és a Procedures. A Tables mellett megint kattints a „+” jelre. Ezzel láthatóvá válik az összes táblád. Ha most szeretnéd a vizuális lekérdezés-tervezőt használni, csak húzd a használni kívánt táblákat a tervező jobb oldali ablakába.







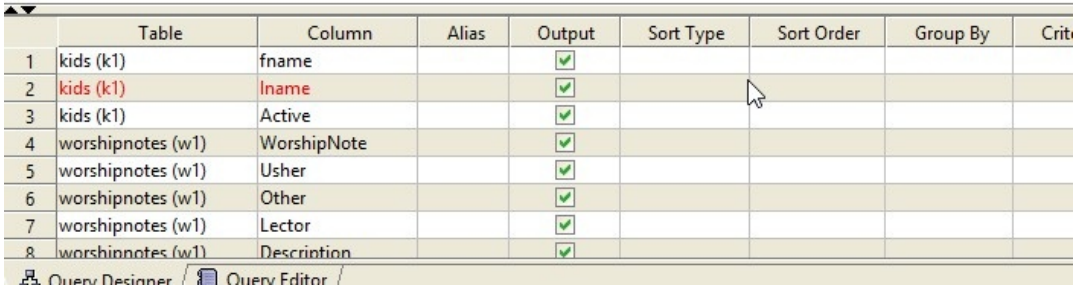


	Table	Column	Alias	Output	Sort Type	Sort Order	Group By	Crit
1	kids (k1)	fname		☑				
2	kids (k1)	lname		☑				
3	kids (k1)	Active		☑				
4	worshipnotes (w1)	WorshipNote		☑				
5	worshipnotes (w1)	Usher		☑				
6	worshipnotes (w1)	Other		☑				
7	worshipnotes (w1)	Lector		☑				
8	worshipnotes (w1)	Description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

Query Designer | Query Editor

Amint az összes szükséges tábla ott van, létrehozhatod köztük a kapcsolatokat.

Ebben a példában két tábla van, egyikben a hittant tanuló gyerekek, egy másikban az istentiszteletekről készült feljegyzéseik. Az utóbbi nem tartalmazza a gyerekek nevét, csak egy azonosítót, mely a másik táblára mutat. Egy fogd és vidd művelettel összekapcsoltam a kidID mezőt a pkID-vel a gyerek táblában. Kiválasztottam a mezőket, melyeket az eredményben szerettem volna látni: a gyerekek vezeték- és keresztnevét, egy aktív (nem törölt) jelzőt a gyerek táblából, és néhányat a megjegyzésekből. Lent láthatóak a mezők, hogy melyik táblából származnak és egyéb információk is.

Beállíthatjuk, hogy egy mező szerepeljen a kimenetben (pl. Active = 1) vagy sem, továbbá a rendezés típusát és sorrendjét is. Ha ezzel végeztél, kattints a lenti fülre és láthatod az eredményt.

```
1  SELECT
2      kl.fname,
3      kl.lname,
4      wl.WorshipNote,
5      wl.Usher,
6      wl.Other,
7      wl.Lector,
8      wl.Description,
9      wl.CountAs,
10     wl.Acolyte
11 FROM
12     kids kl,
13     worshipnotes wl
14 WHERE
15     wl.kidID = kl.pkID AND
16     kl.Active = 1
17 ORDER BY
18     kl.lname,
19     kl.fname
```

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		☑				

tod az aktuális SQL lekérdezést.

A lekérdezés teszteléséhez lépj a futó alakra és (ha mindent jól csináltál) megkapod az eredményeket a szerkesztő alatt egy táblában. Ha akarsz, kézzel is hozzáadhatsz sorokat. Ha esetleg szeretnéd összevonni a kereszt- és vezetékneveket (fname és lname) egy teljes névbe, ezt megteheted a következő sor beszúrásával a „k1.lname” után:

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

A „||” karakterek az összefűzés műveletet jelölik, eredményként a FullName mezőt kapjuk, mely a két nevet tartalmazza, köztük szóközzel. Ne felejtsd le a vesszőt a sor végéről! Ha elégedett vagy a lekérdezéssel, a mentés gombra kattintva elmentheted tetszőleges nevet adva neki.

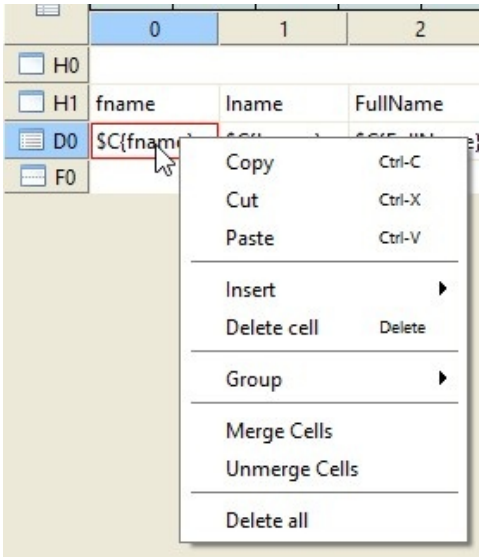
Menj a fában a Query elemre és jobb gombbal a most létrehozott lekérdezésre. Válaszd a New Report from Query-t. A lekérdezéstervező ablaka helyén a riporttervező jelenik meg.

