به نام او

آموزش برنامه‌نویسی

PYTHON 3.9

مقدمه‌ای بر اصول پایه‌ای پایتون
(ویرایش سوم)

فابریزیو رومانو
هنریش کراکر

ترجمه: مهندس حسین یعسوبی
انتشارات پندار پارس

سرشناسه	:	رومانو، فابریزیو، ۱۹۷۵- م. Romano, Fabrizio, 1975 -
عنوان و نام پدیدآور	:	آموزش برنامه‌نویسی Python 3.9: مقدمه‌ای بر اصول پایه‌ای پایتون/ فابریزیو رومانو، هنریش کراکر؛ ترجمه حسین یعسوبی.
مشخصات نشر	:	تهران: پندار پارس، ۱۴۰۱.
مشخصات ظاهری	:	۵۱۰ ص.: مصور، جدول.
شابک	:	978-622-7785-07-4
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	عنوان اصلی: Learn Python Programming : An in-Depth Introduction to the Fundamentals of Python, 3rd.ed, 2021.
یادداشت	:	ترجمه‌ی ویراست قبلی کتاب حاضر با عنوان " آموزش برنامه‌نویسی PYTHON 3.7: راهنمای مبتدی‌ها در برنامه‌نویسی، علم داده، و توسعه وب با پایتون ۳.۷" توسط همین ناشر در سال ۱۳۹۸ منتشر شده است.
عنوان دیگر	:	مقدمه‌ای بر اصول پایه‌ای پایتون.
عنوان دیگر	:	آموزش برنامه‌نویسی PYTHON 3.7: راهنمای مبتدی‌ها در برنامه‌نویسی، علم داده، و توسعه وب با پایتون ۳.۷.
موضوع	:	پایتون (زبان برنامه‌نویسی کامپیوتر) Python (Computer program language)
شناسه افزوده	:	کروگر، هنریش، ۱۹۸۱- م.
شناسه افزوده	:	Kruger, Heinrich, 1981-
شناسه افزوده	:	یعسوبی، حسین، ۱۳۵۲ -، مترجم
رده بندی کنگره	:	QA۷۶/۷۳
رده بندی دیویی	:	۱۳۳/۰۰۵
شماره کتابشناسی ملی	:	۸۸۰۴۳۸۶
اطلاعات رکورد کتابشناسی	:	فیپا

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۱۲۲۴۵۲۳۴۸ **info@pendarepars.com**

نام کتاب : **آموزش برنامه‌نویسی Python 3.9، مقدمه‌ای بر اصول پایه‌ای پایتون**

ناشر : **انتشارات پندار پارس**

تالیف : **فابریزیو رومانو، هنریش کراکر (Fabrizio Romano, Heinrich Kruger)**

ترجمه : **حسین یعسوبی**

چاپ نخست : **فروردین ۱۴۰۱**

شمارگان : **۲۰۰ نسخه**

طرح جلد : **رامین شکرالهی**

چاپ، صحافی : **روز** **انتشار با کاغذ یارانه‌ای وزارت ارشاد**

قیمت : **۱۹۰.۰۰۰ تومان** **شابک : ۴-۰۷-۷۷۸۵-۶۲۲-۹۷۸**

***هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد ***

سخن مترجم (ناشر)

هر بار که کتابی را برای ترجمه یا تألیف به دست می‌گیرم، در میانه راه، به خود می‌گویم این آخرین باری است که این کار را می‌کنم. چراکه یکی از سخت‌ترین کارهای موجود است. به‌ویژه اگر هدف، ارائه کاری با کیفیت درخور باشد. اما پس از به چاپ رسیدن اثر، گویی فرزند نداشته‌ات را در آغوش می‌گیری و در چشمانت می‌نگرد و تو هم نوازشش می‌کنی. نکته‌اش در این است که این نوزاد نابکار، زیر لب به تو می‌گوید که از هم اینک در اندیشه خلق خواهر و برادرم باش و به هیچ وجه، به مشکلات این کار نمی‌اندیشد!

به هر روی، سال ۹۹ که دنیا درگیر بیماری کرونا بود، و کسب و کارهای زیادی همچون ناشران، با کاهش تولید روبه‌رو بودند، دست‌کم برای من فرصتی شد که دو کتاب را ترجمه کنم و به چاپ برسانم.

کتابی که در دست دارید، اگر بگویم ترجمه بهترین کتاب آموزش برنامه‌نویسی پایتون موجود است، گزاف نگفته‌ام. با کمی گشت و گذار در وب، به این خواهید رسید.

با انتشار نسخه ۳.۹ پایتون، فابریزیو نیز اقدام به ویرایش کتاب خود کرد و این‌بار، از دوستش هنریش کراکر نیز کمک گرفت و دستی به سروروی کتاب کشیدند. در این سوی آب‌ها، زحمت ما را هم زیاد کردند و موجب شد، پاراگراف به پاراگراف، تغییرات ایجاد شده را در کتاب نسخه ۳.۷ پیاده‌سازی کنیم ☺. خوشبختانه پاره‌ای ایرادهای کتاب نیز در این حین رفع و رجوع شد.

پایتون زبانی است که جای خود را به‌خوبی در همه زمینه‌های فنآورانه، دانشگاهی، کسب‌وکار و فضای مجازی باز کرده است و کمتر کسی است که این را نداند. بر خود لازم دانستم این اثر را به بهترین شکل ممکن ترجمه کنم تا شاید بتواند راهگشای برنامه‌نویسان و دانشجویان و حتی دانش‌آموزانی که قصد فراگیری یک زبان همه‌کاره را دارند باشد. هرچند، هیچ اثری به‌دور از اشکال نیست. کافیست بر ما منت نهید و نظرها و یا اشکال‌های احتمالی که از چشم ما به‌دور مانده را در صفحه این کتاب در سایت انتشارات پندارپارس مطرح فرمایید.

حسین یعسوبی

مدیر مسئول انتشارات پندار پارس

بهار ۱۴۰۱

فهرست

پیش‌گفتار ۱۵
کتاب برای چه کسانی مفید است ۱۶
محتویات کتاب ۱۶
بهره‌برداری بیشتر از کتاب ۱۸
دانلود فایل‌های کد مثال‌ها ۱۸
فصل ۱؛ معرفی کوتاه پایتون ۱۹
معرفی پایتون ۲۰
ورود به پایتون ۲۲
درباره پایتون ۲۳
قابل‌حمل بودن ۲۳
انسجام ۲۳
بهره‌وری توسعه‌دهنده ۲۳
یک کتابخانه وسیع ۲۴
کیفیت نرم‌افزار ۲۴
یکپارچگی نرم‌افزار ۲۴
رضایت‌مندی و لذت ۲۴
موانع چیست؟ ۲۵
امروزه چه افرادی از پایتون استفاده می‌کنند؟ ۲۵
تنظیم محیط ۲۶
پایتون ۲ در مقابل پایتون ۳ ..... ۲۶
نصب پایتون ۲۷
تنظیم مفسر پایتون ۲۷
درباره محیط‌های مجازی ۲۹
نخستین محیط مجازی‌تان ۳۰
نصب کتابخانه‌های شخص ثالثی ۳۳
دوستمان کنسول ۳۴
چگونگی اجرای یک برنامه پایتون ۳۴
اجرای اسکریپت‌های پایتون ۳۵
اجرای پوسته تعاملی پایتون ۳۵
اجرای پایتون به‌شکل یک سرویس ۳۷
اجرای پایتون به‌شکل یک برنامه‌کاربردی GUI ۳۷
کدهای پایتون چگونه سازماندهی می‌شود ۳۸
چگونه از ماژول‌ها و پکیج‌ها استفاده کنیم؟ ۴۰
مدل اجرایی پایتون ۴۲

نام‌ها و فضاهای نام ۴۲
قلمروها ۴۴
اشیاء و کلاس‌ها ۴۸
راهنمایی‌هایی درباره نحوه کدنویسی صحیح ۵۰
فرهنگ پایتون ۵۲
نکته‌ای از IDEها ۵۳
خلاصه ۵۴
فصل ۲؛ انواع داده‌های توکار ۵۵
هر چیزی یک شئ است ۵۵
تغییرپذیر یا تغییرناپذیر؟ پرسش این است ۵۶
اعداد ۵۸
اعداد صحیح ۵۸
اعداد بولین ۶۱
اعداد حقیقی ۶۲
اعداد مختلط (complex) ۶۳
اعداد کسری و اعشاری ۶۴
ترتیب‌های تغییرناپذیر ۶۵
رشته‌ها و بایت‌ها ۶۵
Encoding و Decoding رشته‌ها ۶۶
ایندکس کردن و برش دادن رشته‌ها ۶۷
فرمت‌بندی رشته ۶۸
تاپل‌ها ۷۰
ترتیب‌های تغییرپذیر ۷۱
لیست‌ها ۷۱
آرایه‌های Byte ۷۵
انواع set ۷۶
انواع Mapping- دیکشنری‌ها ۷۸
Dates and Times ۸۳
کتابخانه استاندارد ۸۳
کتابخانه‌های شخص ثالثی ۸۷
ماژول collections ۸۹
namedtuple ۸۹
defaultdict ۹۱
ChainMap ۹۲
Enums ۹۳
نکات پایانی ۹۴
اندکی کش کردن ۹۴
چگونگی انتخاب ساختارهای داده‌ها ۹۵

درباره indexing و slicing ۹۶
درباره نام‌ها ۹۷
خلاصه ۹۸
فصل ۳؛ تکرارکردن و تصمیم‌سازی ۹۹
برنامه‌نویسی شرطی ۹۹
یک else ویژه: elif ۱۰۰
عملگر مبنای سه ۱۰۳
حلقه‌زنی (Looping) ۱۰۴
حلقه for ۱۰۴
تکرار روی یک بازه ۱۰۵
تکرار روی یک ترتیب ۱۰۵
تکرارکردنی‌ها و تکرارپذیرها ۱۰۷
تکرار کردن روی چند ترتیب ۱۰۸
حلقه while ۱۱۰
شکست (break) و ادامه‌ی (continue) گزاره‌ها ۱۱۳
یک بند else ویژه ۱۱۵
عبارت‌های تخصیص (assignment expressions) ۱۱۷
گزاره‌ها و عبارت‌ها ۱۱۷
استفاده از عملگر شیر دریایی ۱۱۸
یک گوشزد ۱۱۹
درج همه اینها با همدیگر ۱۲۰
یک تولیدکننده عدد اول ۱۲۰
اعمال تخفیف‌ها ۱۲۲
نگاهی گذرا به ماژول itertools ۱۲۶
تکرارکننده‌های infinite ۱۲۶
پایان یافتن تکرارکننده‌ها روی کوتاه‌ترین ترتیب ورودی ۱۲۷
مولدهای ترکیبی ۱۲۸
خلاصه ۱۲۸
فصل ۴؛ توابع، بلوک‌های ساختمانی کد ۱۲۹
چرا از توابع استفاده می‌کنیم؟ ۱۳۰
کاهش کدهای تکراری ۱۳۰
تفکیک یک وظیفه‌ی پیچیده ۱۳۱
پنهان‌سازی جزئیات پیاده‌سازی ۱۳۲
بهبود خوانایی ۱۳۲
بهبود قابلیت ردیابی ۱۳۳
قلمروها و وضوح نام ۱۳۴
گزاره‌های global و nonlocal ۱۳۶
پارامترهای ورودی ۱۳۸

