Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

Programozói sorozat – Különkiadás

Programozói sorozat Különkiadás

Programozzunk Pythonban

8. 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” 44–48. részének** az újabb kiadása (a magazin 73–78. 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!

Hogyanok

Írta Greg Walters

Programozzunk Python-ban - 44. rész

Ez alkalommal teszünk egy kis kitérőt, most nem a TVRage-ről lesz szó. Egy olvasóm azt kérte, hogy beszéljünk a QT Creator-ról, arról, hogy hogyan használható Python programok felhasználói felületének tervezésére.

Sajnos az alapján amit tudok, azt kell mondanom, a QT Creator még nem áll készen a Python támogatására. Dolgoznak ugyan rajta, de a dolog nagyon nincs még kész.

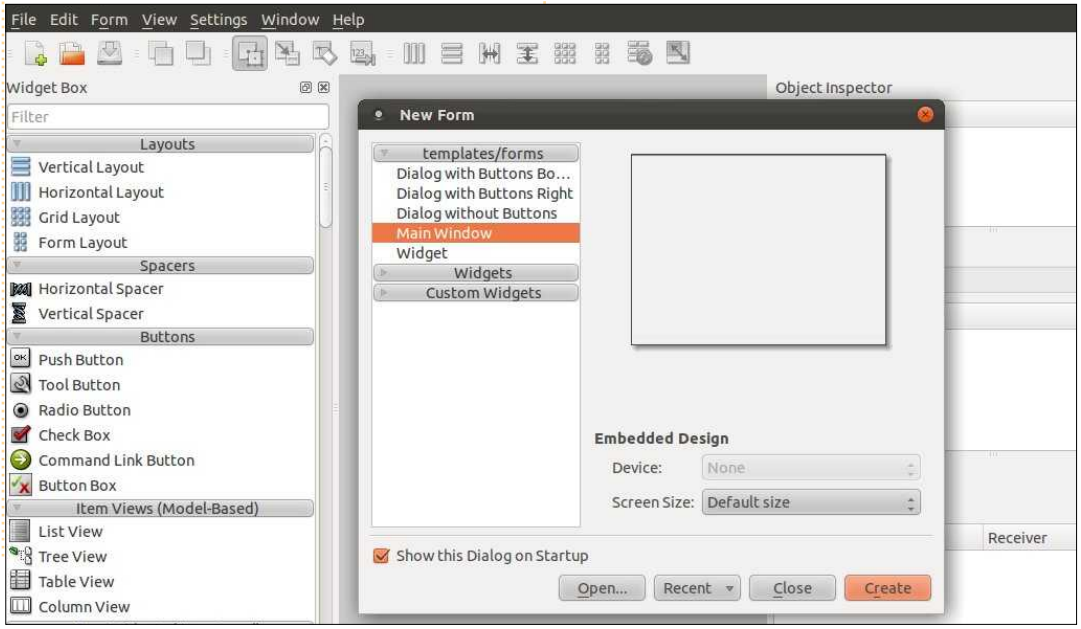
Mégis, szeretném felkészíteni magunkat egy jövőbeli cikkre, így ma a QT4 Designer-rel fogunk dolgozni. A python-qt4, qt4-dev-tools, python-qt4-dev, pyqt4-dev-tools és libqt4-dev csomagokra lesz szükségünk, ha hiányoznak a gépedről, kérlek telepítsd fel őket.

Ha ez megvan, a QT4 Designert az Alkalmazások | Programozás alatt találod. Rajta, indítsd el. Valami ehhez hasonlót kellene látnod:

Válaszd ki a 'Main Window' ablakot és kattints a 'Create' gombra. Így lesz egy üres űrlapunk, amelyet fogd és vidd módszerrel fel tudunk tölteni.

Első dolgunk a fő ablak átméretezése, legyen 500x300-as. Ezt a Property Editor menüpont alatt tudjuk módosítani, amely a tervező ablak jobb oldalán található. Most görgessünk le a tulajdonság szerkesztő listamezőjén egészen a 'windowTitle'-ig. Változtassuk meg a szöveget 'MainWindow'-ról 'Python Test1'-re és a tervező ablakunk címsora meg fog változni 'Python Test1 - untitled*'-re. Mentsük most el a projektünket, legyen a neve 'pytest1.ui'. A folytatásban egy gombot helyezünk el az űrlapunkon. Ez egy kilépés gomb lesz, megnyomásával leállítjuk a teszt-prog- ramunkat. A tervező ablak bal oldalán az összes elérhető vezérlőt megtaláljuk. Keressük meg a 'Buttons' szekciót és húzzuk be a 'Push Button' vezérlőt az űrlapra. Az általunk korábban használt GUI tervezőkkel ellentétben a QT4 tervező- ben vezérlő gombjainknak nincs szükségük rácsra. Mozgassuk a gombot az űrlap közepére-aljára. A geometry alatti Property Editort megnézve valami ilyet fogunk látni:

```
[(200,260), 97x27]
```



A zárójelekben az objektum (ebben az esetben egy nyomógomb) X és Y pozíciója szerepel, ezt annak szélessége és magassága követi. Én átállítottam 200,260-ra.

E fölött az objectName tulajdonság szerepel - ami alapértelmezett esetben 'pushButton'-ra van állítva. Változtassuk ezt meg 'btn Exit'-re. Most görgessünk le a Property Editor listában a 'QAbstractButton' részig és a 'text' tulajdonságot állítsuk 'Exit'-re. Az űrlapon látható, hogy a gombon lévő szöveg megváltozott.

