قسمت سیزدهم

این مار خوش‌خط‌وخال

نفوذ به دنیای موسیقی

» احمد شریف‌پور

شاید شما نیز بسیاری از موسیقی‌های مورد علاقه‌تان را در قالب فایل‌های mp3 روی هارددیسک کامپیوترتان ذخیره کرده‌اید. تا زمانی که تعداد این فایل‌ها به هزار نرسیده است، مدیریت و به خاطر سپردن محل آن‌ها ساده خواهد بود. اما اگر تعداد فایل‌هایتان بیشتر باشد، کار بسیار مشکل خواهد شد. چند سال پیش‌تر، بحث اصلی بر سر فضای ذخیره‌سازی بود، اما اکنون مشکل اصلی این است که به یاد بیاورید کدام موسیقی در چه فایلی و با چه نامی در کجا ذخیره شده است. در این شماره سعی خواهیم کرد، کاتالوگی برای آرشیو موسیقی‌مان به‌وجود آوریم و در این راه از مهارت‌هایی که در کار با پایگاه‌های داده کسب کردیم، استفاده خواهیم کرد و با مفاهیم تازه‌ای نیز آشنا خواهیم شد.

متادیتا

نخستین نکته‌ای که باید بدانید این است که فایل‌های mp3 می‌توانند اطلاعاتی را درباره خود فایل ذخیره کنند. اطلاعاتی نظیر عنوان آهنگ، عنوان آلبوم و نام خواننده از جمله این اطلاعات هستند. این اطلاعات در برچسب‌های ID3 ذخیره می‌شود که به اصطلاح فرا داده یا «متادیتا» نامیده می‌شوند. در نخستین روزهای ظهور فایل‌های mp3 تنها مقدار بسیار کمی از این اطلاعات در فایل‌های mp3 ذخیره می‌شد. در آن زمان این اطلاعات در قسمت پایانی فایل‌های mp3 ذخیره می‌شد و بلوکی متشکل از ۱۲۸ بایت بود. به‌واسطه حجم اندک این بلوک، تنها ۳۰ کاراکتر به هر یک از اطلاعاتی نظیر نام آهنگ و آلبوم و خواننده اختصاص داده شده بود. برای بسیاری از فایل‌ها همین مقدار کافی بود، اما اگر با آهنگی با نام طولانی، نام آلبومی طولانی و... روبه‌رو می‌شدیم، مشکل خودنمایی می‌کرد. این برچسب‌های ID3 تحت‌عنوان استاندارد ID3v1 طبقه‌بندی شدند و برای تکمیل آن نسخه جدیدی از این استاندارد با نام ID3v2 معرفی شد. این استاندارد جدید امکان ذخیره اطلاعاتی با طول‌های متغیر را آن هم در ابتدای فایل فراهم می‌آورد. با استفاده از این استاندارد، اطلاعات ID3v1 هنوز برای حفظ سازگاری با پخش‌کننده‌های قدیمی در انتهای فایل حفظ می‌شدند. به کمک این استاندارد جدید، متادیتا می‌توانست حجمی تا حداکثر ۲۵۶ مگابایت داشته باشد! در این استاندارد هر گروه از اطلاعات در یک چهارچوب یا فریم (Frame) نگه‌داری می‌شود و هر فریم یک معرف یا Identifier دارد. در نسخه فعلی (۲/۴) این استاندارد، طول این معرف چهار کاراکتر است.

موتاژن

پیش‌تر برای کار با این متادیتاها، باید فایل را به‌صورت باینری بازمی‌کردیم و در آن اطلاعات موردنظر را جست‌وجو می‌کردیم. این کار اگرچه ممکن و ساده بود، اما به زمان و کدنویسی بسیاری نیاز داشت. اما اکنون دیگر می‌توانیم از کتابخانه‌های متعددی که برای این کار تهیه شده‌اند، استفاده کنیم. ما از کتابخانه‌ای به نام موتاژن (Mutagen) در این پروژه استفاده خواهیم کرد. برای ادامه کار در اوبونتو در مدیر بسته Synaptic موتاژن را جست‌وجو کرده و آن را نصب کنید. اگر از سیستم‌های ویندوزی برای کدنویسی استفاده می‌کنید، می‌توانید این کتابخانه را از آدرس http://code.google.com/p/mutage دریافت و نصب کنید. البته، موتاژن به غیر از فایل‌های mp3 قابلیت کار با فرمت‌های ASF، Ogg Vorbis، M4A و بسیاری فرمت‌های دیگر را هم دارد. در این پروژه علاوه بر موتاژن مانند دو قسمت قبل به کتابخانه apsw نیز برای کار با پایگاه‌های داده نیاز خواهیم داشت.

```
1  import os , sys
2  from os.path import join , getsize , exists
3  from mutagen.mp3 import MP3
4  import apsw
5  def MakeDatabase():
6          pass
7  def S2HMS(t):
8          pass
9  def WalkThePath(musicpath):
10         pass
11 def error(message):
12         pass
13 def main():
14         pass
15 def usage():
16         pass
17 if __name__ == '__main__':
18         main()
```