پاس‌دادن آرگومان ۱۳۸
تخصیص به نام پارامترها ۱۳۹
تغییر یک شئ تغییرپذیر ۱۴۰
پاس دادن آرگومان‌ها ۱۴۱
آرگومان‌های جایگاهی ۱۴۱
آرگومان‌های کلیدواژه‌ای ۱۴۱
آنپک کردن تکرارپذیری (iterable unpacking)..... ۱۴۲
آنپک‌کردن دیکشنری (dictionary unpacking) ۱۴۳
ترکیب انواع آرگومان‌ها ۱۴۳
تعریف پارامترها ۱۴۴
پارامترهای اختیاری ۱۴۵
پارامترهای جایگاه متغیر ۱۴۵
پارامترهای کلیدواژه متغیر ۱۴۶
پارامترهای فقط جایگاهی..... ۱۴۸
پارامترهای فقط کلیدواژه‌ای ۱۴۹
ترکیب پارامترهای ورودی ۱۵۰
مثال‌های امضا ۱۵۱
پرهیز از تله‌ی پیش‌فرض‌های تغییرپذیر ۱۵۲
مقادیر بازگشتی ۱۵۴
بازگرداندن چند مقدار ۱۵۵
چند نکته مهم..... ۱۵۶
توابع بازگشتی..... ۱۵۷
توابع بی نام ۱۵۸
خصوصیات تابع (Function attributes) ۱۵۹
توابع توکار (پیش‌ساخته)..... ۱۶۰
مستندسازی کد ۱۶۱
درون‌ریزی اشیاء ۱۶۲
درون‌ریزی‌های وابسته (Relative Imports)..... ۱۶۴
مثال پایانی..... ۱۶۵
خلاصه ۱۶۶
فصل ۵؛ خلاصه لیست‌ها و مولدها (COMPREHENSIONS & GENERATORS)۱۶۷
توابع map، zip و filter..... ۱۶۹
Map..... ۱۶۹
Zip ۱۷۲
فیلتر ۱۷۳
Comprehensions (خلاصه لیست‌ها) ۱۷۴
خلاصه‌های تودرتو..... ۱۷۵
فیلتربندی یک comprehension..... ۱۷۶

dict comprehensions (خلاصه دیکشنری) ۱۷۸
Set comprehensions (خلاصه ست) ۱۷۹
مولدها (generators) ۱۸۰
توابع مولد ۱۸۰
رفتن به آن‌سوی next ۱۸۳
عبارت yield from ۱۸۶
عبارت‌های مولد ۱۸۷
چند نکته اجرایی ۱۹۰
افراطی نکردن comprehensionها و مولدها ۱۹۳
نام‌گذاری موضعی ۱۹۷
رفتار مولد در توکارها ۱۹۸
آخرین مثال ۱۹۹
خلاصه ۲۰۱
فصل ۶؛ شئ‌گرایی، دکوراتورها، و تکرارکننده‌ها ۲۰۳
دکوراتورها ۲۰۳
یک کارخانه دکوراتور ۲۱۰
برنامه‌نویسی شئ‌گرا (OOP) ۲۱۲
ساده‌ترین کلاس پایتون ۲۱۲
فضاهای‌نام شئ و کلاس ۲۱۳
پنهان‌کردن خصیصه ۲۱۴
آرگومان self ۲۱۶
آماده‌سازی آغازین یک نمونه ۲۱۷
OOP درباره استفاده دوباره کد است ۲۱۸
وراثت و ترکیب ۲۱۸
دسترسی یک کلاس مبنا ۲۲۳
چند–وراثتی ۲۲۵
Method resolution order (MRO) ۲۲۸
متدهای کلاس و ایستا ۲۳۰
متدهای ایستا (static methods) ۲۳۰
متدهای کلاس ۲۳۲
متدهای اختصاصی و name mangling ۲۳۴
دکوراتور property ۲۳۶
دکوراتور cached_property ۲۳۸
اضافه‌بار دادن عملگر (Operator overloading) ۲۴۰
چند ریختی– یک بازبینی مختصر ۲۴۱
کلاس‌های data ۲۴۲
نوشتن یک تکرارکننده سفارشی ۲۴۳
خلاصه ۲۴۵

فصل ۷؛ استثناها و مدیران محتوا ۲۴۷
استثناها ۲۴۷
بالا آمدن استثناها ۲۴۹
تعریف استثناهای شخصی ۲۵۰
Tracebacks ۲۵۰
رسیدگی به استثناها ۲۵۱
نه فقط برای خطاها ۲۵۵
مدیران محتوا (Context Managers) ۲۵۶
مدیران محتوای کلاس–محور ۲۵۹
مدیران محتوای مولد–محور ۲۶۰
خلاصه ۲۶۲
فصل ۸؛ ماندگاری فایل‌ها و داده‌ها ۲۶۳
کار با فایل‌ها و دایرکتوری‌ها ۲۶۳
بازکردن فایل‌ها ۲۶۴
استفاده از یک مدیر محتوا برای بازکردن یک فایل ۲۶۵
خواندن و نوشتن در یک فایل ۲۶۶
خواندن و نوشتن در حالت باینری ۲۶۷
محافظت دربرابر بازنویسی یک فایل موجود ۲۶۸
بررسی موجود بودن فایل و دایرکتوری ۲۶۸
دست‌کاری فایل‌ها و دایرکتوری‌ها ۲۶۹
دست‌کاری نام مسیرها ۲۷۱
فایل‌ها و دایرکتوری‌های موقتی ۲۷۲
محتویات دایرکتوری ۲۷۳
فشرده‌سازی فایل و دایرکتوری ۲۷۴
تبادل فرمت‌های داده‌ها ۲۷۵
کار با JSON ۲۷۶
انکدینگ/دکدینگ سفارشی با JSON ۲۷۹
IO، جریان‌ها و درخواست‌ها ۲۸۳
استفاده از یک جریان درون–حافظه‌ای ۲۸۳
ایجاد درخواست‌های HTTP ۲۸۴
ایستادگی داده‌ها روی دیسک ۲۸۷
سریالی‌کردن داده‌ها با pickle ۲۸۷
ذخیره داده‌ها با shelve ۲۸۹
ذخیره‌سازی داده‌ها در یک دیتابیس ۲۹۱
خلاصه ۲۹۷
فصل ۹؛ رمزنگاری و توکن‌ها ۲۹۹
ضرورت رمزنگاری ۲۹۹
راهنمایی‌های مفید ۳۰۰

Hashlib ۳۰۰
HMAC ۳۰۴
Secrets ۳۰۵
اعداد تصادفی ۳۰۵
تولید توکن ۳۰۶
تطبیق digest ۳۰۸
توکن‌های JSON Web ۳۰۸
ادعاهای رجیسترشده ۳۱۱
ادعاهای زمان–محور ۳۱۱
ادعاهای Auth-related ۳۱۳
استفاده از الگوریتم‌های نامتقارن (کلید–عمومی) ۳۱۴
مراجع مفید ۳۱۶
خلاصه ۳۱۶
فصل ۱۰؛ تست کردن ۳۱۷
آزمایش برنامه‌کاربردی خود ۳۱۷
تشریح آناتومی یک آزمایش ۳۲۰
خط مشئ آزمایش کردن ۳۲۱
آزمایش یونیت (واحد) ۳۲۲
نوشتن یک آزمایش واحد ۳۲۳
اشیاء ساختگی و وصله‌بندی ۳۲۵
بیانیه‌ها ۳۲۵
آزمایش یک مولد CSV ۳۲۵
مرزبندی‌ها و دانه دانه بودن ۳۳۵
آزمایش تابع export ۳۳۶
نقطه نظرهای پایانی ۳۳۹
توسعه آزمایش–محور ۳۴۱
خلاصه ۳۴۳
فصل ۱۱؛ دیباگ و رفع اشکال ۳۴۵
تکنیک‌های دیباگ‌کردن ۳۴۶
دیباگ‌کردن با چاپ ۳۴۶
دیباگ کردن با یک تابع سفارشی ۳۴۷
استفاده از دیباگر پایتون ۳۵۰
تفتیش logها ۳۵۳
تکنیک‌های دیگر ۳۵۶
خواندن tracebackها ۳۵۶
تأکید (Assertions) ۳۵۷
محل یافتن اطلاعات ۳۵۸
راهنمایی‌های رفع اشکال ۳۵۸

جایی برای سرکشی..... ۳۵۸
استفاده از آزمایش‌ها برای دیباگ..... ۳۵۹
مانیتورینگ..... ۳۵۹
پروفایل کردن پایتون..... ۳۵۹
چه زمانی پروفایل بگیریم؟..... ۳۶۳
اندازه‌گیری زمان اجرا..... ۳۶۴
خلاصه..... ۳۶۵
فصل ۱۲؛ GUIها و اسکریپت‌ها..... ۳۶۷
نخستین روش؛ اسکریپ‌گرفتن..... ۳۶۹
درون‌ریزی‌ها (imports)..... ۳۷۰
تجزیه آرگومان‌ها..... ۳۷۱
منطق تجاری..... ۳۷۳
روش دوم: یک برنامه‌کاربردی GUI..... ۳۷۷
درون‌ریزی‌ها (imports)..... ۳۷۹
منطق طرح‌بندی (layout logic)..... ۳۸۰
منطق تجاری..... ۳۸۴
قاپیدن صفحه وب..... ۳۸۴
ذخیره‌سازی تصاویر..... ۳۸۶
خبر کردن کاربر..... ۳۸۹
شیوه بهبود بخشیدن به برنامه‌کاربردی..... ۳۹۰
از اینجا به کجا برویم؟..... ۳۹۲
ماژول turtle..... ۳۹۲
wxPython، Kivy، و PyQt..... ۳۹۲
اصل کمترین حیرت..... ۳۹۳
ملاحظات نخ‌کشی..... ۳۹۴
خلاصه..... ۳۹۴
فصل ۱۳؛ مختصری درباره علم داده..... ۳۹۵
IPython و Jupyter Notebook..... ۳۹۶
استفاده از Anaconda..... ۳۹۸
آغاز کار با Notebook..... ۳۹۹
سروکله زدن با داده‌ها..... ۴۰۰
تنظیم Notebook..... ۴۰۰
آماده‌سازی داده‌ها..... ۴۰۰
پاک‌سازی داده‌ها..... ۴۰۴
ایجاد DataFarme..... ۴۰۶
آنپک کردن نام کمپین..... ۴۰۹
آنپک کردن داده‌های کاربر..... ۴۱۰
پاک‌سازی هر چیزی..... ۴۱۴

