



# from hello world to web developer

مؤلف:

مهندس پارسا پاکدل لاهیجی



مؤسسه آموزشی تألیفی ارشدان

**from hello world to web developer**

پارسا پاکدل لاهیجی  
آموزشی تألیفی ارشدان  
اول  
اول ۱۴۰۴  
[www.irantypist.com](http://www.irantypist.com)  
[www.irantypist.com](http://www.irantypist.com)  
۱۰۰۰  
[www.arshadan.com](http://www.arshadan.com)  
[www.arshadan.net](http://www.arshadan.net)  
۰۲۱۴۷۶۲۵۵۰۰  
تومان

■ نام کتاب:  
■ تألیف:  
■ ناشر:  
■ ویرایش:  
■ نوبت چاپ:  
■ حروفچینی و صفحه آرایی:  
■ طراح و گرافیست:  
■ شابک:  
■ شمارگان:  
■ مرکز خرید آنلاین:  
■ مرکز پخش و توزیع:  
■ قیمت:

پیشگفتار ناشر:

به نام ایزد دانا که آغاز و انجام از آن اوست

هرگز دل من زعلم محروم نشد      کم ماند زاسرار که مفهوم نشد  
اکنون که به چشم عقل در می نگرم      معلوم شد که هیچ معلوم نشد

ای دانای بی همتا، ای بخشنده‌ای که ناخواسته عطا فرمایی و هر نیازمندی را به عدالت بی‌نیاز گردانی، مگر اینکه نالایق باشد و آن عنایت را به بازگونه از دست دهد. در عرصه پیشرفت تکنولوژی در هزاره سوم، هنوز نیاز بر مطالعه کتاب در کنار استفاده از منابع کامپیوتری و اینترنت احساس می‌شود. از این بابت خوشحالیم که می‌توانیم در جهت اعتلای علم، دانش و فرهنگ کشور قدمی هر چند کوچک برداریم.

و من الله التوفيق

دکتر شمس الدین یوسفیان

مدیر مسئول انتشارات ارشدان



## فهرست مطالب

۹	فصل اول: مقدمه .....
۱۰	۱-۱- مقدمه.....
۱۱	۱-۲- از حجم کتاب نترسید.....
۱۱	۱-۳- در تماس باشید.....
۱۳	فصل دوم: آموزش جامع صفر تا صد پایتون .....
۱۴	۲-۱- چرا برنامه نویسی؟.....
۱۸	۲-۲- نصب و راهاندازی پایتون.....
۲۰	۲-۳- ساختار کدها و متغیرها.....
۲۴	۲-۴- دستورات شرطی و حلقهها.....
۳۱	۲-۵- توابع.....
۳۸	۲-۶- ساختار پایه دادهها.....
۴۸	۲-۷- کلاس‌ها و مازول‌ها.....
۶۲	۲-۸- کار با فایل‌ها و خطاهای .....
۷۰	۲-۹- معرفی چند کتابخانه مهم پایتون.....
۷۵	فصل سوم: آشنایی با خط فرمان لینوکس BASH .....
۷۶	۳-۱- تعریف BASH .....
۷۶	۳-۲- اسکریپت‌نویسی .....
۷۷	۳-۳- مثال‌های اسکریپت .....
۹۱	فصل چهارم: آشنایی با پایگاه‌های داده .....
۹۲	۴-۱- MY SQL .....
۱۰۳	۴-۲- SQLLITE3 .....
۱۰۷	۴-۳- SQLALCHEMY .....

۱۱۳.....	<b>فصل پنجم: HTTP &amp; SSH</b>
۱۱۴.....	۱-۵ ساختار پروتکل HTTP
۱۲۳.....	۲-۵ آشنایی با SSH و اتصال به سرور
۱۲۷.....	<b>فصل ششم: GIT &amp; GITHUB</b>
۱۲۸.....	۶-۱ سیستم کنترل نسخه
۱۳۱.....	۶-۲ دستورات اصلی گیت
۱۳۶.....	۶-۳ آشنایی با گیت‌ها
۱۳۹.....	<b>فصل هفتم: HTML &amp; CSS</b>
۱۴۱.....	۷-۱ ساختار صفحات وب
۱۴۲.....	۷-۲ تگ‌های مهم HTML
۱۵۵.....	۷-۳ طراحی ظاهر با CSS
۲۶۱.....	<b>فصل هشتم: فریم ورک قدرتمند توسعه وب جنگو</b>
۲۶۵.....	۸-۱ مقدمه
۲۶۹.....	۸-۲ داکر
۲۸۱.....	۸-۳ POSTGRE SQL
۲۹۱.....	۸-۴ پروژه کتابفروشی آنلاین
۲۹۹.....	۸-۵ PAGES APP
۳۰۷.....	۸-۶ ثبت‌نام مقدماتی کاربر
۳۱۹.....	۸-۷ ثبت‌نام پیشرفته کاربر
۳۲۹.....	۸-۸ متغیرهای اینوایرنمنت
۳۳۵.....	۸-۹ بخش نهم
۳۴۰.....	۸-۱۰ اپ BOOKS
۳۵۵.....	۸-۱۱ دسترسی‌ها
۳۶۰.....	۸-۱۲ جستجو
۳۶۷.....	۸-۱۳ کارایی

۳۷۷	۱۴-۸- امنیت.....
۳۹۴	۸-۱۵- توسعه پژوهش.....
۴۰۳	کلام آخر.....



# فصل اول: مقدمه

## ۱-۱- مقدمه

در دنیای امروز آشنایی با مفاهیم مرتبط با توسعه وب نه تنها یک مهارت بلکه ابزاری برای خلق نوآوری و حل مسئله است

بسیاری از منابع آموزشی به صورت پراکنده در دسترس هستند و یافتن مسیر یادگیری منسجم دشوار است.

هدف از نگارش این کتاب ارائه مسیر جامع و مرحله به مرحله برای یادگیری طراحی سایت و توسعه وب با تمرکز ویژه بر زبان پایتون و فریم ورک جنگو است. در این مسیر تلاش شده تا مفاهیم از پایه ترین مباحث مانند بش و گیت آغاز شده و به تدریج به مباحث پیشرفته تری نظیر پایگاه های داده طراحی ای پی آی استفاده از داکر و توسعه سیستم های تحت وب با جنگو برسیم.

ساختار کتاب به گونه ای تنظیم شده که مخاطب بتواند بدون پیش نیاز خاص تنها با داشتن علاقه و پشتکار مفاهیم را یه صورت کاربردی و هدفمند بیاموزد. این کتاب حاصل تجربه شخصی در مسیر یادگیری ساعت های طولانی مطالعه تمرین و آزمون خط بوده است و سعی شده از بیان پیچیده مفاهیم اجتناب شده و هر فصل به زبانی ساده اما دقیق نگارش شود.

امید است این کتاب بتواند راهنمایی موثر برای علاقه مندان به حوزه وب باشد و گامی کوچک در جهت گسترش دانش و مهارت های فنی در این زمینه محسوب شود.

با آرزوی موفقیت و پیشرفت

با تجدید احترام

پارسا پاکدل

## ۱-۲- از حجم کتاب نترسید

شاید در نگاه اول حجم این کتاب کمی زیاد بنظر برسد طبیعی است که حس کنید قرار است با یک کوه از اطلاعات روبرو شوید

اما خبر خوب اینجاست قرار نیست همه چیز را یک شبه یاد بگیرید. کتاب را به چشم یک مسیر یادگیری بینید نه یک مانع.

هدف من این نیست که شما همه چیز را حفظ کنید بلکه می خواهم شما را به درک عمیق و عملی از مفاهیم برسانم. شما با پیشرفت در مطالعه خواهید دید که هر فصل به طور منطقی به فصل بعد منتقل می شود و در نهایت مفاهیم به طور یکپارچه در ذهن شما نقش می بندد. پس بدون استرس فصل به فصل جلو بروید توقف کنید تمرین کنید سوال بپرسید و اگر لازم شد چند بار مرور کنید

این بخشی از مسیر طبیعی یادگیری است.

مطمئن باشید هیچ مطلبی از نظر شما پیچیده و دور از دسترس نخواهد بود هر زمان نیاز به راهنمایی داشتید در این کتاب پاسخ پرسش های خود را خواهید یافت.

## ۱-۳- در تماس باشید

در طول مطالعه این کتاب اگر با پاسخی مواجه شدید یا بخشی از مطالب برایتان روشن نبود خوشحال می شوم از طریق ایمیل با من در ارتباط باشید. نظرات انتقادات و پیشنهادات شما می تواند به بهبود نسخه های بعدی این کتاب کمک شایانی کند. همچنین برای مشاهده نمونه پژوهه ها و فعالیت های من در حوزه‌ی برنامه نویسی می توانید به صفحه گیت هاب من مراجعه کنید.

Parsapakdel290@gmail.com

Github.com/parsapakdellahidji



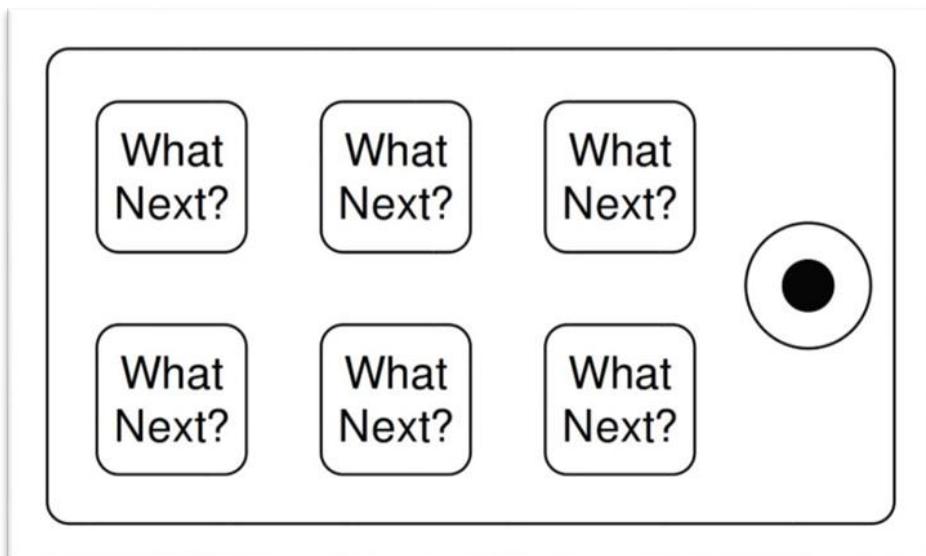
# فصل دوم: آموزش جامع صفر تا صد پایتون

## ۲- چرا برنامه نویسی؟

نوشتن برنامه یا همان برنامه نویسی، یک فعالیتیست که ارزشش را دارد و در جایی به خودتان می‌آید و میبینید که خسته کننده نیست، حوصله تان هم سر نمیرود، و تازه از انجامش لذت هم میبرید. هر کسی دلایل خودش را برای برنامه نویسی دارد که اتفاقاً کم هم نیستند. شاید بخواهید از طریقش کسب درآمد کنید و یا با استفاده از آن مساله‌ی سختی را در خصوص آنالیز دادهها حل کنید؛ شاید تنها به قصد تفریح برنامه نویسی کنید و در این بین مسائل دیگران را با آن حل کنید. به هر حال این کتاب فرض میکند که همه نیاز دارند که برنامه نویسی را یاد بگیرند. وقتی که مهارت‌ش را کسب کردید، میتوانید به این فکر کنید که خب حالا چه استفاده‌های از آن کنم؟.

