Full Circle

AZ UBUNTU LINUX KÖZÖSSÉG FÜGGETLEN MAGAZINJA

Programozói sorozat – Különkiadás

Programozói sorozat Különkiadás

Programozzunk Pythonban

7. 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” 39–43. részének** az újabb kiadása (a magazin 68–72. 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

Programozzunk Pythonban – 39. rész

Írta: Greg Walters

Sok-sok hónappal ezelőtt egy Weather Underground nevű API-val dolgoztunk. Ez még a 11. részben történt, ami az FCM 37. számában jelent meg. Nos, most újra egy API-hoz nyúlunk, ez alkalommal a TVRage (http://tvrage.com) weboldal API-jához. Amennyiben nem ismernéd az oldalt, tulajdonképpen TV showkal (műsorokkal) foglalkozik. Eddig még nem találtam olyan műsort, amely ne lenne benne a rendszerükben. A következő cikkekben szó lesz XML-ről, API-król és Element-Tree-ről, (ezek segítségével átalakító programkönyvtárakat fogunk létrehozni), a cél pedig egy olyan saját kis programkönyvtár létrehozása, amellyel leegyszerűsíthetjük a kedvenc show műsoraink TV-s adatainak letöltését.

Szóba került az átalakító programkönyvtár. Mi is ez? Egyszerűen fogalmazva, egy átalakító programkönyvtár létrehozása vagy használata során olyan kódokat használunk, amelyek „befedik” a weboldal API-ját és egy könnyen használható függvénytárat biztosít számunkra. Mielőtt hozzákezdenénk, szeretnék tisztázni néhány dolgot. Először is, ez egy ingyenes szolgáltatás, de az API használatáért a fejlesztők adományokat kérnek. Ha úgy érzed, hogy ezért a szolgáltatásért érdemes fizetni, kérlek fontolj meg egy $10-os, vagy akár annál nagyobb összegű támogatást. Másodszor, regisztrálj a weboldalukon, így hozzájuthatsz a saját API kulcsodhoz. Ingyenes, szóval igazán semmi nem szól a regisztráció ellen, főleg, ha az itt megtalálható információt használni is szeretnéd a jövőben. Ráadásul regisztrált felhasználóként néhány további információmezőhöz is hozzájutunk, mint például a sorozatok és epizódok összefoglalóihoz, ezek a nem regisztrált változatban elérhetetlenek. Harmadszor, az API frissítésébe rengeteg energiát ölnek. Ez azt jelenti, hogy mire ezt a cikket olvasod, az API jó eséllyel már meg is változott. Mi most olyan nyilvános hírforrásokat fogunk használni, amelyek 2012 decemberében mindenki számára szabadon hozzáférhetőek voltak. Az API honlapja: http://services.tvrage.com/info.php?page=main, itt találunk néhány példát is arra nézve, hogy milyen információhoz férhetünk hozzá.

Nézzük hát meg ezt az API-t és fejtsük meg, hogyan tudjuk használni.

Az API használatával a show-val és/vagy az adott epizóddal kapcsolatos információhoz juthatunk. TV műsorok esetében az információszerzésnek tulajdonképpen három lépése van, ezek az alábbiak:

- Név alapján rákeresünk a műsorra az adatbázisban, további adatok lekérdezéséhez az így kapott egyedi azonosítót (ID) használjuk. A showid értéke egy kulcs, amely az adatbázisban egyértelműen kijelöl egy adatsort.
- A Show ID birtokában további, a műsorral kapcsolatos információhoz jutunk.
- Végül egy kiválasztott epizód adatait is lekérdezzük. Ezt egy olyan listában találjuk, amely tartalmazza az összes eddig leadott rész adatait.

A felsorolt adatokat három egyszerű webes lekérdezéssel érhetjük el. Először egy keresésre lesz szükségünk, ezután a show adatait kérjük le, végül pedig az epizód-listából választjuk ki a kedvünkre valót.

Az alábbiakban láthatók azok a függvényhívások, amelyeket használni fogunk...

- ShowID keresése a műsor neve alapján - `http://services.tvrage.com/feeds/search.php?show={SomeShow}`

- **A show adatainak lekérése Show ID (sid) alapján -** `http://services.tvrage.com/feeds/showinfo.php?sid={SomeShowID}`

- **Epizódlista lekérdezése Show ID (sid) alapján -** `http://services.tvrage.com/feeds/episode_list.php?sid={SomeShowID}`

Eredményül egy-egy XML formátumú adatfolyamot kapunk.

```
<?xml version="1.0" encoding="UTF-8" ?>
<ROOT TAG>
   <PARENT TAG>
      <CHILD TAG 1>DATA</CLOSING CHILD TAG 1>
      <CHILD TAG 2>DATA</CLOSING CHILD TAG 2>
      <CHILD TAG 3>DATA</CLOSING CHILD TAG 3>
   </CLOSING PARENT TAG>
</CLOSING ROOT TAG>
```

Emlékezzünk vissza, hogy hogyan is épül fel az XML. Az első sor mindig valahogy úgy néz ki, mint amit az alábbiakban bemutatok:

Az adat minden eleme egy definiáló és egy záró címkével (tag-gel) van ellátva. Időnként gyermek-címkékkel is találkozhatunk, ez egy szülő-címkébe ágyazott címke. Valahogy így...

```
<CHILD PARENT TAG>

<CHILD TAG 1>DATA</CLOSING
CHILD TAG 1>

</CLOSING CHILD PARENT TAG>
```

Találkozhatunk olyan címkével is, amely bizonyos jellemzővel (attribútummal) rendelkezik:

```
<TAG INFORMATION = VALUE>

<CHILD TAG>DATA</CLOSING
CHILD TAG>

</CLOSING TAG>
```

Néha egy címkéhez nincs adat hozzárendelve. Például...

```
<prodnum/>
```

Azokat a címkéket, amelyek nem tartalmaznak információt, időnként el is hagyják. A Te programodnak ezért tudnia kell kezelni ezeket az eshetőségeket.

XML adatok kezelése során tehát egy root címkével nyitunk, feldolgozzuk az összes többi címkét és ezek segítségével megtaláljuk a számunkra érdekes adatot. Előfordul, hogy egyszerűen mindenre szükségünk van, de bizonyos esetekben az összes információnak csak egy töredékét akarjuk felhasználni.

Nézzük most meg az első hívást és azt, hogy mivel tér vissza. Tegyük fel, hogy a keresett show a Buffy, a vámpírok réme. A műsorra az alábbi módon kereshetünk rá...

```
http://services.tv-
rage.com/feeds
/search.php?show=buffy
```

Az eredményül kapott XML fájl így nézne ki:
http://pastebin.com/Eh6ZtJ9N.

Megjegyzés: a sorbehúzásokat én magam eszközöltem a nyers adat olvashatóbbá tétele érdekében. Nézzük át az XML fájlt, mit is találunk benne.

<Results> - ez a címke a teljes XML adatot körbeöleli, az utolsó sorban a záró </Results> címkének kell szerepelnie. Lehet nulla, de akár ötven eredmény (result) is.
<show> - ez a szülő pont, amely megmondja, hogy „ami következik (egészen a záró show címkéig), az

egy tv műsorról szóló információ”. Az előzőekhez hasonlóan a záró címke itt is egy </show>. Bármi, ami e két címke között található, a műsorral kapcsolatos információ.
<showid>2930</show> - Ez a showid címke, amely tartalmazza az sid-t. Az sid-ot felhasználva kaphatunk információt a műsorról, ebben az esetben ez az azonosító a 2930.
<name>Buffy the Vampire Slayer</name> ez a műsor címe
<link>...</link> ez egy link lenne, amely közvetlenül a show-ra mutat a TVRage oldalon (vagy egy epizód esetén az adott résszel kapcsolatos információra)
<country>...</country> Az ország, ahol a műsort eredetileg leadták
…
</show>
</Results>

A mi programunk esetében tulajdonképpen csak két mezőre van szükség, a <showid>-re és a <name>-re. Talán érdemes figyelembe vennünk még a <started> mezőt is, főleg azért, mert ritkán kapunk vissza egyetlen találatot, főleg akkor, ha nem a műsor teljes nevére kerestünk rá. Például ha a „The Big Bang Theory”-ról szeretnénk adatokat és a kereséshez csak annyit írunk be, hogy „Big Bang”, több mint húsz adatfolyamot kapnánk vissza, hiszen minden olyan cím, amely tartalmazza a „big” és a „bang” szót, megfelel a keresési feltételeknek. Ha az „NCIS”-re keresünk rá, szintén sok találatunk lenne. Néhány olyan is, amelyre egyáltalán nem számítunk. Az „NCIS”, az „NCIS: Los Angeles”, valamint a „The real NCIS” mellett megjelenne még a „The Streets of San Francisco” és a „Da Vinci’s Inquest” is (sok egyéb műsor mellett), hiszen ezek mindegyike tartalmazza az „N” „C” „I” és „S” karaktereket, megfelelő sorrendben.

Ha már tudjuk a sorozat azonosítóját, akkor az adatokat közvetlenül az ID alapján is kérhetjük. Az eredmény nagyon fog hasonlítani a

keresésnél korábban kapott adatsorhoz, de lényegesen több részletet fog tartalmazni. Használjuk újra Buffy-t, ahogy a kereséses példában. Jobbra az XML fájl kivonatos tartalmát láthatjuk.