ذخیره‌سازی DataFrame در یک فایل ۴۱۵
مصورسازی نتایج ۴۱۵
از اینجا به کجا برویم؟ ۴۲۲
خلاصه ۴۲۴
فصل ۱۴؛ مقدمه‌ای بر توسعه API ۴۲۵
وب چیست؟ ۴۲۶
وب چگونه کار می‌کند؟ ۴۲۶
کدهای وضعیت Respons ۴۲۸
مختصری درباره Type hiniting ۴۲۸
چرا تعیین‌گر نوع؟ ۴۳۰
مختصری از تعیین‌گر نوع ۴۳۰
مقدمه‌ای بر APIها ۴۳۳
API چیست؟ ۴۳۳
هدف از یک API چیست؟ ۴۳۳
پروتکل‌های API ۴۳۴
فرمت‌های تبادل–داده API ۴۳۵
API خط آهن ۴۳۵
مدلینگ دیتابیس ۴۳۷
تنظیم اصلی و پیکربندی ۴۴۳
افزودن تنظیمات ۴۴۴
Endpointهای ایستگاه ۴۴۵
خواندن داده‌ها ۴۴۵
ایجاد داده‌ها ۴۵۲
به‌روزرسانی داده‌ها ۴۵۵
حذف داده‌ها ۴۵۸
تصدیق کاربر ۴۵۹
مستندسازی API ۴۶۲
مصرف یک API ۴۶۳
فراخوانی API از Django ۴۶۳
از اینجا به کجا برویم؟ ۴۷۰
خلاصه ۴۷۱
فصل ۱۵؛ پکیج‌بندی برنامه‌های کاربردی پایتون ۴۷۳
Python Package Index (PyPI) ۴۷۳
پروژه زمان‌بندی قطار ۴۷۵
پکیج‌بندی با setuptools ۴۸۰
فایل‌های مورد نیاز ۴۸۰
Pyproject.toml ۴۸۰
License ۴۸۲

README ۴۸۲
Changelog ۴۸۲
Setup.cfg ۴۸۳
Setup.py ۴۸۳
MANIFEST.in ۴۸۵
متادیتای پکیج ۴۸۵
دسترسی به متادیتا در کد ۴۸۹
تعریف محتویات پکیج ۴۹۱
دسترسی به فایل‌های داده‌ای پکیج ۴۹۲
تعیین وابستگی‌ها ۴۹۴
نقاط ورودی (entry points) ۴۹۶
ساخت و انتشار پکیج‌ها ۴۹۷
اسکریپت build ۴۹۸
انتشار ۴۹۹
توصیه‌ای برای آغاز کردن پروژه‌های جدید ۵۰۲
ابزارهای جایگزین ۵۰۲
خلاصه ۵۰۴

پیش‌گفتار

نخستین ویرایش این کتاب، مصادف با تولد چهلم‌امین سالگی من بود. احساس می‌کنم همین دیروز بود، اما در اصل، ۶ سال پیش بود (۲۰۱۵). در طی چند هفته، کتاب جزو پرفروش‌ترین کتاب‌ها شد، و تا به امروز، خوانندگان آن با دوست‌داشتنی‌ترین پیام‌ها و ایمیل‌ها از سرتاسر جهان، مرا مورد لطف خود قرار داده و می‌دهند.

دو سال بعد، ویرایش دوم آنرا نوشتم. به کتاب بهتری از نظر محبوبیت و فروش منجر شد. و اکنون که **ویرایش سوم** را می‌نویسم تنها حکم بازگویی این داستان را ندارد، چون برای این ویرایش، از دوست و همکار عزیزم، **هنریچ کراگر** کمک گرفته‌ام.

به‌کمک همدیگر، روی ساختار کتاب کار کردیم. آنچه را احساس کردیم کاربرد زیادی ندارد، حذف کردیم و مطالبی که حس کردیم کاربرد بیشتری دارد به آن افزودیم. مطالبی را در هم آمیختیم، فصل‌های قدیمی را اصلاح کردیم، و موارد جدید را جایگزین ساختیم. مطمئنیم که با این همکاری دو نفری، بهترین ایده‌های ما در هر صفحه‌ای که می‌خوانید نگارش شده است. از این بابت بسیار خرسندیم.

همیشه می‌خواستم با هنریچ، روی پروژه‌ای این چنینی کار کنم، چون از وقتی او را می‌شناسم احترام زیادی برایش قائلم. او برای این کتاب، دیدگاه منحصر به فردش را آورده، استعداد خارق‌العاده‌اش به عنوان یک توسعه دهنده نرم‌افزار، و این به‌همراه زبان انگلیسی‌اش کمک شایانی به من کرد.

همه چیز به Python 3.9 به‌روزرسانی شده است، اما البته بیشتر کدها هنوز با هر نسخه اخیر از Python 3 کار می‌کند. فصل ترسناکی که درباره همزمانی بود، حذف شده، و فصلی که درباره برنامه‌نویسی Web بود، با فصل دیگری که مفهوم APIها را توضیح می‌دهد جایگزین شده است. همچنین، فصل جدید کاملی درباره اپلیکیشن‌های پایتون پکیجینگ اضافه کرده‌ایم که احساس می‌کنیم راه خوبی برای خاتمه کتاب است.

مطمئنیم این ویرایش از کتاب، به‌مراتب بهتر از ویرایش پیشین است؛ داستان را به‌شکل کامل‌تری بیان می‌کند و شما را با خود به جاهای خوبی می‌برد.

از این بابت خوشحالم که روح کتاب هنوز یکی است. این تنها یک کتاب درباره پایتون نیست. این نخستین و بهترین کتاب درباره برنامه‌نویسی است. کتابی که تا حد امکان، به انتقال اطلاعات به شما کمک می‌کند و گاهی به دلایل مشخصی، شما را به صفحاتی از وب هدایت می‌کند که به درک عمیق‌تری از مطلب برسید.

با وجودی که چندسالی از طراحی اولیه کتاب می‌گذرد، اما مفاهیم و اطلاعات را به خوبی به خواننده منتقل می‌کند و از نظر بعد زمانی، تأثیری در مفاهیم کتاب ندارد. در این باره بسیار اندیشیده‌ایم و از این بابت مطمئن هستیم.

لازم است به سختی کار کنید. کدها برای دانلود در دسترس شماست و توصیه می‌کنیم با آنها کار کنید، آزمایش کنید، تغییر دهید، بسط دهید، بشکنید، و به چیزهایی دست یابید. دوست داریم موارد حیاتی را توسعه دهید. می‌خواهیم شما در کدنویسی، مستقل و صاحب اختیار باشید. امیدواریم در هر کجا هستید، این کتاب به کمک شما بیاید و دست‌کم، به برنامه‌نویس بهتری تبدیل شوید.

هنگامی که پیش‌نویس‌های ویرایش دوم را برای آغاز کار روی ویرایش سوم ملاحظه کردیم، وقتی متوجه شدم نمی‌توانم خودم را در آن صفحات پیدا کنم شگفت‌زده شدم. آن صفحات نشان می‌داد چگونه افکارم و در نتیجه نوشتنم پس از چند سال دچار تغییر شده است.

تغییر در تمام اجزای هستی تنیده شده است. هر چیزی همیشه تغییر می‌کند. بنابراین امیدوارم شما نیز هرگز روی دیدگاه‌های خود ثابت‌قدم نباشید، تا هرگز دچار رخوت نشوید. در عوض، امیدواریم کارمان و روشی که به شما ارائه می‌دهیم در انعطاف‌پذیر ماندن شما، ذکاوت، واقع‌بینی، و وفق‌پذیری شما کمک کند.

هدفم این بود که این کتاب، تنها درباره زبان نباشد بلکه درباره برنامه‌نویسی باشد. در واقع، هنر برنامه‌نویسی، چند دیدگاه را دربر می‌گیرد و زبان، تنها یکی از آنهاست. جنبه قاطع دیگر برنامه‌نویسی، استقلال است. توانایی متوقف نکردن خودتان وقتی به دیوار بلندی برخورد می‌کنید و نمی‌دانید برای حل مشکل موجود، چه کنید. کتابی برای آموزش آن وجود ندارد و فکر کردم به‌جای اینکه سعی کنم این جنبه را آموزش دهم، با آزمایش کردن، خوانندگان را در این باره آموزش دهم.

کتاب برای چه کسانی مفید است

پایتون، مشهورترین زبان آموزشی مقدماتی در دانشگاه‌های تراز بالای علوم رایانه در ایالات متحده است، پس اگر تازه وارد دنیای برنامه‌نویسی شده‌اید یا اگر تجربه اندکی دارید و دوست دارید در این راه گام بگذارید، این زبان و این کتاب، آن چیزی است که به آن نیاز دارید.

اگر پیش‌تر با پایتون یا زبان دیگری کار کرده‌اید، باز هم این کتاب برایتان مفید است، هم به‌عنوان یک مرجع بنیادی و اصلی و هم برای ارائه تجربه‌ها و پیشنهادهای بسیار گسترده‌ای که در دو دهه اخیر، به‌دست آمده و در این کتاب در اختیارتان می‌گذاریم.

محتویات کتاب

فصل ۱، معرفی کوتاه پایتون؛ با مفاهیم بنیادی برنامه‌نویسی پایتون آشنا می‌شوید. شما را تا بالا آوردن و اجرای پایتون روی سیستم‌تان راهنمایی می‌کنیم و با چند دستور آن آشنا خواهید شد.

فصل ۲، Data Typeهای درون‌ساخت؛ انواع داده‌های پایتون که به‌شکل پیش‌ساخته در آن موجود است را معرفی می‌کند. پایتون دارای مجموعه بسیار گسترده‌ای از انواع داده‌های بومی و محلی است و این فصل، با یک مثال کوتاه، هر یک از آنها را توصیف می‌کند.

فصل ۳، شروط و تکرار؛ می‌آموزد با بازرسی شرط‌ها، اعمال منطق، و پیاده‌سازی حلقه‌ها، چگونه جریان کد را کنترل کنیم.

فصل ۴، توابع، ساخت بلوک‌های کد؛ شیوه نوشتن توابع را آموزش می‌دهد. توابع، کلیدهایی برای استفاده دوباره از کدها و کاهش زمان دیباگ کردن است و عموما برای نوشتن بهتر کد است.

فصل ۵، خلاصه‌لیست‌ها و مولدها؛ دیدگاه‌های عملیاتی برنامه‌نویسی پایتون را به شما معرفی می‌کند. این فصل، شیوه نوشتن خلاصه‌لیست‌ها و مولدها را می‌آموزد که ابزارهای مفیدی است که می‌توان برای افزایش سرعت کد و ذخیره حافظه، از آنها استفاده کرد.

فصل ۶، OOP، Decorators، و Iterators؛ مبانی برنامه‌نویسی شئ‌گرا با پایتون را آموزش می‌دهد. مفاهیم کلیدی و همه پتانسیل‌های این پارادایم را نشان می‌دهد. همچنین، یکی از محبوب‌ترین ویژگی‌های پایتون، یعنی دکوراتورها را نشان می‌دهد. در آخر نیز مفاهیم تکرارشدنی‌ها یا همان تکرارکننده‌ها را پوشش می‌دهد.

فصل ۷، استثناها و مدیران محتوا؛ مفهوم استثناها را معرفی می‌کند که بیانگر خطاهایی است که در برنامه رخ می‌دهد و چگونگی رسیدگی به آنها را مطرح می‌کند. همچنین، با مفهوم مدیرها آشنا می‌شوید که در کار با منابع، بسیار مفید است.

فصل ۸، ماندگاری فایل‌ها و داده‌ها؛ می‌آموزد چگونه با فایل‌ها، جریان‌ها، فرمت‌های تبادل داده‌ها و دیتابیس‌ها سروکار داشته باشیم.

