پایتون برای همه

نویسنده: دکتر چارلز سورنس
مترجم: دیانا مظهری

فصل 1

چرا باید برنامه نوشتن را یاد بگیرید؟

نوشتن برنامه (یا همان برنامه نویسی) یک کار بسیار خلاقانه است و موجب احساس موفقیت می شود. شما می توانید برای دلایل متفاوتی برنامه بنویسید، از کسب درآمد گرفته تا حل یک مسئله سخت بررسی داده ها تا سرگرمی و تفریح با کمک به دیگران برای حل مسئله. این کتاب فرض می کند که همه باید بتوانند برنامه بنویسند و وقتی که برنامه نویسی را یاد بگیرید، شما به آن کاری که می خواهید انجام دهید پی میبرید.

اطراف ما در زندگی روزانه پر از کامپیوتر ها گوناگونی است، از لپ تاپ ها تا تلفن های همراه. ما می توانیم این کامپیوتر ها را به عنوان "دستیار شخصی" بدانیم که می تواند کارهایمان را برای ما انجام دهد. نرم افزار های امروزه به طوری ساخته شده اند که به طور پیوسته از ما بپرسند، "چه کار دیگری می خواهی انجام بدهم؟"

برنامه نویس ها یک سیستم عامل و یک دسته اپلیکیشن به سخت افزار اضافه می کنند و ما یک دستیار شخصی دیجیتال به دست می آوریم که به طور قابل توجهی مفید و می تواند به ما در انجام کار های بسیار کمک کند.

کامپیوتر های ما سریع هستند و دارای حافظه عظیمی هستند و می توانستند برای ما خیلی سودمند باشند اگر تنها ما زبان آنها را می دانستیم تا با آنها حرف بزنیم و به آن بگوییم که میخواهیم چه کاری را برای ما انجام دهد. اگر ما این زبان را می دانستیم، می توانستیم به کامپیوتر بگوییم که چه کار های تکراری را از طرف ما انجام دهد. جالب است، فعالیت هایی که ماشین های کامپیوتری به بهترین حالت انجام می دهند، کار های مکرری هستند که انسان ها انجام آن را خسته کننده می دانند.

شکل 1.1: دستیار شخصی دیجیتال
شکل 1.2: حرف زدن برنامه نویس ها با شما

برای مثال، به سه پاراگراف اول این فصل نگاه کنید و به من پرتکرار ترین کلمه و آن چند بار استفاده شده را بگویید. با اینکه شما توانستید متن را در عرض چند ثانیه بخوانید و درک کنید، شمردن آنها کار بسیار کسل کننده ای است چون این کاری نیست که ذهن انسان برای انجام آن ساخته شده است. برای یک کامپیوتر، برعکس آن درست است، خواندن و درک متن از روی یک تکه کاغذ برای یک کامپیوتر سخت است ولی شمردن کلمات و اعلام این که چند بار این کلمه تکرار شده است برای کامپیوتر آسان است:

```
python words.py
Enter file: words.txt
to 16
```

"دستیار شخصی بررسی مشخصات" ما به سرعت به ما گفت که کلمه "--" – دفعه در سه پاراگراف اول این فصل استفاده شده است.

همان این نکته که کامپیوتر ها در انجام کار هایی که انسان ها توانایی انجام آن را ندارند، دلیل بر آن است که شما نیاز دارید که در صحبت کردن با "زبان کامپیوتر" باید مهارت پیدا کنید. هنگامی که شما این زبان جدید را یاد بگیرید، شما می توانید که این فعالیت روزمره و تکراری را به عهده همکارتان بگذارید (همان کامپیوتر) و زمان بیشتری برای انجام کار هایی که به طور خاص مناسب شما است داشته باشید. شما به این همکاری خلاقیت، حس و توانایی کشف اضافه می کنید.

1.1 خلاقیت و انگیزه

با اینکه این کتاب برای برنامه نویس های ماهر نیست، برنامه نویسی جدی می تواند کاری بسیار مفید هم از نظر مادی و هم از نظر شخصی باشد. ساختن برنامه های مفید، زیبا و هوشمند برای دیگران فعالیتی خلاقانه است. کامپیوتر شما یا همان دستیار شخصی دیجیتال معمولا دارای چندین برنامه مختلف از چندین گروه برنامه نویسان به طوری که هر یک از آنها در حال رقابت برای توجه و علاقه شما هستند. آنها بیشترین تلاش خود را می کنند تا نیاز های شما را برطرف کنند و تجربه کاربر مناسبی برای شما ایجاد کنند. در برخی شرایط، وقتی شما یک نرم افزاری را انتخاب می کنید، برنامه نویس ها مستقیما هزینه دریافت می کنند.