Láthatod, hogy az itt szereplő információ nagy része valóban benne volt a keresésnél kapott adatfolyamban. Ugyanakkor az olyan adatok, mint a hálózat (csatorna), annak országa, a futási idő, a leadás napja és időpontja korábban nem jelent meg.

Listázzuk most ki az epizódokat. Ha a műsor csak egy évadot élt meg és csak hat részből áll, ez az adatfolyam rövid lesz. Nézzük meg inkább azt, hogy mi a helyzet az egyik kedvenc tv sorozatom, a Ki vagy, doki? (Doctor Who) esetében. A Doctor Who egy brit TV sorozat, amely eredetileg 1963-ban indult és 26 szezont élt meg, egészen 1989-ig futott. Az első évad 42 epizódból állt, az ezt követő évadok pedig általában 24 részesek voltak. Feldolgozandó adatunk tehát van bőven.

Az epizódlista lekérése során a következő oldalon látható adatokat kapjuk (ismét visszatértünk Buffy-hez). Vessünk egy pillantást az adatfolyam egy részére és máris lesz egy elképzelésünk arról, hogy mivel van dolgunk.

Hadd ismételjem meg: az az információ, amire valójában szükségünk van, a sorozat címe alapján történő keresésnél a showid egyértelmű meghatározásához, az az alábbi három címkében található:

```
<showid>
<name>
<started>
```

A Show Information adatfolyamból általában az alábbi dolgokra van szükségünk:

```
<seasons>
<started>
<start date>
<origin_country>
<status>
<genres>
<runtime>
<network>
<airtime>
<airday>
<timezone>
```

Az epizód lista esetén pedig az alábbiakra:

```
<Season>
<episode number>
<season number>
<production number>
<airdate>
<link>
<title>
```

Figyelmeztetés: az évad

```
<Showinfo>
   <showid>2930</showid>
   <showname>Buffy the Vampire Slayer</showname>
   <showlink>http://tvrage.com/Buffy_The_Vampire_Slayer</showlink>
   <seasons>7</seasons>
   <started>1997</started>
   <startdate>Mar/10/1997</startdate>
   <ended>May/20/2003</ended>
   <origin_country>US</origin_country>
   <status>Canceled/Ended</status>
   <classification>Scripted</classification>
   <genres>
      <genre>Action</genre>
      <genre>Adventure</genre>
      <genre>Comedy</genre>
      <genre>Drama</genre>
      <genre>Mystery</genre>
      <genre>Sci-Fi</genre>
   </genres>
   <runtime>60</runtime>
   <network  country="US">UPN</network>
   <airtime>20:00</airtime>
   <airday>Tuesday</airday>
   <timezone>GMT-5 -DST</timezone>
   <akas>
      <aka country="SE">Buffy &amp;amp; vampyrerna</aka>
      <aka country="DE">Buffy - Im Bann der Dämonen</aka>
      <aka country="NO">Buffy - Vampyrenes skrekk</aka>
      <aka country="HU">Buffy a vámpírok réme</aka>
      <aka country="FR">Buffy Contre les Vampires</aka>
      <aka country="IT">Buffy l'Ammazza Vampiri</aka>
      <aka country="PL">Buffy postrach wampirów</aka>
      <aka country="BR">Buffy, a Caça-Vampiros</aka>
      <aka country="PT">Buffy, a Caçadora de Vampiros</aka>
      <aka country="ES">Buffy, Cazavampiros</aka>
      <aka country="HR">Buffy, ubojica vampira</aka>
      <aka country="FI">Buffy, vampyyrintappaja</aka>
      <aka country="EE">Vampiiritapja Buffy</aka>
      <aka country="IS">Vampírubaninn Buffy</aka>
   </akas>
</Showinfo>
```

(season) és az epizódszám nem feltétlenül az, amire elsőre gondolnánk. A TVRage adatok esetén az évadszám a sorozat kiválasztott részének sorszáma, az adott évadon belül. Az epizódszám az adott rész sorszáma úgy, hogy a sorozat összes korábbi évadának részeit is figyelembe vesszük. A produkciós szám egy, a készítés során használt belső azonosító, ez a legtöbb embernek semmilyen információértéket nem hordoz.

Felfrissítettük hát az XML fájlstruktúrákkal kapcsolatos emlékeinket és megismerkedtünk a TVRage API hívásaival, így most már készen állunk saját kódunk megírására, de erre a következő számig várnunk kell.

Addig is kellemes szünetet kívánok mindenkinek. (Az eredeti angol cikk 2012. decemberében jelent meg – a ford.)

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.

```
<Show>
   <name>Buffy the Vampire Slayer</name>
   <totalseasons>7</totalseasons>
   <Episodelist>
      <Season no="1">
         <episode>
            <epnum>1</epnum>
            <seasonnum>01</seasonnum>
            <prodnum>4V01</prodnum>
            <airdate>1997-03-10</airdate>
            <link>http://www.tvrage.com/Buffy_The_Vampire_Slayer/episodes/28077</link>
            <title>Welcome to the Hellmouth (1)</title>
         </episode>
         <episode>
            <epnum>2</epnum>
            <seasonnum>02</seasonnum>
            <prodnum>4V02</prodnum>
            <airdate>1997-03-10</airdate>
            <link>http://www.tvrage.com/Buffy_The_Vampire_Slayer/episodes/28078</link>
            <title>The Harvest (2)</title>
         </episode>
         <episode>
            <epnum>3</epnum>
            <seasonnum>03</seasonnum>
            <prodnum>4V03</prodnum>
            <airdate>1997-03-17</airdate>
            <link>http://www.tvrage.com/Buffy_The_Vampire_Slayer/episodes/28079</link>
            <title>Witch</title>
         </episode>
         ...
      </Season>
   </Episodelist>
</Show>
```

Hogyanok

Írta: Greg Walters

Programozzunk Pythonban – 40. rész

A legutóbbi alkalommal részletesen áttekintettük a TVRAGE web API-ját. Most olyan kódot fogunk írni, amellyel az így kapott adatokat fel is tudjuk használni.

Ebben a részben olyan újrahasznosítható modulokat fogunk írni, amelyek megkönnyítik az API elérését számunkra és a későbbiekben tetszőleges python programokba importálhatók.

A TVRAGE API jó néhány dologra felhasználható és ez fokozottan igaz a regisztrált változatra, de mi most csak három függvényhívásra koncentrálunk:
1. Műsor keresése név alapján és a ShowID lekérése
2. Műsorinformációk lekérdezése ShowID alapján
3. Epizód-specifikus információk lekérdezése ShowID alapján

A múltkor olyan API hívásokat mutattam be, amelyek regisztráció nélkül is elérhetők. Ezúttal regisztráció-köteles hívásokról lesz szó, ehhez pedig a saját regisztrációs kulcsomat fogom használni. Ezt a kulcsot megosztom ugyan veletek (a TVRAGE-t előre tájékoztattam erről), de szeretnék mindenkit arra kérni, hogy amennyiben szeretnétek használni ezt az API-t, regisztráljatok és használjátok hozzá a saját kulcsotokat, ne éljetek vissza a lehetőséggel. Kérlek fontoljátok meg a projekt (anyagi) támogatását is.

Három fő rutint készítünk a hívások végrehajtására és az információ feldolgozására, hármat az info megjelenítésére (itt élünk azzal a feltételezéssel, hogy a programot „önálló” módban használjuk) és egy fő rutint a munka elvégzésére – ismét csak feltételezve, hogy a program önálló módban fut.

Itt látható azon rutinok listája, amelyeket el fogunk készíteni (igaz, van amit csak egy későbbi alkalommal. Másnak is szeretnék helyet hagyni ebben a számban).

```
def FindIdByName(self,
showname, debug = 0)

def GetShowInfo(self, showid,
debug = 0)

def GetEpisodeList(self,
showid, debug = 0)

def DisplaySearchResult(self,
ShowListDict)


def DisplayShowInfo(self,
dict)

def DisplayEpisodeList(self,
SeriesName, SeasonCount,
EpisodeList)

def main()
```

A FindIdByName rutin egy karakterláncot (string) kap (showname), megteszi a szükséges API hívást, feldolgozza az XML választ és azon műsorok listáját adja vissza, amelyek adatai egyeznek a szótárban. A GetShowInfo az előző rutinban használt showID-ket kapja meg és az ezekről szóló információval tér vissza. A GetEpisodeList szintén az előző rutin showID-jét használja, és az összes epizódra vonatkozó információk szótárával tér vissza.

A kulcs és a bázis URL tárolásához stringek sorozatát fogjuk használni. Nézzük az alábbi kódot (később részletezzük még):

```
self.ApiKey =
"Itnl8IyY1hsR9n0IP6zI"
self.FindSeriesString =
"http://services.tvrage.com/m
yfeeds/search.php?key="
```

A függvényhívás, amit el kell küldenünk (ahhoz, hogy a sorozat idhoz tartozó információ listáját megkapjuk) így néz ki:

http://services.tvrage.com/myfeeds/search.php?key=Itnl8IyY1hsR9n0IP6zI&show={ShowName}

A karakterláncot az alábbi módon adjuk meg…

```
strng = self.FindSeriesString
+ self.ApiKey + "&show=" +
showname
```

Tesztelés céljából a „Continuum” sorozatot fogom használni, ha még nem hallottál róla, ez egy csodálatos scifi sorozat, Kanadában a Showcase hálózaton nézhető. Több oka is van annak, hogy ezt választottam. Először is, (az írás pillanatában) csak két sorozat van, amely a „Continuum”-ra történő keresés során előkerül, így a debuggolás könnyű lesz. Másodszor, jelenleg csak egy évad, azaz tíz rész áll rendelkezésünkre a sorozatból.