فصل ۹، رمزنگاری و توکن‌ها؛ به مفاهیم امنیت، هش‌ها، رمزنگاری، و توکن‌ها می‌پردازد که بخشی از برنامه‌نویسی روزانه‌ای است که ارائه می‌شود.

فصل ۱۰، آزمایش؛ روش‌های اصلی تست کد را با ارائه مثال‌هایی از نحوه پیاده‌سازی آن نشان می‌دهد تا به کدی قوی‌تر، سریع‌تر، و قابل اتکاتر برسیم.

فصل ۱۱، دیباگ‌کردن و رفع اشکال؛ روش‌های اصلی دیباگ‌کردن کد را با ارائه مثال‌هایی از نحوه پیاده‌سازی آن نشان می‌دهد.

فصل ۱۲، GUIها و اسکریپت‌ها؛ با ارائه مثالی از دو جنبه متفاوت نما، شما را راهنمایی می‌کند: یک جنبه اجرا، یک اسکریپت است و دیگری، یک برنامه‌کاربردی GUI (رابط کاربری گرافیکی) مناسب است.

فصل ۱۳، علم داده؛ چند مفهوم کلیدی و یک ابزار بسیار ویژه را معرفی می‌کند، Jupyter Notebook.

فصل ۱۴، مقدمه‌ای بر توسعه API؛ اصول بنیادی توسعه API و تعیین‌گر–نوع در پایتون را معرفی می‌کند. همچنین مثال‌های مختلفی از نحوه استفاده از API را بیان می‌دارد.

فصل ۱۵، پکیج‌کردن برنامه‌های پایتون؛ شما را با فرایند آماده‌سازی یک پروژه برای انتشار آشنا می‌سازد و نشان می‌دهد چگونه نتیجه را روی Python Package Index (PyPI) بارگزاری کنید.

بهره‌برداری بیشتر از کتاب

با دنبال کردن مثال‌های کتاب، آموزش بهتری می‌گیرید. به این منظور، به یک رایانه، یک ارتباط اینترنت، و یک مرورگر نیاز دارید. **کتاب برای نسخه ۳.۹ پایتون نوشته شده، اما بیشتر بخش‌های آن با هر نسخه جدید Python 3.* باید کار کند.** راهنمای نصب پایتون روی سیستم‌عامل هم ارائه شده است. رویه‌های نصب همیشه درحال تغییر است، پس لازم است به به‌روزترین راهنمای نصب که در وب یافت می‌شود مراجعه شود. همچنین شیوه نصب کتابخانه‌های اضافه به‌کار رفته در انواع مثال‌ها را توضیح داده‌ایم و درصورت برخورد با هر مشکلی حین نصب آنها، پیشنهادهایی نیز ارائه کرده‌ایم. برای تایپ کد، نیاز به هیچ ویرایشگر خاصی نیست؛ هرچند، پیشنهاد می‌شود آنهایی که در مثال‌های پیش رو معرفی شده است را مدنظر داشته باشید تا متناسب با محیط کدنویسی ما باشد. در فصل یک پیشنهادهایی برای این موضوع ارائه شده است.

دانلود فایل‌های کد مثال‌ها

فایل‌های کد مثال‌های کتاب را از سایت انتشارات پندار پارس (صفحه این کتاب، برگه سورس‌کد و ضمائم) بردارید. همچنین، در GitHub به نشانی زیر نیز موجود است:

https://github.com/PacktPublishing/Learn-Python-Programming-Third-Edition

همچنین در نشانی زیر، مجموعه‌ای از کتاب‌ها و ویدئوهای مرتبط با کتاب را برایتان گذاشته‌ایم:

https://github.com/PacktPublishing/

ضمنا توجه داشته باشید که ورودی و خروجی‌های خط فرمان (کامندلاین) را به‌صورت زیر، یعنی با سه فلش مقابلشان، آورده‌ایم:

```
>>> import sys
>>> print(sys.version)
```

فصل ۱

معرفی کوتاه پایتون

به اختصار، برنامه‌نویسی کامپیوتری یا همان کدنویسی، به رایانه می‌گوید با استفاده از یک زبانی که می‌شناسد، کاری را انجام دهد. رایانه‌ها ابزارهای بسیار مفیدی هستند، اما شوربختانه نمی‌توانند برای خودشان فکر کنند. آنها نیاز دارند هر چیزی به آنها گفته شود: چگونه یک کار انجام شود، چگونه برای تصمیم‌گیری برای مسیری که باید دنبال شود، شرطی را بررسی کند، چگونه داده‌هایی که از یک وسیله (دیوایس) می‌آید رسیدگی شود، مانند شبکه یا یک دیسک، و وقتی مورد پیش‌بینی نشده‌ای رخ دهد چه واکنشی نشان داده شود؛ مثلا عملیاتی شکست بخورد یا گم شود.

به سبک‌ها و زبان‌های گوناگونی می‌توان کدنویسی کرد. چه بسیاری از آنها سخت و پیچیده هستند. هر کسی می‌تواند چگونگی کدنویسی را بیاموزد و شما نیز می‌توانید. اما، اگر می‌خواستید شاعر شوید آیا نوشتن به تنهایی کافی بود؟ مجبور بودید یک سری مهارت کسب کنید و تلاش بیشتری انجام دهید و طبع شعر هم که زیربنای کار است.

در آخر، همه چیز بستگی به این دارد که قصد دارید در این جاده حرکت کنید. کدنویسی، کنار هم چیدن یک سری دستور که کار کند نیست. چیزی خیلی بیشتر از این است.

کد خوب، باید کوتاه، سریع، زیبا و باسلیقه باشد، خواندن و فهم آن آسان باشد، بسط و تغییرش ساده باشد، پیمایش و تعمیرش ساده باشد، و به آسانی آزمایش شود. نوشتن کدی که هم‌زمان، همه این موارد کیفی را داشته باشد نیاز به زمان و کسب تجربه بالا دارد، اما خبر خوب این است که با خواندن این کتاب، نخستین گام به سمت این ایده‌آل‌ها را خواهید پیمود و تردیدی ندارم که از پس آن بر می‌آیید. در حقیقت، هرکسی می‌تواند، همه ما همیشه درحال برنامه‌نویسی هستیم، تنها از آن آگاه نیستیم. به مثال زیر توجه کنید:

فرض کنید می‌خواهید یک فنجان قهوه دم کنید. برای این کار به یک فنجان، یک ظرف قهوه‌دان، یک قاشق، آب، و کتری نیاز دارید. حتی اگر این کار را بلد نباشید، سعی می‌کنید مقداری داده را ارزیابی کنید. مطمئن می‌شوید آب در کتری هست و زیر کتری روشن است، فنجان تمیز است، و قهوه کافی در قهوه‌دان موجود است. سپس، آب را جوش می‌آورید و شاید در این حین، مقداری قهوه در فنجان بریزید. وقتی آب جوش آماده شد، آنرا در فنجان می‌ریزید و آنرا با قاشق به‌هم می‌زنید.

برنامه‌نویسی این مثال چگونه است؟

ما منابع را گردآوری کردیم (کتری، قهوه، آب، قاشق، و فنجان) و برخی شرایط مربوط به آنها را فراهم کردیم (زیر کتری را روشن کردیم، تمیز بودن فنجان، و وجود داشتن قهوه). سپس دو کار را آغاز کردیم (جوش آوردن آب و ریختن قهوه در فنجان)، و هنگامی که هر دوی آنها انجام شد، با ریختن آب در فنجان و هم زدن، رویه را به پایان رساندیم.

می‌توانید آنرا ببینید؟ تنها یک کارکرد سطح بالای یک برنامه قهوه را توضیح دادیم. کار دشواری نبود اما این همان کاری است که مغز ما هر روز انجام می‌دهد: ارزیابی شرطها، تصمیم برای انجام کارها، انجام وظایف، تکرار برخی از آنها، و توقف در برخی نقاط. اشیاء را تمیز کن، آنها را سرجایشان بگذار، و غیره.

همه آن چیزی که لازم است اکنون بیاموزید این است که همه کارهایی که به‌شکل خودکار در زندگی روزمره خود انجام می‌دهید را خرد کنید تا یک رایانه بتواند برخی از آنها را واقعا درک کند. و لازم است یک زبان را برای ساختن آن، به خوبی فراگیرید.

این کتاب برای همین منظور است. نحوه انجام این کار را خواهیم گفت و تلاش می‌کنیم با تعریف چند مثال ساده اما متمرکز (از نوع مورد علاقه‌ام)، این کار را انجام دهیم.

در این فصل، موارد زیر را پوشش می‌دهیم:

- ویژگی‌ها و زیست‌بوم پایتون
- راهنمایی‌هایی برای شیوه اجرا و کار با پایتون و محیط‌های ویژه
- شیوه اجرای برنامه‌های پایتون
- شیوه سازماندهی کد و مدل اجرایی پایتون

معرفی پایتون

هنگامی که به آموزش کدنویسی می‌پردازم دوست دارم به جهان واقعی ارجاع دهم؛ باور دارم که حقایق موجود در جهان واقعی، به شناخت بهتر مفاهیم کمک می‌کند. هرچند، اینک زمان آن رسیده که کمی جدی‌تر باشیم و کدنویسی را از زاویه فنی‌تر ببینیم.

هنگامی که کدی را می‌نویسیم، دستوری را برای انجام کارهایی به رایانه می‌دهیم. اتفاق در کجا می‌افتد؟ در خیلی جاها: حافظه رایانه، درایوهای سخت، کابل‌های شبکه، CPU و غیره. این یک جهان کامل است که بیشتر وقت‌ها بیانگر زیرمجموعه‌ای از جهان واقعی است.

اگر یک قطعه کد نرم‌افزاری بنویسید که امکان خرید آنلاین لباس را به مردم بدهد، مجبورید افراد واقعی، لباس‌های واقعی، برندهای واقعی، اندازه و غیره را درون مرزهایی از یک برنامه، ارائه کنید.

پس برای انجام این کار، نیاز به ایجاد و رسیدگی به اشیاء در برنامه‌ای که می‌نویسید دارید. هر شخص می‌تواند یک شئ باشد. هر ماشین، یک شئ است. یک جفت جوراب هم یک شئ است. خوشبختانه، پایتون اشیاء را به خوبی می‌شناسد.

دو ویژگی اصلی که هر شئ دارد، مشخصه‌ها (properties) و متدها (methods) است. اجازه دهید یک شخص را به‌عنوان مثال شئ درنظر بگیریم. نوعا در یک برنامه رایانه‌ای، افراد را به‌عنوان مشتری یا کارمند معرفی می‌کنید. مشخصه‌هایی که برای آنها نگاه می‌دارید چیزهایی مانند نام، SSN، سن، آیا دارای گواهی‌نامه رانندگی هستند، نشانی ایمیل، جنسیت، و غیره است. در یک برنامه رایانه‌ای، همه داده‌هایی که به‌منظور استفاده از یک شئ برای رسیدن به هدف خود نیاز دارید را ذخیره می‌کنید. اگر درحال کدنویسی یک وب‌سایت فروش لباس هستید، شاید بخواهید قد و وزن افراد را هم در کنار دیگر مقیاس‌های مشتریان خود ذخیره کنید تا بتوانید لباس‌های مناسب را به آنها پیشنهاد دهید. پس **مشخصه‌ها، ویژگی‌های یک شئ هستند**. همیشه از آنها استفاده می‌کنیم: می‌توانید آن قلم را به من بدهید؟___ کدام یکی؟___ قلم مشکی را. در اینجا ما از مشخصه مشکی یک قلم برای شناسایی آن استفاده کردیم.

