این مار خوش خط و خال

قسمت هشتم

برنامه‌نویسی به زبان پایتون

» احمد شریف‌پور

سکون و ایستایی در دنیای نرم‌افزار مترادف مرگ است. همه نرم‌افزارها برای ادامه حیات خود محتاج به روزآوری و افزوده شدن قابلیت‌ها و ویژگی‌های جدید هستند. در این میان با گسترش کاربرد و نفوذ کامپیوترها در جنبه‌های مختلف زندگی ما، زبان‌های برنامه‌نویسی بیش از سایر نرم‌افزارها به این پویایی و تحول نیاز دارند. بر همین اساس، زبان پایتون نیز به‌صورت مداوم در حال بازبینی و بهینه‌سازی است. در این مورد خاص و با توجه به حجم عظیم تغییرات ایجاد شده در نسخه جدید، بنیاد نرم‌افزار پایتون (Python Software Foundation) تصمیم گرفته، ارتقا به سری جدید را به تدریج انجام دهد، یعنی روند توسعه سری جدید در کنار سری جاری و به‌صورت همزمان انجام می‌پذیرد تا زمانی که بنیاد به این نتیجه برسد، بیشتر کاربران آمادگی انتقال به سری جدید را دارند. در این قسمت تصمیم گرفتیم، به معرفی مختصر خصوصیات و تفاوت‌های نسل جدید پایتون با نسل قبل بپردازیم.

نسل سوم

همان‌گونه که پیش‌تر اشاره شد، در حال‌حاضر توسعه پایتون در دو مسیر موازی در حال انجام است. گروه نخست توسعه‌دهندگان همچنان به کار روی سری دوم پایتون (2.x) مشغول هستند که به‌دلیل قدمت و سابقه بیشتر، پرطرفدارتر است. در حال حاضر بیشتر فریم‌ورک‌ها و کتابخانه‌های رایج و پراستفاده برای این سری توسعه داده شده و پیاده‌سازی می‌شوند. تمام مثال‌ها و برنامه‌هایی که تاکنون معرفی کرده‌ایم نیز از همین سری استفاده می‌کنند.

این نسل از زبان پایتون در اکتبر سال ۲۰۰۰ معرفی شد و اکنون به نسخه ۲/۷/۱ رسیده است. گروه دوم توسعه‌دهندگان به کار روی نسل جدید این زبان، یعنی پایتون 3.x می‌پردازند که تاکنون نسخه ۳/۲ آن نیز منتشر شده است. در این سری وان روسوم (GuidovanRossum) خالق پایتون، تصمیم گرفته به بازنگری اساسی در تمام جنبه‌های این زبان برنامه‌نویسی و حذف BackwardCompatibility (پشتیبانی از اجرای کدهای نوشته شده با نسل دوم این زبان) بپردازد.

هدف این است که سری جدید پایتون، با بنیادی متفاوت و با شروعی دوباره، علاوه بر ساده‌سازی دستور زبان افزایش خوانایی، امکانات و ویژگی‌های جدیدی را نیز به ارمغان آورد.

در این نسخه سینتکس بسیاری از دستورها برای کارایی بهتر، یادگیری ساده‌تر و یکنواخت شدن با سایر اجزای پایتون تغییر کرده و پاره‌ای از دستورات با همان سینتکس قدیمی، کارایی تازه‌ای پیدا کرده‌اند. ما در این قسمت تنها به معرفی برخی تغییرات اساسی خواهیم پرداخت که به صورت خاص به دستوراتی مربوط می‌شوند که در قسمت‌های قبلی از آن‌ها استفاده کرده‌ایم.

برای آشنایی عمیق‌تر با جنبه‌های نو و تغییرات نسل سوم زبان پایتون می‌توانید از منابع فراوانی که در اینترنت موجود است، استفاده کنید. در انتهای این قسمت آدرس تعدادی از این مراجع را ذکر خواهیم کرد.

دستور PRINT

یکی از مهم‌ترین و پرکاربردترین بخش‌های پایتون که در این نسخه دستخوش تغییرات اساسی شده، دستور print است. در نسخه‌های 2.x این زبان به راحتی می‌توانستیم آیتم مورد نظر را به فرم زیر چاپ کنیم:

```
print “This is a test”
```

اما در سری جدید پایتون، چنین دستوری با پیغام خطا مواجه خواهد شد. در واقع نسخه جدید دستور print را همانند یک تابع پیاده‌سازی کرده است و همانند هر تابع دیگری برای ارسال آرگومان‌ها به تابع باید از پرانتز استفاده کنیم. یک دستور ساده print در نسخه سوم به فرم زیر است:

```
print (“This is a test”)
```

هرچند این تغییر به ظاهر ساده است، اما در ادامه خواهیم دید که تغییرات دستور print به همین موارد خلاصه نخواهد شد.

```
>>> scores = ["10" , "100" , "1000"]
>>> times = [30 , 60 , 90 , 120]
>>> print "You get %s scores in about %d seconds." %(scores[1],times[2])
You get 100 scores in about 90 seconds.
>>>
```