ما با کامپیوترها محاصره شده ایم. حالا این کامپیوترها فقط آنهایی نیستند که جعبه‌ی بزرگی دارند و یک مانیتور روی میز جا خوش کرده، بلکه از لپتاپ گرفته تا گوشی موبایلی که همیشه در جیبتان است، یک کامپیوتر به حساب می‌آیند. این کامپیوترهای کوچک و بزرگ کارهای زیادی را برای ما و به جای ما انجام میدهند.

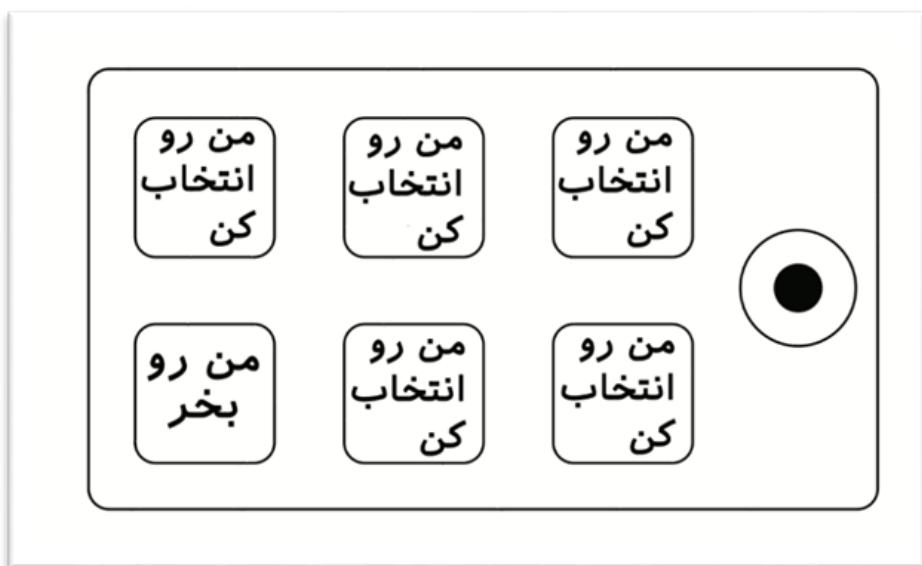
سخت افزاری که در کامپیوترهای امروزی به کار رفته به طریقی ساخته شده که دائم از شما میپرسد «حالا باید چه کاری انجام دهم رئیس؟



کار برنامه نویسان اضافه کردن سیستم عامل و دسته هایی از ابزارها و برنامه ها به سخت افزار است. چیزی که در نهایت تبدیل به دستیار شخصی دیجیتالی میشود. این دستیار شخصی دیجیتال، قادر به انجام کارهای زیادی برای ماست.

### خلاقیت و انگیزه

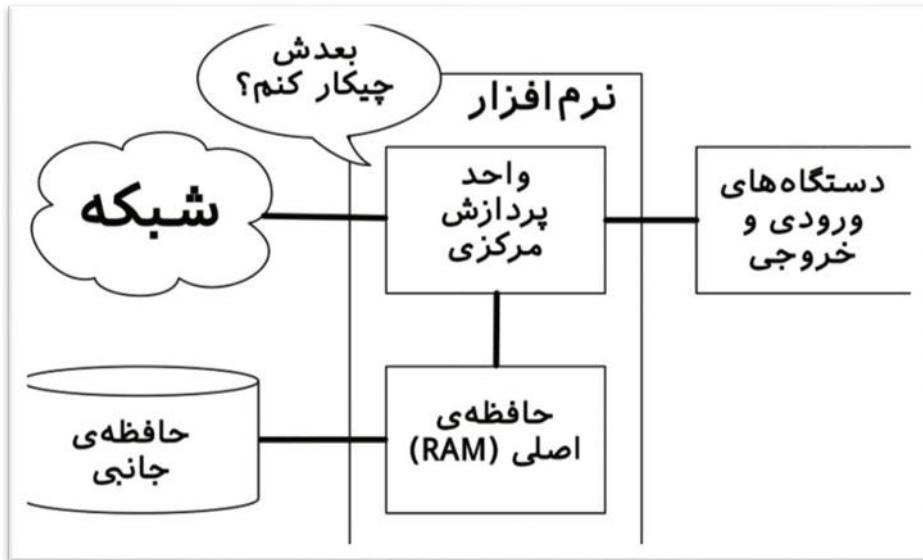
برنامه نویسی حرفه ای یک شغل ارزشمند است که نه تنها انجامش به شما حس خوبی را میدهد که از نظر اقتصادی هم کار مناسبی به حساب می آید. ساخت برنامه های هوشمندانه، زیبا و مفید برای دیگران، یک فعالیت خلاقانه محسوب می شود. اگر به برنامه ها به عنوان خروجی خلاقیت یک دسته برنامه نویس نگاه کنیم، تصویر زیر قیاس خوبیست برای نشان دادن آن چیزی که بر صفحه‌ی موبایل شما نمایش داده میشود. تصویری که در پنج حالت میگوید «مرا انتخاب کن» و در یک حالت میگوید «مرا بخر» که خوب با خرید شما، خیر مادی به برنامه نویسان خواهد رسید.



### معماری و سخت افزار کامپیووتر

قبل از اینکه زبان سخن گفتن با کامپیووتر برای اجرای دستورالعمل هایی را، که از او میخواهیم، یاد بگیریم، بد نیست کمی در خصوص چگونگی ساخت و کارکرد خود کامپیووتر بیاموزیم. اگر قرار باشد

که کامپیوتر یا موبایل خودتان را به قطعه های کوچکی تقسیم کنید، وقتی که در کیس را باز میکنید با این قطعه های اصلی رو برو خواهید شد:



معنای سطح بالای این قسمتها به شرح زیر است:

الف) سی پی یو: قسمتی از کامپیوتر است که به صورت وسواس گونه و پشت سر هم میپرسد «بعدش چیه؟ حالا چیکار کنم؟!» اگر پردازنده‌ی سیستم شما سرعتیش ۳.۰ گیگا هرتز است، این بدان معناست که سی پی یو شما ۳ میلیارد بار در ثانیه آن را از شما می‌پرسد به عبارتی بهتر است یاد بگیرید که خیلی سریع حرف بزنید تا به گرد پای پردازندگان بررسید.

ب) حافظه‌ی اصلی یا RAM اطلاعات را برای استفاده‌ی CPU که خیلی هم عجله دارد نگهداری و مهیا می‌کند. حافظه‌ی اصلی هم سرعتی نزدیک به پردازنده دارد؛ ولی مشکلش این است که تا کامپیوتر را خاموش میکنید، اطلاعات موجود در آن هم دود میشوند و به هوا میروند.

پ) حافظه‌ی جانبی نیز به مانند حافظه‌ی اصلی، اطلاعات را در خود ذخیره می‌کند ولی با این تفاوت که خیلی کنُدتر است. مزیتش چیست؟ حتی با خاموش کردن کامپیوتر، اطلاعات روی حافظه‌ی جانبی باقی میمانند. هارد دیسکها، فلش‌ها از انواع حافظه‌های جانبی به حساب می‌آیند.

ت) دستگاههای ورودی و خروجی شامل صفحه‌ی نمایش، چاپگر، کیبورد، میکروفون، بلندگو، تاج پد و غیره می‌شوند. دستگاههایی که به نحوی تعامل شما با کامپیوتر را برقرار می‌کنند، دستگاههای ورودی یا خروجی‌اند.

ث) این روزها، اکثر کامپیوتراها یک اتصال به شبکه را برای دریافت اطلاعات از طریق شبکه با خود دارند. ما میتوانیم به شبکه به عنوان مکانی نگاه کنیم که داده‌ها را از طریق آن ذخیره و دریافت می‌کنیم. البته سرعت نقل و انتقال داده‌ها کم است و همیشه هم در دسترس نیستند. به عبارتی شبکه، حافظه‌ای جانبی است که گند است و نمی‌شود با اطمینان بر آن تکیه زد.

### درک برنامه‌نویسی

در ادامه‌ی این کتاب ما تلاش می‌کنیم تا شما را تبدیل به شخصی کنیم که در هنر برنامه‌نویسی مهارت لازم را کسب کرده‌است. در انتهای شما یک برنامه‌نویس خواهید بود حالا نه برنامه‌نویس حرفه‌ای ولی حداقل مهارت بررسی یک مساله با آنالیز داده‌ها و اطلاعات و توسعه‌ی برنامه‌های برای حل آن را پیدا خواهید کرد.

در کل شما به دو مهارت برای برنامه‌نویس شدن نیاز دارید:

الف) باید یک زبان برنامه‌نویسی را یاد بگیرید (در این کتاب: پایتون) باید که با فرهنگ لغات و اصول نگارش آن آشنا شوید. بایستی بتوانید که کلمه‌ها را درست بنویسید و جمله‌های خوبی با آنها به زبان جدید بسازید.

ب) باید یک داستان سر هم کنید. داستانسرایی چیست؟ ترکیب کلمه‌ها و جمله‌ها و پرده برداشتن از ایده‌های برای خواننده. هنر و مهارت نقش اساسی را در ساخت یک داستان ایفا می‌کنند. مهارت داستان نویسی با دست به قلم شدن و دریافت بازخورد از خواننده‌ها یا حتی خودتان با مرور دوباره میسر می‌شود. در برنامه‌نویسی، برنامه‌ی ما همان داستان است و مساله‌های که قرار است حل کنیم، ایده‌ای از این داستان.