متدها چیزهایی هستند که یک شئ می‌تواند انجام دهد. من شخصا متدهایی مانند صحبت کردن، راه رفتن، خوابیدن، خوردن، خواندن و نوشتن و غیره دارم. **هر آن چیزی که بتوانم انجام دهم می‌تواند به عنوان متدهای اشیائی دیده شود که مرا تعریف می‌کند**.

پس اکنون که می‌دانید اشیاء چیست و اینکه آنها متدهایی که می‌توانید اجرا کنید و مشخصه‌هایی که می‌توانید رسیدگی کنید را عرضه می‌کنند، آماده آغاز کدنویسی هستید. کدنویسی در حقیقت، به‌سادگی مدیریت آن اشیائی است که در زیرمجموعه جهانی که ما در حال بازتولید در نرم‌افزارمان هستیم زندگی می‌کنند. اشیاء را می‌توان به دلخواه خود، ایجاد، استفاده، استفاده مجدد و حذف کرد.

با توجه به فصل مدل داده‌ای در مستند رسمی پایتون:

"اشیاء، تجرد داده‌های پایتون هستند. همه داده‌ها در یک برنامه پایتون، توسط اشیاء یا ارتباط میان آنها بیان می‌شوند."

در فصل ۶ نگاه دقیق‌تری به اشیاء پایتون می‌اندازیم. اینک لازم است بدانید که **هر شیئی در پایتون دارای یک شناسه یا ID، یک نوع، و یک مقدار است**.

ID یک شئ پس از ایجاد هرگز تغییر نمی‌کند. این یک شناساننده یکتا برای آن است و پایتون برای بازیابی شئ وقتی می‌خواهیم از آن استفاده کنیم در پشت صحنه از آن استفاده می‌کند.

نوع یا type هم به هیچ وجه تغییر نمی‌کند. نوع می‌گوید چه عملیاتی توسط شئ پشتیبانی می‌شود و چه مقادیر ممکنی می‌تواند به آن تخصیص یابد. در فصل ۲ مهم‌ترین انواع داده پایتون را خواهیم دید.

مقدار یا value می‌تواند تغییر کند یا نکند. اگر بتواند تغییر کند، به آن شئ تغییرپذیر، ناپایدار، یا mutable می‌گوییم؛ وگرنه به آن تغییرناپذیر، پایدار، یا immutable می‌گوییم.

چگونه می‌توان از یک شئ استفاده کرد؟ البته که یک نام به آن می‌دهیم! با نام‌گذاری آن می‌توانیم هر بار برای بازیابی و استفاده از آن، از آن نام استفاده کنیم.

در یک وضعیت عمومی‌تر، اشیائی همچون اعداد، رشته‌ها (متن)، کلکسیون‌ها و غیره، با یک نام مرتبط هستند. معمولا می‌گوییم که این نام، نام یک متغیر است. **متغیر را مانند یک جعبه‌ای ببینید که برای نگهداری داده‌ها می‌توانید استفاده کنید.**

بنابراین، همه اشیائ مورد نیاز را دارید؛ اینک چه؟ خب باید از آنها استفاده کنیم، درست است؟ شاید بخواهیم آنها را از طریق یک ارتباط شبکه‌ای انتقال دهیم یا در یک دیتابیس ذخیره کنیم. شاید آنها را روی یک صفحه وب نمایش دهیم یا آنها را درون یک فایل بنویسیم. به منظور این کار لازم است با پر کردن یک فرم کاربری، یا فشار یک دکمه، یا بازکردن یک صفحه وب و انجام یک جست‌وجو، واکنش نشان دهیم. این کارها را با اجرای کد خود، ارزیابی شرط‌ها برای انتخاب اینکه کدام اجزا اجرا شود، چند بار، و در چه شرایطی، عملی می‌کنیم.

و برای انجام همه اینها، به‌شکل اساسی نیاز به زبان داریم. در اینجاست که پایتون به کارمان می‌آید. پایتون زبانی است که در این کتاب برای دستور دادن به رایانه در انجام کاری برایمان، استفاده خواهیم کرد.

مقدمه چینی کافیست، اجازه دهید وارد گود شویم.

ورود به پایتون

پایتون، تولید شگفت‌انگیز Guido Van Rossum، یک ریاضی‌دان و دانشمند هلندی علوم رایانه است که تصمیم گرفت با پرژه‌ای که در کریسمس ۱۹۸۹ آغاز شد به جهانیان هدیه کند. این زبان در حدود سال ۱۹۹۱ به‌شکل عمومی منتشر شد و امروز، یکی از برترین زبان‌های برنامه‌نویسی در کل دنیاست.

من وقتی ۷ سال داشتم برنامه‌نویسی را روی یک کومودور VIC-20 آغاز کردم که بعدها با برادر بزرگترش یعنی کومودور ۶۴ جایگزین شد. آن زبان، بیسیک بود. سپس به سراغ زبان پاسکال، اسمبلی، C، C++، جاوا، جاواسکریپت، ویژوال بیسیک، PHP، ASP، ASP.NET، C# و دیگر زبان‌های کوچکی که حتی به‌خاطر نمی‌آورم رفتم، اما تنها وقتی با پایتون آشنا شدم احساس کردم که گمگشده خود را یافته‌ام.

وقتی همه اجزای بدن شما در حال نعره زدن است، و می‌گوید این را بخر! این برایمان بهترین است!

این مرا یاد روزی انداخت که خواستم از آن استفاده کنم. سینتکس آن کمی متفاوت از آنچه استفاده کرده بودم بود، اما پس از گذراندن احساس عجیب آغازین (مانند داشتن لباس‌های نو)، احساس کردم عاشقش شده‌ام. عمیقا. اجازه دهید بگویم چرا.

درباره پایتون

پیش از اینکه وارد جزئیات کار شویم به حسی بپردازم که چرا فردی می‌خواهد از پایتون استفاده کند (پیشنهاد می‌کنم صفحه پایتون در ویکیپدیا را برای جزئیات بیشتر بخوانید).

به نظر من، پایتون دارای نقاط قوت زیر است:

قابل‌حمل بودن

پایتون در هر جایی اجرا می‌شود و حمل‌ونقل یک برنامه از لینوکس به ویندوز یا مکینتاش، معمولا تنها در حد موضوع فیکس کردن مسیرها و تنظیمات است. پایتون برای قابل حمل بودن طراحی شده و در پشت رابط کاربری، مراقب تغییرات ناگهانی سیستم‌عامل مشخص می‌باشد تا مجبور نباشید سختی کدنویسی مربوط به یک پلتفرم مشخص دیگر را به جان بخرید.

انسجام

پایتون به‌شدت منطقی و منسجم است. می‌توانید ببینید که توسط یک دانشمند خبره علوم رایانه طراحی شده است. اگر متدی را نشناسید، بیشتر وقت‌ها کافیست تنها حدس بزنید چگونه آن متد فراخوانی می‌شود.

شاید اکنون به اهمیت آن پی نبرید، به‌ویژه اگر مبتدی باشید، اما این یک ویژگی اصلی آن است. یعنی کمتر موجب آشفتگی افکارتان و به‌هم ریختگی و زیرورو کردن اسنادتان می‌شود و کمتر نیاز به رجوع به مغزتان حین کدنویسی می‌شود.

بهره‌وری توسعه‌دهنده

مطابق با نظر Mark Lutz (در ویرایش پنجم کتاب Learning Python انتشارات اورلی)، یک برنامه پایتون، معمولا یک پنجم تا یک سوم از اندازه محیط کد جاوا یا C++ را می‌گیرد. یعنی اجرای سریع‌تر برنامه. و سریع‌تر بودن عالی است. سریع‌تر بودن یعنی یک پاسخ سریع‌تر به بازار. کد کمتر نه تنها به معنای نوشتن کد کمتر است بلکه کد کمتری باید خوانده شود (و کدزن‌های حرفه‌ای، بسیار بیشتر از آنکه کدنویسی کنند کدخوانی می‌کنند)، کد کمتری تعمیر و نگهداری شود و پالایش شود.

جنبه مهم دیگر این است که پایتون بدون نیاز به گردآوری زمان‌بر و طولانی و به‌هم پیوستگی گام‌ها، اجرا می‌شود، بنابراین مجبور نیستید منتظر دیدن نتایج کار خود بمانید.

یک کتابخانه وسیع

پایتون دارای یک کتابخانه استاندارد عظیم باورنکردنی است. اگر جواب کارتان را هم ندهد، جامعه پایتون در سرتاسر جهان دارای کتابخانه‌های تنومند شخص‌سومی برای نیازهای شخصی است که می‌توان آزادانه (رایگان) در Python Package Index (PyPI) به آن دسترسی داشت. هنگامی که کدهای پایتون را می‌نویسید و متوجه می‌شوید که نیاز به یک ویژگی (فیچر) مشخصی دارید، در بیشتر موارد دست‌کم یک کتابخانه موجود است که آن ویژگی، از پیش برای شما پیاده‌سازی شده است.

کیفیت نرم‌افزار

پایتون به‌شدت روی خوانا بودن، انسجام، و کیفیت متمرکز است. یکنواختی زبان، امکان خوانا بودنش را افزایش می‌دهد و امروزه، بسیار مهم است که برنامه‌نویسی، یک تلاش جمعی باشد تا یک تلاش انفرادی، و به‌شکل گروهی انجام پذیرد. جنبه مهم دیگر پایتون، طبیعت ذاتی چند الگویی آن است. از آن می‌توان به‌عنوان یک زبان اسکریپتی استفاده کرد، اما می‌توان از سبک‌های شئ‌گرایی، دستوری، و برنامه‌نویسی فانکشنال نیز بهره‌برداری کرد. در کل، استعدادش بالاست.

یکپارچگی نرم‌افزار

جنبه مهم دیگر پایتون این است که می‌تواند با بسیاری دیگر از زبان‌ها توسعه داده شده و یکپارچه شود که به معنای این است که حتی وقتی یک شرکت درحال استفاده از زبان دیگری به‌عنوان ابزار اصلی کارش است، پایتون می‌تواند وارد شود و به‌عنوان یک عامل چسب، میان برنامه‌های کاربردی پیچیده‌ای که به هر روشی نیاز به صحبت با همدیگر دارند، عمل کند. این نوعی از یک مبحث پیشرفته است اما در جهان واقعی، این ویژگی بسیار مهم است.

رضایت‌مندی و لذت

کار با پایتون، جذاب و سرگرم‌کننده است. من می‌توانم ۸ ساعت کدنویسی کنم و خوشحال و راضی، شرکت را ترک کنم و این مغایر با تقلای برنامه‌نویس‌های دیگری است که مجبورند تحمل کنند، زیرا از زبان‌هایی استفاده می‌کنند که فاقد همین مقدار ساختارهای داده‌ای و دستورهای خوش طرح هستند. تردید نکنید که پایتون، کدنویسی را بسان سرگرمی می‌کند و این، انگیزه و کیفیت کارتان را بالا می‌برد.