فهرست۱ چهارچوب اصلی برنامه

```
1  def usage():
2          message = (
3          '====================================\n'
4          'mCat finds all mp3 files in a folder\n'
5          'and subfolders. Reads the ID3 tags and \n'
6          'writes the information to a SQLite database.\n'
7          'usage:\n'
8          '\t %s <foldername>\n'
9          '\t WHERE <foldername> is the path to the mp3 files.\n'
10         '====================================\n'
11         ) %sys.argv[0]
12         print type(sys.argv)
13         error(message)
14         sys.exit(1)
```

فهرست۲ کدهای مورد نیاز برای تکمیل تابع Usage

شروع

فایل جدیدی با نام mCat.py ایجاد کنید و کدهای فهرست ۱ را که چهارچوب اصلی برنامه ما را می‌سازند، در آن وارد کنید. اگرچه این فایل در این وضعیت هیچ کاری انجام نمی‌دهد، اما با اجرای آن و در صورت عدم دریافت پیغام خطا می‌توانید مطمئن شوید که تمام کتابخانه‌های مورد نیاز را به درستی نصب کرده‌اید.

در این فهرست ما اسکلت مربوط به توابع مورد نیاز را پیاده کرده‌ایم. تنها نکته عجیب شاید آخرین قسمت این کد یعنی خطوط ۱۷ و ۱۸ باشد. هنگامی که یک کد به اجرا در می‌آید، چه به صورت مستقیم و چه با import شدن در سایر کدها، پایتون نامی را به آن نسبت می‌دهد و با آن همانند یک شیء رفتار می‌کند. زمانی که کد به تنهایی اجرا شود، نامی که به آن نسبت داده می‌شود __main__ است. با دستور if موجود در خط ۱۷ ما کنترل می‌کنیم که آیا این فایل به‌صورت مستقیم اجرا شده است یا خیر؟ اگر فایل به صورت مستقیم اجرا شده بود، تابع اصلی برنامه یعنی ()main اجرا خواهد شد. اما اگر فایل در یک کد دیگر import شده باشد، هیچ کدی به‌صورت مستقیم اجرا نخواهد شد و برنامه منتظر می‌ماند تا توابع موجود در آن توسط برنامه import کننده فراخوانده شود.

تعریف توابع

حال به تکمیل این توابع می‌پردازیم. در ابتدا به سراغ تابع ()usage می‌رویم. این تابع نحوه کار برنامه ما را برای کاربر توضیح خواهد داد. همچنین با اعلام بروز خطا اجرای برنامه را متوقف خواهد کرد. کدهای این تابع را در فهرست ۲ مشاهده می‌کنید.

در این کدها که باید جایگزین خطوط ۱۵ و ۱۶ فایل اصلی (فهرست ۱) شوند، ابتدا رشته‌ای با نام message در خطوط ۲ تا ۱۱

```
def main():
        global connection
        global cursor
        if len(sys.argv) != 2:
                usage()
        else:
                StartFolder = sys.argv[1]
                if not exists(StartFolder):
                        print “Folder %s does not exist. Exiting.” %StartFolder
                        sys.exit(1)
                else:
                        print “\nWorking on %s” %StartFolder
                        connection = apsw.Connection(“mCat.db3”)
                        cursor = connection.cursor()
                        MakeDataBase()
                        WalkThePath(StartFolder)
                        cursor.close()
                        connection.close()
                        print “\nFinished.”
```