Most hozzunk létre egy másik gombot és helyezzük a 200,200 pozícióba. Változtassuk meg az objectName-et 'btnClickMe'-re, a szöveget pedig 'Click Me!'-re.

Most adjunk hozzá egy címkét. Ezt a bal oldali toolbox menü 'DisplayWidgets' pontján tehetjük meg. Helyezzük el az űrlap közepe környékén (pl. 210, 130) és az objectName legyen lblDisplay. Változtassuk meg a méretét nagyobbra, mondjuk 221 x 20-ra. A property editorban görgessünk le a 'Qlabel'

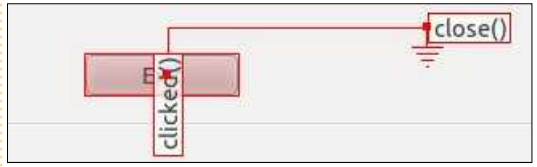
részig és állítsuk be a vízszintes igazítást 'AlignHCenter'-re. A szöveg mezőt hagyjuk üresen, ezt a kódban majd később állítjuk be - a btnClickMe megnyomásakor. Mentsük el a projektet újra.

Slots & Signals

A következő rész talán egy kicsit nehezen lesz érthető, főleg ha régóta olvasod már az írásaimat és használtál már más GUI tervezőket. Máshol az események akkor keletkeztek, ha egy objektumra rákattintottunk, mint például egy gombra. A QT4 tervezőben az eseményeket "Signal"-nak nevezik és a függvény, amit a signal meghív a Slot. Tehát a Kilépés gombunk esetében a Click signal-t használjuk a Main Window Close slot meghívásához. Sikerült teljesen összezavarnom téged? Amikor először használtam QT-t, én sem értettem de egy idő után rájöttem, hogy van mögötte értelem.

Az előre definiált slotok és signalok szerencsére nagyon könnyen használhatóak. A billentyűzet F4 gombjának megnyomásával az Edit Signals & Slots módba kerülünk (innen az F3 lenyomásával tudunk kilépni. Most tartsuk lenyomva az Exit gombot, mozgassuk kicsit fölfelé és jobbra az űrlapon és engedjük el. Egy párbeszédablak fog felugrani, valami ehhez hasonló:

Ezzel a kattintás jelét könnyen az űrlaphoz kapcsolhatjuk. Válasszuk bal oldalon az első opciót: 'clicked()'. Ezzel az ablak jobb oldala is aktívvá válik, itt válasszuk a 'close()' opciót, majd kattintsunk az



'OK'-ra. Az alábbiakat kapjuk:

A kattintás signal (esemény) tehát összekapcsolódik a fő ablakunk Close függvényével.

A btnClickMe összekapcsolását programkódban oldjuk meg.

Mentsük el a fájlt újra. Lépjünk ki a QT4 Designer-ből és nyissunk egy terminált. Lépjünk be abba a könyvtárba, ahová az előző fájlt mentettük. Most a pyuic4 segítségével legeneráljuk a python fájlt. Ez olvasni fogja a .ui fájlt. A kiadandó parancs így néz ki:

```
pyuic4 -x pytest1.ui -o pytest1.py
```

A -x paraméter a kód lefuttatásához és az UI megjelenítéséhez kell. A -o azt mondja, hogy az eredmény egy fájlba, és ne csak egyszerűen a standard kimenetre (stdout) kerüljön. Mielőtt a python fájlt lefuttatnánk győződjünk meg arról, hogy a QT4 Designerbe mindent beleírtunk, amire szükségünk van. A fájl ugyanis teljesen át fog íródni és ha utólag akarunk rajta módosítani, akkor kezdhetjük az egészet a nulláról.

Ha ezzel megvagyunk, a python fájlunk elkészült. Nyissuk meg a kedvenc szövegszerkesztőnkkel.

A fájl maga csak 65 sor, a kommenteket is beleértve. Nem szeretném az egész kódot kivesézni most, a nagy részét most már valószínüleg magadtól is meg tudod érteni. Kisebb módosítsokat azért végre fogunk még rajta hajtani.

Az első teendőnk a signal & slot sor másolása és módosítása. A 47. sor környékén valami ilyesmi kell, hogy szerepeljen:

```
QtCore.QObject.connect(self.btnExit,
QtCore.SIGNAL(_fromUtf8("clic ked()")), MainWindow.close)
```

Másoljuk át ezt a részt rögtön az eredeti alá, majd hajtsuk rajta végre az alábbi módosításokat:

```
QtCore.QObject.connect(self.btnClickMe,
QtCore.SIGNAL(_fromUtf8("clic ked()")),
```

```
self.SetLabelText)
```

Ezzel létrehozzuk a signal/slot kapcsolatot a saját függvényünkkel, amely beállítja majd a címke szövegét. A retranslateUi rutin alá írjuk be a következőt:

```
def SetLabelText(self):

self.lblDisplay.setText(_fromUtf8("That Tickles!!!"))
```

A címke setText információt a setupUi rutin kezdősorából nyertem ki.

Most futtasd le a kódot. Elvileg minden úgy működik, ahogy azt vártuk.

Ez egy NAGYON egyszerű példa ugyan, de biztos vagyok benne, hogy elég képzett vagy már a QT4 Designerrel való további játékra és hogy látod ez az eszköz miért rendkívül nagy erejű.

A következő hónapban visszatérünk az előző számok témájához és újra a TVRage programunkkal fogunk játszani.