اینها جنبه‌های اصلی است که چرا پایتون را به هرکسی پیشنهاد می‌کنم. البته، ویژگی‌های فنی و پیشرفته بسیاری هست که می‌توان درباره آنها صحبت کرد که جای آنها در این پیش‌گفتار نیست و فصل به فصل که جلو برویم خواهید دید.

موانع چیست؟

شاید تنها مانعی که کسی بتواند در پایتون بیابد که ربطی به سلیقه‌های شخصی ندارد، سرعت اجرایش است. نوعا، پایتون آهسته‌تر از برادران کامپایل شده‌اش است. پیاده‌سازی استاندارد محصولات پایتون، وقتی یک برنامه‌کاربردی را اجرا می‌کنید، یک نسخه‌ی کامپایل شده از سورس کد به‌نام بایت کد است (با پسوند pyc.)، که سپس توسط مفسر پایتون اجرا می‌شود. مزیت این رویه در قابل حمل بودن آن است که ما بهای این کاهش سرعت ناشی از این حقیقت که پایتون مانند زبان‌های دیگر، در سطح ماشین کامپایل نمی‌شود را می‌پردازیم.

هرچند، امروزه سرعت پایتون به‌ندرت یک مشکل تقلی می‌شود، و پهنه کاربردش توجهی به این ویژگی منفی آن ندارد. اتفاقی که می‌افتد این است که در زندگی واقعی، هزینه سخت‌افزار، یک مشکل دائمی نیست، و معمولا دست‌یابی به سرعت با کارهای موازی‌سازی، به‌اندازه کافی آسان است. افزون بر این، بسیاری از برنامه‌نویسان، بخش وسیعی از زمان انتظار را صرف تکمیل عملیات IO می‌کنند.؛ بنابراین سرعت اجرای خام، اغلب یک عامل دوم در بازدهی کلی به‌شمار می‌رود. حتی وقتی پای خرد کردن کد به چند قطعه به میان می‌آید، یکی می‌تواند روی پیاده‌سازی‌های سریع‌تر پایتون همچون PyPy سوئیچ کند، که یک متوسط افزایش سرعت پنج–لا را با پیاده‌سازی تکنیک‌های کامپایل پیشرفته، ارائه می‌دهد.

هنگام پرداختن به علم داده، بیشتر تمایل به کتابخانه‌هایی دارید که با پایتون استفاده کنید، مانند Pandas و NumPy، و به سرعت محلی ناشی از روشی که آنها پیاده‌سازی می‌شوند دست‌یابید.

اگر آن استدلال به‌اندازه کافی خوب نبود می‌توانید همواره درنظر بگیرید که پایتون برای هدایت درب پشتی سرویس‌هایی همچون Spotify و Instagram به‌کار رفته است که بازدهی، امر نگران کننده‌ای است. با این حال، پایتون کار خودش را به بهترین شکل ممکن انجام داده و می‌دهد.

امروزه چه افرادی از پایتون استفاده می‌کنند؟

هنوز قانع نشده‌اید؟ اجازه دهید نگاهی به شرکت‌هایی که امروزه از پایتون استفاده می‌کنند بیاندازیم: گوگل، یوتیوب، دراپ‌باکس، یاهو!، Zope Corporation، Industrial Light & Magic، Walt Disny Feature Animation، Blender 3D، Pixar، NASA، NSA، Red Hat، Nokia، IBM، Netflix، Yelp، Intel، Cisco، HP، Qualcomm و JPMorgan Chase تنها چند مورد از آنهاست.

حتی بازی‌هایی همچون Battlefield 2، Civilization IV و QuArK با استفاده از پایتون پیاده‌سازی شده‌اند.

پایتون در مفاهیم پیچیده‌ای همچون برنامه‌نویسی سیستمی، برنامه‌نویسی وب، برنامه‌های کاربردی GUI، بازی‌سازی و روباتیک، نمونه‌سازی سرعت، تعامل و یکپارچه‌سازی سیستم، علم داده، برنامه‌های کاربردی دیتابیس، و موارد دیگر کاربرد دارد. چندین دانشگاه معتبر نیز پایتون را به عنوان زبان اصلی در رشته علوم کامپیوتر تدریس می‌کنند.

تنظیم محیط

پیش از اینکه به نصب پایتون روی سیستم‌تان بپردازیم اجازه دهید کمی درباره نسخه‌ای از پایتون که در این کتاب استفاده می‌کنیم صحبت کنیم.

پایتون ۲ در مقابل پایتون ۳

پایتون دارای دو نسخه اصلی ۲ و ۳ است که نسخه ۳ پس از نسخه ۲ آمده است. این دو، شباهت زیادی به هم دارند اما از برخی جهات، کمی با هم ناسازگارند.

در دنیای واقعی، پایتون ۲ که در سال ۲۰۰۰ منتشر شد فاصله زیادی با قدیمی شدن دارد. کوته سخن، باوجودی‌که پایتون ۳ از سال ۲۰۰۸ بیرون آمد، فاز دگرگونی آن از نسخه ۲، هنوز به پایان نرسیده است. این تقریبا ناشی از این واقعیت است که پایتون ۲ به‌شکل گسترده در صنعت به‌کار گرفته شده است و البته، شرکت‌ها آنقدر مشتاق به به‌روزرسانی سیستم‌های خود با نسخه جدید نبودند، که دلیلش مشکل نداشتن با نسخه ۲ بود. با جست‌وجو در وب می‌توانید به تفاوت‌های اساسی این دو نسخه پی برید.

مشکل دیگری که مانع این تغییر است، توانایی کتابخانه‌های شخص سومی است. معمولا، یک پروژه پایتون به ده‌ها کتابخانه خارجی تکیه دارد و البته وقتی پروژه جدیدی را آغاز می‌کنید نیاز دارید از پیش، برای هر الزام تجاری که شاید به دنبال آن بیاید، از موجود بودن کتابخانه سازگار با نسخه ۳ مطمئن شوید. اگر این مورد نباشد آغاز یک پروژه شاخه–جدید در پایتون ۳ به معنای تولید یک ریسک نهفته است، که یعنی شرکت‌ها از برداشتن آن خشنود نیستند.

هرچند، هنگام نوشتن این کتاب، شمار بسیار زیادی از کتابخانه‌های استفاده شده، به پایتون ۳ پورت شده‌اند و آغاز یک پروژه در پایتون ۳ در بیشتر موارد، امن است. بسیاری از کتابخانه‌ها از نو نوشته شده‌اند تا بیشتر تحت فشار قدرت کتابخانه six (این نام، از ضرب ۲ در ۳ می‌آید که ناشی از پورت کردن از نسخه ۲ به ۳ است)، با هر دو نسخه سازگار باشند، که به درون‌گرایی و وفق دادن رفتار متناسب با نسخه به‌کار رفته، کمک می‌کند. با توجه به PEP 373[1]، پایان عمر[2] پایتون ۲.۷ در سال ۲۰۲۰ تنظیم شده و نسخه ۲.۸ آن وجود نخواهد داشت، پس این زمانی است که شرکت‌هایی که دارای پروژه‌های اجرایی در پایتون ۲ هستند لازم است تا دیر نشده، به‌فکر تدبیر مناسب برای رفتن به پایتون ۳ باشند.

در سیستم من (MacBook Pro)، این آخرین نسخه پایتونی است که دارم:

```
>>> import sys
>>> print(sys.version)
```

[1] https://legacy.python.org/dev/peps/pep-0373
[2] End of life (EOL)

```
3.9.2 (default,  Mar 1 2021, 23:29:21)
[Clang 12.0.0 (clang-1200.0.32.29)]
```

همان‌گونه که می‌بینید یک نسخه 3.9.2 است که در یکم مارس ۲۰۲۱ انتشار یافته است. متن آغازین بالا کمی کد پایتون است که در یک کنسول تایپ شده است. در مورد این کمی جلوتر صحبت خواهیم کرد.

همه مثال‌های این کتاب با استفاده از پایتون ۳.۹ اجرا خواهد شد. اگر دوست دارید مثال‌ها را دنبال کنید و سورس‌کد را دانلود کنید، مطمئن شوید که از همین نسخه استفاده می‌کنید.

نصب پایتون

واقعا هرگز به داشتن یک بخش setup در یک کتاب متقاعد نشده‌ام، باوجودی که مجبورید حتما فرایند نصب را انجام دهید. بیشتر وقت‌ها، میان زمانی که نویسنده، دستورها را می‌نویسد و زمانی که شما آنها را پیاده‌سازی می‌کنید ماه‌ها می‌گذرد. و این بستگی به شانس شما دارد. یک نسخه تغییر می‌کند و شاید مواردی از آن با روشی که در کتاب آمده، کار نکند. خوشبختانه ما وب را داریم، پس برای کمک به نصب و اجرای پایتون، تنها به اهداف و نکات مهم فرایند نصب می‌پردازم.

می‌دانم که بیشتر خوانندگان ممکن است ترجیح دهند کتاب به‌شکل راهنمای گام به گام باشد. شک دارم این روش، کارشان را آسان‌تر کند، پس به‌شدت باور دارم که اگر می‌خواهید کار با پایتون را آغاز کنید باید نخستین تلاشتان در جهت آشنایی با زیست‌بوم آن باشد. این بسیار مهم است و موجب افزایش اعتماد به نفس شما در فصل‌های پیش رو خواهد شد. هر کجا هم که گیر کردید، دوست خوبمان گوگل را فراموش نکنید.

تنظیم مفسر پایتون

پیش از هر چیز، اجازه دهید درباره سیستم‌عامل شما صحبت کنیم. پایتون به احتمال زیاد، از پیش در تقریبا هر توزیع لینوکسی به‌شکل پایه‌ای نصب و به‌شکل کامل، با آن یکپارچه شده است. اگر دارای سیستم‌عامل مک هستید، احتمال دارد پایتون از پیش روی آن باشد (هرچند، ممکن است تنها پایتون ۲.۷ باشد)، اما اگر از ویندوز استفاده می‌کنید به احتمال زیاد، نیاز به نصب آن دارید.

دریافت پایتون و کتابخانه‌های مورد نیاز و اجرای آن نیاز به کمی کار دستی دارد. به‌نظر می‌رسد لینوکس و مک، سیستم‌عامل‌های کاربرپسندتری برای برنامه‌نویسان پایتون باشند؛ اما ویندوز، بیشترین تلاش را می‌طلبد.

کار را از وب‌سایت رسمی پایتون به نشانی https://www.python.org آغاز کنید که میزبان مستندات رسمی پایتون و بسیاری منابع مفید دیگر است. زمانی را به بررسی آن تخصیص دهید.

توجه: وب‌سایت دیگری که **پر** از منابع عالی پایتون و زیست بوم آن است http://docs.python-guide.org است. راهنمایی‌های نصب پایتون در دیگر سیستم‌عامل‌ها با روش‌های مختلف را در آن خواهید دید.

بخش دانلود را بیابید و نصب کننده سیستم‌عامل خود را انتخاب کنید. اگر روی ویندوز هستید، مطمئن شوید وقتی نصب کننده را اجرا می‌کنید، کنار گزینه install pip تیک زده باشید (واقعا پیشنهاد می‌کنم برای امنیت بیشتر، یک نصب کامل از همه کامپوننت‌هایی که نصب کننده ارائه می‌دهد را انجام دهید). درباره pip در ادامه خواهیم گفت.