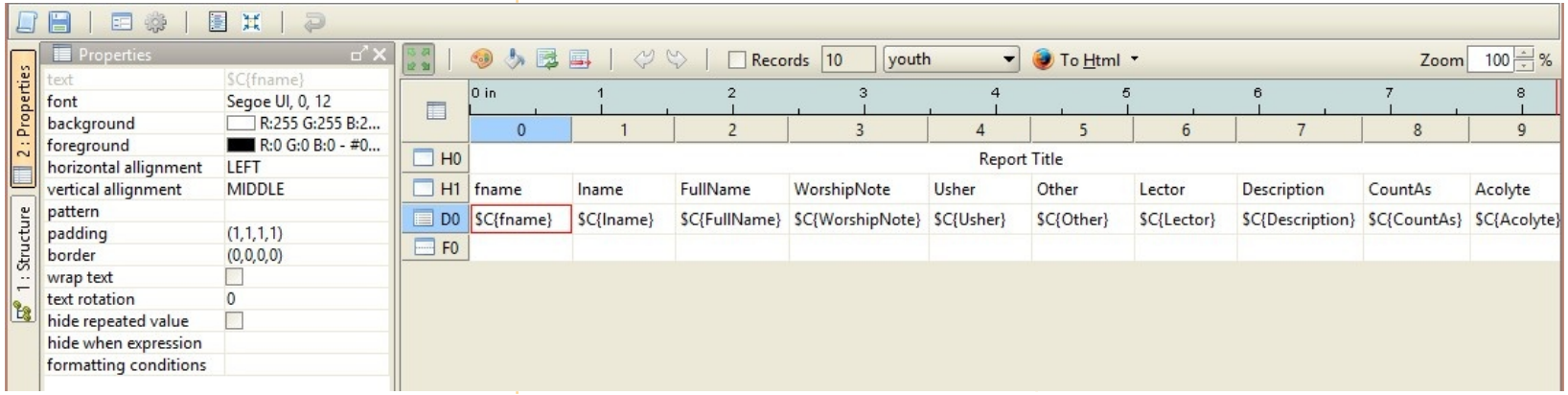
Bal oldalt a tulajdonságok ablak a mezők vagy a teljes riport adatait mutatja. Jobbra van a riporttervező. Hasonlít egy táblázathoz. Minden sor egy sáv, mely információkat tartalmaz a riport egy soráról. Ez esetben van négy sor, ebből kettő fejléc, egy részletek és egy lábléc. Hozzáadhatsz vagy törölhetsz sorokat szükség szerint. Kötöttebb módszer, mint némely más riport tervezőé, de nagyon tetszetős és áttekinthető riportot eredményez.

A két fejléc sorban van a riport címe és az oszlopok fejléce. A részletek sorban minden riportban szereplő mező szerepel. A lábléc sor a lábléc. Nézzük, hogyan fest a riport alapértelmezetten. Megtekintéséhez kattints a sáv tetején a To Html gombra. (A gyerekek vezetékneveit én töröltem ki, ez nem a generátor hibája.)

Egy összecsapott riporthoz képest elég jó, de csinosítsunk rajta egy kicsit. Készítsünk egy csoportot, mely a gyerekek minden adatát a nevük alatt helyezi el.

Jobb kattintás az adat sor első oszlopára, válaszd a Group-ot, majd az Add-ot!





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

Egy ablak jelenik meg, itt megadhatod, mely mezők szerepeljenek a csoportban. Én a FullName--et választottam és az OK gombra kattintottam. Most van egy csoportosító szakaszunk. A részletek közül kivehetjük a három mezőt (fname, lname, FullName), mivel a csoport sávon kiírjuk majd a nevet. Kattints rájuk jobb gombbal és válaszd a Delete Cellt. Most méretezd át a három üres cellát a bal oldalon, hogy csökkentsd a rést!

Egy gyors pillantást vetve a riportra, láthatjuk, hogy a gyerekekhez tartozó információk szépen össze vannak csoportosítva.

Így már jobb, de most csináljunk valami mókásat! A nullák és egyesek nyilván igent és nemet jelentenek. Így azonban nagyon unalmasan festenek a riportban, adjunk hozzá egy feltételes utasítást ezekhez a mezőkhöz, hogy egy négyzet mutassa az állapotukat: bejelölt az igen (vagyis 1), üres a nem (vagyis 0). Ezt igen egyszerű megcsinálni, de úgy tűnik, mintha napokat szántál volna a riportra. A Windows Wingdings fontjából a 0x6F (0168) lesz az üres négyzet, a 0xFE (0254) a bejelölt.