Mint mindig, a kód most is megtalálható pastebin-en. Az ui kód itt: http://pastebin.com/98fSasdb, a python kód pedig itt: http://pastebin.com/yC30B885.

Találkozzunk legközelebb!

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.

Az én rövid történetem

írta: Anthony Venable

Valamikor 2010 elején nagyon rossz volt az anyagi helyzetem. Muszáj volt egy ingyenes operációs rendszert találnom, ami az otthoni gépeimen futott volna. Keresgéltem a neten, de sokáig nem találtam semmi használhatót. Aztán egyik nap elmentem a Barnes and Noble-ba és találtam ott egy Linuxos magazint. (Hallottam már korábban a Linuxról, de azt hittem, soha az életben nem lennék képes használni.) Megkérdeztem olyanokat, akikről tudtam, hogy számítógép-szakértők, és ők mind azt mondták, hogy a Linux a kockáknak való, jobb, ha nem is foglalkozom vele, mert nehéz. Egyszer nem mondott róla senki semmi pozitívumot. Őszintén szólva, meglepődtem, hogy korábban nem jutott az eszembe.

Elolvastam a magazint, így ismerkedtem meg az Ubuntu 9.10 Karmic Koalával. Nagyon tetszett az elnevezése, tisztára olyan volt, mintha mindig is erre vágytam volna. Izgatottan vittem haza és nagy meglepetésemre olyan könnyen sikerült feltelepítenem a gépemre, hogy úgy döntöttem, együtt fog futni a Windows XP-vel, dual boot rendszerben. Semmi mást nem csináltam, csak betettem a live CD-t a meghajtóba, követtem lépésről lépésre az instrukciókat. Nagyon lassúnak és körülményesnek kell lenni ahhoz, hogy az ember ne értse a dolgok miértjét.

Azóta összességében nagyon meg vagyok elégedve az Ubuntuval és kipróbáltam már újabb verziókat is, mint pl. a 10.04 (Maverick Meerkat) és a 10.10 (Lucid Lynx). Mindig kíváncsian vártam az újabb kiadásokat, hogy lássam, hogyan integrálódik a multi-touch a 10.04-es változathoz képest.

Ez is azt mutatja, hogy az ember a legklasszabb dolgokat sokszor véletlenül fedezi fel.

Hogyanok

Programozzunk Pythonban – 45. rész

Írta: Greg Walters

Ez alkalommal az adatbázisos programunkat fogjuk tovább csiszolgatni, melyet korábban (a 41., a 42. és a 43. részben) készítettünk. A következő néhány számban QT-t fogunk alkalmazni a felhasználói felület elkészítéséhez.

Először nézzük meg, hogy a már létező alkalmazásunk hogyan működik. Íme egy rövid áttekintés róla:

- kapcsolat létrehozása az adatbázissal – szükség esetén az adatbázis létrehozása
- sormutató létrehozása az adatbázishoz
- tábla létrehozása (ha még nem létezik)
- a videó mappa/mappák hozzárendelése egy változóhoz
- videó fájlok keresése a mappákban
- fájlnév, sorozatcím, évad, sorszám, epizódszám azonosítása
- ellen rzés az adatbázisban: az adott epizód szerepel-e már benne
- ha nem, hozzáadjuk az adatbázishoz „-1”-es TvRage ID-vel
- adatbázis újbóli átnézése, show id és állapot megadása szükség ese-tén és az adatbázis frissítése.

Át fogjuk tervezni az adatbázisunkat, hozzáadunk egy új táblát, és a már meglévő adattáblákat is módosítjuk. Először létrehozzuk az új táblát Series névvel, mely minden, a rendszerünkben elérhető tv sorozat adatait tartalmazza majd. Az alábbi mezők fognak benne szerepelni:

- Pkid
- Sorozat címe
- TvRage Sorozat ID
- Évadok száma
- Kezdés dátuma
- „Ended Flag”
- Származási ország
- Sorozat állapota (véget ért, jelenleg fut, stb.)
- Besorolás (megírt, „reality”, stb.)
- Sorozat összefoglaló
- M fajok
- Sorozat hossza (percben)
- Hálózat/Csatorna
- Megjelenés napja
- Megjelenés id pontja
- Elérési út a sorozathoz

Új táblánk létrehozásához használhatjuk a MakeDataBase rutint. A már meglévő kód elé írjuk be a jobbra látható sorokat.

```
sql = 'CREATE TABLE IF NOT EXISTS Series (
        pkid INTEGER PRIMARY KEY AUTOINCREMENT,
        SeriesName TEXT,
        SeriesID TEXT,
        Seasons TEXT,
        StartDate TEXT,
        Ended TEXT,
        OriginCountry TEXT,
        Status TEXT,
        Classification TEXT,
        Summary TEXT,
        Genres TEXT,
        Runtime TEXT,
        Network TEXT,
        AirDay TEXT,
        AirTime TEXT,
        Path TEXT);'
    cursor.execute(sql)
```

Az SQL utasításnak („sql = …”) egy sorban kellene lennie, de a könnyebb érthetőség kedvéért itt most tördeltem. A meglévő tábla módosítását későbbre hagyjuk.

Most módosítsuk a WalkThePath függvényt úgy, hogy a sorozat neve és az elérési út bekerüljön a Sorozat táblába.