Nem árt, ha van róla fogalmunk, hogy a feldolgozó rutinok során mit csinálunk, így a teljes URL hívásokat összegyűjtöttem itt, ezeket ki is próbálhatod, mielőtt saját kód írásába kezdenél.

Keresés a sorozat neve alapján…
http://services.tvrage.com/myfeeds/search.php?key=Itnl8IyY1hsR9n0IP6zI&show=continuum

Sorozat-információk lekérése ShowID (sid) alapján
http://services.tvrage.com/myfeeds/showinfo.php?key=Itnl8IyY1hsR9n0IP6zI&sid=30789

Epizódlista és információk lekérése ShowID (sid) alapján
http://services.tvrage.com/myfeeds/episode_list.php?key=Itnl8IyY1hsR9n0IP6zI&sid=30789

Most, hogy ez megvan, kezdjünk hozzá a saját kódunk megírásához.

Hozzunk létre egy fájlt „tvrage.py” néven. A következő egy-két számban ezt fogjuk használni és módosítani.

Az importálásokkal kezdünk…

Az XML feldolgozáshoz az ElementTree-t fogjuk használni, az urllib-re az internetes kommunikációhoz lesz szükségünk. A sys programkönyvtár a sys.exit miatt kell.

A fő ciklust most írjuk meg, hogy a későbbi programrészeket akár azonnal le tudjuk tesztelni (jobbra, lent). Ne feledjük, ez a forrásfájlunk végén kell szerepeljen.

Ahogy azt korábban már említettem, az első négy sor azokat a karakterláncokat tartalmazza, amelyekből a függvényhíváshoz szükséges URL-t összerakjuk. (a GetEpisodeListString-nél az egésznek egy sorban kellene lennie.) Az utolsó négy sor a később használt lista előkészítéséhez szükséges.

```
#=============================================================
#                        IMPORTS
#=============================================================
from xml.etree import ElementTree as ET
import urllib
import sys
```

ELőször létrehozzuk a karakterláncot, amelyet URL-ként fogunk használni. Ezután beállítjuk a foglalat (socket) időkorlátját 8 másodpercre és az általunk generált

```
    def FindIdByName(self,showname,debug = 0):
        strng = self.FindSeriesString + self.ApiKey + "&show=" + showname
        urllib.socket.setdefaulttimeout(8)
        usock = urllib.urlopen(strng)
        tree = ET.parse(usock).getroot()
        usock.close()
        foundcounter = 0
        self.showlist = []
```

```
#=============================================================
#     Main loop
#=============================================================
if __name__ == "__main__":
    main()
```

Írjuk meg az osztályunkat. Legyen a neve “TvRage”. Megírjuk most az __init__ rutinunkat is.

```
class TvRage:
    def __init__(self):
        self.ApiKey = "Itnl8IyY1hsR9n0IP6zI"
        self.FindSeriesString = "http://services.tvrage.com/myfeeds/search.php?key="
        self.GetShowInfoString = "http://services.tvrage.com/myfeeds/showinfo.php?key="
        self.GetEpisodeListString =
"http://services.tvrage.com/myfeeds/episode_list.php?key="
        self.ShowList = []
        self.ShowInfo = []
        self.EpisodeList = []
        self.EpisodeItem = []
```

URL-vel meghívjuk az urllib.urlopen-t, eredményül pedig (remélhetőleg) megkapjuk a kért xml fájlt az usock objektumban. Meghívjuk az ElementTree setup-ot, ennek segítségével pedig feldolgozzuk az xml adatokat. (Amennyiben elvesztetted volna itt a fonalat, kérlek olvasd el újra az XML-ről szóló cikkeimet [10., 11. és 12. rész az FCM 36., 37. és 38. számában]). Ezután bezárjuk a foglalatot (socket), és inicializálunk egy számlálót, amely a találatok számát mondja majd meg nekünk és visszaállítjuk alapállapotba a „showlist" listát, üressé.

Most az xml információkon futunk végig a „show" címkét használva. Emlékezzünk, hogy a visszakapott adatszerkezet valahogy így néz ki, mint itt jobbra, fent.

Sorra vesszük a „show" szülőelem egyes információcsoportjait és feldolgozzuk az így nyert adatokat. Gyakorlatilag csak a műsor nevére (<name>) és az azonosítójára (<showid>) van szükségünk, de most a többi infót sem hanyagoljuk el.

Az elsőt most megbeszéljük, a többit ez alapján megérted majd. Végighaladva az információn, olyan címkéket keresünk, amelyekre ráillenek a mi feltételeink. Ha találunk ilyet, mindegyiket hozzárendeljük egy ideiglenes változóhoz és ezt eltároljuk a szótárban: egy értéket és egy hozzá tartozó kulcsot. A fenti esetben „showid" címkét keresünk az XML adatban. Ha találunk ilyet, akkor ennek értékét hozzárendeljük a szótár „ID" kulcsához.

A következő rész a műsor műfajával foglalkozik. A fenti XML részlet alapján ehhez a műsorhoz négy különböző műfaj rendelhető: akció, krimi, dráma és Sci-Fi. Ezek mindegyikét le kell kezelnünk.

Végül megnöveljük a found-counter változó értékét és hozzácsatoljuk a szótárhoz a „showlist" listába. Ezután az egészet újrakezdjük az elejétől egészen addig, amíg elfogy az XML adat. Ha elkészültünk, a szótárak listájával térünk

```
for node in tree.findall('show'):
    showinfo = []
    genrestring = None
    dict = {}
    for n in node:
        if n.tag == 'showid':
            showid = n.text
            dict['ID'] = showid
```

```
<Results>
    <show>
        <showid>30789</showid>
        <name>Continuum</name>
        <link>http://www.tvrage.com/Continuum</link>
        <country>CA</country>
        <started>2012</started>
        <ended>0</ended>
        <seasons>2</seasons>
        <status>Returning Series</status>
        <classification>Scripted</classification>
        <genres>
            <genre>Action</genre>
            <genre>Crime</genre>
            <genre>Drama</genre>
            <genre>Sci-Fi</genre>
        </genres>
    </show>
    ...
</Results>
```

```
        elif n.tag == 'name':
            showname = n.text
            dict['Name'] = showname
        elif n.tag == 'link':
            showlink = n.text
            dict['Link'] = showlink
        elif n.tag == 'country':
            showcountry = n.text
            dict['Country'] = showcountry
        elif n.tag == 'started':
            showstarted = n.text
            dict['Started'] = showstarted
        elif n.tag == 'ended':
            showended = n.text
            dict['Ended'] = showended
        elif n.tag == 'seasons':
            showseasons = n.text
            dict['Seasons'] = showseasons
        elif n.tag == 'status':
            showstatus = n.text
            dict['Status'] = showstatus
        elif n.tag == 'classification':
            showclassification = n.text
            dict['Classification'] = showclassification
```

vissza.

A kód nagy része önmagáért beszél, mi most arra a for ciklusra koncentrálunk, amelyet az információ kiíratásánál használunk. Végigfutunk a szótárak listájának összes elemén és kiíratunk egy számláló változót, a műsor nevét (c['Name']) és az id-t. Az eredmény így fog kinézni…

```
Enter Series Name -> continu-
um
2 Found
------------------------
1 - Continuum - 30789
2 - Continuum (Web series) -
32083
Enter Selection or 0 to exit
->
```

Jegyezzük meg, hogy az elemek listájának számozása nullával kezdődik, így ha a felhasználó „1”-et ír be, valójában a szótár 0. számú elemére kíváncsi. Minderre azért van szükség, mert egy „átlagos” felhasználó 1-től és nem 0-tól kezd el számolni. Ezért a 0-t tulajdonképpen használhatjuk a függvényből való kilépésre is a „Q”, „q” vagy „-1” helyett.

Most következik a „fő” (main) rutin, amely az egészet egyben tartja számunkra.

Ma csak hozzákezdünk a megírásához, a következő alkalommal folytatjuk.

Legközelebb további rutinokkal bővítjük a programunkat. Az eddig megírt kód megtalálható a http://pastebin.com/6iw5NQrW linken.

Hamarosan folytatjuk.

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

```
elif n.tag == 'genres':
    for subelement in n:
        if subelement.tag == 'genre':
            if subelement.text != None:
                if genrestring == None:
                    genrestring = subelement.text
                else:
                    genrestring += " | " + subelement.text
    dict['Genres'] = genrestring
```

```
  def main():
      tr = TvRage()
      #--------------------
      # Find Series by name
      #--------------------
      nam = raw_input("Enter Series Name -> ")
      if nam != None:
          sl = tr.FindIdByName(nam)
          which = tr.DisplayShowResult(sl)
          if which == 0:
              sys.exit()
          else:
              option = int(which)-1
              id = sl[option]['ID']
              print "ShowID selected was %s" % id
```

```
            foundcounter += 1
            self.showlist.append(dict)
        return self.showlist
#=========================================================
```

Következő lépésben egy olyan rutint készítünk, amely az összes eredményt megjeleníti.

```
    def DisplayShowResult(self, ShowListDict):
        lcnt = len(ShowListDict)
        print "%d Found" % lcnt
        print "------------------------"
        cntr = 1
        for c in ShowListDict:
            print "%d - %s - %s" % (cntr,c['Name'],c['ID'])
            cntr += 1
        sel = raw_input("Enter Selection or 0 to exit -> ")
        return sel
```

Programozzunk Pythonban – 41. rész

A múlt alkalommal hozzákezdtünk a TVRAGE web API-jának használatához szükséges saját könyvtárunk megírásához. Ebben a számban folytatjuk a megkezdett munkát. Kérlek nyisd meg a múlt alkalommal készített fájlt, vagy töltsd le pastebinről (http://pastebin.com/6iw5NQrW), mert most ezt fogjuk bővíteni, módosítani.