Mielőtt folytatnám, van egy dolog, amit a Windows jobban tud a Linuxnál: az Alt és numerikus billentyűzet segítségével speciális karaktereket tudsz bevinni. Linuxon ezt nem lehet. Egy kerülő megoldás a Ctrl+Shift+U kombinációt használta a kívánt Unicode karakter bevitelére. De ez nem működött minden gépen. Linuxon erre a legkönnyebb megoldás, amit találtam: nyisd meg a Character Map-et, keress rá a kívánt Unicode karakterre, dupla kattintással másold be a Text to copy: dobozba, kattints a Copy-ra és illeszd be a dokumentumba. A keresett Unicode karakterek a 2610 (üres doboz) és 2611 (doboz pipával) a WingDings 2 fontkészletben. Biztos vagyok benne, hogy még vannak más, könnyebb módszerek ennek a megoldására, de nem volt több időm. (Ellenőrizd, hogy a Script listából kiválasztottad a Commont.)

A WorshipNotes mezővel kezdjük. A részletek sorban jobb kattintás a mezőre, melyet módosítani akarsz. Ez most a $C{WorshipNote}. Válaszd az Insert-et, majd az Expression-t. Még egy jó dolog, hogy a NextReportsban majdnem mindent megtehetünk kevés gépeléssel. Nézd meg az Operators részt az ablak közepén. Kattints kétszer az „if..else..”-re, és az bekerül mintaként a szerkesztőbe, így nem ejtessz hibát.

Most be kell tenni a WorshipNotes mezőt a szerkesztő zárójelei közé. Csak kattints a két zárójel közé a kurzor áthelyezéséhez, aztán dupla katt a mezőre, amit oda akarsz helyezni. Bumm! Ki is van töltve. Most a szerkesztőben lépj a mező neve mögé, aztán dupla katt az „== (eq)” operátorra. Végül írj be egy “1”-est, ekkor a szerkesztő sorában ez látható:

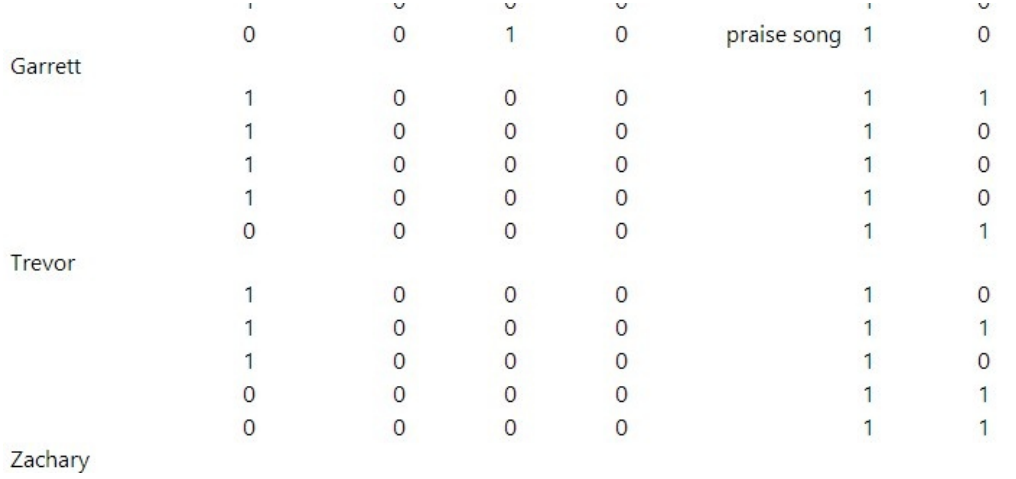
```
if ( $C_WorshipNote == 1 ) {
; } else { ; }
```

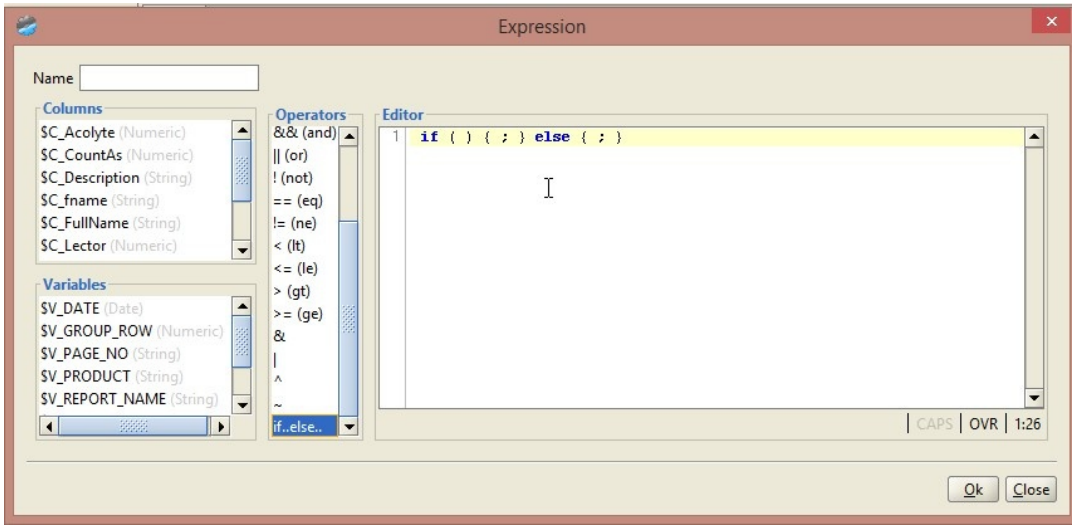
Már majdnem kész a kifejezés. Az első pár kapcsos zárójel adja meg, hogy mit kell tenni, ha a kifejezés igaz, a második pedig, ha hamis. A CharMap-et használjuk (Windowson, Linuxon is van ilyen, Gnome alatt például gucharmapnek hívják), hogy a karaktereket a szerkesztőbe illesszük. Vagy Windows alatt lenyomod az Alt billentyűt és beírod a 0168-at az üres dobozért, és a 0254-et a jelöltért. Most a következőképp fest a kifejezésünk (legalábbis Windows alatt):