Módosítsuk az alábbi sort:

```
sqlquery = 'SELECT
count(pkid) as rowcount from
TvShows where Filename =
"%s";' % fl
```

erre:

```
sqlquery = 'SELECT
count(pkid) as rowcount from
series where seriesName =
"%s";' % showname
```

Emlékeztetőül: ezzel azt ellenőrizzük, hogy a sorozat bekerült-e már a táblába. Most találjuk meg az alábbi két sort:

```
sql = 'INSERT INTO TvShows
(Series,RootPath,Filena-
me,Season,Episode,tvrageid)
VALUES (?,?,?,?,?,?)'

cursor.execute(sql,(showna-
me,root,fl,season,episode,-
1))
```

és cseréljük le ket erre:

```
sql = 'INSERT INTO Series
(SeriesName,Path,SeriesID)
VALUES (?,?,?)'

cursor.execute(sql,(showna-
me,root,-1))
```

Ezzel a sorozat neve (showname), az elérési útvonal a sorozathoz, valamint egy „-1” mint TvRage ID kerül be a táblánkba. A „-1”-et jelzőként használjuk, ez mutatja, hogy a sorozatról további információkra van szükségünk a TvRage-től.

Most változtassunk a WalkTheDatabase függvényen úgy, hogy az adott sorozatok (ahol a SeriesID = -1) hiányzó adatait lekérdezze és beírja a táblánkba.

Változtassuk meg a lekérdezés karakterláncát erről:

```
sqlstring = "SELECT DISTINCT
series FROM TvShows WHERE
tvrageid = -1"
```

erre:

```
sqlstring = "SELECT pkid,Se-
riesName FROM Series WHERE
SeriesID = -1"
```

Ez egy olyan eredményhalmazt ad, melyet minden sorozatnál felhasználhatunk a TvRage lekérdezésekhez. Most cseréljük le az alábbi két sort:

```
seriesname = x[0]

searchname = string.cap-
words(x[0]," ")
```

erre:

```
pkid = x[0]

seriesname = x[1]

searchname = string.cap-
words(x[1]," ")
```

A pkID-t a frissítés állapotára fogjuk használni. Módosítanunk kell az UpdateDatabase függvény meghívásán is, hogy benne legyen a pkid. Újabb cserékre lesz szükség, őt:

```
UpdateDatabase(seriesname,id)
```

erre:

```
UpdateDatabase(seriesna-
me,id,pkid)
```

és őt:

```
GetShowStatus(seriesname,id)
```

erre:

```
GetShowData(seriesna-
me,id,pkid)
```

```
def GetShowData(seriesname,id,pkid):
    tr = TvRage()
    idcursor = connection.cursor()
    dict = tr.GetShowInfo(id)
```

```
    seasons = dict['Seasons']
    startdate = dict['StartDate']
    ended = dict['Ended']
    origincountry = dict['Country']
    status = dict['Status']
    classification = dict['Classification']
    summary = dict['Summary']
```

Ezt az új függvényt hamarosan megírjuk.

Előbb azonban változtassuk meg az UpdateDatabase függvényt erről:

```
def UpdateDatabase(seriesna-
me,id):
```

erre:

```
def UpdateDatabase(seriesna-
me,id,pkid):
```

A lekérdezésen is módosítanunk kell:

```
    sqlstring = 'UPDATE tvs-
hows SET tvrageid = ' + id +
' WHERE series = "' + series-
name + '"'
```

helyett legyen:

```
    sqlstring = 'UPDATE Se-
ries SET SeriesID = ' + id +
' WHERE pkID = %d' % pkid
```

Most szükségünk van a GetShowData függvényre. Kiszedjük az információt a TvRage-ből és elrakjuk a Series táblánkba.

Újabb emlékeztető: a TvRage függvények egy példányát hozzuk létre, valamint egy szótárt, ez tartalmazza majd a sorozataink adatait. Ezután jöhetnek a változók, melyekből később az adatokat a táblába írjuk.

Ne feledjük, hogy a műfajok (Genres) alelemként szerepelnek és egy vagy több műfaj is megjelenhet a felsorolásban. Szerencsére a TvRage függvényeket úgy írtuk meg, hogy a megfelelő karakter-

lánc tetszőleges mennyiségű műfajt tartalmazhat:

```
genres = dict['Genres']

runtime = dict['Runtime']

network = dict['Network']

airday = dict['Airday']

airtime = dict['Airtime']
```

Végül a lekérdezés-sztringet is létrehozzuk, amely elvégzi a frissítést. Ezt is valójában egy sorba kellene írni, de a megértés könnyítésére ismét csak tördeltem a sort.

A {szám} rész (ismét emlékeztetőül) olyan, mint a "%s" formázási opció. A lekérdezési karakterláncban a {szám} helyett az a valódi adat fog látszani, melyet szeretnénk. Ezeket a mezőket már korábban definiáltuk mint szöveg, ezért idézőjelbe rakjuk őket.

Végül pedig beírunk az adatbázisba.

Ennyi lenne mára. Legközelebb innen folytatjuk. Addig is minden jót!

Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta foglalkozik programozással. Szeret főzni, túrázni, kedveli a zenét és szívesen tölti idejét családjával. Honlapja: www.thedesignatedgeek.net.

```
    try:
        idcursor.execute(sqlstring)
    except:
        print "Error Adding Series Information"
```

```
sqlstring = 'Update Series SET Seasons = "{0}", StartDate = "{1}", Ended = "{2}",
OriginCountry = "{3}", Status = "{4}", Classification = "{5}",
Summary = "{6}", Genres = "{7}", Runtime = "{8}", Network = "{9}",
AirDay = "{10}",AirTime = "{11}"
WHERE pkID ={12}'.format(seasons,startdate,ended,
origincountry,status,classification,summary,
genres,runtime,network,airday,airtime,pkid)
```

Programozzunk Pythonban – 46. rész

A cikkeim általában hosszúak, egészségügyi okokból most azonban elég rövid írással sikerült előrukkolnom. Mindenesetre ma a médiakezelő programunkat fogjuk továbbfejleszteni.