فهرست ۳ کدهای مورد نیاز برای تکمیل تابع main

```
def MakeDataBase():
    sql = ‘CREATE TABLE IF NOT EXISTS mp3 (pkID INTEGER PRIMARY KEY,’
    sql = sql + ‘ title TEXT, artist TEXT, album TEXT, bitrate TEXT, ‘
    sql = sql + ‘genre TEXT, playtime TEXT, track INTEGER, year TEXT,’
    sql = sql + ‘ filesize TEXT, path TEXT, filename TEXT);’
    cursor.execute(sql)
```

فهرست ۴ کدهای مورد نیاز برای تکمیل تابع Make Data Base (1)

تعریف شده است. دو نکته کوچک در تعریف این متغیر وجود دارد. نخست این‌که اگر تعدادی رشته را بدون هیچ عملگری پشت سرهم ردیف کنیم، پایتون آن‌ها را به ترتیب به هم افزوده و یک رشته طولانی‌تر به‌وجود خواهد آورد؛ دوم این‌که در خط ۸ ما با %s جایی را برای درج یک رشته خالی گذاشته‌ایم. این جای خالی در خط ۱۲ و از طریق sys.argv[0]% با نام فایل حاوی برنامه پر شده است. با import کردن ماجول sys ما به متغیری به نام argv دسترسی خواهیم داشت. این متغیر که از جنس لیست است، حاوی کل آرگومان‌هایی است که در هنگام اجرای برنامه از طریق خط فرمان واردشده‌اند. برای نمونه، اگر برای اجرای فایل mCat در خط فرمان از دستور زیر استفاده شود:

```
python mCat.py /usr/Music Ali 1234
```

مقدار ['mCat.py', '/usr/Music', 'Ali', '1234'] در متغیر argv ذخیره خواهد داشت. در این متغیر آرگومان اول یا argv[0] همواره نام فایل اجرا شده خواهد بود.

در خط ۱۳ ما تابع error را برای اطلاع‌دادن به کاربر فراخوانده‌ایم و پس از آن در خط ۱۴ با استفاده از sys.exit(1) از برنامه خارج شده‌ایم. تابع exit از ماجول sys برای پایان بخشیدن به اجرای برنامه‌ها به کار می‌رود و در صورتی که مقدار ارسال شده به آن 0 باشد، به سیستم اعلام می‌کند که اجرا با موفقیت به پایان رسیده است. در غیر این صورت مشخص خواهد شد که اجرای برنامه به‌واسطه خطا متوقف شده است.

حال نوبت به تعریف تابع error می‌رسد. برای تکمیل این تابع عبارت زیر را جایگزین کلمه pass در خط ۱۲ فهرست ۱ کنید.

```
print >> sys.stderr , message
```

در این دستور print، ما از قابلیت تغییر مسیر خروجی استفاده‌کرده‌ایم و پیغام را به خروجی استاندارد تعریف شده برای پیغام‌های خطا هدایت‌کرده‌ایم. توضیح ضروری این‌که در سیستم‌عامل به‌صورت استاندارد، خروجی‌هایی برای چاپ اطلاعات معمول، اطلاعات مربوط به خطاها و ورودی اطلاعات کاربر در نظر گرفته می‌شود که به ترتیب stdout، stderr و stdin نامیده می‌شوند. زمانی که شما از دستور print به‌صورت عادی استفاده می‌کنید، اطلاعات شما به‌صورت خودکار به stdout منتقل می‌شود که در پایتون این خروجی روی ترمینال تعریف شده است. خروجی stderr نیز به‌صورت پیش‌فرض روی همان ترمینال است، اما می‌توان آن را به محل‌های دیگری (مثلاً یک فایل متنی برای ذخیره کل پیغام‌های خطا) نیز هدایت کرد. از این خروجی‌ها و ورودی‌های متفاوت می‌توان برای تهیه logهای دقیق و کامل از اجرای نرم‌افزار استفاده کرد.

اکنون باید به سراغ تابع اصلی یا main برویم. این تابع که کدهای آن را در فهرست ۳ مشاهده می‌کنید، ابتدا اتصال یا connection و مکان‌نما یا cursor مورد نیاز را برای کار با پایگاه داده تعریف کرده و پس از آن با کنترل آرگومان‌هایی که کاربر هنگام اجرای برنامه وارد کرده و در صورتی که آرگومان‌ها درست باشند، اعمال اصلی برنامه را به انجام می‌رساند.