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

Adj nevet a kifejezésnek (én a WNotes-ot választottam) és mentsd el. A mező tulajdonságai között válaszd ki a betűtípust (most a WingDings-et használtam) és így fog kinézni az eredmény (lásd a következő oldalon).

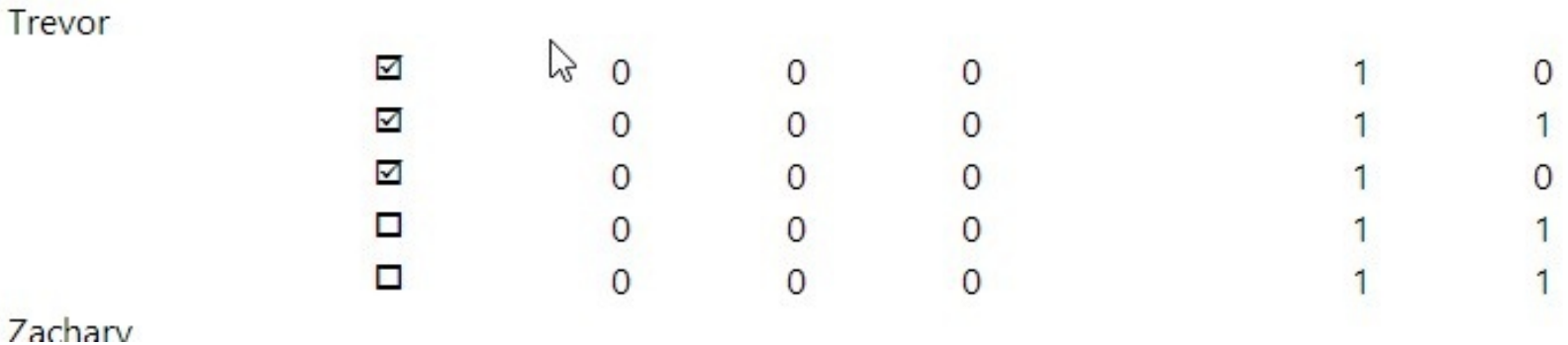
Itt vannak a mi kis dobozkáink. Más mezőkkel hasonlóan egyszerű ezt megcsinálni.





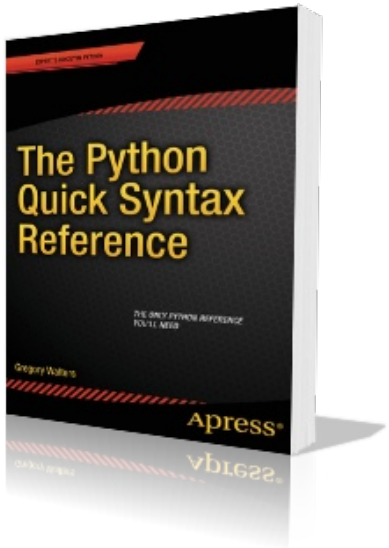
Csak három órát játszadoztam a programmal, és ennél még tovább jutottam. Komolyan mondom, hogy sokkal többet tanultam, de a továbbiakat meghagyom későbbre. Sablonokkal színezheted a riportot, hozzáadhatsz képeket és még sok mást megtehetsz.

Legközelebb elmondom, hogyan ágyazhatjuk be ezeket a riportokat egy Python programba. Addig próbálgassátok ezt a nagyszerű SZABAD szoftvert.



Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta programozik. Szeret főzni, túrázni, szereti a zenét és az idejét a családjával tölteni. Honlapja: www.thedesignatedgeek.net.

Programozzunk Pythonban – 52. rész



Mielőtt rátérnénk az éppen aktuális python cikk tárgyára, hadd említsek meg egy kiadványt. December végén/január elején megjelent az első, Pythonról szóló könyvem az Apress kiadónak köszönhetően, melynek címe „The Python Quick Syntax Reference". Több helyen is elérhető, például az Apress oldalán (http://www.apress.com/9781430264781), a Springer.com-on (http://www.springer.com/computer/book/978-1-4302-6478-1) és Amazonon (http://www.amazon.com/The-Python-Quick-Syntax-Reference/dp/1430264780/).

