این مار خوش خط و خال

آشنایی و کار با زبان برنامه‌نویسی پایتون

قسمت سوم

» احمد شریف پور

عده‌ای عقیده دارند که دوران برنامه‌های دسکتاپ دیگر به پایان رسیده است. محاسبات ابری، سرویس‌های رایگان اینترنتی، عرضه سیستم عامل‌های مبتنی برسرویس‌های وب و سایر تحولات دوره کنونی که گاهی آن را دوران Post-Desktop می‌نامند، همگی نشانه‌ای از تأثیر و قدرت نفوذ اینترنت در دنیای محاسبات هستند. اگرچه هنوز چندان در استفاده از زبان برنامه‌نویسی پایتون ماهر نشده‌ایم، اما این امر نمی‌تواند مانع استفاده ما از این دریای بی‌پایان امکانات باشد. در این قسمت از مجموعه، به معرفی ساده یک ماجول برای کار با داده‌های قالب XML خواهیم پرداخت و از طریق یک برنامه ساده با کمک APIهای یک سرویس‌دهنده وب اطلاعات آب و هوایی محل موردنظرمان را به‌دست خواهیم آورد.

شما به یقین اصطلاح XML را شنیده‌اید، هرچند ممکن است درباره آن اطلاعات زیادی نداشته باشید. در این قسمت آموزش زبان پایتون روی این موضوع متمرکز خواهیم شد. هدف این قسمت آشنا کردن شما با XML، نشان دادن روش خواندن و نوشتن فایل‌های XML در برنامه‌ها و کسب آمادگی برای نوشتن برنامه‌های مبتنی بر وب است.

XML (سرنام Extensible Markup Language) به لحاظ ساختاری بسیار شبیه HTML است. هدف از طراحی زبان XML ایجاد ساختاری برای ذخیره و انتقال بهینه اطلاعات از طریق اینترنت و سایر سیستم‌های ارتباطی است. یک فایل XML به طور معمول یک فایل متنی ساده است که به کمک برچسب‌ها یا تگ‌های دلخواه شما قالب‌بندی شده است که این برچسب‌ها و قالب‌ها باید تا حد ممکن گویا باشند. چون قالب فایل متنی است، به سادگی قابل فشرده شدن بوده و همین امر امکان انتقال سریع‌تر و راحت‌تر آن را از طریق اینترنت فراهم می‌آورد. برخلاف HTML قالب XML به تنهایی کاری انجام نمی‌دهد. نحوه نمایش اطلاعات را مشخص نمی‌کند و همچنین شما را به استفاده از تگ‌های خاص محدود نمی‌کند. کد زیر نمونه کلی یک فایل XML را نمایش می‌دهد.

```
<root>
        <node1> Data Here </node1>
        <node2  attribute=”something”>
        Node 2 Data
        </node2>
        <node3>
        <node3sub1>  more data </node3sub1>
        </node3>
</root>
```

فهرست ۱ نمونه یک فایل XML

نخستین نکته قابل توجه این است که تورفتگی‌ها و فاصله‌گذاری‌ها تنها در جهت ساده کردن درک کد برای انسان پیش‌بینی شده‌اند. کد فایل‌های XML به سادگی با فرم زیر نیز کار خواهد کرد.

```
<root><node1>
 Data Here </node1><node2  attribute=”something”>Node 2
Data</node2<node3><node3sub1>  more data </node3sub1>
</node3></root>
```

فهرست ۲ فایل XML فرمت خاص نوشتاری ندارد.

نشانه‌هایی که در علامت‌های “<>” قرار گرفته‌اند، تگ نامیده شده و از قوانین خاصی تبعیت می‌کنند. نخست این‌که باید تک کلمه‌ای باشند. دوم این‌که هر تگ آغازین باید تگ انتهایی متناظر خود را داشته باشد که با علامت “/” قبل از نام تگ مشخص می‌شود. همچنین به خاطر داشته باشید که تگ‌ها به بزرگی و کوچکی حروف حساس هستند. بنابراین <Node1> و <noDe1> و <NODE1> هریک تگ‌های متفاوتی بوده و باید تگ انتهایی جداگانه‌ای داشته باشند. نام تگ‌ها می‌تواند متشکل از حروف، اعداد و سایر کاراکترها باشد، اما نباید با اعداد یا نشانه‌های خاص شروع شوند. همچنین در نام تگ‌ها از علامت‌های “.” و “-” و “:” استفاده نکنید، زیرا برخی نرم‌افزارها این علامت‌ها را به عنوان

دستور تلقی می‌کنند. به هریک از تگ‌ها عنصر یا المان هم گفته می‌شود. به طور معمول بین هر تگ آغازین و تگ پایانی مرتبط با آن داده‌هایی وجود خواهد داشت. درواقع وظیفه تگ‌های XML معرفی و طبقه‌بندی این داده‌ها است.