در اینجا همانند دو شماره پیشین، متغیرهای عمومی connection و cursor را برای کار با پایگاه داده تعریف کرده‌ایم. پس از آن پارامترهایی را که کاربر در خط فرمان وارد کرده است، کنترل کرده‌ایم. ما ورودی کاربر را برای یافتن ۲ پارامتر کنترل می‌کنیم که نخستین مورد، نام برنامه اجراشده و دومی آدرس پوشه موردنظر است. به خاطر داشته باشید که اگر آدرس شما حاوی کاراکتر فضای خالی (space) باشد، به عنوان دو پارامتر جدا در نظر گرفته شده و باعث بروز خطا می‌شود. در این حالت باید آدرس را در علامت‌های کوتیشن قرار دهید. اگر کاربر آدرس پوشه‌ای را وارد نکرده یا بیش از یک آدرس را وارد کرده باشد، با پیغام خطا روبه‌رو خواهد شد. اگر ورودی کاربر درست باشد، ابتدا در خط ۹ کنترل می‌کنیم که آیا چنین پوشه‌ای موجود است یا خیر. تابع ()exists که از ماجول os آورده شده است، با گرفتن یک آدرس در سیستم فایلی کامپیوتر وجود یا عدم‌وجود آن را کنترل می‌کند.

پس از آن و در خط ۱۶ تابع ساخت پایگاه‌داده (MakeDataBase) فراخوانده شده است. این تابع که در فهرست ۴ آورده شده، درصورت عدم وجود جدول mp3 در فایل mCat.db3 آن را بامشخصات مورد نظر ایجاد می‌کند. پس از آن با فراخوانی تابع ()WalkThePath کل پوشه داده شده بررسی و فایل‌های mp3 آن شناسایی شده و اطلاعات به پایگاه داده منتقل می‌شود. پس از آن هر دو شیء connection و cursor بسته شده‌اند. آوردن کدهای تابع WalkThePath، به دلیل طولانی بودن امکان‌پذیر نیست، اما می‌توانید آن را به صورت کامل از سایت ماهنامه شبکه به آدرس www.shabakeh-mag.com/Data/Files/Items/2012/1/WalkThePath.zip دریافت کنید.

در ایـن تابع ما ابتدا سـه شـمارنده برای ردگیری تعداد خطاها، پوشه‌ها و فایل‌ها ایجاد کرده‌ایـم. پـس از آن برای نگه‌داری سـوابق خطاهای احتمالی، فایلی با نام errors.log ایجاد شده است. پس از آن به کمک تابعی که ماجول os در اختیار ما قرار می‌دهد، به بررسی پوشه مورد نظر پرداخته‌ایم. سیستم کار تابع ()walk به این ترتیب است که کار را از پوشه داده شده توسط کاربر شـروع کرده و به تمام پوشه‌ها و زیرپوشـه‌های موجود در آن به ترتیب وارد شده و این روند را برای هر یک از زیرپوشه‌ها نیز تکرار می‌کند. در هرکدام از زیرپوشـه‌ها، ما به دنبال فایل‌هایی گشـته‌ایم که پسوندشان mp3 باشـد. پـس از آن بـه سـراغ هریک از فایل‌های یافت شـده رفته‌ایم و ابتدا متغیرهای محلـی مربوط به آن را از محتـوای قبلی پاک کرده‌ایـم. پـس از آن از تابـع ()join ماجـول os.path اسـتفاده کرده‌ایم و آدرس کامل فایل را برای موتاژن آمـاده کرده و آن را به عنوان آرگومان تابع ()MP3 موتاژن مورد اسـتفاده قرار می‌دهیم تا وهله‌ای از شیء audio را برای ما ایجاد کند. پس از آن کل برچسب‌های ID3 آن را خوانده و برچسب‌های موردنظرمان را جدا کرده و در متغیرهـای موقتی ذخیره می‌کنیم. آن‌گاه از این متغیرها برای ساختن دستور SQL و وارد کردن اطلاعات در پایگاه داده اسـتفاده می‌کنیم. تنها نکته قابل توجه شاید فرم دستور SQL باشـد. در اینجا نیز ما با فرمتی شـبیه %s در دسـتور print، محل مقادیر را با ؟ مشخص کرده‌ایـم و بعدتر هنگام اجرای دسـتور SQL ایـن متغیرها را در محـل موردنظر جایگذاری می‌کنیم.