Ahogy a cím is sugallja, ez a könyv egy olyan szintaxis referencia, amely leginkább azok számára lehet hasznos, akik Python mellett más nyelvekben is programoznak: segít megjegyezni, hogy egy adott parancs hogyan működik és mik az előfeltételei. Amennyiben módodban áll, kérlek támogasd egy szegény öreg programozó megélhetését a könyv megvásárlásával.

No de most lépjünk tovább a nagyobb és jobb dolgok irányába.

A fent reklámozott könyvemen dolgozva újra felfedeztem egy olyan SQL parancsot, melyre korábbi SQL adatbázisokkal foglalkozó Python cikkeim során még nem tértem ki. Szóval arra gondoltam, most a CREATE TABLE AS SELECT utasítást vesézzük ki, mellyel már meglévő táblából (vagy összekapcsolt táblákból) készíthetünk egy újat. Az általános szintaxis így néz ki:

```
CREATE TABLE [IF NOT EXISTS]
{New Table Name} AS SELECT
{query}
```

A szögletes zárójelben lévő rész (IF NOT EXISTS) opcionális és arra szolgál, hogy a táblát csak akkor hozzuk létre, ha még nem létezik. A kapcsos zárójelben lévő rész viszont kötelező. Az első az új tábla nevét, a második pedig azt a lekérdezést jelöli, mellyel az új táblát létre akarjuk hozni.

Tegyük fel, hogy az adatbázisunkat több tábla alkotja. Az egyik tábla neve legyen „study" és tartalmazza a fogadó műveleteket. Hat mezőből áll.

Az első adathalmaznak, melyet létrehozunk a nyers adatokból, a csomagok és a napok számát kell tartalmaznia, feltételezve, hogy hétköznapokról van szó (hétfőtől péntekig) és hogy a napok nem ünnepnapok (ünnepnapokon tipikusan kevesebb csomag szokott lenni). A lekérdezésünk így néz ki:

```
SELECT pkgs, Count(DOW) as CountOfDOW FROM study
   WHERE (Holiday <> 1)
      AND DayName in ("Monday","Tuesday","Wednesday","Thursday","Friday")
   GROUP BY pkgs
```

Az eredményül kapott adat szerkezete:

```
pkID - Integer, Primary Key, AutoIncrement
DOM -  Integer - Day of the month (1-31)
DOW -  Integer - Day of week (1-7 (Sunday = 1, Monday = 2, etc))
pkgs - Integer - Number of packages received that day
DayName - TEXT - "Sunday","Monday", etc
Holiday - Integer 0 or 1 (Is this day considered a holiday or not) 1 means yes
```

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

48                3
```

Az adatok tehát azt mutatják, hogy a tanulmányozott 65 nap közül csak egy hétköznapon volt 31 csomagunk, 3 munkanapon 48 és így tovább. Hasonló lekérdezéseket kedvünk szerint készíthetünk, melyek akár a munkaszüneti napokat és a hétvégéket is tartalmazzák.

A lekérdezés után az adatokat egy adathalmazként kapjuk vissza és van, amikor további analíziseket is szeretnénk végrehajtani az adaton, ezért azt egy táblába mentjük. A következő példában létrehozunk egy „weekdays” táblát a fent bemutatott lekérdezéssel.

Ezek után ha egy adott hétköznap adataira szükségünk van, egyszerűen csak futtatunk egy lekérdezést a weekdays táblán.

Ha már tudjuk, mit szeretnénk és teszteltük a lekérdezésünket, megírhatjuk a kódunkat. Tegyük fel, hogy a study táblát már létrehoztuk és feltöltöttük. Ezek után

```
CREATE TABLE IF NOT EXISTS weekdays AS
   SELECT pkgs, Count(DOW) as CountOfDOW FROM study
   WHERE (Holiday <> 1)
      AND DayName in ("Monday","Tuesday","Wednesday","Thursday","Friday")
   GROUP BY pkgs
```

Python segítségével a fő adatbázisunkban létrehozhatjuk az új tábláinkat. Ezt a feladatot az APSW SQLite csomag segítségével fogom elvégezni.

Nyitnunk kell egy kapcsolatot az SQLite adatbázishoz és létrehoznunk egy kurzort. Erről a korábbi számokban már sokszor esett szó.

Ha ez megvan, szükségünk lesz egy olyan rutinra, amely a lekérdezés során kapott adathalmazból létrehozza az új táblát, ezt megfelelően feldolgozza és elvégez rajta néhány számítást.

Ahogy az látható, létre-

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

hozunk egy újabb kurzort, melyet a kód utolsó részében fogunk majd használni. Ha a tábla már létezik, ejtjük és a lekérdezésünket a „study” táblán hajtjuk végre.

Három újabb oszlopot hozunk létre a weekdays táblában: „probability”, „lower” és „upper”. Ezt az „ALTER TABLE” SQL paranccsal tudjuk megtenni:

Ezután összegezzük az adatokat a CountOfDOW mezőben.