اگر ما برنامه ها را نتیجه خلاقانه گروه هایی از برنامه نویس ها در نظر بگیریم، شکل مقابل ورژن منطقی تری از دستیار شخصی دیجیتال ما است:

اکنون، انگیزه اصلی ما برای درآمد یا رضایت کاربر ها نیست، ولی انگیزه ما این است که از وقتمان به خوبی برای کار با داده ها و مشخصاتی که در طول زندگی با آنها مواجهه می شویم. هنگامی که شما شروع می کنید، شما هم برنامه نویس و هم کاربر نهایی خواهید بود. با گذشت زمان با کسب مهارت بیشتر به عنوان برنامه نویس و برنامه نویسی باعث افزایش احساس خلاقیت شما می شود، شاید شما بیشتر به ساختن برنامه برای دیگران فکر کنید.

شکل 1.3: معماری سخت افزار

1.2.معماری سخت افزار کامپیوتر

قبل از این که ما یاد گرفتن زبانی که برای راهنمایی و حرف زدن با کامپیوتر برای ساخت نرم افزار را شروع کنیم، ما باید کمی درباره روش ساخت کامپیوتر ها را یاد بگیریم. اگر شما تکه های کامپیوتر یا موبایلتان را جدا کنید، موارد زیر را داخل آن پیدا می کنید:

- واحد پردازش مرکزی (یا CPU) همان بخشی از کامپیوتر است که برای پرسش پیوسته سوال "بعد چی؟" ساخته شده است. اگر سرعت کامپیوتر شما ۳.۰ گیگاهرتز باشد، یعنی پردازنده در هر ثانیه سه میلیارد بار می‌پرسد: «کار بعدی چیست؟». شما باید یاد بگیرید که سریع صحبت کنید تا بتوانید با پردازنده همگام شوید.

- حافظه اصلی برای ذخیره اطلاعات و مشخصاتی است که CPU به سرعت به آن نیاز دارد. سرعت حافظه اصلی تقریبا برابر CPU است. اما اطلاعات ذخیره شده در حافظه اصلی پس از خاموش کردن کامپیوتر از بین می رود
- حافظه ثانویه نیز برای ذخیره اطلاعات استفاده می شود ولی بسیار کندتر از حافظه اصلی است. نقطه قوت حافظه ثانویه این است که می تواند اطلاعات را حتی وقتی برقی به کامپیوتر نمی رسد. مثال‌هایی از حافظه ثانویه شامل درایوهای دیسک یا حافظه فلش (که معمولاً در فلش‌مموری‌ها و پخش‌کننده‌های موسیقی قابل حمل یافت می‌شوند) است.
- وسیله های ورودی و خروجی، به طور ساده صفحه های دیجیتال ما هستند، کیبورد، موس، میکروفون، بلندگو، پد لمسی و مانند آنها. اینها تمام روش هایی هستند که ما با کامپیوتر ها ارتباط برقرار می کنیم.
- امروزه همه کامپیوتر ها یک شبکه ارتباطی دارند تا اطلاعات را در طی یک شبکه بازیابی کنند. ما می توانیم از این شبکه به عنوان مکانی بسیار کند برای ذخیره و بازیابی اطلاعات داده هایی که شاید برای همیشه وجود نداشته باشد. به عبارتی، این شبکه، شکل آهسته تر و بعضی مواقع غیر قابل اعتماد تر از حافظه ثانویه است.

با اینکه اکثر این نکات ریز که این بخش ها کار می کنند را بهتر است به عهده سازنده های کامپیوتر بگذاریم. این به ما کمک بسیاری می کند که واژه نامه ای داشته باشیم که در مورد این بخش های مختلف در عین نوشتن برنامه حرف بزنیم.