هر فایل XML درواقع شبیه یک درخت است که از تگ Root یا ریشه (تگی که تمام تگ‌های دیگر در داخل آن قرار گرفته‌اند) شروع شده و سپس در قسمت‌های مختلف چندشاخه می‌شود. تمام فایل‌های XML باید یک تگ ریشه (نه لزوماً با نام root) داشته باشند که درواقع والد هر چیز دیگری در داخل فایل است. در نمونه فوق تگ Root سه تگ فرزند دارد که به ترتیب node1 و node2 و node3 هستند. و تگ node3 خود والد تگ node3sub1 است.

همچنین به node2 هم توجه کنید که به غیر از داده‌های مرتبط، از چیزی به نام خاصیت یا attribute استفاده می‌کند. این روزها بیشتر توسعه‌دهندگان از خاصیت‌ها پرهیز می‌کنند، زیرا خود تگ‌ها به تنهایی کارآمد هستند، اما هنوز هم استفاده از خاصیت‌ها رواج دارد.

بهتر است مثال مفیدتری را بررسی کنیم. به فهرست ۳ نگاه کنید. در این فهرست شما تگ ریشه people را مشاهده می‌کنید که خود دارای دو تگ فرزند با نام person است. هر تگ فرزند person خود شش تگ فرزند با نام‌های firstname، lastname، gender، address، city و state دارد. این فایل در نگاه نخست ممکن است شبیه یک پایگاه داده به نظر برسد و در واقع برخی نرم‌افزارها نیز از فایل‌های XML به عنوان یک ساختار داده ساده استفاده می‌کنند.

```
<people>
        <person>
                <firstname>Samantha</firstname>
                <lastname>Pharoh</lastname>
                <gender>Female</gender>
                <address>123 Main St.</address>
                <city>Denver</city>
                <state>Colorado</state>
        </person>
        <person>
                <firstname>Steve</firstname>
                <lastname>Levon</lastname>
                <gender>Male</gender>
                <address>332120 Araphoe Blvd.</address>
                <city>Denver</city>
                <state>Colorado</state>
        </person>
</people>
```

فهرست ۳ کد فایل xmlsample1.xml

اکنون دیگر می‌توانیم به نوشتن یک برنامه ساده برای خواندن این فایل بپردازیم. شما به راحتی می‌توانید فایل را باز کرده، خط به خط بخوانید و براساس المان‌ها با داده‌ها کار کنید و سپس فایل را ببندید. اما در اینجا برای سهولت کار ما از یک ماجول کتابخانه‌ای به نام ElementTree استفاده می‌کنیم. شما می‌توانید به سادگی با مراجعه به آدرس http://effbot.org/downloads/#elementtree نسخه مورد نیاز را برای سیستم عامل خود دانلود کنید. ما از نسخه elementtree-1.2.6-20050316.tar.gz استفاده می‌کنیم. پس از اتمام دانلود، آن را در یک پوشه موقتی از حالت فشرده خارج کنید، سپس در پنجره ترمینال به آن پوشه رفته و دستور زیر را وارد کنید:

```
sudo python setup.py install
```

این دستور فایل‌های مربوط به این ماجول را در پوشه اصلی پایتون قرار می‌دهد و از آن پس می‌توان آن را در تمام برنامه‌های پایتون استفاده کرد. کاربران ویندوز به جای این دستورات باید فایل elementtree-1.2.6-20050316.win32.exe را از همان آدرس دانلود کرده و اجرا کنند.

برای پروژه این شماره یک پوشه بسازید و در ویرایشگر متن دلخواه خود، کدهای فهرست ۳ را تایپ کرده و با نام xmlsample.xml در آن پوشه ذخیره کنید. ابتدا با نوشتن کد فهرست ۴ از صحت نصب ماجول ElementTree اطمینان حاصل می‌کنیم. این کد را با نام code1.py ذخیره و اجرا کنید.

```
import elementtree.ElementTree as ET
tree=ET.parse('xmlsample1.xml')
ET.dump(tree)
```