```
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
               ("Monday","Tuesday","Wednesday","Thursday","Friday")
               GROUP BY pkgs'''
     cursor.execute(query)
```

Csak egyetlen rekordot kapunk vissza, de úgyis ott van még a for ciklusunk. Emlékezzünk, hogy a „CountOfDow” mező tartalmazza azoknak a napoknak a számát, amikor a study szerint megfelelő számú csomag érkezett. Ez tehát egy olyan érték, ami tartalmazza az összes „CountOfDow” összegét. Az én konkrét példámban ez a szám 44.

```
upquery = "SELECT * FROM
weekdays"

c1 = cursor.execute(upquery)
```

Itt egy „SELECT all” lekérdezést hajtottunk végre, így az adatbázis összes rekordja a „c1” kurzorban lesz. Az egész adathalmazon végig fogunk sétálni, lekérve az összes pkgs (row[0]) és CountOfDow (row[1] adatot változókba.

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

Most az összes napi csomagszámnak meghatározzuk a valószínűségét és kiszámoljuk a felső (upper) és alsó (lower) értékeket, melyeket a későbbiekben fogunk felhasználni. Ellenőrizzük azt is, hogy a LastUpper változó értéke tartalmazza-e a „.0”-t. Ha igen, beállítjuk a valószínűség értékét, máskülönben a alsó (lover) + valószínűség (prob) értéket adjuk meg.

Végül frissítjük az SQL utasítást úgy, hogy az új, számolt értékek is bekerüljenek az adatbázisba.

Végül kapunk egy csomagszámot (pkgs), a napok számát, amikor a megadott csomagszám érkezett, ennek a valószínűségét a teljes study-ra vonatkozóan (31 csomag egyetlen napon érkezett a 44-ből (hétköznapok a 60+ napot tartalmazó study-ban), mely 0.02-es valószínűséget jelent.).

A valószínűség értékeket összeadva a táblában 1.0-át kell, hogy kapjunk.

A felső és alsó értékek egy olyan, 0 és 1 közötti lebegőpontos számot fognak adni, amely egy tetszőleges csomagszám valószínűségét jellemzi az adott határokon belül. Ez a szám felhasználható az adat statisztikai analízisére. Életszerű példa lenne az autómosóba érkező autók számának jóslása korábbi adatok alapján. Ha szeretnéd jobban megérteni a dolgot, vess egy pillantást az alábbi linkre:

```
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
#====================================
```

http://www.algebra.com/algebra/homework/Probability-and-statistics/Probability-and-statistics.faq.question.309110.html.

A két rutin kódja ez alkalommal is elérhető a következő helyen:

http://pastebin.com/kMc9EXes

Találkozunk legközelebb.

Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta programozik. Szeret főzni, túrázni, szereti a zenét és idejét a családjával tölteni. Honlapja: www.thedesignatedgeek.net.

Hogyanok

Írta: Greg D. Walters
Fordította: Tulipán Attila

Programozzunk Pythonban – 53. rész

Ebben a hónapban arra gondoltam, hogy egy olyan függyvényt írok, amely kulcsot generál egy emailből. Mindannyian nagyon jól tudjuk, hogy miért van szükségünk titkosítási kulcsra. Amennyiben szükséged van egy gyorsan összedobott függvényre, akkor használhatod ezt. Ne felejtsd, hogy a Python nyelv egy szkriptelő nyelv, ami azt jelenti, hogy a forráskódod olvasható. Ez más módon orvosolható, de ezt egy későbbi cikk- ben fogom tárgyalni. Nézzük át a kód logikáját nagy vonalakban, mielőtt kódolni kezdünk.

Először is bekérünk egy email címet, és két részre bontjuk: a lokális (a „@” jel előtti rész) és a tartomány (a „@” karakter utáni rész a domén) részre. Vannak bizonyos szabályok az érvényes email címre vonatkozóan, ami tovább bővítve igen bonyolulttá is válhat. A céljainknak elegendő lesz néhány szabály, és azokat is csak a lokális részre alkalmazzuk. További információért rákereshetsz az interneten az akutális szabályokra. A kódunkban csak a következő szempontokat fogjuk figyelembe venni:

- kisbetűs karakterek
- nagybetűs karakterek
- számok 0 és 9 között
- speciális karakterek (!#$%&'*+-/=?^_`{|}~.)
- a pont („.”) karakter engedélyezett, de csak ha nem ismétlődik egymás után (pl.: ...)

Miután ellenőríztük az email cím érvényességét, készítünk egy „ellenőrző kódot”, amely az ASCII tábla karakter-értékeit veszi alapul. Az email cím minden egyes karakterének ASCII értékét összeadjuk, amit aztán elosztunk a teljes email cím karaktereinek számával. Például használjuk a nagyon különleges fredjones@someplace.com email címet. Ha bejárjuk az email cím karaktereit, akkor az ord() függvénnyel megkaphatjuk minden egyes karakter ASCII értékét. Miután ezeket összeadtuk, megkapjuk összegként az 1670-et. Ezt aztán elosztjuk az email cím hosszával (23). Ekkor megkapjuk a 72-t. Ne felejtsd, hogy itt az integer adattípussal osztunk, így az eredmény is integer típusú lesz.

Most, hogy megvan az ellenőrző kódunk, kivonunk belőle 68-at (a „D” karakter ASCII értéke), ami az eltolás mértékét fogja adni. Ezt az eltolást a titkosításhoz használjuk, amikor az email karaktereit „titkosítjuk”. A visszafejtés nehezítésére a hosszat (az eltolással együtt) a 2. karakterként, az ellenőrző kódot pedig a 4. karakterként adjuk meg.

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

Tehát a fredjones@someplace.com-re a következő titkosítási kulcsot kapjuk:

```
j[vHihnsriwDwsqitpegi2gsq
```

Kezdjünk is a kódoláshoz. Mivel ez már a cikksorozat 53. része, nem fogok olyan részletesen leírni minden egyes lépést.

Először is az importálás jön.

```
import sys
```

Készítünk (jobbra, fent) egy karakterláncot (string típus), amely tartalmazza az összes érvényes karakterünket, amelyet az IsValidEmail függvényben használunk. Szétdaraboltam őket három karakterláncra, hogy a magazinban is szépen látszódjanak. Majd összefűzzük őket az IsValidEmail függvényben. Megadunk egy globális „Offset” (eltolás) változót is 0 értékkel. Ezt az értéket fogjuk hozzáadni (később) minden egyes karakter értékéhez, amikor a titkosítást végezzük.

```
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
```

Most jön az első függvényünk. Ez pedig az IsValidEmail függvény. Alapvetően „s” változóként átadjuk az emailt a függvénynek, továbbá egy opcionális debug szintet („debug flag” – hibaüzenet kiíratásának a szintje). Ez a debug flag felelős azért, – ahogyan azt a korábbiakban is használtuk –, hogy a kód futása közben információt kapjunk, hogy mégis mi történik. Általában az „s” változónak 1-es értéket adunk, ha a folyamat üzeneteit részletesen szeretnénk látni.

Először is a paraméterként megkapott email értékét az „email” változónak adjuk és megkeressük benne a „@” karaktert, amely elválasztja az email lokális és tartományi részét. Aztán az email lokális részét (értelemszerűen) a „local” változóhoz rendeljük, a tartományi részt pedig a „domain” változóhoz. Ezután beállítjuk az „isgood” logikai változót hamis értékre. Aztán pedig létrehozzuk a „localvalid” változót, amely a korábban készített 3 rövidebb karatkerláncból áll.

Ezek után pedig egyszerűen bejárjuk az email lokális részének karaktereit a listánkban szereplő érvényes karakterekkel az „in” kulcsszó segitségével. Ha valamelyik karakter az email lokális karakterei közül nem felel meg az ellenőrzés során, akkor kilépünk a ciklusból, miközben az „isgood” változót „False”-ra (hamis) állitjuk.

Végezetül rákeresünk az egymás utáni pont („.”) karakterláncra. Ehhez a string.find függvényt használjuk, amely megtalál minden „..”, vagy „...”, stb. előfordulást. Mivel lusta programozó vagyok, csak egy „dupla pont” ellenőrzést használtam, ami működik a többire is.

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

Az utolsó dolog, amit a függvényben csinálunk az, hogy a return utasítással visszaadjuk az „isgood” logikai változó értékét.

```
return isgood
```

A következő függvény a CheckSum függvény, ami viszonylag rövid. Bejárjuk az email karaktereit és generálunk egy futó összegzést az ASCII karakterek alapján, amelyhez a beépített ord tipusú átváltást használjuk. Ahogyan említettem korábban ezt az összértéket fogjuk osztani az email cím hosszával. Ekkor a return utasítás segítségével visszaadjuk a checksum értékét és a checksum karaktereit.