زمانیکه یک زبان برنامه‌نویسی مثل پایتون را یاد گرفتید، فرا گرفتن دومین و چندین زبان برنامه‌نویسی جاوا اسکریپت یا C++ بسیار ساده‌تر می‌شود. درست است که در زبان جدید بایستی که کلمه‌ها دستور زبان آن را مجدداً یاد بگیرید، ولی مهارت اصلی حل مساله در بین همه‌ی زبانهای برنامه‌نویسی یکسان است. لغات و جملات پایتون را خیلی سریع یاد خواهید

گرفت ولی برای نوشتن برنامه‌های منسجم که مسائل جدید را حل کنند به زمان بیشتری احتیاج خواهید داشت. در این کتاب ما سعی میکنیم که برنامه‌نویسی را شبیه به نویسنده‌گی به شما یاد دهیم. ابتدا شروع به خواندن و توضیح برنامه‌ها میکنیم، سپس برنامه‌های ساده ای خواهیم نوشت و بعد از آن به پیچیدگی برنامه‌هایی که مینویسیم اضافه میکنیم. در نهایت به جایی میرسید که سبک خودتان را در برنامه نویسی و دیدن الگوها پیدا خواهید کرد. مسائل را به روش خودتان میبینید و برنامه‌های برای حل آن به شیوه‌ی ویژه خودتان خواهید نوشت؛ و آن موقع است که به خودتان می‌آید و میبینید که چقدر نوشتند برنامه‌ها برایتان لذت‌بخش است و روند خلاقانه‌ای دارد. ما با لغات و ساختار برنامه‌های پایتون شروع میکنیم. ممکن است که در ابتدای کمی سادگی مسائل، حوصله سربر باشد و خاطره‌های سالهای اول ابتدایی را برایتان زنده کند، ولی بایستی که صبور باشید و آرام آرام آنها را فرا بگیرید.

### مسیر یادگیری

گاهی در مسیر یادگیری با خود فکر میکنید که «مطلوب این کتاب با هم جور در نمی‌آید؛ چرا سر در نمی‌آورم؟» درست مانند یادگیری «حرف زدن» است. یکی دو سالی طول کشید تا فقط یاد گرفتید که صدا در بیاورید. سپس شش ماهی طول کشید که از یک کلمه‌ی ساده به یک جمله برسید و شاید ۲ الی ۵ سال زمان برد تا از آن جمله‌های کوتاه یک پاراگراف بسازید. در نهایت هم باز چند سالی وقت صرف کردید تا بالاخره توانستید یک داستان کوتاه‌جذاب را بنویسید.

ولی هدف ما این است که شما پایتون را خیلی سریعتر یاد بگیرید برای همین در چند فصل ابتدایی همه‌ی این موارد را دَرَهم به شما یاد خواهیم داد. ولی مانند یادگیری یک زبان جدید، درک مفاهیم و اُخت گرفتن با آنها زمان میبرد.

## ۲-۲- نصب و راهاندازی پایتون

### کلمات

کلماتی که در پایتون رزرو شده هستند به شرح زیر است:

```

and      del      global     not      with
as       elif     if         or       yield
assert   else     import    pass]
break   except   in        raise
class   finally  is        return
continue for    lambda   try
def     from    nonlocal while

```

با این کلمات در زمان خودشان به طور مفصل آشنا خواهیم شد.

### نصب و راه اندازی

حالا که چند کلمه ای پایتونی یاد گرفتیم بایستی یاد بگیریم که چطور با استفاده از کلمات و جمله ها شروع به صحبت کردن با پایتون کنیم. پس بد نیست که توانایی خودمان در زبان جدید را به بوته ی آزمایش بگذاریم. قبل از اینکه بتوانید با او صحبت کنید، بهتر است که به خانه تان دعوتش کنید. پس لازم است که مراحل نصب آن روی کامپیوتربان را با هم مرور کنیم. ما در این قسمت در خصوص نصب پایتون و راه اندازی آن روی ویندوز صحبت خواهیم کرد. کاربران لینوکس حتما میدانند که چطور پایتون رو از طریق مخازن نرم افزاری خود به سادگی نصب کنند.

برای نصب پایتون وارد سایت [python.org](http://python.org) شده دو نسخه ۳۲ و ۶۴ بیتی در دسترس است که با توجه به معماری سیستم خود یکی از آنها را دریافت کنید. من در این کتاب از پایتون ۳ استفاده میکنم پس مطمئن باشید که نسخه مرتبط را دانلود میکنید. در زمان نگارش این کتاب نسخه پایتون آن ۳,۱۳,۳ است.

وقتی نصب ظاهر شد گزینه Add python to PATH را تیک بزنید و نصب را ادامه دهید. حال کافیست از منوی استارت ویندوز Cmd را اجرا کنید و زمانی که باز شد عبارت python را بنویسید و وارد شوید.

اعلان مفسر >>> پایتون نشان دهنده ی آمادگی کدنویسی است.

## نصب کد ادیتور

کد ادیتورهای زیادی وجود دارد که ما در اینجا با یکی از بهترین و محبوب ترین شان به نام vscode کار میکنیم. برای دانلود وارد سایت visual studio.com شده و در تاب دانلود را دانلود میکنیم و به سادگی نصب میکنیم.

پس از باز کردن vscode بر روی آیکن افزونه کلیک میکنیم و عبارت python را تایپ و نصب میکنیم. با نصب این افزونه کد ادیتور ما به یک IDE قادرمند تبدیل خواهد شد.

## ۳-۲- ساختار کدها و متغیرها

یک مقدار مثل یک حرف یا یک عدد پایه ای ترین چیزیست که برنامه با آن کار میکند. مثلا ۲ یک عدد صحیح (Integer) و ۳.۶ یک عدد اعشاری (Float) "Hello World!" یک رشته (String) است. ولی چرا "رشته"? چون شامل رشته هایی از حروف میشود. حالا شما یا مفسر چگونه تشخیص میدهید که "Hello World!" یک رشته است؟ پاسخ ساده است: رشته ها بین علامت های کوتویشن قرار میگیرند.

تابع print چه برای رشته چه برای عدد صحیح برای نمایش دادن به کار می رود.

```
python
>>> print(4)
4
```

اگر میخواهید نوع یک مقدار را از مفسر بپرسید، میتوانید از type استفاده کنید:

```
>>> type('Hello, World!')
<class 'str'>
>>> type(17)
<class 'int'>
```

## متغیرها

متغیر یا variable نامی است که به یک مقدار اشاره می کند یک Statement Assignment یا گزاره گمارشی، یک متغیر را میسازد؛ البته اگر پیشتر موجود نباشد؛ سپس مقداری را به آن نسبت یا اختصاص میدهد.

>>> message = 'And now for something completely different'

>>> pi = 3.1415926535897931

در مثال بالا ما دو مقدار را به دو متغیر اختصاص داده‌ایم. اولین خط، یک پیغام را به متغیر message، و دومی مقدار تقریبی عدد pi را به متغیر pi اختصاص داده است.

## گزاره‌ها

یک گزاره، واحدی از کد است که مفسر پایتون توانایی اجرایش را داشته باشد. ما تا اینجا دو نوع گزاره دیده‌ایم: print برای چاپ یک عبارت و گزاره‌ی مربوط به گمارش یا اختصاص چیزی به متغیر (Assignment).

وقتی شما یک گزاره را در حالت تعاملی وارد می‌کنید، مفسر آن را اجرا و سپس نتیجه را نمایش میدهد. البته اگر نتیجه‌ی قابل مشاهده‌ای داشته باشد یک اسکریپت معمولًا شامل یک سری گزاره می‌شود. اگر بیش از یک گزاره در اسکریپت داشته باشیم، نتیجه در زمانیکه گزاره در داخل اسکریپت اجرا می‌شود، نمایش داده می‌شود. به عنوان مثال:

Print(1)

X = 2

Print(x)

به ترتیب خروجی ۱ و ۲ را خواهد داشت. به دو نکته در اسکریپت بالا توجه داشته باشید. اول اینکه گزاره‌ها به ترتیب از بالا به پایین خوانده شدن و دوم اینکه دومین گزاره یعنی  $x=2$  در اسکریپت بالا یک گزاره‌ی گمارشی است و خروجی قابل نمایش ندارد.

## قرارداد در متغیرها

الف) نام‌ها باید با معنی باشد

ب) تفاوتی ندارد از حروف کوچک یا بزرگ استفاده شود اما طبق قرارداد از حروف کوچک استفاده می‌کنیم.

پ) متغیر‌های دو بخشی با \_ از هم جدا می‌شوند.

ت) اسم متغیر نمی‌تواند با عدد شروع شود

## عملوند و عملگر

**عملوند:** در ریاضیات و برنامه نویسی رایانه، یک عملوند هدف یک عملیات ریاضی است. هر عبارت که بین دو عملگر قرار بگیرد و یا بعد از یک عملگر باید یک عملوند محسوب می‌گردد.

**عملگر:** در ریاضیات، یک عملگر تابعی است که بر تابعی دیگر اعمال می‌شود. در مواردی ممکن است یک عملگر را یک عمل ریاضی گویند مانند «عمل جمع» که در اصل عملگر جمع می‌باشد. در علوم کامپیوتر پس از تعریف متغیرها و مقدار دادن به آنها. عملیاتی روی آنها انجام می‌شود. انجام عملیات توسط عملگر صورت می‌گیرد.

عملگرهای + و - و \* و / و // و \*\* به ترتیب جمع، تفریق، ضرب، تقسیم، تقسیم مسطح و بهتوانرساندن را انجام میدهند.

در پایتون اولویت ابتدا با پرانتز سپس توان بعد ضرب و تقسیم و در نهایت جمع و تفریق است.

رشته

رشته یا string دنباله‌ای از کارکتر است.

```
name = 'ali'
```

در پایتون اگر بخواهیم متنی را به خط بعد ببریم راه‌های زیادی وجود دارد اما بهترین روش استفاده از سه تا کوتیشن است:

```
X = " ali
```

```
Y= akbari "
```

```
>>> ali
```

```
akbari
```

به مثال‌های زیر با دقت نگاه کنید

```
x = 'parsa'
print(len(x)) >>> 5
print(x[2])>>>>r
print(x[0:3])>>>par
print(x[-3])>>>>s
```

در خط اول متغیر `parsa` را تعریف کردیم، در خط دوم باتابع `len` طول کارکتر را بدست آوردم. در خط سوم سومین کارکتر را بدست آوردیم ( دقت کنید تعداد کارکتر از صفر شروع می شود ) در خط چهارم تعداد کارکتر مبدا تا مقصد را خودمان تعیین کردیم و در خط پنجم کارکتر سوم از آخر را بدست آوردیم.