فهرست ۴ برنامه code1.py

در صورتی که همه مراحل با موفقیت انجام شده باشد باید خروجی شما دقیقاً مطابق فهرست ۳ باشد. تنها کاری که ما انجام دادیم وادار کردن ElementTree به باز کردن فایل، تحلیل آن و چاپ دوباره نتایج تحلیل (در اینجا دقیقاً همانند فایل اصلی) است. حال کد فهرست ۵ را تایپ کرده و با نام code2.py ذخیره و اجرا کنید. نتیجه باید مطابق فهرست ۶ باشد.

```
1 import elementtree.ElementTree as ET
2 tree=ET.parse('xmlsample1.xml')
3 person=tree.findall('.//person')
4 for p in person:
5       for dat in p:
6               print "Element: %s \
7               Data: %s" %(dat.tag,dat.text)
```

فهرست ۵ برنامه code2.py

```
Element: firstname Data: Samantha
Element: lastname Data: Pharoh
Element: gender Data: Female
Element: address Data: 123 Main St.
Element: city Data: Denver
Element: state Data: Colorado
Element: firstname Data: Steve
Element: lastname Data: Levon
Element: gender Data: Male
Element: address Data: 332120 Araphoe Blvd.
Element: city Data: Denver
Element: state Data: Colorado
```

فهرست ۶ خروجی برنامه code.2

شکل ۱ نمونه نتایج معمول Wunder Ground برای شهر تهران

در اینجا هر داده‌ای را با المان مربوط به آن ردیف کرده‌ایم. در خط ۱، ماجول ElementTree را import کرده‌ایم. در خط ۲ از ElementTree خواسته‌ایم که فایل را تحلیل کرده و نتیجه را در متغیری به نام tree ذخیره کند. در خط بعد از ماجول ElementTree خواسته‌ایم که تمام نمونه‌های موجود از تگ person را درون متغیر tree بیابد که در این مثال دو نمونه وجود داشته است.

فایل‌های دیگر شما ممکن است محتوی یک یا هزار تگ person باشد. در خطوط ۴ و ۵ ما یک حلقه ساده for ساخته‌ایم که تمام نمونه‌های person را پویش کرده، اطلاعات آن‌ها را با استفاده از خصوصیات tag. و text. نمایش می‌دهد.

توجه کنید که متد findall ساختار درختی را براساس نام تگ یا آدرس آن جست‌وجو می‌کند. به عبارت دیگر برای جست‌وجوی تگ firstname هم می‌توانید خود عبارت 'firstname' را جست‌وجو کنید و هم از آدرس آن در ساختار فایل XML به شکل 'person//firstname//.' برای جست‌وجو بهره ببرید. فقط به خاطر داشته باشید findall تمام نمونه‌های یافت شده را (حتی اگر تنها یک نمونه باشد) در قالب یک لیست برمی‌گرداند. به همین دلیل توانستیم در حلقه for خط ۴ از متغیر person عنوان شمارنده حلقه استفاده کنیم.

یک مثال واقعی

اکنون که با فرمت XML و ماجول ElementTree کم‌وبیش آشنا شده‌ایم، می‌توانیم از دانش خود برای استفاده از سرویس‌های اینترنتی که معمولاً بر پایه XML بنا شده‌اند، بهره ببریم. در مثال بعدی ما از APIهای ارائه شده برای سایت http://wunderground.com استفاده می‌کنیم. شاید بپرسید API چیست؟

API سرنام (Application Programming Interface) است. این اصطلاح عجیب و غریب یکی از راه‌های تعامل برنامه‌ها با یکدیگر است. به کتابخانه‌هایی که در مثال‌های قبل import کردیم، فکر کنید. برخی از این کتابخانه‌ها می‌توانند به تنهایی اجرا شوند، اما اگر آن‌ها را import کنید می‌توانید از برخی توابع آن به‌صورت توکار استفاده کنید.

APIها نقش فراهم کردن دسترسی را بین برنامه‌های مختلف با زبان‌ها و سیستم‌های مختلف بر عهده دارند. عده‌ای به API به دید درهای مخفی در برنامه‌ها نگاه می‌کنند که از طریق آن‌ها می‌توانیم یک برنامه را مجبور کنیم عملیات مورد نظرمان را انجام دهند. در این مثال از یک url خاص به همراه یک کلمه کلیدی برای جست‌وجو و استخراج اطلاعات مربوط به وضعیت آب و هوای شهر موردنظرمان از طریق نرم‌افزارهای سایت http://wunderground.com استفاده خواهیم کرد. مرورگر دلخواهتان را باز کرده و آدرس این سایت را در آن وارد کنید. در جعبه جست‌وجوی بالای صفحه کدپستی یا نام شهر، استان یا نام کشور را وارد کنید.