A kód jelenlegi állapotában egy terminál ablakban megadhatjuk a keresett TV-műsor címét, a kérés után pedig a kiválasztott műsorról információkat kapunk. Legutóbb a Continuum műsort használtuk próbaként. Az <Enter> lenyomására a program meghívja az API-t és rákeres a műsor nevére, majd visszaad egy listát azon műsorok nevével, amelyekre ráillik a keresett kulcsszó. Ezután a listából, a számának beírásával kiválaszthatjuk a keresett elemet, majd megjelenik egy „ShowID selected was 30789” felirat. Most megírjuk azt a kódrészt, amely a ShowID felhasználásával információkat szolgáltat az adott műsorról. Tartsuk észben: a megjelenítő függvények leginkább arra szolgálnak, hogy bizonyítsák a függvények működését. Alapvető célunk itt egy újrahasznosítható programkönyvtár létrehozása, amelyet később felhasználhatunk egy GUI-s programnál. Nyugodtan módosíts rajtuk kedved szerint.

```
def GetShowInfo(self,showid,debug=0):
    showidstr = str(showid)
    strng = self.GetShowInfoString + self.ApiKey + "&sid=" + showidstr
    urllib.socket.setdefaulttimeout(8)
    usock = urllib.urlopen(strng)
    tree = ET.parse(usock).getroot()
    usock.close()
    dict = {}
```

Utoljára a „DisplayShowResult” függvényt hoztuk létre. A folytatásban rögtön ez után, de még a „main” elé fogunk írni. A visszaadott információ egy szótárban lesz eltárolva és (amennyiben elérhető) az alábbi dolgokat tartalmazza majd:

- Show ID
- A műsor címe
- Link a műsorra
- eredeti bemutató ország
- szezonok száma
- kép a sorozatról
- kezdés éve
- kezdés dátuma
- befejezés dátuma
- állapot (éppen futó, visszatérő, törölt, stb.)
- besorolás (megírt, reality, stb.)
- sorozat összefoglaló
- műfaj(ok)
- műsoridő percekben
- a premiert leadó csatorna/hálózat neve
- a hálózat országa
- megjelenés ideje
- a vetítés napja a héten
- időzóna

Fent látható a kód kezdete.

A kód nagy része ismerős lehet az előző számból. Valójában nem sok minden változott, menjünk tovább (lent).

Ahogy láthatod, itt sincs semmi új, ehhez hasonló dolgokat láttunk már a korábbi számokban. Egy for

```
for child in tree:
    if child.tag == 'showid':
        dict['ID'] = child.text
    elif child.tag == 'showname':
        dict['Name'] = child.text
    elif child.tag == 'showlink':
        dict['Link'] = child.text
    elif child.tag == 'origin_country':
        dict['Country'] = child.text
    elif child.tag == 'seasons':
        dict['Seasons'] = child.text
    elif child.tag == 'image':
        dict['Image'] = child.text
    elif child.tag == 'started':
        dict['Started'] = child.text
    elif child.tag == 'startdate':
        dict['StartDate'] = child.text
```

```
        elif child.tag == 'ended':
            dict['Ended'] = child.text
        elif child.tag == 'status':
            dict['Status'] = child.text
        elif child.tag == 'classification':
            dict['Classification'] = child.text
        elif child.tag == 'summary':
            dict['Summary'] = child.text
```

ciklust használunk, ellenőrizzük az XML fájlban szereplő összes specifikus értékkel rendelkező címkét. Ha megtaláljuk a keresettet, hozzáadjuk a szótár elemhez.

A dolgok most egy kicsit bonyolódnak. A „genres” címkéket ellenőrizzük. Ez alatt olyan gyermek címkék találhatóak, amelyek tartalmazzák a „genre” (műfaj) megnevezését. Minden műsorhoz több címkét is hozzá lehet rendelni. A műfajokat ezért egy karakterlánchoz (sztring) fogjuk hozzáfűzni és egymástól két szóköz karakter közé írt függőleges vonallal választjuk el, így: “ | “.

Visszatértünk a „normális” kódhoz, ehhez hasonlóval korábban már találkoztunk. Az egyetlen dolog ami kicsit más a „network” címke, amely rendelkezik egy „country” attribútummal. Az attribútum adatokat a „child.attrib[‘attributetag’]” segítségével nyerjük ki a „child.text” helyett.

Ez lenne a rutin vége. Most a nehezen megszerzett információt kellene valahogy megjelenítenünk. Egy „DisplayShowInfo” névre hallgató rutint fogunk létrehozni.

Most frissítenünk kell a „main” rutinunkat, hogy az előbb megírt két új rutint is használja. Az egész main függvényt bemásolom ide, de a változtatásokat **feketével** kiemeltem.

A következő oldalon balra, alul a „Continuum” műsor kiválasztása esetén a „DisplayShowInfo” kimenete látható.

```
elif child.tag == 'genres':
    genrestring = None
    for subelement in child:
        if subelement.tag == 'genre':
            if subelement.text != None:
                if genrestring == None:
                    genrestring = subelement.text
                else:
                    genrestring += " | " + subelement.text
    dict['Genres'] = genrestring
```

```
     elif child.tag == 'runtime':
         dict['Runtime'] = child.text
     elif child.tag == 'network': # has attribute
         dict['NetworkCountry'] = child.attrib['country']
         dict['Network'] = child.text
     elif child.tag == 'airtime':
         dict['Airtime'] = child.text
     elif child.tag == 'airday':
         dict['Airday'] = child.text
     elif child.tag == 'timezone':
         dict['Timezone'] = child.text
 return dict
```

Az időzónára vonatkozó adatokat itt nem jelenítettem meg, de igény esetén nyugodtan használd fel azt is.

A folytatásban az epizódlista rutinokon fogunk dolgozni. A munka nagy részét elvégző függvényt „GetEpisodeList”-nek nevezzük el

```
    def DisplayShowInfo(self,dict):
        print "Show: %s" % dict['Name']
        print "ID: %s  Started: %s  Ended: %s  Start Date: %s  Seasons: %s" %
(dict['ID'],dict['Started'],dict['Ended'],dict['StartDate'],dict['Seasons'])
        print "Link: %s" % dict['Link']
        print "Image: %s" % dict['Image']
        print "Country: %s  Status: %s   Classification: %s" %
(dict['Country'],dict['Status'],dict['Classification'])
        print "Runtime: %s  Network: %s   Airday: %s   Airtime: %s" %
(dict['Runtime'],dict['Network'],dict['Airday'],dict['Airtime'])
        print "Genres: %s" % dict['Genres']
        print "Summary: \n%s" % dict['Summary']
```

és az alábbi információkat adja majd vissza:

- Évad
- Epizód száma
- Az epizód évadon belüli száma
- Produkciós szám
- Megjelenés dátuma
- Link
- Cím
- Összefoglaló
- Értékelés
- Képernyőkép az epizódról (ha elérhető)

Mielőtt hozzákezdenénk a kódoláshoz, érdemes újra áttekintenünk, hogy az API-hoz intézett epizódlista kérés mit ad vissza. Valahogy úgy néz ki, mint a következő oldalon jobbra fent.

```
ShowID selected was 30789
Show: Continuum
ID: 30789  Started: 2012  Ended: None  Start Date:
May/27/2012  Seasons: 2
Link: http://www.tvrage.com/Continuum
Image: http://images.tvrage.com/shows/31/30789.jpg
Country: CA  Status: Returning Series  Classification:
Scripted
Runtime: 60  Network: Showcase  Airday: Sunday
Airtime: 21:00
Genres: Action | Crime | Drama | Sci-Fi
Summary:
Continuum is a one-hour police drama centered on Kiera
Cameron, a regular cop from 65 years in the future who
finds herself trapped in present day Vancouver. She is
alone, a stranger in a strange land, and has eight of the
most ruthless criminals from the future, known as Liber8,
loose in the city.

Lucky for Kiera, through the use of her CMR (cellular
memory recall), a futuristic liquid chip technology
implanted in her brain, she connects with Alec Sadler, a
seventeen-year-old tech genius. When Kiera calls and Alec
answers, a very unique partnership begins.

Kiera's first desire is to get "home." But until she
figures out a way to do that, she must survive in our
time period and use all the resources available to her to
track and capture the terrorists before they alter
history enough to change the course of the future. After
all, what's the point of going back if the future isn't
the one you left?
```

```
def main():
    tr = TvRage()
    #--------------------
    # Find Series by name
    #--------------------
    nam = raw_input("Enter Series Name -> ")
    if nam != None:
        sl = tr.FindIdByName(nam)
        which = tr.DisplayShowResult(sl)
        if which == 0:
            sys.exit()
        else:
            option = int(which)-1
            id = sl[option]['ID']
            print "ShowID selected was %s" % id
    #-----------------------------
    # Get Show Info
    #-----------------------------
    showinfo = tr.GetShowInfo(id)
    #-----------------------------
    # Display Show Info
    #-----------------------------
    tr.DisplayShowInfo(showinfo)
```

Az epizódokra vonatkozó információk az „episode” címkében találhatók – amely a „Season” gyermeke – amely az „Episodelist” gyermeke – amely a „Show” gyermeke. Figyeljünk oda a feldolgozásnál. A már meglévő tudásunkkal a kód első néhány sora könnyen értelmezhető.