1.4: تو کجا هستی؟

به عنوان یک برنامه نویس، وظیفه شما استفاده و تنظیم هر یک از این منابع است تا این مسئله را که نیاز دارید را حل کنید و با این داده های به دست آمده، آن را حل و بررسی کنید. به عنوان برنامه نویس شما به احتمال زیاد با CPU حرف می زنید و به آن می گویید که بعدا چه کاری را انجام دهد. گاهی شما به CPU می گویید که از حافظه اصلی، حافظه ثانویه، شبکه، یا وسایل ورودی و خروجی استفاده کنید.

شما باید همان کسی باشید که به سوال «کار بعدی چیست؟» پاسخ دهد. اما اگر شما را به اندازه 5 میلی متر تبدیل کنیم و داخل کامپیوتر قرار دهیم فقط برای اینکه بتوانید در هر ثانیه سه میلیارد بار فرمانی را صادر کنید، برای شما بسیار اذیت کننده می شود. پس در عوض، باید راهنمایی هایتان را از قبل بنویسید. ما به این اطلاعاتی که از پیش نوشته می شوند، برنامه می گوییم و عمل نوشتن این راهنمایی ها و به خصوص درست نوشتن آنها برنامه نویسی نام دارد.

1.3 درک برنامه نویسی

در ادامه این کتاب، ما تلاش می کنیم شما را به یک فردی تبدیل کنیم که در هنر برنامه نویسی مهارت یافته است. نهایتا شما یک برنامه نویس خواهید شد - شاید نه برنامه نویس حرفه ای، اما حداقل توانایی کافی خواهید داشت که به یک مسئله داده یا اطلاعات نگاه کنید و برنامه ای بسازید که مسئله را حل کند.

به عبارتی، برای یک برنامه نویس شدن شما به دو مهارت نیاز دارید:

- اولا، شما باید زبان برنامه نویسی را بدانید (پایتون) - شما باید با لغات و قواعد آن آشنایی کافی داشته باشید. شما باید نوشتن صحیح این کلمات را در این زبان جدید بدانید و روش نوشتن جملات با ساختاری مناسب در این زبان را یاد بگیرید.

- ثانیا، شما باید یک داستان را تعریف کنید. در نوشتن داستان، شما واژه ها و جملات را در کنار هم قرار می دهید تا تصورات خود را به خواننده کتاب برسانید. ساختن این داستان نیازمند هنر و مهارت نیاز است و مهارت در داستان نویسی از راه نوشتن و دریافت بازخورد پیشرفت می کند. در برنامه نویسی، برنامه ی ما همان «داستان» و مسئله ای که شما سعی میکنید حل کنید همان «ایده» است.

هنگامی که شما یک زبان برنامه نویسی مانند پایتون را یاد بگیرید، یاد گرفتن زبانی دیگر مانند جاوا (Javascript) و سی پلاس پلاس (C++) برای شما آسان تر می شود. این زبان برنامه نویسی جدید دارای لغات و قواعد متفاوتی است ولی راه و روش حل مسئله در تمام زبان های برنامه نویسی یکسان خواهد بود.
شما «لغات» و «جملات» پایتون را به سرعت یاد خواهید گرفت. نوشتن برنامه ای منطقی و کامل برای حل یک مسئله، زمان بیشتری خواهد برد. ما برنامه نویسی را همانند نوشتن یاد می دهیم. ما با خواندن و توصیف برنامه های شروع میکنیم، بعد برنامه های ساده می نویسیم، در نهایت به تدریج شروع به نوشتن برنامه های سخت تر و پیچیده تر شروع می کنیم. در برهه ای از زمان شما الهام خود را می یابید و الگو ها را به تنهایی پیدا می کنید و بیشتر به طور طبیعی مشاهده می کنید که چگونه یک مسئله را درک کنید و یک برنامه ای برای حل آن بنویسید. وقتی به آن سطح می رسید، برنامه نویسی به یک فرایندی لذت بخش و خلاقانه تبدیل می شود.
ما با لغتنامه و ساختار پایتون شروع می کنیم. در یاد گرفتن صبور باشید که مثال های ساده یاد آور زمانی خواهد بود که برای اولین بار نوشتن را یاد می گیرید.