در این صورت با حجم انبوهی از اطلاعات آب و هوایی درباره محل موردنظر روبه‌رو خواهید شد. شکل ۱ نتیجه جست‌وجوی شهر تهران را نمایش می‌دهد. برای استفاده از امکانات این سایت مجبور نیستید، همواره از طریق مرورگر اینترنت خود به آن مراجعه کنید. به جای این کار می‌توانید از APIهای این سایت استفاده کنید.

برای کسب اطلاعات بیشتر درباره APIهای این سایت، نحوه استفاده و... به آدرس زیر مراجعه کنید:

```
http://wiki.wunderground.com/index.php/API_-_XML
```

بـا مطالعه ایـن صفحه متوجه خواهید شـد که می‌تـوان به کمک تعـدادی آدرس خـاص اینترنتـی و افزودن نام شـهر یا کدپسـتی به انتهای آن‌ها، اطلاعات مربوط به یک شـهر یا ایسـتگاه هواشناسـی را جست‌وجو کرد.

به عنـوان مثال، تابـع GeoLookupXML مشـخصات جغرافیایی یـک محل، کـد فرودگاه‌های نزدیـک آن و طول و عـرض جغرافیایی محـل را در قالـب یـک فایـل XML به مـا بازخواهـد گرداند یـا تابع WXCurrentObXML شـرایط فعلی آب و هوای یـک منطقه را در قالب XML برگشـت خواهـد داد. بـرای اسـتفاده از این تابع یـا API کافی اسـت نام شـهر یا محل موردنظـر را پس از علامت مسـاوی انتهای url، جایگزیـن KSFO کنیـد. بـه عنوان مثـال، مراجعه بـه آدرس زیر اطلاعـات آب و هوایی شـهر تهران را همانند شـکل ۲ برای ما نمایش خواهد داد.

```
http://api.wunderground.com/auto/wui/geo/
WXCurrentObXML/index.xml?query=Tehran
```

در صفحـه باز شـده به راحتـی می‌توان با کنتـرل تگ‌هایی نظیر <wind_string> یـا <temp_c> بـه اطلاعاتـی نظیر سـرعت وزش بـاد و دما دسـت یافـت. حال بـرای اسـتفاده از این قابلیـت برنامه فهرسـت ۷ را تایپ، با نام weather.py ذخیره و اجرا کنید. پس از اجرا، این برنامه نام شـهر موردنظر کاربر را پرسـیده و با ارسال آن به API سایت wunderground اطلاعات آب و هوایی نظیر دما و سرعت وزش باد را نمایش می‌دهد.

اگر نام شـهر مورد نظر اشـتباه وارد شـود یا چنین شهری دارای ایستگاه هواشناسی نباشـد، پیغامی مبنی بر عدم وجود اطلاعات چاپ خواهد شد و با وارد کردن کلمه end برنامه به اتمام خواهد رسید. لازم به یادآوری نیسـت که برای اجرای درست برنامه دسترسی به اینترنت ضروری است.

```
#! /usr/bin/env python
import urllib
import elementtree.ElementTree as ET
### Creating the request
api="http://api.wunderground.com/auto/wui/geo/WXCurrentObXML/
index.xml?query="
items_name = ["wind_string" , "temp_c","weather"]
### Functions
def getWeather(loc):
   request=urllib.urlopen(loc)
   weather=ET.parse(request)
   return weather
def extractItems(weather,items_name):
   items=[]
   for i in items_name:
      items.append(weather.findall(i))
   return items
def printResult(items):
   if items[1][0].text == None:
      print "\nWe found nothing about this city . . ."
   else:
      print
      print "============================================="
      print "==       Weather of %-22s==" %location.upper()
      print "============================================="
      for i in range(len(items)):
         print "== %-11s: %-25s ==" %(items[i][0].tag,items[i]
[0].text)
         print "============================================="
### Main Loop
while True:
   location = raw_input("\nEnter city name: ")
   if location.upper()=="END":
      break
   loc = api + location
   weather = getWeather(loc)
   items=extractItems(weather,items_name)
   printResult(items)
```

فهرست ۷ کد برنامه weather.py

شکل ۲ نمونه فایل XML برگشت داده شده از API سایت http://wunderground.com

```
Enter city name: Tehran

==============================================
==         Weather of TEHRAN                ==
==============================================
==  wind_string:  From the NW at 5 MPH      ==
==============================================
==  temp_c     :  12                        ==
==============================================
==  weather    :  haze                      ==
==============================================

Enter city name: Dezful

We found nothing about this city . . .

Enter city name: Shiraz

==============================================
==         Weather of SHIRAZ                ==
==============================================
==  wind_string:  Calm                      ==
==============================================
==  temp_c     :  8                         ==
==============================================
==  weather    :  Scattered Clouds          ==
==============================================

Enter city name: end
```