Most nézzük meg a „Show” gyökércímke alatt lévő „name” és „totalseasons” címkéket. Ha velük végeztünk, az „Episodelist” és a „Season” következik. Vegyük észre, hogy a „Season” címkének van egy attribútuma. Talán az is feltűnt,

```
        def GetEpisodeList(self,showid,debug=0):
            showidstr = str(showid)
            strng = self.GetEpisodeListString + self.ApiKey
+ "&sid=" + showidstr
            urllib.socket.setdefaulttimeout(8)
            usock = urllib.urlopen(strng)
            tree = ET.parse(usock).getroot()
            usock.close()
            for child in tree:
```

```
            if child.tag == 'name':
                ShowName = child.text
            elif child.tag == 'totalseasons':
                TotalSeasons = child.text
            elif child.tag == 'Episodelist':
                for c in child:
                    if c.tag == 'Season':
                        dict = {}
                        seasonnum = c.attrib['no']
                        for el in c:
```

hogy a „Showname” vagy „Totalseasons” adatokat nem használjuk a szótárban. Egy olyan változóhoz rendeljük hozzá őket, amely a rutin végén visszaadja azt az őt meghívó kódnak.

Most, hogy megvan a kódnak ez a része is, áttérünk az epizódspecifikus adatokra.

Már csak az epizódspecifikus információk (amit a szótárban eltároltunk) listánkhoz való hozzáfűzése maradt és mehetünk tovább. Ha az összes epizóddal végeztünk, visszatérünk a hívó függvényhez és ahogy azt korábban már jeleztem,

```
        if el.tag == 'episode':
            dict={}
            dict['Season'] = seasonnum
            for ep in el:
                if ep.tag == 'epnum':
                    dict['EpisodeNumber'] = ep.text
                elif ep.tag == 'seasonnum':
                    dict['SeasonEpisodeNumber'] = ep.text
                elif ep.tag == 'prodnum':
                    dict['ProductionNumber'] = ep.text
                elif ep.tag == 'airdate':
                    dict['AirDate'] = ep.text
                elif ep.tag == 'link':
                    dict['Link'] = ep.text
                elif ep.tag == 'title':
                    dict['Title'] = ep.text
                elif ep.tag == 'summary':
                    dict['Summary'] = ep.text
                elif ep.tag == 'rating':
                    dict['Rating'] = ep.text
                elif ep.tag == 'screencap':
                    dict['ScreenCap'] = ep.text
```

```
<Show>
<name>Continuum</name>
<totalseasons>2</totalseasons>
<Episodelist>
<Season no="1">
<episode>
<epnum>1</epnum>
<seasonnum>01</seasonnum>
<prodnum/>
<airdate>2012-05-27</airdate>
<link>
http://www.tvrage.com/Continuum/episodes/1065162187
</link>
<title>A Stitch in Time</title>
<summary>
Inspector Kiera Cameron loses everything she has and finds
herself on a new mission when she and eight dangerous
terrorists are transported from their time in 2077 back to
2012 during the terrorist's attempt to escape execution.
She takes on a new identity and joins the VPD in order to
stop the terrorists' reign of violence. Along the way, she
befriends Alec Sadler, the 17 year old who will one day
grow up to create the technology her world is built upon.
</summary>
<rating>8.8</rating>
<screencap>
http://images.tvrage.com/screencaps/154/30789/1065162187.p
ng
</screencap>
</episode>
```

három adatelemmel térünk vissza: „ShowName”, „TotalSeasons” és a szótárak listájával.

Készítsük el a megjelenítő rutinunkat is. Ez elég egyszerű lesz, talán csak az „if e.has_key (‘keynamehere’)“ sor szorul némi magyarázatra. Ez egy olyan ellenőrzés, amellyel megbizonyosodunk arról, hogy a „Rating” és „Summary” változók tartalmaznak adatot. Vannak olyan műsorok, amelyek esetében ezek a mezők üresek.

```
self.EpisodeItem.append(dict)
        return ShowName,TotalSeasons,self.EpisodeItem
```

Már csak a „main” rutin frissítése maradt hátra. Álljon hát itt a teljes „main” rutin, az új részeket ismét csak **kiemelve** (a cikk végén látható).

Most ha elmented a kódot és lefuttatod a programot, a „GetEpisodeList” és a „DisplayEpisodeList” kimenete megfelelően fog működni. Jobbra lent egy részlet az Epizód adatairól.

Ennyit erre a hónapra. Mint mindig, a kódot most is megtalálod pastebinen: http://patbin.com/kSEfs2E.

Remélem te is élvezed a programkönyvtárral való játszadozást. Az API-val további adatokhoz is hozzá tudunk férni. Emlékezz rá, hogy a TVRage mindezt ingyen biztosítja a felhasználók számára, kérlek fontold meg az erőfeszítéseik és a kemény munkájuk anyagi támogatását adomány formájában.

Találkozunk legközelebb.

```
    def DisplayEpisodeList(self,SeriesName,SeasonCount,EpisodeList):
        print "----------------------------------"
        print "Series Name: %s" % SeriesName
        print "Total number of seasons: %s" % SeasonCount
        print "Total number of episodes: %d" % len(EpisodeList)
        print "----------------------------------"
        for e in EpisodeList:
            print "Season: %s" % e['Season']
            print "   Season Episode Number: %s - Series Episode Number: %s" %
(e['SeasonEpisodeNumber'],e['EpisodeNumber'])
            print "   Title: %s" % e['Title']
            if e.has_key('Rating'):
                print "   Airdate: %s   Rating: %s" % (e['AirDate'],e['Rating'])
            else:
                print "   Airdate: %s   Rating: NONE" % e['AirDate']
            if e.has_key('Summary'):
                print "   Summary: \n%s" % e['Summary']
            else:
                print "   Summary: NA"
            print "============================"
        print "------------ End of episode list ------------"
```

```
----------------------------------
Series Name: Continuum
Total number of seasons: 2
Total number of episodes: 10
----------------------------------
Season: 1
   Season Episode Number: 01 - Series Episode Number: 1
   Title: A Stitch in Time
   Airdate: 2012-05-27   Rating: 8.8
   Summary:
Inspector Kiera Cameron loses everything she has and finds herself on a new mission when
she and eight dangerous terrorists are transported from their time in 2077 back to 2012
during the terrorist's attempt to escape execution. She takes on a new identity and
joins the VPD in order to stop the terrorists' reign of violence. Along the way, she
befriends Alec Sadler, the 17 year old who will one day grow up to create the technology
her world is built upon.
============================
```

```
def main():
    tr = TvRage()
    #--------------------
    # Find Series by name
    #--------------------
    nam = raw_input("Enter Series Name -> ")
    if nam != None:
        sl = tr.FindIdByName(nam)
        which = tr.DisplayShowResult(sl)
        if which == 0:
            sys.exit()
        else:
            option = int(which)-1
            id = sl[option]['ID']
            print "ShowID selected was %s" % id
    #-----------------------------
    # Get Show Info
    #-----------------------------
    showinfo = tr.GetShowInfo(id)
    #-----------------------------
    # Display Show Info
    #-----------------------------
    tr.DisplayShowInfo(showinfo)
    #-----------------------------
    # Get Episode List
    #-----------------------------
    SeriesName,TotalSeasons,episodelist = tr.GetEpisodeList(id)
    #-----------------------------
    # Display Episode List
    #-----------------------------
    tr.DisplayEpisodeList(SeriesName,TotalSeasons,episodelist)
    #-----------------------------
```

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.

Hogyanok

Írta: Greg Walters

Programozzunk Pythonban – 42. rész

Tegyük fel, hogy egy multimédia központot szeretnél létrehozni a nappaliban. Van egy számítógéped erre a célra, azon pedig egy csodálatos program, az XBMC. Napokon keresztül rippeled a DVD filmjeid és sorozataid a számítógépre. Utánanéztél a dolgoknak és a fájlokat már a megfelelő módon át is nevezted. Tegyük fel, hogy az „NCIS” a kedvenc sorozatod és megvan az összes DVD-n elérhető része. Találtál egy helyet, ahol az aktuális epizódok is elérhetőek. Szeretnéd megtudni, hogy mi lesz a következő epizódban és hogy mikor adják a TV-ben. Szeretnél továbbá egy listát is az összes epizódról, amivel elkápráztathatod a barátaidat.

Körvonalakban erről szól a következő projektünk, amelyet ebben a hónapban kezdünk el. Először beleássuk magunkat a TV-műsorokat tartalmazó mappába, kibányásszuk a sorozatok nevét és az összes epizódot – annak címét, évadát és az epizód számát. A kinyert adatokat egy adatbázisban tároljuk el.

Az XBMC elvárja, hogy a sorozat részeit az alábbi módon nevezzük el:

```
Tv.Show.Name.SxxExx.Episode
name here if you
care.extension
```

Példának használjuk az NCIS legelső epizódját. AVI formátum esetén a fájl neve így nézne ki:

```
NCIS.S01E01.Yankee White.avi
```

a legutolsó epizódé pedig:

```
NCIS.S10E17.Prime Suspect.avi
```

Ha a sorozat címe több szóból áll, azt így adhatjuk meg:

```
Doctor.Who.2005.S07E04.The
Power of Three.mp4
```

A mappaszerkezet az alábbi módon kell, hogy kinézzen:

```
TVShows
      2 Broke Girls
            Season 1
                  Episode 1
                  Episode 2
                  ...
            Season 2
            ...
      Doctor Who 2005
            Season 1
            ...
            Season 2
```

és így tovább. Most hogy már ismerjük a struktúrát, lépjünk tovább.