در انتها در قسـمت کنتـرل خطاها با نوع دیگـری از قالب‌بنـدی رشـته‌ها و جایگذاری مقادیر مواجه می‌شـوید که شاید کمی عجیب باشد. این syntax برنامه‌نویسی یکی از اجزای اجباری سـری 3.x پایتون است. در این شیوه

```
1  def S2HMS(t):
2      if t > 3600:
3          h = int(t/3600)
4          r = t-(h*3600)
5          m = int(r / 60)
6          s = int(r-(m*60))
7          return '%d:%02d:%02d'.format(h,m,s)
8      else:
9          m = int(t / 60)
10         s = int(t-(m*60))
11         return '%d:%02d'.format(m,s)
```

فهرست ۵ کدهای مورد نیاز برای تکمیل تابع S2HMS

مقادیر {0} و {1}... با مقادیر متناظر در پرانتز format جایگزین می‌شوند.

کدهای مربوط به تابـع MakeDataBase را در فهرسـت ۴ مشـاهده می‌کنید. در اینجا با توجه به نکته‌هایی که در دو شماره قبل درباره ایجاد پایگاه‌های‌داده‌گفتیم، نباید در درک روند انجام کار این تابع مشـکلی وجود داشته باشد. در این مورد هم به سـادگی یک دسـتور SQL برای سـاخت یک جدول بـا فیلدهایی نظیر نام آلبوم، نام هنرمند، ژانر موسـیقی و... ساخته و اجرا شده است.

نکتـه مهمی که باید بـه آن توجه کنید این است که شـما باید پیش از اجرای برنامه، فایل پایگاه داده را به‌شـخصه سـاخته و در پوشه برنامـه قـرار دهیـد. کدهای این فهرسـت را جایگزیـن خطوط ۵ و ۶ فهرسـت ۱ کنید. البته می‌توانید به دلخواه خود کدهایی را به این تابع و تابـع ()main بیافزایید تا آدرس محل ذخیره پایگاه داده یا نام آن نیز از کاربر پرسیده شود و فایل موردنظر نیز درسـت همزمان با اجرای برنامه ایجاد شود.

در نهایـت، بـه بررسـی تابـع S2HMS می‌پردازیـم. ایـن تابـع که کدهـای آن را در فهرست ۵ مشـاهده می‌کنید، مدت هر آهنگ را که توسط موتاژن به صورت یک عدد اعشاری برگردانده می‌شود به فرمت ساعت:دقیقه:ثانیه تبدیـل می‌کند. به نحـوه مرتب کردن خروجی تابـع در دسـتورات return در خطوط ۷ و ۱۱ توجـه کنیـد. با عبـارت %02d مـا از پایتون می‌خواهیـم که عدد جایگزیـن را حتی اگر تک رقمـی بود، دو رقمی منظور کند و به جای رقم نخست آن 0 قرار دهد.

استفاده

با تکمیل آخرین تابع دیگر زمان اسـتفاده از برنامه فرا رسـیده اسـت. حال می‌توانید با دستور زیر برنامه را اجرا کنید:

```
python mCat.py <folder>
```

فقـط آدرس پوشـه محل ذخیره‌سـازی موسـیقی‌هایتان را جایگزین قسمت <folder> کنید. در این صورت پایگاه‌داده‌ای حاوی اطلاعات تمام فایل‌های موسیقی در اختیار خواهید داشت. بـا اندکی صبـر و حوصلـه و در صورت نیاز مراجعه به قسمت‌های قبلی، می‌توانید به سادگی کدی بنویسـید که بتواند آهنـگ یا آلبوم مورد نظر شـما را در این پایگاه داده جست‌وجو کرده و محل ذخیره و سایر اطلاعات آن را برای شما به نمایش درآورد.

کـد کامـل شـده برنامـه را نیـز می‌توانیـد از سـایت ماهنامـه و از آدرس www.shabakeh-mag.com/Data/Files/Items/2012/1/mCat.zip دریافت کنید.