A program – többek között – azt fogja tudni, hogy jelzi, ha az adatbázisunkban bármely sorozatból van egy hiányzó epizód. Nézzünk egy konkrét példát. Van egy sorozat, legyen mondjuk a „That 80's Show", amely három évadot élt meg. A 2. évadban 15 rész volt, nekünk mégis csak 13 van meg belőle a gyűjteményünkben. Hogyan derítsük ki – programozóként –, hogy melyik részek hiányoznak?

A legegyszerűbben listákkal és set-ekkel. Listákhoz már több alkalommal is volt szerencsénk az elmúlt néhány évben, set adattípust viszont még nem használtunk, így ma ezt a kérdéskört fogjuk megvizsgálni. A Python „hivatalos dokumentációja" szerint (docs.python.org) a set definíciója:

„A set egy olyan rendezetlen gyűjtemény, amelyben az elemek nem ismétlődhetnek. Használható például tagság tesztelésére vagy ismétlődő bejegyzések kiszűrésére. A set objektumok az olyan matematikai műveleteket is támogatják, mint az unió, metszet, különbség vagy szimmetrikus különbség."

Egy, a dokumentációban is szereplő példán szeretném bemutatni, ez hogyan is működik.

```
>>> Basket =
['apple','orange','apple','pe
ar','orange','banana']

>>> fruit = set(basket)

>>> fruit

set(['orange','pear','apple',
'banana'])
```

Vegyük észre, hogy a basket változóban eredetileg kétszer szerepelt az alma (apple) és a narancs (orange), de amint ebből set-et készítettünk, a másolatok eltűntek. Az így kapott set-et vizsgálva könnyen kideríthetjük, hogy egy adott elem része e a gyümölcsnek (fruit), vagy bármilyen más set-nek. Erre szolgál az „in" operátor.

```
>>> 'orange' in fruit

True

>>> 'kiwi' in fruit

False

>>>
```

Ez ugye egyszerű és gondolom már látod, hogy hová akarok kilyukadni. Tegyük fel, hogy van egy bevásárlólistánk, amelyen szerepel sok gyümölcs is. A boltban állva szeretnénk megtudni, mi hiányzik még a kosarunkból. Mondjuk így:

```
>>> shoppinglist =
['orange','apple','pear','ban
ana','kiwi','grapes']

>>> basket =
['apple','kiwi','banana']

>>> sl = set(shoppinglist)

>>> b = set(basket)

>>> sl-b

set(['orange', 'pear', 'grapes'])

>>>
```

Létrehozzuk a két listánkat: a bevásárlólistát (shoppinglist) és a kosarunk tartalmát (basket). Mindkettőből egy set-et hozunk létre és alkalmazzuk rájuk a különbség operátort (mínusz jel). Eredményül azokat az elemeket fogjuk kapni, amelyek rajta vannak a bevásárlólistán, de nincsenek a kosárban.

Ugyanezt a logikát használva egy olyan függvényt hozunk létre, amely a hiányzó epizódjainkat találja meg, legyen ennek a neve „FindMissing" és kapjon két változót. Az első egy egész szám, amely megadja a hiányzó epizódok számát az évadban, a második pedig egy lista a hiányzó epizódok sorszámaival.

A rutin lefutása után az alábbi eredményt kapjuk: [5, 8, 15]. Most nézzük meg a kódot. Az első sor létrehozza az EpisodesNeeded set-et, amely egész számok listája a megadott tartományon belül. Ne feledjük, hogy a range függvényünk 0-tól indul, így amikor 16-ot adunk meg neki (expected (=15) + 1), a lista valójában 0-tól 15-ig megy. A range függvénynek azt is meg-

mondjuk, hogy 1-ről induljon, így a 16 elemű lista hiába kezdődik 0-ról, mi csak 15 értékét fogjuk használni (1-től 15-ig).

Ez után létrehozunk egy set-et abból a listából, amit a have változón keresztül megadtunk a függvényünknek, azaz a meglévő epizódok listáját.

Nem maradt más dolgunk, előállítjuk a keresett listát a kettő különbségéből, majd a list.sort() segítségével sorba rendezzük. Megadhatjuk az eredményt, mint a fügvény visszatérési értékeként is, de egyelőre csak kiíratjuk az eredményt a képernyőre.

Erre a hónapra sajnos csak ennyi tellett tőlem. Remélem, hogy az itt elmondottakat fel fogjuk tudni használni a médiakezelő programunkban.

Minden jót, találkozzunk legközelebb!