Nagyon régen készítettünk egy olyan programot, amely az MP3 fájljainkat rendezte egy adatbázisba. Ha jól emlékszem ez a 35. számban volt, a Python sorozat 9. részeként. Egy WalkThePath névre keresztelt rutint hívtunk meg a mappák kezdőmappából történő rekurzív bejárásához és kikerestük az összes „.mp3” kiterjesztésű fájlt. Ennek a kódnak a nagy részét most újra fogjuk hasznosítani, persze kisebb módosításokkal. Ez alkalommal videófájlokat fogunk keresni, amelyek az alábbi kiterjesztések valamelyikével rendelkeznek:

```
.avi
.mkv
.m4v
.mp4
```

A számítógépes média világában ezek a formátumok rendkívül elterjedtek.

A projekt első részét „tvfilesearch.py” néven fogjuk elmenteni. Ha a végére értünk, ne felejtsd majd el elmenteni, a következő számban ott folytatjuk, ahol ma abbahagyjuk.

Kezdjük az import utasításokkal:

```
import os
from os.path import join,
getsize, exists
import sys
import apsw
import re
```

Importáljuk az os, sys és apsw programkönyvtárakat. Ezeket korábban már használtuk. Ez alkalommal szükség lesz a re programkönyvtárra is a reguláris kifejezések használatához. Erről ma csak felületesen lesz szó, de a következő számban részletezni fogom.

Jöjjön a két utolsó függvény. Minden egyéb, amit később hozzáteszünk a kódhoz, az import utasítások és e két függvény között szerepel majd.

Ez lesz a fő függvényünk, itt hozzuk létre a kapcsolatot az SQLi-

te adatbázissal, amelyet az apsw biztosít. Létrehozunk egy sormutatót, majd pedig meghívjuk a MakeDatabase függvényt és ezzel – ha még nem létezik – létrehozzuk az adatbázist.

Az én TV-fájljaim két meghajtón vannak eltárolva, ezért csináltam egy listát, amely az útvonalakat tartalmazza. Ha neked csak egy mappád van, módosíthatod ezt a három sort az alábbiak szerint:

```
    startfolder = "/file-
path/folder/"
    WalkThePath(startfolder)
```

Most létrehozzuk a saját „standard” if__name__ függvényünket.

```
#============================
if __name__ == '__main__':
    main()
```

Túl vagyunk az unalmas részen, jöhet a hús és a krumpli. A MakeDataBase függvénnyel kezdünk. Ezt tegyük rögtön az import utasítások után.

A fenti függvényt az MP3 szkenner esetén már átbeszéltük, így most csak emlékeztetnélek rá, hogy itt ellenőrizzük a tábla létezését, ha pedig még nem létezik, létrehozzuk.

Most létrehozzuk a WalkThePath függvényt:

A függvény meghívása során (ahogy azt már pedzegettem) megadjuk a fájlok elérési útját, pontosabban a gyökérmappát, ahonnan a keresésünket indítani szeretnénk. Töröljük a showname változó értékét, amelyet majd később fogunk használni. Megnyitjuk az error log fájlt és hagyjuk, hogy a függvény tegye a dolgát. A hívás után (os.walk) egy három elemből álló rekordot kapunk vissza (mappa útvonala, mappanevek, fájlnevek). A mappaútvonal egy karakterlánc lesz, amely az elérési utat adja meg. A mappanevek az almappák nevéből álló listát adnak, a fájlnevek pedig a nem-mappák neveit foglalja össze. Ezeket a listákat elemezve megkeressük a kiválasztott kiterjesztésű fájlokat.

```
    for file in [f for f in
files if f.endswith
(('.avi','mkv','mp4','m4v'))]
:
```

```
#========================================
def main():
    global connection
    global cursor
    # Create the connection and cursor.
    connection = apsw.Connection("TvShows.db3")
    cursor = connection.cursor()
    MakeDataBase()
```

```
#========================================
def MakeDataBase():
    # IF the table does not exist, this will create the table.
    # Otherwise, this will be ignored due to the 'IF NOT EXISTS' clause
    sql = 'CREATE TABLE IF NOT EXISTS TvShows (pkID INTEGER PRIMARY KEY, Series TEXT,
RootPath TEXT, Filename TEXT, Season TEXT, EPISODE TEXT);'
    cursor.execute(sql)
```

```
#========================================
def WalkThePath(filepath):
    showname = ""
    # Open the error log file
    efile = open('errors.log',"w")
    for root, dirs, files in
os.walk(filepath,topdown=True):
```

```
    #========================================
    # Set your video media paths
    #========================================
    startfolder = ["/extramedia/tv_files/","/media/freeagnt/tv_files_2/"]
    for cntr in range(0,2):
        WalkThePath(startfolder[cntr])
        # Close the cursor and the database
    cursor.close()
    connection.close()
    print("Finished")
```

Felbontjuk a fájlneveket kiterjesztésre és fájlnévre (kiterjesztés nélkül). Ezután meghívjuk a GetSeasonEpisode függvényt, amely a fájlnév alapján megadja az évad- és epizódszámot (itt feltételezzük, hogy az elnevezés megfelelően formázott).

```
 OriginalFilename,ext =
os.path.splitext(file)

fl = file

isok,data = GetSeasonEpi-
sode(fl)
```

A GetSeasonEpisode egy bool változót és egy listát (ebben az esetben „data”) ad vissza, amely tartalmazza a sorozat nevét, az évadot és az epizódszámokat. Ha a fájlnév nem megfelelően formázott, az „isok” bool változó értéke hamis.

A folytatásban ellenőrizzük, hogy a fájl része-e az adatbázisnak. Ha igen, nem kétszerezzük meg. Egyszerűen csak a fájlnevet ellenőrizzük. Áshatnánk mélyebbre és megvizsgálhatnánk az útvonalak egyezését is, de ez alkalommal ettől eltekintünk, így is jó lesz.

Ha minden megfelelően működik, a lekérdezésre adott válasz egy 1-es vagy 0-ás lesz. Ha 0, akkor még nincs benne az adatbázisban, így most hozzáadjuk, ha pedig már benne van, akkor csak továbblépünk. Vegyük észre a Try Except parancsot. Ha valami rosszul sül el, például olyan karakter van a szövegben, amelyet az adatbázis nem tud kezelni, a program nem fog leállni. A hibát mindenesetre a logfájlban jelezzük, hogy később azt ellenőrizhessük.

```
                    if isok:
                showname = data[0]
                season = data[1]
                episode = data[2]
                print("Season {0} Episode {1}".format(season,episode))
            else:
                print("No Season/EPisode")
                efile.writelines('--------------------------\n')
                efile.writelines('{0} has no series/episode information\n'.format(file))
                efile.writelines('--------------------------\n\n')
```

```
            sqlquery = 'SELECT count(pkid) as rowcount from TvShows where Filename =
"%s";' % fl
            try:
                for x in cursor.execute(sqlquery):
                    rcntr = x[0]
                if rcntr == 0:  # It's not there, so add it
```

```
                    try:
                        sql = 'INSERT INTO TvShows (Series,RootPath,Filename,Season,Episode)
VALUES (?,?,?,?,?)'
                        cursor.execute(sql,(showname,root,fl,season,episode))
                    except:
                        print("Error")
                        efile.writelines('--------------------------\n')
                        efile.writelines('Error writing to database...\n')
                        efile.writelines('Filename = {0}\n'.format(file))
                        efile.writelines('--------------------------\n\n')
                except:
                    print("Error")
                print('Series - {0} File - {1}'.format(showname,file))
```

Itt egyszerűen csak elhelyezünk egy új rekordot az adatbázisba, vagy az error fájlba írunk.

```
 # Close the log file
   efile.close
# End of WalkThePath
```

Most nézzük a GetSeasonEpisode függvényt.

```
#============================
def GetSeasonEpisode(file-
name):
filename = filename.upper()
```

```
resp = re.search(r'(.*).S\d\dE\d\d(\.*)', filename, re.M|re.I)
```

A re.search rész a re programkönyvtár része. Egy minta karakterláncot használ, amely a mi esetünkben a fájlnév, amit elemezni szeretnénk. A re.M|re.I paraméterek, amelyek megadják, hogy több- soros típusú keresést (re.M) szeretnénk ignore-case (re.I) típussal kombinálni. Erről, azaz a reguláris kifejezések kezeléséről többet is megtudhatsz a következő számból. Most a keresett mintánk a „.S” karakterekből áll, amelyet két decimális szám követ, valamint egy „E” betű és két további szám, végül pedig egy pont. A következő fájnév – „tvshow.S01E03.avi" – például megfelel a keresési feltételeknek. Néhányan „tvshow.s01e03.avi", vagy „tvshow.103.avi" néven kódolják az információt, amit nyilvánvalóan nehezebb értelmeznünk. A következő hónapban úgy módosítjuk majd a függvényt, hogy a legtöbb esetben működőképes maradjon. Az „r” segítségével nyers karakterláncot is használhatunk a keresés során.