فهرست ۱ قالب‌بندی و جای‌گذاری متغیرها در پایتون 2.X

```
>>> scores = ["10" , "100" , "1000"]
>>> times = [30 , 60 , 90 , 120]
>>> print("You get %d scores in about %d seconds.",format(scores[1],times[2]))
You get 100 scores in about 90 seconds.
>>>
```

فهرست ۲ قالب‌بندی و جای‌گذاری متغیرها در پایتون 3.X

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [M
Type "copyright", "credits" or "license()" for m
>>> name = input("Enter your name: ")
Enter your name: Ali

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    name = input("Enter your name: ")
  File "<string>", line 1, in <module>
NameError: name 'Ali' is not defined
>>>
```

فهرست ۳ در پایتون 2.X دستور input تنها ورودی عددی را می‌پذیرد.

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59)
Type "copyright", "credits" or "license()" for
>>> 12 > "Ali"
False
>>>
```

فهرست ۴ در سری 2.X می‌توان انواع داده نامربوط را با هم مقایسه کرد.

```
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02)
Type "copyright", "credits" or "license()" for
>>> 12 > "Ali"
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    12 > "Ali"
TypeError: unorderable types: int() > str()
>>>
```

فهرست ۵ در پایتون 3.X تنها انواع داده مشابه قابل مقایسه هستند.

```
Python 2.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (Intel)] on
Type "copyright", "credits" or "license()" for more information.
>>> newSerie="3.x"
>>> oldSerie="2.x"
>>> print("Using %s syntax in %s environment.".format(newSerie , oldSerie))
Using 3.x syntax in 2.x environment.
>>>
```

فهرست ۶

قالب‌بندی و جای‌گذاری متغیرها

در نسخه قبلی پایتون ما از کدهایی برای قالب‌بندی و جای‌گذاری متغیرها در یک رشته استفاده می‌کردیم (همانند شکل ۱). اگرچه این عبارت در پایتون ۳/۲ نیز به درستی کار خواهد کرد، اما این وضعیت در حال تغییر است، زیرا توابع قالب‌بندی %s و %d در نسخه‌های آینده حذف خواهند شد و سیستم نگارشی {x} جایگزین آن خواهد (شکل ۲). در این روش عبارت‌های معرفی شده به عنوان آرگومان‌های متد format، به ترتیب جایگزین {x}ها خواهند شد. همان‌طور که مشاهده می‌کنید، خوانایی این روش جدید بسیار بیشتر از حالت قبلی ارجاع به متغیرها و قالب‌بندی آن‌ها است.

اعداد

در سری 2.x پایتون عبارت 5/2.0 حاصلی برابر با2.5 داشت. اما اگر از عبارتی مانند 5/2 استفاده می‌کردید، به‌واسطه تبدیل نتیجه به فرمت عدد صحیح (به‌واسطه عدد صحیح بودن هر دو عملوند عبارت 5/2) حاصلی برابر 2 دریافت می‌کردید. در سری 3.x پایتون استفاده از هر دو فرم قبلی نتیجه‌ای برابر با 2.5 ایجاد خواهد کرد. و در صورتی که نیاز داشته باشید تقسیم را به‌صورت صحیح و بدون اجزای اعشاری انجام دهید، باید از فرمت 5//2 استفاده کنید.

دستور INPUT

در سری 2.x پایتون، دستور input تنها ورودی‌های عددی را می‌پذیرفت و در صورت ورود رشته‌ها همانند شکل ۳ با پیغام خطا روبه‌رو می‌شدیم. برای دریافت رشته‌ها از کاربر مجبور بودیم از دستور raw_input استفاده کنیم. اما در سری 3.x دستور input با تمام ورودی‌ها همانند رشته رفتار می‌کند و در صورتی که به دریافت اعداد نیاز داشته باشید، باید ورودی‌های دستور input را به صورت دستی به عدد تبدیل کنید.

نامساوی و شرطی‌ها

در سری 2.x پایتون برای کنترل نامساوی‌بودن عبارت‌ها از عملگر <> استفاده می‌شد، اما این شیوه در سری 3.x دیگر مجاز نیست و به جای آن باید از عملگر != استفاده کرد. نکته دیگر این‌که در سری 2.x در صورت مقایسه دو عملوند از نوع مختلف (مثلاً مقایسه یک رشته با یک عدد صحیح) همانند شکل ۴ نتیجه‌ای برابر با false یا نادرست بازگردانده می‌شد. به عبارت دیگر، خطایی رخ نمی‌داد. اما در سری 3.x باید حتماً، از یکسان بودن (به عبارت کلی‌تر، قابل مقایسه بودن) عملگرها اطمینان حاصل کرد. در غیر این صورت، همانند شکل ۵ با خطای Type Error مواجه خواهیم شد.