1.4 کلمات و جملات

برخلاف زبان های انسان ها، لغتنامه ی پایتون در واقع بسیار محدود است. ما به این «لغتنامه»، کلمه های خاص یا کلید واژه ها می گوییم. اینها کلماتی هستند که معانی بسیار خاصی برای پایتون دارند. وقتی پایتون این کلمات را در یک برنامه ی پایتون می بیند، اینها فقط و فقط یک معنی برای پایتون دارند. با گذر زمان، شما با نوشتن برنامه های مختلف، کلمات خاص خودتان را می سازید که معنی خاصی برای شما دارند که به آنها متغیر ها می گویند. شما در انتخاب اسم های متغیر هایتان آزادی زیادی خواهید داشت، ولی نمیتوانید از هیچ یک از کلمات خاص پایتون به عنوان اسم متغیر هایتان استفاده کنید.
وقتی ما یک سگ را تربیت می کنیم، ما از کلمات خاصی مانند «بشین»، «بایست» و «بیار» استفاده می کنیم. هنگامی که شما با یک سگ صحبت می کنید و از هیچ یک از کلمات خاص استفاده نمی کنید، آنها فقط با یک نگاه پیچیده به شما نگاه می کنند تا اینکه از یک کلمه خاص استفاده کنید. به عنوان مثال، اگر شما بگویید:« ای کاش افراد بیشتری برای بهبود سلامت خود پیاده روی می کردند.»، چیزی که بیشتر سگ ها به احتمال زیاد می شنوند این است، « فلان فلان فلان پیاده روی فلان فلان فلان». این به همان دلیل است که «پیاده روی» یک کلمه خاص در زبان سگ است. خیلی ها ممکن است بگویند که زبان بین انسان ها و گربه ها هیچ کلمات خاصی ندارد.
کلمات خاص (یا همان کلمات رزرو شده) در زبانی که انسان ها با پایتون صحبت می کنند شما موارد زیر است:

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try

as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

همین ها هستند، و برخلاف یک سگ، پایتون از ابتدا به طور کامل تربیت شده است. وقتی شما می گویید «try»، پایتون هر دفعه که شما می گویید بدون اشتباه امتحان می کند.
ما درباره این کلمات رزرو شده و راه استفاده از آنها در زمان مناسب یاد می گیریم، اما اکنون ما بر روی معادل «صحبت کردن» (در زبان انسان به سگ) برای پایتون تمرکز می کنیم. نکته ی خوب در مورد فرمان دادن به پایتون که حرف بزند این است که ما حتی می توانیم با قرار دادن پیاممان داخل علامت های نقل قول (""):

```
print('Hello world!')
```

الان ما حتی اولین جمله نحوی صحیح خود را نوشته ایم. جمله ی ما با تابع ***print*** شروع شده و به دنبال آن، رشته متنی دلخواه قرار دارد که در علامت نقل قول تکی قرار دارد. رشته متن ها در عمل print داخل علامت های نقل قول قرار دارند. علامت های نقل قول تکی و دوتایی کار یکسانی می کنند؛ اکثر افراد از نقل‌قول تکی استفاده می‌کنند، مگر در مواردی که نقل‌قول تکی (که همان آپاستروف هم هست) درون رشته وجود داشته باشد.

1.5 صحبت کردن با پایتون

اکنون که ما یک کلمه و یک جمله از پایتون را می دانیم، ما باید بدانیم که چطور مکالمه‌ای با پایتون را آغاز کنیم تا مهارت های زبانی خود را امتحان کنیم.
قبل از اینکه بتوانید با پایتون صحبت کنید، شما باید ابتدا نرم افزار پایتون را دانلود کنید و یاد بگیرید که چگونه پایتون را در به کار بیندازید. این کار مراحل زیادی برای این فصل دارد پس من پیشنهاد می کنم به سایت www.py4e.com مراجعه کنید که در آنجا من مراحل دانلود و نصب پایتون را از صفر تا صد با تصاویر برای سیستم های ویندوز و مک توضیح داده‌ام. در مقطعی، شما وارد یک ترمینال یا پنجره دستورات می شوید و واژۀ python را تایپ می کنید و مترجم پایتون در حالت تعاملی (interactive mode) اجرا می شود و چیزی شبیه به زیر ظاهر خواهد شد:

```
 Python 3.11.6 (main, Nov 2 2023, 04:39:43)
      [Clang 14.0.3 (clang-1403.0.22.14.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

این علامت <<< شیوه مترجم پایتون برای پرسیدن این سوال از شماست: «می خواهی در مرحله‌ی بعد چه کاری انجام دهم؟» پایتون برای مکالمه با شما آماده است. تنها چیزی که لازم است بدانید این است که چگونه به زبان پایتون حرف بزنید.
بیایید تصور کنیم برای مثال شما حتی ساده‌ترین کلمات و جملات زبان پایتون را نمی دانید. شما شاید تمایل داشته باشید که این جمله‌ی استاندارد و معروفی که فضانوردان هنگام فرود روی سیاره‌ای دوردست و تلاش برای صحبت با ساکنان آن سیاره می گویند:

```
>>> I come in peace, please take me to your leader
File "<stdin>", line 1
```

```
      I come in peace take me tou your leader
        ^^^^
SyntaxError: invalid syntax
>>>
```

این خیلی خوب پیش نمی رود. مگر اینکه سریع دربارهٔ چیزی فکر کنید، ساکنان این سیاره به احتمال زیاد شما را با نیزه هایشان زخمی می کنند، روی سیخی می گذارند، روی آتش کباب می کنند و برای شام می خورند.
خوشبختانه شما یک نسخه از این کتاب را در سفرتان آورده‌اید و حالا به همین صفحه رجوع می کنی و دوباره تلاش می کنی:

```
 >>> print('Hello world!')
Hello world!
```

الان این خیلی بهتر است، پس شما تلاش می کنید که بیشتر مکالمه کنید:

```
 >>> print('You must be the legendary god that comes from the sky')
You must be the legendary god that comes from the sky
>>> print('We have been waiting for you for a long time')
We have been waiting for you for a long time
>>> print('Our legend says you will be very tasty with mustard')
Our legend says you will be very tasty with mustard
>>> print We will have a feast tonight unless you say
      File "<stdin>", line 1
            print 'We will have a feast tonight unless you say
                  ^
SyntaxError: unterminated string literal (detected at line 1)
>>>
```

این مکالمه خیلی خوب پیش می رفت تا اینکه شما کوچکترین اشتباه را در زبان پایتون کردید که پایتون نیزه ها را باری دیگر در آورد.
در این مرحله، باید فهمیده باشید که با اینکه پایتون به طور فوق العاده پیچیده، قدرتمند و نسبت به نحوهٔ نگارش حساس است، اما پایتون هوشمند نیست. در واقع شما فقط دارید با خودتان حرف می زنید ولی با نگارش درست.
به عبارتی، زمانی که شما از یک برنامه‌ای استفاده می کنید که توسط شخصی دیگر نوشته شده است، مکالمه بین شما و برنامه‌نویس های آن برنامه است و پایتون یک واسطه است. پایتون روشی است که سازندگان برنامه ها برای بیان اینکه مکالمه چگونه قرار است پیش برود استفاده می کنند. در چند فصل دیگر، شما هم یکی از آن برنامه‌برنامه نویس های خواهید بود که از پایتون برای صحبت کردن با کاربر برنامهٔ خودتان به کار می برید.
قبل از اینکه اولین مکالمهٔ خود را با مترجم پایتون به پایان برسانیم، بهتر است راه مناسب «خداحافظ» گفتن را هنگام ارتباط با ساکنان سیارهٔ پایتون را بدانید:

```
>>> good-bye
Traceback (most recent call last):
      File "<stdin>", line 1, in <module>
NameError: name 'good' is not defined
>>> if you don't mind, I need to leave
      File "<stdin>", line 1
```

```
        if you don't mind, I need to leave
                   ^
SyntaxError: unterminated string literal (detected at line 1)
>>> quit()
```

متوجه خواهید شد که خطا در دو تلاش اول نادرست، با یکدیگر متفاوت است. خطای دوم متفاوت است چون if یک کلمه‌ی خاص است و پایتون این کلمه را دید و گمان کرد که ما سعی داشتیم چیزی بگوییم ولی نگارش جمله را اشتباه گرفتیم.

روش درست خداحافظی با پایتون این است که در علامت تعاملی <<< دستور quit() را وارد کنیم. احتمالا زمان زیادی می برد تا آن را حدس بزنید، پس داشتن یک کتاب در دسترس احتمالا کمک زیادی خواهد کرد.