Greg Walters a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa és 1972 óta foglalkozik programozással. Szeret főzni, túrázni, kedveli a zenét és szívesen tölti idejét családjával. Honlapja: www.thedesignatedgeek.net.

```
def FindMissing(expected,have):
    #=================================
    # 'expected' is the number of episodes we should have
    # 'have' is a list of episodes that we do have
    # returns a sorted list of missing episode numbers
    #=================================
    EpisodesNeeded = set(range(1,expected+1))
    EpisodesHave = set(have)
    StillNeed = list(EpisodesNeeded - EpisodesHave)
    StillNeed.sort()
    print StillNeed

FindMissing(15,[1,2,3,4,6,7,9,10,11,12,13,14])
```

PYTHON SPECIAL EDITIONS:

http://fullcirclemagazine.org/issue-py01/

http://fullcirclemagazine.org/issue-py02/

http://fullcirclemagazine.org/python-special-edition-issue-three/

http://fullcirclemagazine.org/python-special-edition-volume-four/

http://fullcirclemagazine.org/python-special-edition-volume-five/

http://fullcirclemagazine.org/python-special-edition-volume-six/

Programozzunk Pythonban – 47. rész

A múlt hónapban a halmazok használatáról beszéltünk, hogy megmutassák nekünk a hiányzó epizódok számait. Most itt az ideje, hogy a múltkori nyers kódot a gyakorlatba ültessük.

Egy rutint fogunk módosítani, egyet pedig mi fogunk megírni. Először a módosítást végezzük el. A munkaállományban, amelyet az utóbbi néhány hónapban használtál, keresd meg a WalkThePath(filepath) rutint. A negyedik és ötödik sornak így kell kinéznie:

```
efile = open('errors.log',"w")
for root, dirs, files in os.walk(filepath,topdown=True):
```

E két sor közé illesztjük be a következő kódot:

```
lastroot = ''
elist = []
currentshow = ''
currentseason = ''
```

Mostanra már fel kellett ismerned, hogy amit itt teszünk, az a változók inicializálása. Három karakterlánc típusú változó és egy lista van. A listát arra fogjuk használni, hogy az epizódok számait tároljuk (innen származik az elist név).

Vessünk egy pillantást arra, és frissítsük fel a memóriánkat, hogy mit csinálunk a meglévő rutinban, mielőtt bármilyen további módosítást végeznénk.

Itt az első két sor beállítja a dolgokat a walk-the-path rutinhoz, amivel elindulunk egy adott mappából a fájlrendszerben, és rekurzívan minden, az alatt lévő mappát végiglátogatunk, olyan fájlokat keresve, melyek kiterjesztése .avi, .mkv, .mp4 vagy .m4v. Ha van ilyen, akkor ezután ismételten végiglépegetünk ezeknek a fájlneveknek a listáján.

A fenti kód meghívja a GetSeasonEpisode rutint, hogy lekérdezzük a sorozat nevét, az évad számát és az epizód számát a fájlnévből. Ha mindent helyesen dolgoz fel, az isok változót igazra állítja, és az

```
for root, dirs, files in os.walk(filepath,topdown=True):
    for file in [f for f in files if f.endswith (('.avi','mkv','mp4','m4v'))]:
```

```
# Összef  zi az elérési utat és a fájlnevet egy változóba.
fn = join(root,file)
OriginalFilename,ext = os.path.splitext(file)
fl = file
isok,data = GetSeasonEpisode(fl)
```

adatokat, amelyeket keresünk, beteszi egy listába, azután pedig visszaadja nekünk.

Itt egyszerűen vesszük a GetSeasonEpisode rutinból visszakapott adatokat, és külön változókba tesszük őket, amelyekkel eljátszhatunk. Most, hogy tudjuk, hol voltunk, beszéljünk arról, hogy hová tartunk.

Meg akarjuk kapni az egyes fájlok epizódszámát, és beletenni az elist listába. Ha az összes fájllal végeztünk abban a mappában, ahol jelenleg vagyunk, akkor feltételezhetjük, hogy egész jól lépést tartunk a fájlokkal, és a legmagasabb számozású a legfrissebben elérhető epizód. Ahogy a múlt hónapban megtárgyaltuk, ezután létrehozhatunk egy halmazt, amelyet 1-től az utolsó epizódig számozunk, majd a listát is halmazzá konvertálhatjuk, és lekérdezhetjük a különbséget. Bár ez elméletben nagyszerű, egy kicsit sántít, amikor átültetjük a tényleges gyakorlatba. Valójában nem kapunk szép és egyértelmű utalást arra, hogy mikor vagyunk készen bármely megadott mappával. Amink mégis megvan, az annak az ismerete, hogy amikor végzünk az egyes fájlokkal, a „for file in [...” utáni kód rögtön futni fog. Ha tud-

```
if isok:
    showname = data[0]
    season = data[1]
    episode = data[2]
    print("Season {0} Episode {1}".format(season,episode))
```

juk az utoljára meglátogatott és a jelenlegi mappa nevét, összehasonlíthatjuk őket, és ha különböznek, akkor végeztünk a mappával és az epizódlistánk készen van. Ez az, amire a „lastroot” változó való.

Éppen a „for file in[” sor után van az a pont, ahová az új kódunk nagy részét beszúrjuk. Ez csak hét sorból áll. Íme a hét sor. (A vastagbetűs sorok a meglévő sorok, az egyszerűség kedvéért.)