### فرمت های string

به مثال های زیر توجه کنید

```
x = 'ali'
y = 'ahmadi'
print(x+y)>>> aliahmadi
print(x,y)>>>ali ahmadi
print(f'hello mr{x , y} goodbye')>>> hello mr ali ahmadi goodbye
```

در خط سوم اگر از `+ استفاده کنیم دو رشته را به هم می چسباند و در خط چهارم اگر از کاما استفاده کنیم از هم فاصله می دهد. در خط پنجم هر زمان که بخواهیم تغییری در رشته متن ایجاد کنیم از f string استفاده می کنیم.`

### متدهای string

برای رسیدن به متدها از استفاده میکنیم. در زیر به چند مورد از مهمترین متدهای رشته اشاره میکنیم:

(1) `upper()`: برای بزرگ کردن حروف

(2) `lower()`: برای کوچک کردن حروف

(3) `find()`: پیدا کردن شماره حروف

(4) `strip()`: حذف کردن فاصله ها

(5) `replace()`: جایگزینی حروف

به مثال های زیر توجه کنید

```
x = 'alireza'
# from hello world to web developer
print(x.find('e'))>>>4
print(x.replace('v', 'a'))>>>vlirezv
```

در خط سوم و چهارم با توجه به توضیحات قبل یکسری عملیات متعدد روی رشته ها انجام دادیم و نتیجه اش را در جلویشان دیدیم اما چرا در خط دوم هیچ جوابی نگرفتیم؟ در پایتون هر زمان که بخواهیم متنی بنویسیم برای خوانایی بهتر کد که چاپ هم نشود از `# استفاده` میکنیم.

حال اگر بخواهیم

```
x = 'parsa'
print('p'in x)>>>True
print('n'in x)>>>False
```

`True` و `False` در پایتون جزو دسته بولین قرار می گیرند.

در خواست ورودی از کاربر

گاهی لازم است که یک مقدار را از کاربر ورودی بگیریم. در این حالت از تابع `input` استفاده می کنیم.

```
>>> name = input('What is your name?\n') What is your name?
Chuck
>>> print(name) Chuck
```

آن `\n` در انتهای رشته بیانگر این است که ورودی را در «خط جدید» بگیرد و مخفف است. این نویسه مختص شکستن خط و رفتن به خط بعدی است؛ و به همین خاطر است که ورودی کاربر در خط جدید ظاهر میشود. به خط سوم در کد بالا نگاه کنید.

## ۴-۲- دستورات شرطی و حلقه ها

بولی و عبارت های بولی

یک عبارت Boolean چیست؟ خب به فارسی به آن بولی میگوییم و عبارتیست که یا غلط است یا صحیح.

انواع عملگرهای مقایسه‌ای عبارتند از:

$x \neq y$

$x$  با  $y$  برابر نیست:

$x > y$

$x$  از  $y$  بزرگتر است:

$x < y$

$x$  از  $y$  کوچکتر است:

$x \geq y$

$x$  بزرگتر یا مساوی با  $y$  است:

$x \leq y$

$x$  کوچکتر یا مساوی با  $y$  است:

$x \text{ is } y$

$x$  درست همان  $y$  است:

$x \text{ is not } y$

$x$  همان  $y$  نیست.

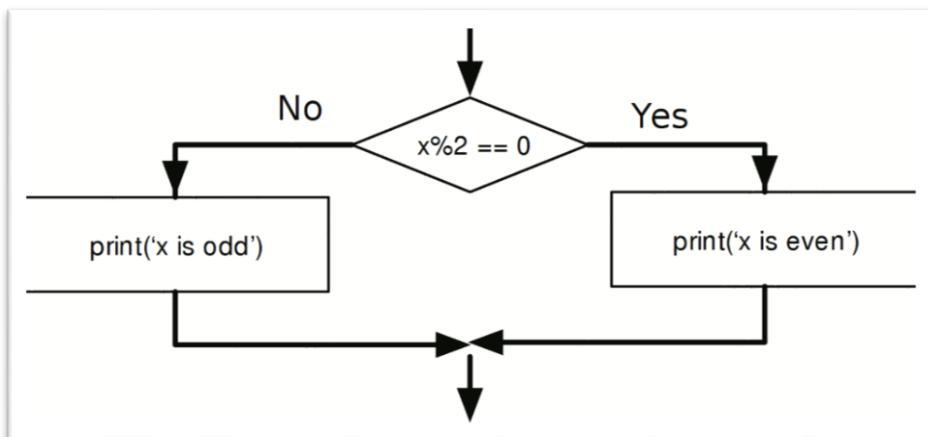
### عملگرهای منطقی

سه عملگر منطقی وجود دارد. این عملگرهای منطقی عبارتند از «`and`» و «`or`» و «`not`». معنای این علمگرها دقیقاً برابر با معنی لفظی آنهاست. مثلاً عبارت زیر را درنظر بگیرید:

`x>5 and x<8` باشد  
`x<3 or x>6` باشد  
`not (x>y)` از وای بزرگتر باشد

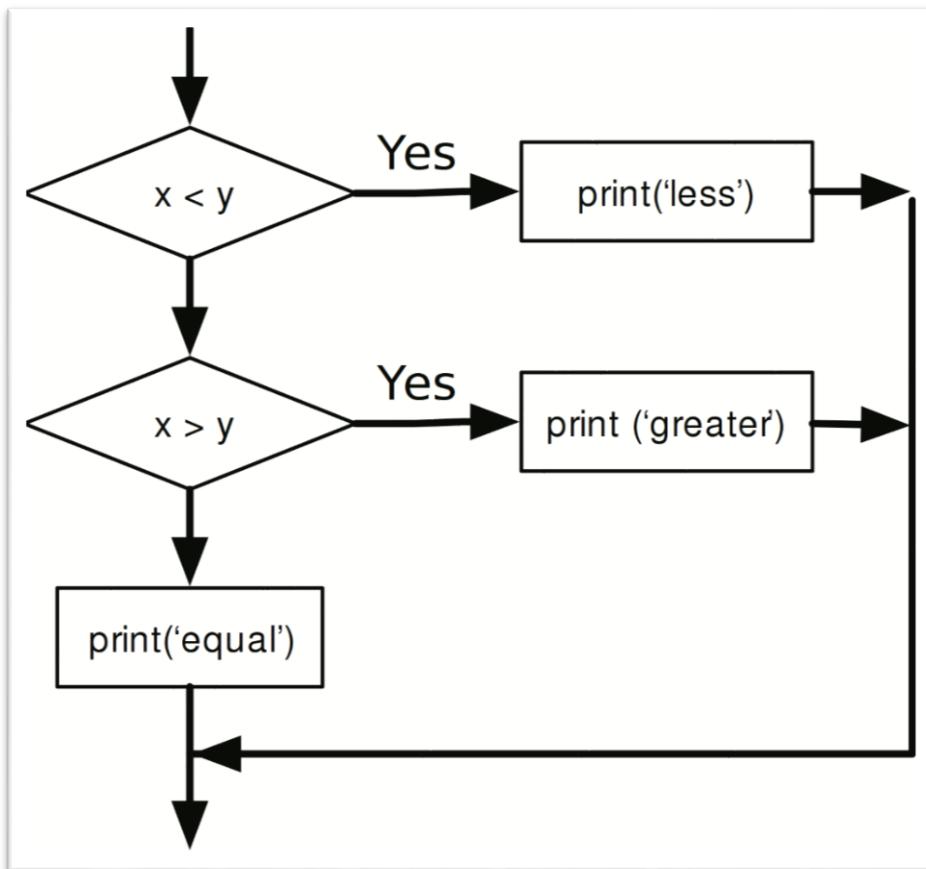
### اجرای شرطی

برای ساختن برنامه های به درد بخور، معمولا همیشه بایستی از شروط استفاده کنیم. بررسی کنیم که اگر فلان، در نتیجه بیسار و برنامه عکسالعمل و رفتار مناسبی را در موقع لازم نشان دهد. ولی چگونه؟ گزارهای شرطی به ما این امکان را می دهد. عبارت بولی بعد از if، شرط خوانده میشود. ما گزارهی if را با کلون یا دو نقطه (:) به پایان میرسانیم و سپس خطوط مربوط به آن شرط را مینویسیم. خطوط بعد از if که جزئیات آن شرط به حساب می آیند به اندازه space فرورفتگی دارند. پایتون به این تورفتگیها بسیار حساس است. اگر شرط منطقی درست باشد، در نتیجه گزارهی تورفته اجرا میشود. در غیر این صورت برنامه از روی آن گزاره پوش میکند.



فرم دیگری از گزارهی if فرم اجرای ثانویست. در این حالت «اگر / » گزارهی شرطی درست نباشد از آن شرط پرش کرده و به «در غیر این صورت / » رفته و گزارهی مربوط به آن را اجرا میکند.

گاهی بیش از دو احتمال وجود دارد و ما نیاز به شاخههای بیشتری در برنامهمان داریم. یکی از راههای نوشتن این مدل شرطها استفاده از شرطهای زنجیروار است.



در پایین یک مثال کامل از دستورات شرطی if میزنیم:

```

x = int(input('enter your income(1-100):'))
if x<10:
    print('you are poor')
elif x<50:
    print('you are medile')
elif x<100:
    print('you are good')
else:
    print( 'you are rich')
  
```

نکته ۱) ورودی input همیشه str است پس نمی شود اعمال ریاضی روی آن انجام داد. در نتیجه باید int آنرا بگیریم و سپس عملیات ریاضی را انجام دهیم.

نکته ۲) در استفاده از if و elif محدودیت نداریم.

### استفاده از try&except برای استثنایها

پیشتر یاد گرفتیم که با input از کاربر، ورودی بگیریم و سپس با استفاده ازتابع `int` آن را به عدد صحیح تبدیل کنیم. یادتان می‌آید که ورودی `input` به عنوان نوع استرینگ «ذخیره می‌شود. زمانی که مار در حال اجرای گزاره‌ای با خطا روبرو شدیم نمی‌خواهیم برنامه مان کرش کند. پس از استفاده از try & except می‌توانیم به `try` و `except` به مثابه یک ضمانتنامه نگاه کنیم. ضمانت اجرای پاره‌های از کدها.

به مثال زیر توجه کنید

```
inp = input('Enter Fahrenheit Temperature:') try:
    fahr = float(inp)
    cel = (fahr - 32.0) * 5.0 / 9.0 print(cel)
except:
    print('Please enter a number')
```

پایتون شروع به اجرای بلوک try می‌کند. اگر همه چیز به خوبی پیش برود از بخش except برش کرده و به اجرای ادامه‌ی اسکریپت می‌پردازد. ولی اگر مورد خاصی در بلوک try حادث شود، از آن قطعه کد خارج شده و به اجرای سلسله گزاره‌های بلوک except می‌پردازد.

### حلقه while

برای عبارات نامعین به کار می‌رود و هر بار دستور را بررسی می‌کند و تا زمانی که دستور درست باشد در حلقه می‌ماند. ما به این جریان اجرا، loop یا حلقه می‌گوییم. ولی چرا حلقه؟ چون جریان اجرا را دور زده و باز به مرحله‌ی اول باز می‌گردد. به هر بار که بدنه‌ی این حلقه را اجرا می‌کنیم یک «تکرار» می‌گوییم.