سایر تغییرات

انواع داده بنیادین در پایتون نیز دستخوش تغییر شده‌اند. دو نوع داده int و long در قالب یک نوع داده، یعنی int ادغام شده‌اند و متدهای مربوط به دیکشنری‌ها نظیر ()dict.keys به جای بازگرداندن یک لیست، نوع جدیدی به نام view را باز می‌گردانند. تابعی نظیر ()range هم به جای یک لیست یک شمارنده یا Iterator را باز می‌گرداند.

همچنین تغییرات فراوانی نیز در کتابخانه‌های استاندارد پایتون ایجاد شده است. به عنوان مثال، می‌توان به حذف توابع مربوط به گوفر (gopherlib) و جایگزین شدن md5 با hashlib اشاره کرد. البته، همان‌طور که در ابتدا نیز اشاره شد، تغییرات ایجاد شده بسیار زیاد بوده و توضیح کامل همه آن‌ها در این مقاله نمی‌گنجد.

تبدیل برنامه‌ها به سری جدید

سری 3.x پایتون با ابزاری برای تبدیل کدهای سری‌های قبلی به سری جدید همراه شده است. اگرچه این ابزار در همه موارد به درستی عمل نمی‌کند، اما در بسیاری از موارد شما را به هدف بسیار نزدیک خواهد کرد. این ابزار نام معنی‌دار 2to3 را یدک می‌کشد. در لینوکس برای استفاده از این ابزار از طریق ترمینال دستور زیر را وارد می‌کنیم:

```
2to3 [-w] <input_file.py<
```

توجه داشته باشید که این دستور به شرطی عمل خواهد کرد که سری 3.x پایتون روی لینوکس نصب شده باشد. در اوبونتو برای نصب این سری جدید از دستور زیر استفاده کنید:

```
sudo apt-get install python3
```

این ابزار در ماشین‌های ویندوزی با نام 2to3.py در پوشه c:\Python3x\Tools\Scripts قرار دارد و برای استفاده از آن باید از خط فرمان و در پوشه گفته شده دستور زیر را صادر کرد:

```
python.exe  2to3.py [-w] <input_file.py>
```

این برنامه در حالت عادی تنها تغییرات لازم را برای تبدیل کد، روی صفحه نمایش چاپ خواهد کرد و فایل را تغییر نخواهد داد. استفاده از سوییچ اختیاری w– باعث می‌شود، تغییرات مورد نیاز به فایل اصلی اعمال شود. در صورت استفاده از این سوییچ یک نسخه پشتیبان از فایل تهیه شده و تغییرات موردنیاز به طور مستقیم به فایل اعمال می‌شود.

آیا باید به سری 3.x مهاجرت کنیم؟

تغییرات سینتکس در همه نسخه‌های جدید زبان‌های برنامه‌نویسی به چشم می‌خورد. میان برهایی نظیر =+ ممکن است به سادگی کار ما را در هنگام کدنویسی ساده‌تر کنند. اما همین تغییرات ساده سینتکس جنبه‌های منفی نیز در خود دارند. درباره زبان پایتون نیز به دلیل همین تغییرات (گاهی ساده) در سینتکس، بسیاری از ماجول‌ها و کدهایی که قبلاً استفاده می‌کردیم، در برنامه‌هایی که با سری 3.x پایتون نوشته می‌شوند، قابل استفاده نیستند. به عنوان نمونه می‌توان به کتابخانه‌های ElementTree اشاره کرد که در بخش مربوط به xml از آن استفاده کردیم. اما با تمام این مشکلات نباید از مهاجرت به سری 3.x پایتون نا امید شد.

خوشبختانه پایتون سری 2.x (حداقل از نسخه ۲/۶ به بعد) در عین سازگاری با نسخه‌های قبلی، از تمام سینتکس‌های مورد نیاز برای نوشتن کد به سبک سری 3.x پشتیبانی می‌کند (فهرست ۶). پیشنهاد می‌کنیم که از همین حالا نوشتن کد به سبک سری 3.x را آغاز کرده و اگر می‌توانید تنها به کتابخانه استاندارد پایتون بسنده کنید و به ماجول‌های دیگر رجوع نکنید، حتماً به 3.x پایتون مهاجرت کنید. به کمک ابزار انتقالی که پیش‌تر از آن صحبت کردیم، تقریباً می‌توانید به سادگی تمام مجموعه کدهایی را که تاکنون نوشته‌اید، به سری 3.x ارتقا دهید. اما حتی اگر نیازمند کتابخانه‌هایی هستید که هنوز به سری 3.x منتقل نشده‌اند، همواره گوشه چشمی به این سری جدید داشته باشید. زیرا دیر یا زود کدهای موردنظر شما نیز به سری 3.x ترجمه خواهند شد.

مطالعه بیشتر

برای آشنایی بیشتر با تغییرات و نحوه مهاجرت به این سری جدید می‌توانید به منابع زیر مراجعه کنید:

- http://wiki.python.org/moin/Python2orPython3
- http://docs.python.org/release/3.1.2/whatsnew/3.0.html
- http://docs.python.org/library/2to3.html
- http://diveintopython3.org