اکنون که پایتون را روی سیستم خود نصب کردید، باید بتوانید یک کنسول را باز کنید و با تایپ python، پوسته تعاملی پایتون را اجرا کنید.

توجه: لطفا دقت کنید که معمولا به‌جای Python interactive shell (پوسته تعاملی پایتون)، از عبارت "کنسول پایتون" استفاده می‌کنیم.

برای باز کردن این کنسول در ویندوز، به منوی Start ویندوز بروید و در کادر Run، عبارت cmd را تایپ کنید. اگر حین کار با مثال‌های کتاب، با هرچیزی که شبیه یک مشکل مجوز است روبرو شدید، لطفا مطمئن شوید که کنسول را با مدیر سیستم یا همان administrator درستی اجرا کرده‌اید.

در macOS، می‌توانید با رفتن به Application | Utilities| Terminal، یک Terminal را آغاز کنید.

اگر هم روی لینوکس هستید، همه چیز را درباره کنسول می‌دانید.

از واژه کنسول به‌شکل مبادله‌ای برای اشاره به کنسول لینوکس، خط فرمان ویندوز، و ترمینال مکینتاش استفاده خواهم کرد. همچنین با فرمت پیش‌فرض لینوکس به کامند–لاین پرامپت (یا همان خط–فرمان سریع) اشاره خواهم کرد، مانند این:

```
$ sudo apt-get update
```

اگر با این دستور آشنا نیستید لطفا کمی وقت برای یادگیری مبانی کارکرد کنسول بگذارید. به اختصار، پس از نماد $ معمولا دستوری می‌آید که باید تایپ کنید. به بزرگی و کوچکی واژگان و فاصله‌ها دقت کنید که بسیار مهم است.

پس از باز شدن کنسول، python را در پرامپت تایپ کنید و مطمئن شوید پوسته تعاملی پایتون نمایش می‌یابد. برای خروج، ( ) exit را تایپ کنید. فراموش نکنید که اگر پایتون *2. از پیش روی سیستم‌عامل نصب باشد، شاید مجبور شوید پایتون ۳ را مشخص کنید.

پس از اجرای پایتون، باید عبارات زیر را ببینید (برخی جزئیات برپایه نسخه و OS تغییر می‌کند):

```
fab $ python3
Python 3.9.2 (default, Mar 1 2021, 23:29:21)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

اینک که پایتون نصب شده و می‌توانید آنرا اجرا کنید زمان آن است که مطمئن شوید ابزار دیگری دارید که با آن مثال‌های این کتاب را دنبال کنید: یک محیط مجازی یا virtual environment.

درباره محیط‌های مجازی

حین کار با پایتون، استفاده از محیط‌های مجازی، بسیار مرسوم است. اجازه دهید کمی توضیح دهیم که اینها چیست و چرا به اینها نیاز داریم. با یک مثال ساده، توضیح می‌دهیم.

پایتون را روی سیستم نصب می‌کنید و کار با یک وب‌سایت برای Client X را آغاز می‌کنید. یک پوشه پروژه می‌سازید و شروع به کدنویسی می‌کنید. در کنار آن، برخی کتابخانه‌ها را نیز نصب می‌کنید؛ مانند فریمورک Django که در فصل ۱۴ به آن خواهیم پرداخت. فرض کنید نسخه Django 2.2 را برای Project X نصب می‌کنید.

اکنون وب‌سایت شما آنقدر خوب است که مشتری دیگری به‌نام Y می‌گیرید. او می‌خواهد وب‌سایت دیگری بسازید، پس Project Y را آغاز می‌کنید و به همان روش، دوباره نیاز به نصب جانگو دارید. تنها مشکلی که وجود دارد این است که نسخه جانگو اکنون به 3 ارتقاء یافته است و نمی‌توانید آن را روی سیستم خود نصب کنید زیرا جایگزین نسخه پیشین آن که برای Project X نصب کرده بودید خواهد شد. نمی‌خواهید ریسک معرفی مشکلات ناسازگاری را بکنید، بنابراین دو انتخاب دارید: یا درگیر نسخه‌ای شوید که هم‌اکنون روی سیستم دارید، یا آنرا به‌روز کنید و مطمئن شوید نخستین پروژه، باز به‌شکل کامل و صحیح با نسخه جدید کار می‌کند.

اگر صادق باشیم، هیچ کدام از این انتخاب‌ها چندان خوشایند نیست. قطعا نیست. اینجاست که راه‌حل مجازی‌سازی به میان می‌آید: محیط‌های مجازی!

محیط‌های مجازی، محیط‌های ایزوله شده پایتون هستند که هر کدام، پوشه‌ای است حاوی همه اجرا شدنی‌های لازم برای استفاده از پکیج‌هایی که یک پروژه پایتون لازم دارد (فعلا پکیج‌ها را مانند کتابخانه درنظر بگیرید).

پس برای ایجاد یک محیط مجازی برای Project X، همه وابستگی‌ها را نصب کنید و سپس یک محیط مجازی برای پروژه Y بسازید و همه وابستگی‌هایش را بدون کوچکترین نگرانی نصب کنید، زیرا هر کتابخانه‌ای که بعدا نصب می‌کنید، در انتهای مرزهای محیط مجازی مناسب می‌نشیند. در این مثال، پروژه X کتابخانه Django 2.2 را نگه می‌دارد، درحالی که پروژه Y، کتابخانه Django 3.0 را نگه می‌دارد.

توجه: بسیار مهم است که هرگز کتابخانه‌ها را مستقیما در سطح سیستم نصب نکنید. مثلا لینوکس برای وظایف و عملیات گوناگونی به پایتون تکیه دارد و اگر با سیستم نصب پایتون بخواهید آزارش دهید، ریسک به‌هم ریختگی تعامل کل سیستم را بالا خواهید برد (حدس بزنید برای این فرد چه اتفاقی می‌افتد...). پس این را به‌عنوان یک قانون مدنظر داشته باشید که پیش از رفتن به تخت‌خواب، مسواک بزنید: *همیشه، همیشه هنگام آغاز یک پروژه جدید، یک محیط مجازی بسازید.*

برای ساخت یک محیط مجازی روی سیستم‌تان، از چند روش مختلف می‌توانید استفاده کنید. به‌طوری که پایتون ۳.۵، روش ساخت یک محیط مجازی که از ماژول venv استفاده می‌کند را پیشنهاد داده است. در مستند رسمی پایتون به نشانی (https://docs.python.org/3/library/venv.html) می‌توانید آنرا ببینید.

نکته: مثلا اگر از یک توزیع لینوکس Debian–محور استفاده می‌کنید، *لازم* است ماژول venv را پیش از اینکه بتوانید از آن استفاده کنید با فرمان زیر نصب کنید:

```
$ sudo apt-get install python3.9-venv
```

روش مرسوم دیگر ساخت محیط‌های مجازی، استفاده از پکیج پایتون شخص ثالثی virtualenv است. اطلاعات بیشتر آن در وب‌سایت رسمی virtualenv ارائه شده است:

https://virtualenv.pypa.io

در این کتاب، از تکنیک‌های پیشنهادی استفاده خواهیم کرد که ماژول venv را از کتابخانه استاندارد پایتون به‌کار می‌برد.

نخستین محیط مجازی‌تان

ایجاد یک محیط مجازی، بسیار آسان است، اما نسبت به نحوه پیکربندی سیستم و نسخه پایتونی که برای اجرای محیط مجازی می‌خواهید، نیاز به اجرای صحیح فرمان دارید. مورد دیگری که هنگام کار با آن باید انجام دهید، فعال‌سازی آن است. فعال‌سازی محیط‌های مجازی اصولا منجر به تولید یک‌سری مسیر شعبده‌سازی در پشت صحنه می‌شود تا وقتی مفسر پایتون را فراخوانی می‌کنید، در واقع یک محیط مجازی را صدا می‌زنید به‌جای یک سیستم صرف. به یک مثال کامل روی هر دو سیستم Windows و Ubuntu (روی یک Mac که بسیار شبیه اوبنتو است) توجه کنید. می‌خواهیم:

۱. یک پایانه (terminal) باز کنید و به پوشه‌ای (دایرکتوری) که به‌عنوان ریشه پروژه‌های‌مان استفاده می‌کنیم تغییرش دهید (پوشه ما srv است). می‌خواهیم پوشه جدیدی به‌نام my-project بسازیم و به آن تغییرش دهیم.

۲. یک محیط مجازی به‌نام lpp3ed ایجاد کنید.

۳. پس از ایجاد محیط مجازی، آنرا فعال خواهیم کرد. روش‌های موجود، برای هر سیستم‌عاملی اندکی متفاوت است.

۴. سپس مطمئن می‌شویم که با اجرای پوسته تعاملی پایتون، درحال اجرای نسخه دلخواه (3.9.*) هستیم.

۵. در پایان، با استفاده از فرمان deactivate، محیط مجازی را غیرفعال کنید.

توجه: برخی توسعه‌دهندگان ترجیح می‌دهند همه محیط‌های مجازی را با استفاده از یک نام صدا بزنند (مانند venv.). با این روش می‌توانند اسکریپت‌ها را در مقابل هر محیط مجازی تنها با دانستن نام پروژه‌ای که در آن اقامت دارند اجرا کنند. دلیل وجود نقطه در venv. این است که سیستم‌عامل‌های لینوکس و مک، با درج نقطه در نام، آن فایل یا پوشه را نامرئی می‌کنند.

این پنج گام ساده، همه آن چیزی که برای آغاز و استفاده از یک پروژه لازم دارید را نشان می‌دهد.

می‌خواهیم با مثالی روی ویندوز آغاز کنیم (دقت کنید که بسته به OS خود، نسخه پایتون و غیره، شاید نتیجه‌ی اندک متفاوتی بگیرید). در لیست‌بندی زیر، خطوطی که با یک # آغاز می‌شود، کامنت هستند، فاصله‌ها برای خوانایی بیشتر اضافه می‌شود، و فلش ↪ اشاره به جایی دارد که ادامه‌ی محتویات خط به‌دلیل کمبود جا، به خط بعدی منتقل شده است:

```
C:\Users\Fab\srv>mkdir my-project # step 1
C:\Users\Fab\srv>cd my-project
C:\Users\Fab\srv\my-project>where python # check system python
C:\Users\Fab\AppData\Local\Programs\Python\Python39\python.exe
C:\Users\Fab\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\Fab\srv\my-project>python -m venv lpp3ed # step 2

C:\Users\Fab\srv\my-project>lpp3ed\Scripts\activate # step 3

# check python again, now virtual env python is listed first
(lpp3ed) C:\Users\Fab\srv\my-project>where python
C:\Users\Fab\srv\my-project\lpp3ed\Scripts\python.exe
C:\Users\Fab\AppData\Local\Programs\Python\Python39\python.exe
C:\Users\Fab\AppData\Local\Microsoft\WindowsApps\python.exe

(lpp3ed) C:\Users\Fab\srv\my-project>python # step 4
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55)
↪ [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