Haladjunk tovább. A keresés visszaad egy objektumot, amiben nézelődhetünk. A „resp” egy válasz, amely üres abban az esetben, ha nincs egyezés. A visszakapott információ két csoportra osztható. Az első fele a keresett karakterlánc előtti részt tartalmazza, míg a második a keresett mintára ráillő részt is. A fenti esetben például az első rész a „tvshow” lenne, a második pedig “tvshow.S01E03.”.

```
    if resp:
        showname = resp.group(1)
```

A műsor nevét az első csoport alapján határozzuk meg. Meghatározzuk ennek a hosszát, ez alapján pedig összerakjuk a megfelelő karakterláncot.

```
        shownamelength = len(showname) + 1
        se = filename[shownamelength:shownamelength+6]
        season = se[1:3]
        episode = se[4:6]
```

Ez után a műsor nevében szereplő pontokat lecseréljük szóközökre – a szöveg így az emberi szem számára könnyebben olvashatóvá válik.

```
        showname = showname.replace("."," ")
```

Készítünk egy listát, amelyben a műsor neve, az évad és az epizód sorszáma szerepel, ezeket pedig egy Igaz értékű bool változóval egyetemben a függvény visszaadja (a bool változó értéke azt jelzi, hogy minden rendben ment).

```
        ret = [showname,season,episode]
        return True,ret
```

Amennyiben nem találtunk egyezést, a listánkba a műsor címe helyére üres karakterláncot teszünk, a másik két értéket pedig „-1”-re állítjuk. A bool változó értéke Hamis lesz.

```
    else:
        ret = ["",-1,-1]
        return False,ret
```

Ennyi lenne a kód mára. Vessünk egy pillantást a programunk kimenetére. Feltételezve, hogy a fájlstruktúrád pontosan olyan, mint az enyém, az alábbi szöveg jelenik meg a képernyőn...

```
Season 02 Episode 04
SELECT count(pkid) as rowcount from TvShows where Filename = "InSecurity.S02E04.avi";
Series - INSECURITY File - InSecurity.S02E04.avi
Season 01 Episode 08
SELECT count(pkid) as rowcount from TvShows where Filename = "Prime.Suspect.US.S01E08.Underwater.avi";
Series - PRIME SUSPECT US File - Prime.Suspect.US.S01E08.Underwater.avi
```

… és így tovább. Ha szeretnéd, lerövidítheted a megjelenítendő kimenetet. Minden egyes bejegyzés el lesz mentve az adatbázisunkban, valahogy így:

```
pkID | Series | Root Path | Filename | Season | Episode
2526 | NCIS   | /extramedia/tv_files/NCIS/Season 7|NCIS.S07E04.Good.Cop.Bad.Cop.avi | 7 | 4
```

Mint mindig, a teljes kód most is elérhető a PasteBin.com-on: http://pastebin.com/txmmagkL

Legközelebb további Évad|Epizód formátumokkal fogunk foglalkozni és tovább fogjuk bővíteni a ma megírt programunkat.

Találkozzunk 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.

Programozzunk Pythonban – 43. rész

A legutóbbi alkalommal egy olyan projektet kezdtünk el, amely a két számmal ezelőtt megírt TvRage modulunkat használja fel. Most ezt fogjuk folytatni. Javítani fogunk a fájlnevek feldolgozásának módján, az adatbázishoz pedig két újabb mezőt (TvRageId és Status) adunk hozzá.

Először az import sorokat módosítjuk. Azok számára, akik lemaradtak az előző számról, álljanak itt a múltkori include-ok is (jobbra fent).

Az „import re” után következő sorok újak.

A következő dolgunk a GetSeasonEpisode függvény átírása lesz. Szinte mindent le fogunk cserélni abból, amit a múlt alkalommanl létrehoztunk és a kódot flexibilisebbé teszük, hogy a különböző évad/epizód sémákat jobban tudjuk kezelni. Az új módosításokkal az alábbi változatokat fogjuk tudni kezelni:

```
Series.S00E00

Series.s00e00

Series.S00E00.S00E01

Series.00x00

Series.S0000

Series.0x00
```

Mielőtt az adatbázisba írnánk, a „hiányzó vezető nullák” problémáját is orvosolni fogjuk.

Az első mintánk a több epizódból álló fájlok kezelését próbálja megoldani. Több elterjedt módja is van az elnevezésnek, amivel mi most foglalkozunk az „S01E03.S01E04” alakú lesz és az alábbi karakterlánc mintát használjuk: „(.*)\.s(\d{1,2})e(\d{1,2})\.s(\d{1,2})e(\d{1,2})”. Eredményül (remélhetőleg) öt csoportot fogunk kapni, amely az alábbiakból fog állni: a sorozat neve (S[1]), évad (S[2]), epizódszám 1 (S[3]), évad (S[4]) és epizódszám 2 (S[5]). Minden, ami a „.s” előtt van tehát egy csoportba kerül, majd ez után jön két szám, az „e” karaktert kihagyjuk és két újabb szám következik, majd pedig ez utóbbi kettő újra. Ha a fájlunk neve „Monk.S01E05.S01E06.avi”, akkor az egyes csoportokba az alábbi értékek kerülnek…

```
S[1] = Monk

S[2] = 01

S[3] = 05

S[4] = 01

S[5] = 06
```

```
import os
from os.path import join, getsize, exists
import sys
import apsw
import re
#------------------------------
#   NEW LINES START HERE
#------------------------------
from xml.etree import ElementTree as ET
import urllib
import string
from TvRage import TvRage
```

Mi csak az első három csoportot (S[1], S[2], S[3]) fogjuk használni a kódban, de bizonyára látod, hogy hová vezet ez az egész. Ha a keresés során egyezést találunk, a „GoOn” változó értékét igazra állítjuk. Ez segít majd eligazodnunk a későbbi if ágak során.

Itt van tehát a GetSeasonEpisode rutin forráskódja (következő oldal).

Ha eddig eljutottunk, hozzáfogunk a műsor nevének formázásához is a benne szereplő pontok eltávolításával, kinyerjük az évad és epizód információkat, majd pedig visszaadjuk őket. Az évad információt az „S00E00”-szerű mintából bányásszuk ki, itt az évadszámban benne lesz majd a vezető nulla is. Amennyiben a fájlnév „xxx” alakban tartalmazza ezt az információt, akkor feltételezzük, hogy az évad számát az első karakter kódolja, a maradék kettő pedig az epizódét. Kicsit előre gondolkodunk és az évadszámot kétjegyűként tároljuk el, szükség esetén vezető nullával kiegészítve.

A MakeDatabase rutinunkban módosítjuk az adatbázis létrehozásának módját: két újabb mezőt adunk hozzá.

Ismét csak az utolsó két mező definíciója változott az előzőekhez képest.

A WalkThePath függvényünkben csak azok a sorok változtak, amelyek az adatbázisba való beillesztésért felelősek. A módosított kóddal az új struktúrákat is le tudjuk kezelni. Talán még emlékszel, hogy a TV fájljainkat tartalmazó mappá(ka)t adjuk át a függvényünknek. Ez az én esetemben két mappa, amelyeket listaként adunk meg és egy for ciklussal egyesével adjuk őket át a függvényünknek. Minden megadott mappában .avi, .mkv, .mp4 és .m4v kiterjesztésű fájlokat keresünk. Ha találunk ilyet, a GetSeasonEpisode függvénynek adjuk tovább. Ez után leellenőrizzük, hogy az adott fájl szerepel-e már az adatbázisunkban, ha nem, hozzáadjuk. A múltkor megírt függvénynek most csak egy részét írom ide (következő oldalon).

A két új sor vastagon van szedve.

Félig készen is vagyunk. A kódban most néhány segédfüggvény követ-

```
def GetSeasonEpisode(filename):
    GoOn = False
    filename = filename.upper()
```

Ez az első minta ellenőrzőnk.

```
    #Should catch multi episode .S01E01.S01E02 type filenames
    resp = re.search(r'(.*)\.s(\d{1,2})e(\d{1,2})\.s(\d{1,2})e(\d{1,2})',filename,
re.I)
    if resp:
        showname = resp.group(1)
        GoOn = True
    else:
```

A második ellenőrzés során SddEdd vagy sddedd formátumot keresünk…

```
    # Should catch SddEdd or sddedd
        resp = re.search(r'(.*).S(\d\d?)E(\d\d?)(\.*)', filename, re.I)
        if resp:
            showname = resp.group(1)
            GoOn = True
        else:
```

A következő minta a ddxdd formátumra hajaz.

```
       #check for ddxdd
       resp = re.search(r'(.*)\.(\d{1,2})x(\d{1,2})(.*)', filename, re.I)
       if resp:
           showname = resp.group(1)
           GoOn = True
       else:
```

Ez a kódrész Sdddd mintát keres.

```
                #check for Sdddd
                resp = re.search(r'(.*).S(\d\d)(.\d\d?)' , filename, re.I)
if resp:
    showname = resp.group(1)
    GoOn = True
else:
```

És végül megpróbáljuk a ddd szerkezetet is lekezelni.

```
     # Should catch xxx
     resp = re.search(r'(.*)(\d)(.\d\d?)',filename,re.I)
     if resp:
         showname = resp.group(1)
         GoOn = True
```

```
if GoOn:
    shownamelength = len(showname) + 1
    showname = showname.replace("."," ")
    season = resp.group(2)
    if len(season) == 1:
        season = "0" + season
    episode = resp.group(3)
    ret = [showname,season,episode]
    return True,ret
else:
    ret = ["",-1,-1]
    return False,ret
```

```
def MakeDataBase():
    # IF the table does not exist, this will create the table.
    # Otherwise, this will be ignored due to the 'IF NOT EXISTS' clause
    sql = 'CREATE TABLE IF NOT EXISTS TvShows (pkID INTEGER PRIMARY KEY, Series TEXT, RootPath TEXT, Filename TEXT,
Season TEXT, Episode TEXT, tvrageid TEXT,status TEXT);'
    cursor.execute(sql)
```

kezik, ezek a TvRage függvényünket segítik az adatbázis mezők feltöltésében. Az első függvény a WalkThePath függvény után fut le: végignézi az adatbázist, megkeresi a sorozat nevét és a TvRage szervertől lekérdezi az azonosító (id) számát. Miután ez megvan, frissítjük az adatbázist és a következő TvRage lekérdezésnél már ezt az azonosítót használjuk a sorozat jelenlegi állapotának meghatározására. A sorozat lehet új (New), visszatérő (Returning), lemondott (Canceled), befejezett (Ended) és pihentetett („On Haitus"). Erre az információra az új epizódok keresése során lesz szükségünk, hiszen ha a sorozatnak már nem lesz több új része, akkor értelemszerűen nem fogjuk azokat keresni. Megvan hát a sorozat állapota és ezt eltárolhatjuk az adatbázisban.