Az új kódot sorról-sorra véve, íme a logika:

Először megnézzük, hogy a lastroot változónak ugyanaz-e az értéke, mint a root változónak (a jelenlegi mappa neve). Ha igen, akkor ugyanabban a mappában vagyunk, így nem futtatunk semmilyen kódot. Ha nem, akkor átadjuk a jelenlegi mappa nevét a lastroot változónak. Ezután megnézzük, hogy az epizódlistában (elist) van-e bármilyen bejegyzés(len(elist) > 0). Ez arra való, hogy megbizonyosodjunk afelől, hogy nem üres mappában voltunk. Ha vannak elemeink a listában, akkor meghívjuk a Missing rutint. Átadjuk az epizódlistát, a legmagasabb epizódszámot, a jelenlegi évad számát és nevét, hogy ezeket később megjeleníthessük a

```
    for file in [f for f in files if f.endswith (('.avi','mkv','mp4','m4v'))]:
        # Összef  zi az elérési utat és a fájlnevet egy változóba.
        if lastroot != root:
            lastroot = root
            if len(elist) > 0:
                Missing(elist,max(elist),currentseason,currentshow)
            elist = []
            currentshow = ''
            currentseason = ''
        fn = join(root,file)
```

képernyőn. Az utolsó három sor törli a listát, a jelenlegi műsor nevét és a jelenlegi évadot, és megyünk tovább, ahogy eddig.

Ezután módosítanunk kell két sort, és hozzá kell adni egy sort az if isok: kódhoz (itt jobbra). Ismét a vastagbetűs sorok a meglévők:

Itt épp csak visszatértünk a GetSeasonEpisode rutinból. Ha van egy feldolgozható fájlnevünk, meg akarjuk kapni a bemutató nevét és az évad számát, illetve hozzá akarjuk adni a jelenlegi epizódot a listához. Vedd észre, hogy az epizód számát integer típusúvá konvertáljuk, mielőtt hozzáadjuk a listához.

```
        isok,data = GetSeasonEpisode(fl)
        if isok:
            currentshow = showname = data[0]
            currentseason = season = data[1]
            episode = data[2]
            elist.append(int(episode))
        else:
```

Kész is vagyunk a kódnak ezzel a részével. Most már csak hozzá kell adnunk a Missing rutint. Közvetlenül a WalkThePath rutin után a következő, lent látható kódot szúrjuk be.

Ez is egy nagyon egyszerű kódhalmaz, és egész jól átnéztük a múlt hónapban, de a biztonság kedvéért vegyük végig, hátha kimaradtál belőle.

Definiáljuk a függvényt és beállítunk négy paramétert. Átadjuk az epizódlistát (eplist), az epizódok várható számát (shouldhave), amely az epizódlistában lévő legmagasabb epizódszám, továbbá az évad számát (season) és a műsor nevét (showname).

Ezután létrehozunk egy halmazt a beépített range függvény használatával, amely tartalmaz egy számlistát, 1-től a shouldhave + 1 értékig. Majd meghívjuk a difference függvényt – erre a halmazra és

```
#---------------------------------
def Missing(eplist,shouldhave,season,showname):
    temp = set(range(1,shouldhave+1))
    ret = list(temp-set(eplist))
    if len(ret) > 0:
        print('Missing Episodes for {0} Season {1} - {2}'.format(showname,season,ret))
```

az epizódlistából konvertált halmazra (temp-set(eplist)) – és visszakonvertáljuk azt listává. Majd ellenőrizzük, hogy van-e valami a listában – így nem jelenítünk meg üres listát tartalmazó sort, ha pedig van ott valami, azt megjelenítjük.

Ennyi. Az egyetlen hiba ebben a logikában az, hogy ha így dolgozunk, nem tudjuk meg, hogy van-e olyan új epizód, ami még nincs meg nekünk.

Feltettem neked a két rutint a pastebin oldalra, ha csak egy gyors cserét akarnál végezni a működő kódodon. Ezt megtalálhatod a http://pastebin.com/XHTRv2dQ címen.

Legyen jó (hó)napod, és hamarosan találkozunk!

Greg Walters a RainyDay Solutions, LLC, azaz a Colorado állami Aurora településen működő tanácsadó vállalat tulajdonosa, és 1972 óta foglalkozik programozással. Imád főzni, hegyet mászni, imádja a zenét és a családjával való időtöltést. A webcíme 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

Programozzunk Pythonban gyűjtemény:

http://fullcirclemagazine.org/issue-py01/

http://fullcirclemagazine.org/issue-py02/

http://fullcirclemagazine.org/python-special-edition-issue-three/

http://fullcirclemagazine.org/python-special-edition-volume-four/

http://fullcirclemagazine.org/python-special-edition-volume-five/

http://fullcirclemagazine.org/python-special-edition-volume-six/

Programozzunk Pythonban – 48. rész

Üdvözlök újra mindenkit. Nehéz elhinni, hogy 4 éve annak, hogy ezt a sorozatot elkezdtem. Arra gondoltam, hogy jegelem egy kicsit a média kezelő feladatot és visszanyúlok néhány Python programozási alaphoz.

Ebben a hónapban a print parancsot fogom feleleveníteni. Ez a leginkább használt függvények egyike (legalábbis nálam), ami talán sosem kapja meg a kellő figyelmet, amit megérdemelne. Rengeteg dolog van még amire használható az alap „%s %d”-n kívül.