```
    # 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)
```

Az EncodeKey függvény következik. Bár nagyon egyszerűnek tűnik, fontos, hogy megértsük, úgyhogy figyelmet kérek! Az „Offset” változót globális változónak adjuk meg, hogy a függvényen belül is értéket tudjunk neki adni, illetve más függvények is használhassák. Aztán az „Offset” változó értékének a checksum-68 értéket adjuk. Amint a példa is mutatja a cikk elején ez 72-68, ami 4. Aztán bejárjuk az email cím minden karakterét, miközben az offset értékét hozzáadjuk az adott karakter ASCII értékéhez. Tehát az „f” a fredjones-ban 102+4, vagyis 106, ami az „i” karakter. A „cntr” változót használva, megállapítjuk, hogy mit adunk a NewEmail karakterlánchoz, ahogyan haladunk karakterről karakterre. Vegyük észre a kódban, hogy a számláló 0-ról kezd, és haladunk az email hosszáig. Tehát a 0. karakter az „f”. az 1. karakter az „r” és igy tovább. És most jön az a rész, ami összezavarhat néhány embert. Ha a „cntr” változó értéke 1 („r”), akkor beszúrjuk az email hosszának érté-

két+68 és az eltolás értékét, ami a példánk alapján *iYt* lesz. A következő ciklusban a cntr értéke 2, de már van 3 karakterünk az emailben. Itt fogjuk beszúrni az ellenörző kód karakterét („F”) és az eltolás karakter értékét harmadikként. Innentől kezdve csak szépen adogatjuk hozzá minden egyes eltolás értékét a karakterlánchoz. Amikor a ciklus lefut, akkor visszaadjuk a kulcs értékét.

A DecodeKey függvény lényegében visszafele csinálja, amit az EncodeKey függvényben csináltunk Egyetlen dolog, amire érdemes figyelni, hogy az első „if debug” feltételvizsgálatban a „!=0”-t használtam és nem a „==1”-t. Ez a kettő valójában felcserélhető.

A DoIt függvény egy email címet kér be, mégpedig „raw_input” formátumban, aztán pedig meghívja a függvényeket, hogy legenerálja a titkosítási kulcsot.

Így tehát végül meghívjuk a DoIt függvényt.

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

Láthatjuk, hogy az eredmény nem szupertitkos. Sőt, ha valakinek elég időt hagynánk, akkor könnyedén visszafejthetné, hogy hogyan is készitettük a kulcsot. Mégis kezdésnek elég ez, amit aztán módosíthatunk, hogy nehezebben lehessen feltörni. Lehetne például használni véletlen számot a „D” (68) helyett. Ha ezt választanánk, akkor tegyük bele a kódba a kezdeti magot (seed), hogy minden alkalommal ugyanazt a véletlen számot generálja. Vagy lehetne egy kicsit mélyebbre is menni, amikor az eltolás értékét beteszed valahova a kulcsba, például az utolsó karakter helyére, ami a visszafejtő eltolás lehetne.

```
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
```

Mint mindig, a teljes forráskód elérhető a http://pastebin.com/MH9nVTNK címen. Jó szórakozást a kóddal, a következő alkalomig.

Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta progra-mozik. Szeret főzni, túrázni, szereti a zenét és idejét a családjával tölteni. Honlapja: www.thedesignatedgeek.net.

Közreműködnél?

A FULL CIRCLE-nek szüksége van rád!

Egy magazin, ahogy a Full Circle is, nem magazin cikkek nélkül. Szükségünk van játékok, programok és hardverek áttekintő leírására, ezenkívül bármire, amit elmondanátok a *buntu felhasználóknak. A cikkeiteket küldjétek a következő címre: articles@fullcirclemagazine.org

Folyamatosan keressük a cikkeket a magazinba. Segítségül nézzétek meg a **Hivatalos Full Circle Stílus Útmutatót**: http://url.fullcirclemagazine.org/75d471

Véleményed és Linuxos tapasztalataidat a letters@fullcirclemagazine.org címre,
Hardver és szoftver **elemzéseket** a reviews@fullcirclemagazine.org címre,
Kérdéseket a „Kávé” rovatba a questions@fullcirclemagazine.org címre,
Képernyőképeket a misc@fullcirclemagazine.org címre küldhetsz,
... vagy látogasd meg a **fórumunkat** a fullcirclemagazine.org címen.

A Full Circle Magazin beszerezhető:

EPUB - Az utóbbi kiadások megtalálhatók epub formátumban a letöltési oldalon. Ha bármi problémád lenne az epub fájllal, küldj e-mailt a mobile@fullcirclemagazine.org címre.

Google Currents - Telepítsd a Google Currents programot az Android/Apple eszközödre, keresd rá a „full circle”-re (a programon belül) és hozzáadhatod az 55., vagy újabb kiadásokat. Vagy letöltheted az FCM letöltési oldaláról.

Ubuntu Szoftver Központ - Megszerezheted a magazint az Ubuntu Szoftver Központból is https://apps.ubuntu.com/cat/. Keress rá a „full circle”-re, válassz egy kiadást és kattints a letöltés gombra.

Issuu - Olvashatod a Full Circle Magazint online az Issuu-n: http://issuu.com/fullcirclemagazine. Oszd meg és értékeld a magazint, hogy minél többen tudjanak a magazinról és az Ubuntu Linuxról.

Ubuntu One - Letöltheted a kiadásokat a saját Ubuntu One tárhelyedre, ha rákattintasz a „Send to Ubuntu One” gombra, ami elérhető az 51. kiadástól.

A Full Circle Csapat

Szerkesztő – Ronnie Tucker
ronnie@fullcirclemagazine.org
Webmester – Rob Kerfia
admin@fullcirclemagazine.org
Podcast – Les Pounder & Co.
podcast@fullcirclemagazine.org

Szerkesztők és Korrektorok
Mike Kennedy, Lucas Westermann, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred

Köszönet a Canonicalnek, a fordítócsapatoknak a világban és **Thorsten Wilms**-nek az FCM logóért.

Full Circle magazin
Magyar Fordítócsapat

Koordinátor:
Pércsy Kornél

Fordítók:

A fordítók csapata

Lektor:
A fordítócsapat lektorai

Szerkesztő/Korrektor:
Heim Tibor