Itt most megállunk és tüzetesebben megvizsgáljuk az SQL lekérdezésünket. Ez egy kicsit különbözik mindentől, amit eddig csináltunk. A kód így néz ki:

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

Ez azt mondja, hogy mutass nekem egy példát a sorozat nevére, nem számít hány van belőle és a tvrageid-je legyen egyenlő „-1"-el. A Doctor Who 2005-ből például találunk 103 epizódot. A Distinct utasítást használva csak egyetlen rekordot kapunk majd, feltételezve, hogy a TvRageID-nk még hiányzik.

```
    for x in SeriesCursor.execute(sqlstring):
        seriesname = x[0]
        searchname = string.capwords(x[0]," ")
```

A sorozat nevének (x[0]) „megfelelő" módon történő megváltoztatására a stringkönyvtár capwords függvényét használjuk. Erre azért van szükség, mert a TvRage olyan bejegyzést vár, amelyben nem csak nagybetűk vannak. A „THE MAN FROM UNCLE" cím tehát „The Man From Uncle"-re lesz konvertálva. Ezt a

```
            sqlquery = 'SELECT count(pkid) as rowcount from TvShows where Filename =
"%s";' % fl
            try:
                for x in cursor.execute(sqlquery):
                    rcntr = x[0]
                if rcntr == 0:  # It's not there, so add it
                    try:
                        sql = 'INSERT INTO TvShows
(Series,RootPath,Filename,Season,Episode,tvrageid) VALUES (?,?,?,?,?,?)'
                        cursor.execute(sql,(showname,root,fl,season,episode,-1))
                    except:
```

```
def WalkTheDatabase():
    tr = TvRage()
    SeriesCursor = connection.cursor()
    sqlstring = "SELECT DISTINCT series FROM TvShows WHERE tvrageid = -1"
```

FindIdByName hívásnál fogjuk majd használni, ahol a függvény egy a keresési feltételeknek megfelelő műsorok listájával tér vissza. A listát megjeleníti és mi kiválaszthatjuk belőle azt, amelyik megfelelő. A kiválasztás után az adatbázisban frissítjük az id számot és meghívjuk a GetShowStatus függvényt, amely megmondja, hogy a sorozat pillanatnyilag milyen állapotban van.

Az UpdateDatabase függvény egyszerűen csak kulcsként felhasz-

```
def UpdateDatabase(seriesname,id):
    idcursor = connection.cursor()
    sqlstring = 'UPDATE tvshows SET tvrageid = ' + id + ' WHERE series = "' + seriesname + '"'
    try:
        idcursor.execute(sqlstring)
    except:
        print "error"
```

```
def GetShowStatus(seriesname,id):
    tr = TvRage()
    idcursor = connection.cursor()
    dict = tr.GetShowInfo(id)
    status = dict['Status']
    sqlstring = 'UPDATE tvshows SET status = "' + status + '" WHERE series = "' + seriesname + '"'
    try:
        idcursor.execute(sqlstring)
    except:
        print "Error"
```

nálja a sorozat nevét és frissíti a bejegyzéseket a megfelelő TvRage ID-re.

A GetShowStatus is nagyon egyszerű. A TvRage programkönyvtárból meghívjuk a GetShowInfo-t és átadjuk neki a megfelelő id-t, ő pedig visszaadja a sorozat adatait. Talán még emlékszel, hogy a TvRage sok információt tárol, de ezen a ponton csak a műsor állapotára vagyunk kíváncsiak. Mivel egy szótárban minden információt megkapunk, most csak a ['Status'] kulcsra fogunk rákeresni. Miután megvan, ennek megfelelően frissítjük az adatbázist és megyünk tovább.

Majdnem készen vagyunk. Még egy sort hozzá kell adnunk a múlt heti main függvényünkhöz (feketén szedve), amellyel – miután begyűjtöttük az összes fájlnevet – meghívjuk a „WalkTheDatabase" függvényt. A main függvénynek ismét csak egy részét másolom ide, de ez alapján könnyen megtalálod majd az új sor helyét.

Ez lenne hát a mi kódunk. Gondoljuk azért át az elejétől a végéig, hogy mit csinálunk.

```
        print("Requesting information on " + searchname)
        sl = tr.FindIdByName(searchname)
        which = tr.DisplayShowResult(sl)
        if which == 0:
            print("Nothing found for %s" % seriesname)
        else:
            option = int(which)-1
            id = sl[option]['ID']
            UpdateDatabase(seriesname,id)
            GetShowStatus(seriesname,id)
```

Először – ha még nem létezik – létrehozunk egy adatbázist.

Ez után végigjárjuk az előre meghatározott útvonalakat és olyan fájlokat keresünk, amelyeknek a

```
startfolder = ["/extramedia/tv_files","/media/freeagnt/tv_files_2"]
#for cntr in range(0,2):
    #WalkThePath(startfolder[cntr])
WalkTheDatabase()
# Close the cursor and the database
cursor.close()
connection.close()
print("Finished")
```

kiterjesztése az alábbiak közül valami:

```
.AVI, .MKV, .M4V, .MP4
```

Ha találunk ilyet, kezelésbe vesszük a nevét és megfelelően feldolgozzuk: kibányásszuk a sorozat nevét, évadát és az epizód sorszámát. Ezeket az adatokat is elmentjük az adatbázisunkban, ha még nincs ott.

Miután végignéztük a fájlokat, egy adatbázis lekérdezéssel megkeressük azokat a sorozatneveket, amelyekhez nem sikerült TvRage ID-t hozzárendelni. Ez után a TvRage API-t is használva, végrehajtunk egy lekérdezést és megkeressük a megfelelő sorozatot, majd annak ID-jét hozzárendeljük a mi adabázisunkban szereplőhöz. Ezen a folyamaton minden sorozat egyszer fog átesni. A jobbra látható képernyőkép mutat egy példát arra, hogy hogyan néz ez ki a Midsomer Murders tvsorozat esetében.

Ebben az esetben én egy 1-est írtam be, így a sorozat megkapta a TvRage 4466-os ID-jét. Ez bekerült az adatbázisba és a sorozat jelenlegi állapotának lekérdezéséhez ezt az azonosítót használjuk fel. Egy újabb lekérdezés után a TvRage azt mondja, hogy egy visszatérő sorozatról („Returning Series”) van szó. Ezt az információt is eltároljuk az adatbázisban és továbblépünk.

Az első futás során a programnak szüksége lesz a te felügyeletedre, mert minden sorozat esetén rá fog kérdezni az ID azonosítóra. Jó hír, hogy ezt csak egyszer kell végigcsinálnunk. „Normális” esetben csak néhány sorozatról van ugye szó. Nálam ez 157 darab különböző sorozatot jelentett, úgyhogy eltartott egy darabig, mire a végére értem. Mivel a fájlnevek megfelelően voltak megadva (TvRage és TheTvDB.com alapján), így általában az első opció volt a nyerő nálam.

Mellesleg a sorozataim több mint fele vagy véget ért vagy már törölték. Ez sokat sejtet arról, hogy vajon melyik korosztályba tartozom.

Szokás szerint a teljes kód elérhető PasteBin-en: http://pastebin.com/MeuGyKpX

Legközelebb a TvRage további integrációjáról lesz szó. Addig is minden jót!

```
Requesting information on Midsomer Murders
5 Found
------------------------
1 - Midsomer Murders - 4466
2 - Motives and Murders - 31373
3 - See No Evil: The Moors Murders - 11199
4 - The Atlanta Child Murders - 26402
5 - Motives & Murders: Cracking the Case - 33322
Enter Selection or 0 to exit ->
```

Greg a RainyDay Solutions, LLC (Aurora, Colorado) tanácsadó cég tulajdonosa, 1972 óta programozik. Szeret főzni, túrázni, zenét hallgatni, és az idejét a családjával tölteni. Weboldala: 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 Canonical-nek, a fordítócsapatoknak a világban és **Thorsten Wilms**-nek az FCM logóért.

Full Circle Magazin
Magyar Fordítócsapat

Koordinátor:
Pércsy Kornél

Fordítók:

A fordítók csapata

Lektor:
Balogh Péter

Szerkesztő/Korrektor:
Heim Tibor