`x = 100`

`while x>0:`

```
print(x)
x//2
>>>100
50
25
12
6
3
1
```

گاهی تا زمانیکه به اواسط بدنه حلقه نرسید، نمیدانید که باید این حلقه را تمام کنید یا نه. در این زمان میتوانید که یک حلقه‌ی بینهایت را از عمد بنویسید و سپس با استفاده از گزاره break از حلقه بیرون بپرید. فرض کنید که شما میخواهید که کاربر مدام ورودی به برنامه بفرستد و این کار تنها زمانی به پایان برسد که کاربر عبارت done را تایپ کند. کد زیر را ببینید:

while True:

```
line = input('> ')
if line == 'done':
    break
print(line)
print('Done!')
```

شرط اصلی حلقه True است و به این خاطر که یک مقدار ثابت است، حلقه‌ی ما بینهایت خواهد بود. ولی زمانی که شرط داخل بدنه‌ی حلقه درست از آب در بیاید، گزاره‌ی break اجرا شده و ما از حلقه خارج میشویم.

گاهی در حین تکرار حلقه میخواهید که تکرار جاری را تمام کرده و سریع به سراغ تکرار بعدی بروید، بدون اینکه گزاره‌های بعدی موجود در حلقه، در آن تکرار اجرا شوند. اینجاست که از گزاره continue استفاده میکنیم. کد زیر را در نظر بگیرید. این کد، ورودی کاربر را گرفته و تا زمانیکه کاربر done را وارد کند، آنها را چاپ میکند. ولی یک استثنای این وسط وجود دارد. اگر کاربر رشته‌ای وارد کند که با # شروع شده باشد، برنامه آن را چاپ نخواهد کرد و به سراغ دریافت مجدد ورودی از کاربر میرود.

```
while True:
```

```
    line = input('> ') if line[0] == '#':
```

```
        continue
```

```
    if line == 'done': break
```

```
    print(line) print('Done!')
```

تمام خطوط غیر از آن خطی که با علامت هش شروع شده بود، چاپ شدند. دلیلش این است که آن رشته، شرط موجود در بدنهٔ حلقه را محقق کرده و گزارهٔ continue اجرا نمی‌شد. با اجرا شدن continue، جریان اجرا بدون اجرا کردن ادامهٔ گزاره‌های حلقه، به سراغ تکرار بعدی می‌رود. به همین خاطر در آن مورد خاص، برنامه از اجرای گزارهٔ print باز ماند.

## حلقه for

برای تکرار در یک عبارت معین بکار می‌رود بطوریکه روی یک رشته تکرار می‌کند و در هر تکرار یک عنصر از آن رشته را پردازش می‌کند.

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends:
    print('Happy New Year:', friend) print('Done!')
>>>
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

حلقه‌هایی که می‌شمارند و جمع می‌زنند

```
count = 0
```

```
for itervar in [3, 41, 12, 9, 74, 15]:
```

```
    count = count + 1
```

```
    print('Count: ', count)
```

قبل از اینکه حلقه شروع شود، مقدار متغیر count را صفر قرار میدهیم. سپس یک حلقه مینویسیم. این حلقه تک تک آیتم‌های لیست را دور زده و به ازای هر کدام از آنها یک بار

گزاره‌ی موجود در بدن را اجرا می‌کند. در اینجا نام متغیر تکرار itervar است که کنترل حلقه را در دست دارد.

```
total = 0
```

```
for itervar in [3, 41, 12, 9, 74, 15]:
```

```
    total = total + itervar
```

```
    print('Total: ', total)
```

در این حلقه ما از متغیر تکرار استفاده کرده‌یم. به جای اضافه کردن عدد یک به count – به مانند مثال قبل – ما عدد موجود در لیست (3, 41, 12, 9, ...) را به مقدار کل، در حین اجرای آن تکرار اضافه نموده‌ایم. به عبارتی متغیر total حاوی مقداری برابر با جمع مقادیر تا آنجای کار می‌شود. قبل از اینکه حلقه شروع شود مقدار total صفر بود، چرا که هنوز مقدار اختصاص داده شده اولیه را در خود داشت و عملیاتی روی آن صورت نگرفته بود. در حین اجرای حلقه، مقدار total برابر با حاصل جمع مقدارهای موجود در لیست می‌شود و در پایان، مقدار total برابر با جمع تمامی مقادیر موجود در لیست خواهد بود.

همینطور که حلقه اجرا می‌شود، جمع عناصر موجود در معادله را حساب می‌کند؛ یک متغیر که به این صورت مورد استفاده قرار می‌گیرد، accumulate یا انباشتگر خوانده می‌شود.

## ۲-۵- توابع

### احضار توابع

در برنامه‌نویسی، یک فانکشن یا تابع، سلسله‌ای از گزاره‌ها تحت نام یا عنوانیست که یک محاسبه‌ی خاص را انجام میدهد. زمانیکه شما یک تابع را تعریف می‌کنید، در حقیقت یک نامی برای آن انتخاب کرده و سپس تحت آن نام یک سری گزاره را می‌گنجانید. در ادامه می‌توانید آنها را توسط همان نام «کال»

### مثال

```
>>>type(32)
```

```
<class 'int'>
```

نام تابع در مثال بالا type است. عبارتی که در پرانتز آمده «آرگویمنت» آن تابع به حساب می‌آید. اگر آرگویمنت شامل یک متغیر یا مقدار شود، ما آنها را به تابع – به مانند مثال بالا – ارسال می‌کنیم. نتیجه احضار تابع type در مثال بالا، نشان دادن نوع داده آرگویمنتیست که به آن ارسال کردہ‌ایم.

## توابع توکاری شده

پایتون تعدادی از توابع مهم را در خودش دارد و لازم نیست که برای احضارشان، ابتدا آنها را تعریف کنیم. سازنده‌ی پایتون، یک سری تابع برای حل مسائل معمول نوشته و در داخل پایتون برای استفاده‌ی ما قرار داده است. مثلاً تابع min و max بیشترین و کمترین مقدار موجود در یک لیست را برمی‌گرداند.

```
>>> max('Hello world')>>> 'w'  
>>> min('Hello world')>>> ''
```

این توابع فقط محدود به رشته‌ها نیستند و میتوانند روی هر مقداری محاسبات لازم را انجام و جواب پس بدهند.

به یاد داشته باشید که اسم تابع توکاری شده را به عنوان نام متغیر استفاده نکنید. اسم متغیر خود را

## تابع‌های تبدیل نوع

پایتون شامل تابعهایی برای تبدیل یک مقدار به «نوع» دیگر می‌شود. برای نمونه تابع int یک مقدار را از شما گرفته، و در صورت وجود شرایط بودنش، یک عدد صحیح تحویل شما میدهد. اگر هم مقدار وجود شرایط نباشد، این تابع به شما در خصوص مشکل گزارش خواهد داد:

```
>>> int('32')
32
>>> int('Hello')
ValueError: invalid literal for int() with base 10: 'Hello'
```

تابع `int` میتواند یک عدد اعشاری را نیز به عدد صحیح تبدیل کند. البته کارش گرد کردن نیست، بلکه قسمت اعشاری را میبرد:

```
>>> int(3.99999)
3
>>> int(-2.3)
-2
```

تابع `float` پک عدد صحیح یا رشته را گرفته و به عدد اعشاری تبدیل میکند:

### اعداد تصادفی

ساخت یک برنامه‌ای که کاملا در مقابل یک محاسبه جبری و قطعی قرار بگیرد کار ساده‌ای نخواهد بود. حداقل به نظر می‌آید که نیست. یکی از این روشها استفاده از الگوریتمهایی است که اعداد شبه تصادفی ایجاد میکنند. اعداد شبه تصادفی، واقعاً تصادفی نیستند و بر اساس محاسبات جبری به دست می‌آیند ولی برای ما و شما، تشخیص غیرتصادفی بودن آنها غیرممکن است. مازول `random`، توابعی را فراهم میکند که پایتون بتواند این اعداد شبه تصادفی را ایجاد نماید. ما در اینجا به اعداد شبه تصادفی، اعداد تصادفی میگوییم. تابع `random` در پایتون، یک عدد اعشاری بین ۰.۰ و ۱.۰ ایجاد میکند که شامل خود عدد ۱.۰ نمیشود. هر زمان که `random` را فراخوانی کنید، یک عدد تصادفی‌بین‌لند در این محدوده دریافت میکنید. برای نمونه نگاهی به اسکریپت زیر بیندازید (میتوانید این کدها را در حالت تعاملی پایتون نیز وارد کنید):

```
import random
for i in range(10):
    x = random.random()
    print(x)
```

با اجرای این اسکریپت یک عدد تصادفی بین ۰.۰ و ۱.۰ ایجاد می‌شود.

تابع random یکی از چندین توابعیست که کارهای مرتبط با اعداد تصادفی را انجام میدهد.

تابع randint پارامترهای low و high را گرفته و یک عدد صحیح بین آنها برمیگرداند:

```
>>> random.randint(5, 10)
```

5

```
>>> random.randint(5, 10)
```

9

## توابع ریاضی

پایتون یک مازول به اسم math دارد که تابع ریاضی پر استفاده را در اختیار شما قرار میدهد.

قبل از اینکه از آن استفاده کنید، لازم است که درونریزی اش کنید:

```
import math
```

```
print(math.cos(0))
```

```
>>>1
```

## تعریف تابع و استفاده از آن

```
def print_lyrics():
```

```
    print("I'm a lumberjack, and I'm okay.")
```

```
    print('I sleep all night and I work all day.')
```

```
def repeat_lyrics():
```

```
    print_lyrics()
```

```
    print_lyrics()
```

```
repeat_lyrics()
```

برنامه شامل دو تابع تعریف شده به اسمهای print\_lyrics و repeat\_lyrics میشود. تابع repeat\_lyrics مثل سایر گزاره ها اجرا میشوند ولی نتیجه ی اجراشان ساخته شدن فانکشن آبجکتهاست. گزاره های داخل تابع تا زمانیکه تابع فراخوانی نشود، اجرا نخواهند شد و خود تابع هیچ خروجی را نمیسازد.

همانگونه که انتظار می‌رود، قبل از اجرای یک تابع، بایستی که آن را تعریف کرده باشید. به عبارت دیگر، تعریف تابع بایستی که قبل از اولین باری که فراخوانده می‌شود آورده شود.