شکل ۳ نمونه اجرای برنامه weather.py

در خط ۱ مانند مثال‌های شماره قبلی، شی‌بنگ آورده شده است. در خط ۲ و ۳ با دستور import دو کتابخانه urllib برای کار با آدرس‌های اینترنتی و elementtree برای کار با شیء XML به برنامه وارد شده‌اند. در خطوط ۵ و ۷ دو متغیر api (محتوی آدرس اینترنتی API سایت) و items_name (محتوی فهرست تگ‌هایی که می‌خواهیم از فایل استخراج کنیم) تعریف شده‌اند. شما با بررسی ساده فایل XML سایت http://wunderground.com (همانند شکل ۲) می‌توانید تگ دلخواه خودتان نظیر فشار هوا، دمای شبنم و... را نیز به انتهای لیست متغیر items_name اضافه کرده و نتیجه را در خروجی برنامه مشاهده کنید. در خط ۹ تابعی تعریف شده که توسط متد urlopen از کتابخانه urllib، با ارسال درخواست کاربر فایل حاصل از API سایت را دریافت می‌کند. سپس این فایل توسط متد parse از کتابخانه elementtree پویش شده و نتیجه در متغیر weather ذخیره شده و برگشت داده می‌شود. تابع تعریف شده در خطوط ۱۳ تا ۱۷ با چک کردن متغیر weather برای یافتن تگ‌هایی که ابتدای برنامه در متغیر _items name ذخیره شده‌اند، اطلاعات موردنظر را از فایل XML استخراج کرده و نتیجه را به انتهای لیست items اضافه می‌کند. پس از آن items به حلقه اصلی برنامه برگردانده می‌شود.

تابع تعریف شده در خط ۱۸ برای چاپ نتایج جست‌وجو در نظر گرفته شده است. در خط ۱۹ اگر برای آیتم دما مقداری یافت نشده باشد، اعلام می‌شود که شهر موردنظر در پایگاه داده وجود نداشته است و در غیر این صورت با یک حلقه، کل اطلاعات items چاپ خواهد شد.

نخستین نکته قابل توجه این است که خروجی متد findall همواره یک لیست است، حتی اگر تنها یک نمونه از تگ موردنظر در فایل XML موجود باشد. چون در خط ۱۶ این لیست به متغیر item که خود یک لیست دیگر است اضافه شده، ما برای استخراج مقادیر آن از دو اندیس استفاده کرده‌ایم. بنابراین عبارت items[0][1].text به آیتم متن ذخیره شده در نخستین عضو دومین لیست موجود در درون لیست items اشاره می‌کند که همان مقدار دمای محل مورد نظر است (به محل تگ temp_c در متغیر items_name توجه کنید).

نکته دوم این‌که سیستم به کار رفته برای فرمت کردن رشته‌ها و متغیرها با شکل جدید 25s-% ظاهر شده است. در اینجا عدد قبل از s اندازه فضای خالی پیش‌بینی شده را تعیین می‌کند. در چنین حالتی حتی اگر طول متغیر مورد نیاز ۵ کاراکتر باشد، طول آن با افزودن ۲۰ کاراکتر فاصله به ۲۵ خواهد رسید. این فضاهای خالی در حالت عادی در سمت چپ رشته افزوده می‌شوند که علامت منفی آن‌ها را به سمت راست رشته اصلی منتقل می‌کند. بنابراین به عنوان مثال، رشته "Tehran" به رشته " Tehran" تبدیل خواهد شد. تمام اعداد به کار رفته در این مثال با آزمایش و خطا و برای تنظیم جدول ایجاد شده با کاراکترهای "=" به‌دست آمده‌اند.

حلقه اصلی برنامه در خط ۳۱ آغاز شده و نام محل موردنظر از کاربر می‌پرسد و با فراخوانی ترتیبی توابع، چرخه برنامه ادامه می‌یابد تا زمانی که کاربر به جای نام شهر کلمه end را وارد کند. شکل ۳ نمونه اجرای این برنامه را نشان می‌دهد.

اگر مطالب گفته شده در این قسمت و قسمت‌های قبل را دنبال کرده باشید باید بتوانید برنامه را به دلخواه خود برای دریافت اطلاعات دیگر تغییر داده یا با مطالعه راهنماهای موجود در سایت http://wiki.wunderground.com سایر APIهای آن را به کار بگیرید. در قسمت‌های بعدی به مباحث برنامه‌نویسی شیء گرا و رابط گرافیکی کاربر خواهیم پرداخت.