(lpp3ed) C:\Users\Fab\srv\my-project>deactivate # step 5
C:\Users\Fab\srv\my-project>
```

هر گام با یک کامنت نشانه‌گذاری شده است تا بتوانید به سادگی آنها را دنبال کنید.

در یک ماشین لینوکس، گام‌ها یکسان است، اما فرمان‌ها کمی متفاوت هستند. حتی شاید مجبور شوید برای اینکه بتوانید از ماژول venv برای ساخت یک محیط مجازی استفاده کنید، چند گام تنظیماتی دیگر را اجرا کنید. آموزش همه توزیع‌های لینوکس در اینجا غیرممکن است، پس لطفا با نگاهی در وب، آنچه برای توزیع خودتان مناسب است را بیابید.

به محض تنظیم شدن، دستورالعمل‌های لازم برای ایجاد یک محیط مجازی را خواهیم داشت:

```
fab@fvm:~/srv$ mkdir my-project # step 1
fab@fvm:~/srv$ cd my-project

fab@fvm:~/srv/my-project$ which python3.9 # check system python
/usr/bin/python3.9 # <-- system python3.9

fab@fvm:~/srv/my-project$ python3.9 -m venv lpp3ed # step 2
fab@fvm:~/srv/my-project$ source ./lpp3ed/bin/activate # step 3

# check python again: now using the virtual environment's one
(lpp3ed) fab@fvm:~/srv/my-project$ which python
/home/fab/srv/my-project/lpp3ed/bin/python

(lpp3ed) fab@fvm:~/srv/my-project$ python # step 4
Python 3.9.2 (default, Feb 20 2021, 20:56:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

(lpp3ed) fab@fvm:~/srv/my-project$ deactivate # step 5
fab@fvm:~/srv/my-project$
```

مورد دیگری که باید توجه کنید این است که به منظور فعال‌سازی محیط مجازی، نیاز به اجرای اسکریپت lpp3ed/bin/activate داریم که لازم است به‌شکل منبع در آید. هنگامی که یک اسکریپت، sourced می‌شود، یعنی در پوسته جاری اجرا می‌شود و بنابراین در انتها پس از اجرا تأثیر می‌گذارد. این بسیار مهم است. همچنین توجه کنید که چگونه پس از فعال کردن محیط مجازی، پرامپت تغییر می‌کند و نام آنرا در سمت چپ نشان می‌دهد (و چگونه وقتی آنرا غیرفعال می‌کنیم ناپدید می‌شود).

در اینجا باید بتوانید یک محیط مجازی را ایجاد و فعال کنید. لطفا تلاش کنید و محیط مجازی دیگری را بدون راهنمایی من ایجاد کنید. با این رویه به‌خوبی آشنا شوید زیرا این چیزی است که همواره با آن سر و کار خواهیم داشت: عبارت "ما هرگز در پایتون به‌شکل سیستم–گستر کار نمی‌کنیم" را به‌یاد دارید؟ محیط‌های مجازی بسیار مهم است.

سورس‌کد کتاب، حاوی یک پوشه اختصاصی برای هر فصل است. هنگامی که کد نشان داده شده در هر فصل، نیاز به نصب کتابخانه‌های شخص ثالثی دارد، یک فایل requirements.txt را درج خواهیم کرد (یا یک پوشه requirements معادلش با بیش از یک فایل تکست درونش) که بتوانید برای نصب کتابخانه‌های لازم برای اجرای کد، استفاده کنید. پیشنهاد می‌کنیم هنگام آزمایش کدهای هر فصل؛ یک محیط مجازی اختصاصی برای آن فصل بسازید. با این کار، خواهید توانست تمرین‌های خوب و مناسبی در ایجاد محیط‌های مجازی و نصب کتابخانه‌های شخص سومی انجام دهید.

نصب کتابخانه‌های شخص ثالثی

توجه: به‌منظور نصب کتابخانه‌های شخص ثالثی، نیاز به استفاده از Python Package Installer یا همان pip داریم. خوش شانسید اگر از پیش، درون محیط مجازی‌تان موجود باشد، اگر نبود، مستند آنرا در نشانی http://pip.pypa.io بخوانید.

مثال زیر نشان می‌دهد چگونه یک محیط مجازی ایجاد کنیم و یک جفت کتابخانه شخص ثالثی برگرفته از یک فایل ملزومات را نصب کنیم.

```
mpro:srv fab$ mkdir my-project
mpro:srv fab$ cd my-project/

mpro:my-project fab$ python3.9 -m venv lpp3ed
mpro:my-project fab$ source ./lpp3ed/bin/activate

(lpp3ed) mpro:my-project fab$ cat requirements.txt
Django==3.1.7
requests==2.25.1

# the following instruction shows how to use pip to install
# requirements from a file
(lpp3ed) mpro:my-project fab$ pip install -r requirements.txt
Collecting Django==3.1.7
Using cached Django-3.1.7-py3-none-any.whl (7.8 MB)

... much more collection here ...

Collecting requests==2.25.1
   Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)

Installing collected packages: ..., Django, requests, ...
Successfully installed Django-3.1.7 ... requests-2.25.1 ...

(lpp3ed) mpro:my-project fab$
```

همان‌گونه که در انتهای این کد می‌بینید، pip هر دو کتابخانه‌ای که در فایل ملزومات است را به همراه چند مورد دیگر نصب نموده است. دلیلش این است که هم django و هم requests دارای لیست کتابخانه‌های شخص ثالثی خودشان هستند که به آنها وابسته‌اند، و بنابراین، pip آنها را به‌شکل خودکار برایمان نصب خواهد کرد.

بنابراین با داربست‌بندی مسیر، آماده صحبت بیشتری درباره پایتون و نحوه کار با آن می‌شویم. پیش از آن اجازه دهید کمی درباره چند واژه مرتبط با کنسول صحبت کنیم.

دوستمان کنسول

در این سطح از GUIها و وسایل لمسی، به‌نظر می‌رسد کمی مسخره باشد که مجبور شویم در ابزاری همچون کنسول، پی‌درپی رفت و آمد کنیم وقتی هر کاری را تنها با یک کلیک می‌توان انجام داد.

اما حقیقت این است که هر بار برای گرفتن ماوس و حرکت دادن اشاره‌گر روی نقطه‌ای که می‌خواهید کلیک کنید، دست راست خود را از روی کیبورد برمی‌دارید (یا دست چپ‌تان اگر چپ دست هستید) زمان را از دست می‌دهید. انجام کارهایی با کنسول، متضاد ادراک مستقیم حسی است، و شاید به سودمندی و سرعت بالاتری منتهی شود. به این باور دارم، از این نظر به من اعتماد کنید.

سرعت و سودمندی، مهم است و شخصا مشکلی با ماوس ندارم اما دلیل بسیار خوب دیگری برای اینکه شاید بخواهید به شناخت خوبی از کنسول پی ببرید وجود دارد: وقتی کدی را توسعه می‌دهید که روی برخی سرورها باید بالا بیاید، شاید کنسول تنها ابزار موجود باشد. اگر با آن دوست شوید قول می‌دهم هرگز بی‌خیال آن نشوید و تا با آن کار نکنید به اهمیت آن پی نخواهید برد (مثلا وقتی وب‌سایت غیرفعال شده و مجبورید خیلی سریع ببینید چه اتفاقی افتاده است).

بنابراین، این واقعا به شما بستگی دارد. اگر بی طرف هستید لطفا از فایده تردید به من بگویید و آنرا امتحان کنید. این آسان‌تر از آن است که فکرش را کنید و هرگز افسوس آنرا نمی‌خورید. چیزی رقت انگیزتر از یک توسعه‌دهنده خوب نیست که درون یک ارتباط SSH با یک سرور گم شود چون آنها برای مجموعه ابزارهای سفارشی خودشان به‌کار می‌روند و تنها برای آن.

اجازه دهید به پایتون برگردیم.

چگونگی اجرای یک برنامه پایتون

به چند روش مختلف می‌توان یک برنامه پایتون را اجرا کرد.

اجرای اسکریپت‌های پایتون

پایتون می‌تواند به‌عنوان یک زبان اسکریپتی به‌کار رود. در حقیقت، پایتون همواره خودش را خیلی مفید اثبات می‌کند. اسکریپت‌ها فایل‌هایی هستند (معمولا از ابعاد کوچک) که به‌صورت طبیعی برای انجام چیزی مانند یک وظیفه، اجرا می‌شوند. بسیاری از توسعه‌دهندگان، در نهایت به فکر داشتن زرادخانه ابزارهای خود می‌افتند تا وقتی نیاز به انجام یک کار شدند، از آن استفاده کنند. برای نمونه، می‌توان اسکریپت‌هایی برای تجزیه تحلیل داده‌ها (پارس‌کردن) به یک فرمت و رندر کردن آن به فرمتی دیگر داشت. یا می‌توان از یک اسکریپت، برای کار با فایل‌ها و پوشه‌ها استفاده کرد. فایل‌های پیکربندی را می‌توان ایجاد کرد یا تغییر داد و موارد بسیار زیاد دیگر. از لحاظ فنی، کاری نیست که نتوان در یک اسکریپت انجام داد.

داشتن اسکریپت‌های اجرایی در یک زمان ویژه روی یک سرور، کاملا مرسوم است. برای نمونه، اگر دیتابیس وب‌سایت شما نیاز به پاک‌سازی هر ۲۴ ساعت یک‌بار داشته باشد (مثلا جدول نگهدارنده نشست‌های کاربر که به‌سرعت، کاملا منقضی می‌شود اما به‌شکل خودکار پاک‌سازی نمی‌شود)، می‌توانید یک کار Cron تنظیم کنید که اسکریپت را رأس ساعت ۳ صبح هر روز، شلیک کند.

توجه: آنچه در ویکی‌پدیا آمده، نرم‌افزار سودمند Cron، یک کار زمان–محور زمان‌بندی کننده در سیستم‌عامل‌های رایانه‌ای Unix–گونه است. افرادی که محیط‌های نرم‌افزاری را تنظیم و تعمیر و نگهداری می‌کنند، از Cron برای زمان‌بندی کارهایشان (فرمان‌ها یا اسکریپت‌های پوسته) استفاده می‌کنند تا به‌شکل دوره‌ای در زمان‌ها، تاریخ‌ها یا فواصل ثابت اجرا شود.

من اسکریپت‌های پایتونی دارم که همه کارهای دم‌دستی که انجام دستی آنها دقایقی از وقتم را می‌گیرند را انجام می‌دهند و در برخی مقاطع، تصمیم به خودکارسازی آنها می‌گیرم. در فصل ۱۲، GUIها و اسکریپت‌ها، روی اسکریپت‌نویسی با پایتون کار می‌کنیم.

اجرای پوسته تعاملی پایتون

روش دیگر اجرای پایتون، فراخوانی (صدازدن) پوسته تعاملی است. این چیزی است که پیش‌تر هنگامی که python را در خط فرمان کنسول خود تایپ کردیم دیدیم.

بنابراین، یک کنسول را باز کنید، محیط مجازی خود را فعال کنید (که اینک با این باید دومین مورد برایتان باشد، درست است؟) و python را در آن تایپ کنید. چند خط که شاید همانند زیر باشد نمایش داده می‌شود:

```
 (lpp3ed) mpro:my-project fab$ python
Python 3.9.2 (default, Mar 1 2021, 23:29:21)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```