## جريان اجرا

برای اینکه تشخیص دهید تابع را در کجا یک برنامه تعریف کنید، بایستی بدانید که در چه زمانی برای اولین بار فراخوانده می‌شود. به عبارتی بایستی از جریان اجرای یک برنامه مطلع باشید. اجرا همیشه از اولین خط برنامه آغاز و در هر لحظه یک گزاره به ترتیب از بالا به پایین اجرا می‌شود. توابع جریان و روند اجرای یک برنامه را دستکاری نمی‌کنند ولی گزاره‌های داخل یک تابع، تا زمانیکه آن تابع فراخوانی نشود، اجرا نخواهد شد. فراخوانی یک تابع شبیه به یک راه فرعی در درون برنامه و جریان اجراست. به جای اینکه روند اجرای برنامه به خط بعد از تابع برود، مسیرش را به سمت تابع کج کرده و به داخل بدنه‌ی آن می‌پردازد. سپس گزاره‌های موجود در بدنه‌ی تابع را اجرا نموده و در نهایت به مکانی که مسیرش را کج کرده بود بر می‌گردد و جریان اجرا را از سر می‌گیرد.

حالا هدف از این قصه‌ی سر در گم ما چه بود؟ خواستیم بگوییم که زمانیکه یک برنامه را می‌خوانید همیشه از بالا به پایین و به ترتیب نخوانید، بلکه بهتر است که جریان اجرا را دنبال کنید. مثلاً لازم نیست گزاره‌های یک تابع را تا قبل از فراخوانی آن، صرفاً به خاطر اینکه در بالای برنامه قرار گرفته، بخوانید و آنالیز کنید.

## پارامترها و آرگویمنت‌ها

```
def greet(first_name):
    print(f'hi {first_name}')
greet('parsa')
>>> hi ali
```

پارامتر First\_name است.

آرگومان Ali است.

## Keyword arguments

هنگام ورودی به تابع خوانایی کد را بالا می برد.

```
def x (number,by):
    return(number,by)
print(x(number=2,by=1))
>>>3
```

## args\*

تابع می تواند تعداد نامشخصی آرگومان به صورت تاپل (با تاپل در فصل بعد آشنا خواهد شد) بپذیرد.

```
def multiply(*number):
    total=1
    for number in numbers:
        total *= number
    return total
print(multiply(2,4,5,6,7))
```

## \*\*args

تابع می تواند تعداد نامشخصی آرگومان کلید مقدار به صورت دیکشنری (با دیکشنری در فصل بعد آشنا خواهد شد) بپذیرد.

```
def save_user(**user):
    print(user)
save_user(id=1 , name = 'parsa')
>>> {'id':1 , 'name':'parsa'}
```

نکته: این دو تابع برای توابع یا متدهایی هستند که ورودی شان مشخص نیست.

## اسکوپ

محدوده‌ای است به عنوان مرزهایی از کدام بخش و کدام کد به کدام متغیر دسترسی دارد. متغیرها به دو دسته محلی و گلوبال تقسیم می‌شود

```
massage = 'a'
```

```
def greet(name):
```

```
    massage = 'b'
```

```
greet('parsa')
```

```
print(massage)
```

```
>>>a
```

را آورد چون متغیر گلوبال است.

### چرا از توابع استفاده می‌کنیم؟

خرد کردن برنامه به توابع مختلف چه مزیت‌هایی دارد؟ در اینجا به دلایلی که به خاطرش ما یک برنامه را به تابع‌ها می‌شکنیم، مرور می‌کنیم:

۱) ساخت یک تابع جدید به شما امکان نامگذاری روی دستهای از گزاره‌ها را میدهد. به این صورت برنامه‌ی شما خواناتر شده، بهتر درک می‌شود و راحت‌تر می‌تواند اشکال‌زدایی شود.

۲) توابع، اندازه‌ی برنامه‌ی ما را کوچک‌تر می‌کنند. عدم نیاز به تکرار یکسری گزاره که تحت نامی به عنوان تابع در برنامه تعریف کرده‌ایم، به ما این امکان را میدهد که برنامه‌های خود را بدون از دست دادن کارایی کوچک‌تر کنیم.

۳) تقسیم یک برنامه‌ی بزرگ به توابع به شما امکان اشکال‌زدایی قسمتهای مختلف را در آن واحد میدهد. سپس می‌توانید این قسمتها را سر هم کنید و در نهایت کل بدنه‌ی برنامه را بسازید.

۴) توابعی که خوب طراحی شده‌اند، می‌توانند توسط برنامه‌های زیادی مورد استفاده قرار بگیرند. زمانیکه یک تابع را نوشته‌ید و اشکال‌زدایی کردید، می‌توانید با خیال راحت از آن در سایر برنامه‌های خود نیز استفاده کنید.

## ۶-۲- ساختار پایه داده‌ها

این ساختار به چهار دسته تقسیم می‌شود: ۱- لیست ۲- تاپل ۳- مجموعه ۴- دیکشنری

(List)

لیست یک توالی از مقدارها است. در یک لیست مقدارها هر چیزی میتوانند باشند. مقدارها در یک لیست المنتها، آیتمها و یا عناصر لیست خوانده میشوند. چندین راه برای ساخت یک لیست جدید وجود دارد؛ ساده‌ترین راه، قرار دادن عناصر لیست در داخل قلاب است. همانطور که انتظار می‌رود شما میتوانید مقدارهای لیست را به متغیرها نسبت دهید:

X = [10, 'ali', 32.5, False]

### لیست‌های تغییرپذیرند

متن کد برای دسترسی به عناصر یک لیست، دقیقاً مثل سینتکس برای دسترسی به کاراکترهای یک رشته است: عملگر قلاب. عبارت داخل قلاب، شاخص را تعیین می‌کند. به خاطر داشته باشید که شاخصها با عدد 0 شروع می‌شوند: برخلاف رشته‌ها، لیست‌ها تغییرپذیرند؛ به عبارتی شما میتوانید که ترتیب آیتمهای یک لیست را تغییر دهید و یا آیتمها را دوباره به آن اختصاص دهید. زمانیکه عملگر قلاب در سمت چپ یک گمارش قرار می‌گیرد، عنصری از لیست که قرار است مقدار تازه‌ای بگیرد را مشخص می‌کند. بگذارید با یک مثال کمی روشنتر منظورم را بیان کنم:

```
>>> numbers = [17, 123]
```

```
>>> numbers[1] = 5
```

```
>>> print(numbers)
```

```
[17, 5]
```

عنصر یکم در لیست numbers عدد 123 بود؛ با استفاده از دستور گمارش که با علامت = مشخص می‌شود، میتوان این مقدار را تغییر داد. کافیست که در سمت چپ با استفاده از شاخص عنصر آن را مشخص کردیم.

و سپس مقدار جدید را در سمت راست معادله قرار دهیم. به این صورت مقدار جدید با مقدار قبلی عنصر در لیست تعویض میشود.

### پیش رفتن در یک لیست

اگر شما بخواهید که عناصر را آپدیت کنید و چیزی روی آنها بنویسید، نیاز به شاخص خواهد داشت. یک راه حل رایج برای اینکار ترکیب دو تابع range و len برای این کار است:

for i in range(len(numbers)):

    numbers[i] = numbers[i] \* 2

این حلقه در یک لیست پیش میرود و هر عنصر را به صورت جداگانه به روز میکند. len برابر با تعداد عناصر یک لیست خواهد بود. range لیستی از شاخصها از ۰ تا n - ۱ را بر میگرداند. در اینجا n طول یک لیست است. در هر بار چرخش این حلقه i شاخص عنصر بعدی خواهد بود. دستور گمارش در بدنه با استفاده از i شروع به خواندن مقدار قدیمی کرده و سپس مقدار جدید را به آن اختصاص میدهد.

### عملیات‌های لیست

۱) با استفاده از عملگر + میتوانید لیستها را به هم بچسبانید:

۲) با استفاده از عملگر \* میتوانید یک لیست را به تعداد دلخواه تکرار کنید:

```
>>> a = [1, 2, 3]
```

```
>>> b = [4, 5, 6]
```

```
>>> c = a + b>>>
```

```
>>> print(c)
```

```
[1, 2, 3, 4, 5, 6]
```

```
[0] * 4
```

```
[0, 0, 0, 0]
```

```
>>> [1, 2, 3] * 3
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

## قاج زدن لیست (عملگر اسلایس)

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t[1:3] ['b', 'c']
```

```
>>> t[:4]
```

```
['a', 'b', 'c', 'd']
```

```
>>> t[3:]
```

```
['d', 'e', 'f']
```

اگر به جای اولین شاخص چیزی ننویسید، از ابتدای لیست شروع میشود. اگر دومین شاخص را ننویسید، قاج زدن تا انتهای لیست ادامه پیدا خواهد

## متدهای لیست

به چند مورد از مهمترین متدهای لیست می پردازیم:

به آخر لیست اضافه میکند. : append(a

از آخر لیست حذف می کند. : pop(b

تما لیست را پاک می کند. : clear(c

تعداد یک عنصر در لیست را می شمارد. : count(d

دو لیست را به هم می چسباند : extend(e

در جایی که بخواهیم به لیست اضافه میکند : insert(f

خود مقدار لیست را حذف میکند : remove(g

لیست را معکوس می کند : reverse(h

لیست را مرتب می کند. : sort(i

در زیر یک مثال کامل از تمام متدهای ذکر شده در بالا آورده شده است:

```

my_list = [3, 1, 4, 1, 5]
my_list.append(9)
print("از بعد append:", my_list)>>> [3,1,4,1,5,9]
popped_item = my_list.pop()
print("آیتم حذف شده:", popped_item)>>>[3,1,4,1,5,9]
count_of_one = my_list.count(1)
print("تعداد عدد 1 در لیست:", count_of_one)>>>2
my_list.extend([2, 6, 5])
print("از بعد extend:", my_list)>>>[3,1,4,1,5,2,6,5]
my_list.insert(2, 7)
print("از بعد insert:", my_list)>>>[3,1,7,4,1,5,2,6,5]
my_list.remove(1)
print("از بعد remove:", my_list)>>>[3,7,4,1,5,2,6,5]
my_list.reverse()
print("از بعد reverse:", my_list)>>>[5,6,2,5,1,4,7,3]
my_list.sort()
print("از بعد sort:", my_list)>>>[1,2,3,4,5,6,7]
my_list.clear()
print("از بعد clear:", my_list)>>>[]

```

اکنون به تعریف از چند تابع و متدهم در لیست می پردازیم که در سایر ساختمنهای پایه داده نیز بکار می رود.

۱) مپ کردن : یک تابع درونی پایتون است که یک تابع را برای همه عناصر در یک تکرار معین اعمال می کند و نیاز به حلقه زدن نمی باشد.

۲) لامبда: یک تابع یکبار مصرف و ناشناس درونی پایتون است که برای ایجاد توابع کوچک آنها می توانند خوانایی و قابلیت نگهداری کد را با حذف نیاز به تعریف یک تابع جداگانه افزایش داد.

۳) فیلتر: مقدارهای مورد نیاز خودمان را از یک توالی قابل پیمایش صاف میکنیم.

۴) مرتب سازی: از این متدهم برای ترتیب بندی آبجکت های مورد نظر بدون تغییر اطلاعات اولیه استفاده می شود.