A Python 2.x-ben és a Python 3.x-ben különbözik a print függvény szintaxisa, ezért külön fogjuk őket tárgyalni. Emlékezz viszont, hogy a 3.x szintaxist a Python 2.7-ben is használhatod. A legtöbb dolog amit ebben a hónapban bemutatok, az az interaktív shellben lesz végrehajtva. Nyugodtan követheted te is a mi példánkat. A kód így fog kinézni:

```
>>> a = "Hello Python"

>>> print("String a is %s" % a)
```

a kimenet pedig vastagon lesz szedve valahogy így:

```
String a is Hello Python
```

Python 2.x

Nyílván emlékszel, hogy a 2.x-ben a print függvény egyszerű szintaxisa a %s vagy a %d változó behelyettesítést használja az egyszerű stringekhez vagy számokhoz. Azonban sok más formázási lehetőség is rendelkezésre áll. Például ha bevezető nullákkal kell formáznod egy számot, akkor azt így teheted:

```
>>> print("Your value is %03d" % 4)
Your value is 004
```

Ebben az esetben a „%03d” formázó parancsot használjuk ami azt jelenti, hogy „Jelenítsd meg a számot 3 karakter szélesen és ha szükséges töltsd fel balról nullákkal.

```
>>> pi = 3.14159

>>> print('PI = %5.3f.' % pi)

PI = 3.142.
```

Itt a float formázó parancsot használjuk. A „%5.3” azt mondja meg, hogy a kimenet teljes hossza 5 legyen 3 tizedes hellyel. Vedd észre, hogy a tizedes pont egy helyet elfoglal a teljes szélességből.

Egy másik dolog, amivel lehet nem voltál eddig tisztában, az az, hogy a formázó parancs részeként egy dictionary kulcsait is felhasználhatod.

```
>>> info = {"FName":"Fred","LName":"Farkel","City":"Denver"}

>>> print('Greetings %(FName)s %(LName)s of %(City)s!' % info)

Greetings Fred Farkel of Denver!
```

Konvertálás	Jelentés
'd'	Előjeles egész decimális
'i'	Előjeles egész decimális
'u'	Elavult – azonos a 'd'-vel
'o'	Előjeles oktális érték
'x'	Előjeles hexadecimális – kisbetű
'X'	Előjeles hexadecimális – nagybetű
'f'	Lebegőpontos decimális
'e'	Lebegőpontos exponential – kisbetű
'E'	Lebegőpontos exponential – nagybetű
'g'	Lebegőpontos forma – kisbetűs exponenciális formát használ, ha a kitevő kisebb mint -4 vagy nem kisebb mint a pontosság, máskülönben decimális forma.
'G'	Lebegőpontos forma – nagybetűs exponenciális formát használ ha a kitevő kisebb mint -4 vagy nem kisebb mint a pontosság, máskülönben decimális forma.
'c'	Egyetlen karakter
'r'	String (érvényes Python objektumot a repr() használatával konvertál)
's'	String (érvényes Python objektumot a str() használatával konvertál)
'%'	Nincs konvertált argumentum, egy „'%” karakter az eredménye

A következő tábla bemutatja a különböző lehetséges helyettesítő karaktereket és értelmezésüket.

PYTHON 3.X

Python 3.x-ban sokkal több opciónk van (emlékezz, hogy a Python 2.7-ben is használhatjuk ezeket), amikor a print függvény kerül terítékre.

Emlékeztetőül itt van egy egyszerű példa a 3.x print függvényére.

```
>>> print('{0}
{1}'.format("Hello","Python"))
Hello Python

>>> print("Python is {0}
cool!".format("WAY"))
Python is WAY cool!
```

A helyettesítő mezők kapcsos zárójelben vannak: „{" „}". Bármi más ezeken kívül szó szerintinek tekintendő és módosítás nélkül jelenik meg. Az első példában megszámoztuk a helyettesítő mezőket 0-val és 1-el. Ez azt mondja meg a Pythonnak, hogy vegye az első (0) értéket és rakja bele a {0} mezőbe és így tovább. Azonban egyáltalán nem kötelező a számok használata. Ez a lehetőség azt eredményezi, hogy az első érték belekerül az első zárójel párba és így tovább.

```
>>> print("This version of {}
is {}".format("Python","3.3.2"))
This version of Python is
3.3.2
```

Ahogy a TV-reklámokban mondják „DE VÁRJON… EZ MÉG MIND SEMMI". Ha néhány inline formázást szeretnénk használni, a következő lehetőségeink vannak.

```
:<x Balra igazítás x széles-
ségben
:>x Jobbra igazítás x széles-
ségben
:^x Középre igazítás x széles-
ségben
```

Itt egy példa rá:

```
>>>
print("|{:<20}|".format("Left"))
|Left                |

>>>
print("|{:>20}|".format("Right")
)
|               Right|

>>>
print("|{:^20}|".format("Center"))
|       Center       |
```

Az igazítással/szélességgel együtt még egy kitöltő karaktert is megadhatsz.

```
>>>
print("{:*>10}".format(321.40))
*****321.4
```

Ha egy dátum-, időkimenetet szükséges formáznod, akkor valami ilyet írhatsz:

```
>>> d =
datetime.datetime(2013,10,9,10,45,
1)

>>>
print("{:%m/%d/%y}".format(d))
10/09/13

>>>
print("{:%H:%M:%S}".format(d))
10:45:01
```

Egyszerű kiírni az ezreseket vesszővel (vagy bármely más karakterrel) elválasztva.

```
>>> print("This is a big
number
{:,}".format(7219219281))
This is a big number
7,219,219,281
```

Nos, elég is lesz ennyi gondolatébresztő erre a hónapra. Találkozunk

Greg Walters 1972 óta programozik és a Colorádóban található RainyDay Solution konzultációs vállalkozás tulajdonosa. Szeret főzni, kirándulni, zenét hallgatni és szabadidejét a családjával tölteni. A weboldala www.thedesignatedgeek.net.

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 meghal lgathatja.

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

podcast.ubuntu-uk.org

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