مثال:

```

numbers = [5, 12, 7, 18, 3, 21, 8, 15]
doubled_numbers = list(map(lambda x: x * 2, numbers))
print("اعداد ضربدر 2:", doubled_numbers) >>> [10, 24, 14, 36, 6, 42, 16, 30]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print("اعداد زوج:", even_numbers) >>> [12, 18, 8]
sorted_numbers = sorted(numbers, reverse=True)
print("مرتب شده به صورت نزولی:", sorted_numbers) >>> [21, 18, 15, 12, 8, 7, 5, 3]

```

## تاپل (tuple)

اگر مجموعه ای از عتاقر که تصادفی تغییر نکند همان لیستی است که قابل خواندن باشد.  
در این حالت ما از تاپل استفاده می کنیم.

Point = (1,2)

Print(type(point))

>>>class 'int'

اگر پرانتز را نیز برداریم همان تاپل است و اگر یک عضو داشته باشیم و بعد از آن کاما بگذاریم باز هم یک تاپل است.

تاپل ها می توان با هم جمع یا ضرب کرد

تاپل ها تغییر ناپذیرند.

## مقایسه تاپل ها

عملگر مقایسه همانگونه که برای بقیه توالیها کاربرد دارد، برای تاپل هم میتواند مورد استفاده قرار بگیرد. پایتون، مقایسه را با اولین المنت از هر توالی شروع میکند. اگر آنها مساوی بودند به سراغ المنت بعدی میروند و این کار تا زمانیکه المنتهای متفاوتی را پیدا کند، ادامه خواهد داشت. ولی به محض رسیدن به یک تفاوت، المنتهای بعد از آن، هرچقدر هم که بزرگ باشند، دیگر درنظر گرفته نخواهند شد:

فرض کنید که شما لیستی از کلمات دارید و میخواهید که آن را از بزرگتر به کوچکتر مرتب کنید:

```
txt = 'but soft what light in yonder window breaks' words = txt.split()
```

```
t = list()
for word in words: t.append((len(word), word))
t.sort(reverse=True) res = list()
for length, word in t: res.append(word)
print(res)
```

حلقه اول، یک لیست از تاپلها میسازد. هر تاپل دو جزء دارد. جزء اول، طول کلمه است. ولی کدام کلمه؟ کلمه‌ای که جزء دوم تاپل است. به عبارتی تاپل‌ما تشکیل شده است از: 1) طول کلمه و 2) خود کلمه. این بخش D از DSU است. شما یک کلید در ابتدای تاپل دارید و هدفتان مرتب کردن لیست با استفاده از این کلید مقدم است.

متند sort در مقایسه این تاپلها، ابتداء اولین المنش که همان طول کلمه باشد را مقایسه میکند و اگر طول دو کلمه مساوی بود، به سراغ مقایسه المنتهای دوم، که همان کلمه باشد، میرود. کلمه‌ی کلیدی در آرگیومنت آن یعنی reverse=True به sort میگوید که بایستی مقایسه حالت نزولی (از بالا به پایین) را داشته باشد. این بخش S از DSU به حساب می‌آید.

حلقه‌ی دوم در لیست تاپلها پیمایش کرده و لیستی از کلمات، به ترتیبی که طول آنها از زیاد به کم باشد، میسازد. در اینجا با دو کلمه‌ی چهار حرفی what و soft طرف هستیم. با توجه به یکسان بودن المنش اول - طول کلمه - تابع sort به سراغ مقایسه المنش دوم میرود. w از s بزرگتر است و با توجه به اینکه لیست ما از بزرگ به کوچک است، در نتیجه what قبل از soft می‌آید. ساخت یک لیست از المنتهای مرتب شده هم بخش U از DSU است.

خروجی برنامه شبیه به این خواهد بود:

```
['yonder', 'window', 'breaks', 'light', 'what', 'soft', 'but', 'in']
```

### (Set)

مجموعه‌ای از عناصر که در آن تکرار وجود ندارد.  
به مجموعه می‌توان عنصر اضافه یا حذف کرد.

مجموعه از عملیات ریاضی پیروی می‌کند. در زیر به چند مورد از آنها می‌پردازیم:

```
a = {1,3,5}
```

```
b = {2,4,6}
```

```
print(a|b) #union >>>{ 1,2,3,4,5,6 }
print(a&b) #intersection >>>{ }
print(a-b) #difrence >>>{ 1,3,5 }
```

### دیکشنری (dict)

یک دیکشنری چیزی شبیه به یک لیست - ولی کلی تر است. در یک لیست ماجایگاه های شاخص را داریم که با عدد صحیح مشخص میشوند؛ در دیکشنری، عناصر می توانند عدد رشته و... باشند. میتوانید دیکشنری را نقشه های بین دسته های از شاخصها (که به آنها کلید میگوییم) و دسته های از مقادیر در نظر بگیرید. هر کلید، نقشه یا راهی به یک مقدار دارد. ارتباط کلی و مقدار را جفت کلید-مقدار یا گاهی آیتم مینامیم.

اگر کلیدی در دیشکنری موجود نباشد، در تلاش برای چاپ آن با خطأ روبرو خواهد شد:

تابع len برای دیکشنریها نیز کاربردی است و تعداد جفت کلید-مقدار را برمیگرداند:

عملگر in در دیکشنریها هم مورد استفاده قرار میگیرد؛ in به شما میگوید که آیا کلید خاصی در دیشکنری وجود دارد یا خیر (به خاطر داشته باشید که in کلیدها را مرور میکند؛ اگر عبارتی جزو مقدارهای یک دیکشنری باشد، in توجهی به آن نمیکند):

اگر میخواهید که ببینید آیا مقدار خاصی در دیکشنری وجود دارد یا خیر، بایستی از متده استفاده کنید. این متده، مقادیر را به عنوان یک لیست برمیگرداند. در اینجا میتوانید از in برای یافتن مقدار خاصی در لیست ساخته شده بهره ببرید:

### دیکشنری به عنوان دسته ای از شمارندها

فرض کنید که یک رشته را به شما داده اند و از شما میخواهند که تعداد دفعاتی که هر کاراکتر در این رشته نمایان شده را حساب کنید. چندین راه برای این کار وجود دارد:

- ۱) میتوانید ۲۲ متغیر بسازید؛ هر کاراکتر یک متغیر. سپس در رشته پیمایش کرده و برای هر کاراکتر یک عدد به متغیر مورد نظر اضافه کنید. احتمالا با استفاده از شرط زنجیروار میتوانید لیستی با ۲۲ المنش درست کنید سپس هر کدام از کاراکترها را به یک شماره تغییر دهید (با استفاده از تابع توکاریشدهی (ord) و در نهایت با استفاده از شماره - به عنوان شاخص - شمارنده موردنظر را یک واحد افزایش دهید.

۲) میتوانید یک دیکشنری بسازید که کلیدهایش، کاراکترها و مقادیرش، شمارنده‌ی مربوط به هر کلید یا کاراکتر باشد. اولین باری که یک کاراکتر را میبینید، یک آیتم به دیکشنری اضافه میکنید. بعد از آن در مواجه مجدد با کاراکتر مورد نظر، مقدار آیتم موجود را یک واحد افزایش دهید.

هر کدام از این گزینه‌ها، پاسخ یکسانی به سوال ما میدهد ولی پیاده سازی روش رسیدن به پاسخ در هر یک متفاوت است.

پیاده سازی، راهی برای اجرای یک محاسبه است؛ برخی پیاده سازی‌ها از بقیه بهترند. به عنوان مثال، مزیت پیاده سازی به روش سوم و با استفاده از دیکشنری این است که ما نیازی به دانستن حروفی که در رشته ظاهر میشوند نداریم بلکه خود برنامه جای خالی برای آنها دارد. به عبارتی زمانیکه برنامه به هر کدام از حروف برسورد کند، جای خالی برای آنها ساخته و در درون دیکشنری قرارشان میدهد؛ بدون اینکه زحمت تعیین آنها را از قبل کشیده باشیم.

## حلقه‌زدن و دیکشنری‌ها

اگر از یک دیکشنری به عنوان توالی در یک گزاره‌ی `for` استفاده میکنید، گزاره‌ی `for` در بین کلیدهای دیکشنری پیمایش خواهد کرد. برای درک بهتر موضوع به مثال زیر نگاهی بیندازید؛ حلقه‌ی زیر کلیدها و مقدار مرتبط با آنها را یکی چاپ میکند

```
counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
```

```
for key in counts:
```

```
    print(key,counts[key])
```

```
>>> jan 100
```

```
chuck 1
```

```
annie 42
```

## متدهای دیکشنری

```
dict.clear()          ممه عناصر را پاک می کند
dict.copy()          کپی سطحی از دیکشنری ایجاد کرده و برگردانده
dict.get(key)        مقدار مربوط به هر کلید مشخص شده از دیکشنری را برگردانده
dict.items()         لیستی از عناصر دیکشنری را به صورت تابل حاوی جمله‌های کلید و مقدار برگردانده
dict.pop()           عنصر مربوط به کلید مشخص شده را از دیکشنری حذف و برگردانده
dict.values()        لیستی از مقادیر درون دیکشنری را برگردانده
dict.update()        دو دیکشنری را به هم متصل می کند
```

در ادامه یک مثال کامل از تمام نکاتی که درباره دیکشنری زدیم می پردازیم:

```
students = {
```

```
'Ali': {'math': 17, 'physics': 14, 'chemistry': 18},  
'Sara': {'math': 12, 'physics': 19, 'chemistry': 15},  
'Reza': {'math': 19, 'physics': 18, 'chemistry': 17},  
'Mina': {'math': 8, 'physics': 9, 'chemistry': 12},  
'Hossein': {'math': 14, 'physics': 13, 'chemistry': 16},  
'Narges': {'math': 20, 'physics': 20, 'chemistry': 20},
```

```
}
```

```
assign_title = lambda avg: (  
    "Excellent" if avg >= 19 else  
    "Good" if avg >= 17 else  
    "Average")
```

```
# Full functional pipeline  
final_result = sorted(  
    map(  
        lambda item: {  
            "name": item[0],  
            "average": round(sum(item[1].values()) / len(item[1]), 2),  
            "title": assign_title(sum(item[1].values()) / len(item[1]))  
        },  
        filter(  
            lambda item: all(score > 15 for score in item[1].values()),  
            students.items()  
        )
```

```

),
key=lambda x: x["average"],
reverse=True
)

top_student = reduce(
    lambda acc, item: item if acc["average"] < item["average"] else acc,
    final_result
)

print("All Students Sorted:")
print(*final_result, sep="\n")

print("\nTop Student:")
print(top_student)

خروجی کد به صورت زیر خواهد بود:
All student sorted:
{'name': 'narges' , 'average':20.00 , 'title': 'Excellent'}
{'name': 'Reza' , 'average': 18.00 , 'title': good}

Top student:
{'name': 'narges' , 'average':20.00 , 'title:'Excellent}

```

در این مثال ما کل فرایند را بدون حلقه ها و دستورات شرطی اجرا کردیم.

به کمک فیلتر ما ضعیف ها را حذف کردیم با مپ معدل گیری و لقب به دانش آموزان دادیم با سورت مرتب سازی نزولی انجام دادیم و با ردیوس بهترین را پیدا کردیم.

## ۷-۲- کلاس‌ها و ماثول‌ها

### استفاده از آبجکت‌ها

پایتون آبجکتهای توکاری شده فراوانی را فراهم کرده است. در اینجا کد ساده‌های را می‌بینید. چند خط اول بایستی برای شما خیلی ساده و قابل درک باشد.

```
stuff = list()
stuff.append('python')
stuff.append('chuck')
stuff.sort()
print (stuff[0])
print (stuff.__getitem__(0))
print (list.__getitem__(stuff,0))
```

ولی به جای تمرکز روی کارهایی که این خطوط انجام میدهند، بباید ببینیم که از منظیر نامه نویسی شیگرا، زیر پوست این برنامه چه خبر است. اگر منظور من از پاراگراف‌های بعدی را نمی‌فهمید، نگران نباشید چرا که هنوز است بسیاری از اصطلاحات مرتبط با آن برایتان قابل‌فهم نشده است.

خط اول یک آبجکت با نوع «لیست» را «می‌سازد». خط دوم و سوم متدهای append() را فراخوانی کرده و در خط چهارم متدهای sort() احضار می‌شود. خط پنجم مقدار آیتم را در مکان 0 دریافت و چاپ می‌کند. در خط ششم متدهای \_\_getitem\_\_() در لیست stuff با یک پارامتر «صفر» فراخوانی می‌شود.

با استفاده از خروجی تابع dir() می‌توانیم قابلیتهای یک آبجکت را نظاره کنیم:

```
>>> stuff = list()
>>> dir(stuff)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__',
```

```
'__repr__', '__reversed__', '__rmul__', '__setattr__',
['__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count',
'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
```

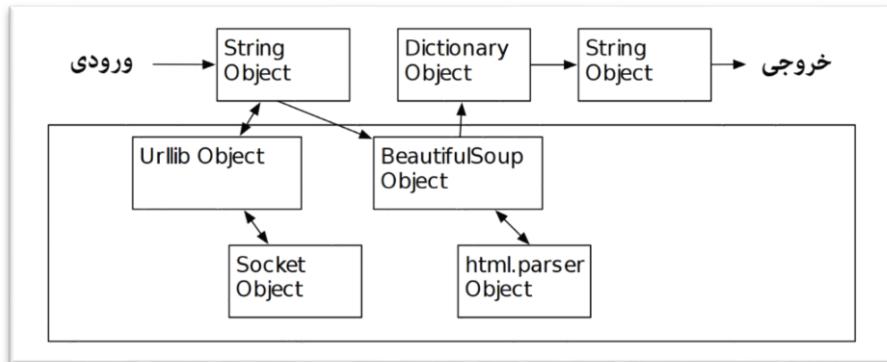
### شروع با برنامه‌ها

در پایه ای ترین حالت، یک برنامه ورودی می‌گیرد، پردازش انجام میدهد و خروجی تولید می‌کند. برنامه «تبدیل طبقه» یک برنامه‌ی کوتاه ولی کامل را در سه مرحله به ما نشان میدهد



زمانیکه داخل یک برنامه ایم تعاملات تعیین شده ای با دنیای خارج خواهیم داشت، اما این تعاملات به خوبی برنامه ریزی شده اند و نیازی به تمرکز روی آنها نیست. در طرف مقابل، زمانیکه در حال نوشتن کد هستیم، عکس این مطلب صدق میکند و ما به جزئیات درون برنامه‌دققت میکنیم. برای نگاه بهتر به برنامه نویسی شیگرا آن را به صورت نقاطی در نظر بگیرید که برنامه‌ی ما را تجزیه و جدا میکند. هر نقطه حاوی مقداری کد و داده میشود (درست شبیه به یک برنامه) و برای تعامل با دنیای بیرون و سایر نقاط به خوبی آماده شده است.

کلید اصلی، فهمیدن کامل این برنامه‌ها و نحوه‌ی کارکردشان نیست، بلکه ساخت یک شبکه از تعاملات بین آبجکتها و تنظیم جریان اطلاعات بین آنها برای ساخت یک برنامه است. شما با نمونه‌ی این برنامه در چند فصل پیش هم آشنا شدید. در آنجا میفهمیدید که برنامه چه کاری را انجام میدهد؛ بدون اینکه نیازی به درک هماهنگی حرکات داده‌ها بین آبجکتها داشته باشید. در آنجا خطوط کد را می‌دیدید که کارشان را انجام میدهند.



### اولین آبجکت پایتونی ما

تعريف آبجکت در ساده ترین حالت عبارت است از: مقداری کد به علاوه‌ی ساختارهای داده که کوچکتر از یک برنامه‌ی کامل است. تعريف کردن یک تابع به ما اجازه میداد تا بسته‌هایی با نام مشخص درست کنیم، سپس مقداری کد درون آن قرار دهیم و هر زمان که لازم بود با فراخوانی نام آن بسته، تابع را احضار کنیم.

یک آبجکت میتواند شامل داده و تعدادی تابع شود (که به آنها متدهای میگوییم). داده‌ها توسط تابع‌های داخل آبجکت مورد استفاده قرار می‌گیرند. ما به آیتمهای مرتبط با داده‌ها در آبجکت صفات یا attributes میگوییم. صفات، در برنامه نویسی به مشخصه‌هایی که به تعريف خواص اشیاء میپردازد گفته می‌شود.

با استفاده از کلیدواژه class داده‌ها و کدی را، که هر کدام آبجکتها را میسازند، تعريف می‌کنیم. این کلیدواژه تشکیل شده است از نام کلاس شروع یک قطعه یا بلاک از کد که تورفته است و شامل صفات (داده‌ها) و متدها (کد) می‌شود.

```

class PartyAnimal:
    x = 0
    def party(self):
        self.x = self.x + 1
        print("So far",self.x)
an = PartyAnimal()
an.party()
an.party()
an.party()
PartyAnimal.party(an)

```

هر متدهایی به یک تابع است و با کلیدواژه `def` آغاز میشود و در نهایت شامل قطعه ای تورفته از کدهاست. در این مثال ما یک صفت (`x`) و یک متدهای (`draw`) داشتیم. متدها یک پارامتر اولیه `x` را دارند که طبق عرف آن را `self` می نامیم. شبیه به کلیدواژه `def` که باعث نمیشود کد موجود در تابع اجرا شود، کلیدواژه `class` یک آبجکت را نمیسازد. در عوض کلیدواژه `class` یک قالبی درست میکند تا نشان دهد چه داده ها و کدی در هر آبجکت با نوع `PartyAnimal` قرار خواهد گرفت.

### Constructor

به دادن آبجکت های اولیه به کلاس می گویند.

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def draw(self):
        print(f'point{self.x},{self.y}')
point=point(1,2)
point.draw()
>>>point(1,2)
```

### Class vs instance attributes

```
class Point:
    default_color = 'blue'
        self.x = x
        self.y = y
    def draw (self):
        print(f'point({self.x},{self.y})')
point=point(1,2)
point.draw()
print(point.default_color)
>>>point(1,2)
blue
```

نکته) این ها اtribut و اینسنس هستند. ما می توانیم کلاس های اtribut را تعریف کنیم که برای تمام آبجکت ها یکسان است. همچنین می توانیم همه به وسیله کلاس و هم آبجکت

بگوییم اگر به وسیله کلاس بگوییم همه کلاس تغییر می کند ولی بوسیله آبجکت فقط آبجکت تغییر می کند.

### مقایسه آبجکت‌ها

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __eq__(self,other):
        return self.x==other.x and self.y==other.y
Point=Point(1,2)
another=Point(1,2)
print(Point==another)
>>>True
```

در اینجا ما دو آبجکتی که در ابتدا آدرس هایشان یکی نیست به طبع برابر هم نیستند را برابر هم قرار دادیم.

### عملگرهای محاسباتی روی آبجکت

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __str__(self):
        return(f'{self.x},{self.y}')
    def __add__(self,other):
        return Point(self.x+ other.x+ self.y+ other.y)
point = Point(1,2)
another=Point(3,4)
print(point+another)
>>>(4,6)
```

### ایجاد ساختمان داده شخصی

در مثال زیر یکسری کلمات را می خواهیم به ساختمان داده های شخصی بدهیم تا انعطاف کدهایمان بیشتر شود.

```
class Bagofworld:  
    def __init__(self):  
        self.words = {}  
    def add(self,word):  
        self.words[word]=self.words.get(word,0)+1  
    def __getitem__(self,word):  
        return self.words.get(word,0)  
    def __setitem__(self,word,count):  
        self.words[word]= count  
    def __len__(self):  
        return len(self.words)  
    def __iter__(self.words):  
        return iter(self.words)  
  
document = Bagofworld()  
document.add('python')  
document['python'] = 20  
print(document['python'])  
>>>20
```

در مثال فوق ما ابتدا کلمه به دیکشنری اضافه کردیم سپس از روی {} مقدار را خواندیم و بعد آنرا صدای زدیم و در آخر طول دیکشنری را پیمایش کردیم.

### property

```
class Circle:  
    def __init__(self, radius):  
        self._radius = radius  
  
    @property  
    def radius(self):  
        return self._radius  
  
    @radius.setter  
    def radius(self, value):  
        if value < 0:  
            raise ValueError("Radius cannot be negative")  
        self._radius = value  
  
    @property  
    def area(self):  
        from math import pi  
        return pi * (self._radius ** 2)  
c = Circle(5)  
print(c.radius) # 5  
print(c.area) # 78.53981633974483  
c.radius = 10  
print(c.area) # 314.1592653589793
```

پر اپرتی روی اتریبیوت می ایستد و مقدار آن را گت یا ست می کند.

## وراثت-Inheritance

می توانیم یک کلاس از کلاس دیگر زیر مجموعه ایجاد کنیم که متدها و اتریبیوت های کلاس فوق را به ارث ببرد.

```

class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return "Some sound"

class Dog(Animal):
    def speak(self):
        return "Woof!"

class Cat(Animal):
    def speak(self):
        return "Meow!"

a = Animal("Generic Animal")
d = Dog("Buddy")
c = Cat("Whiskers")
print(a.name, "says", a.speak())
print(d.name, "says", d.speak())
print(c.name, "says", c.speak())

```

باز تعریف متدها

```

class Vehicle:
    def __init__(self, brand):
        self.brand = brand

    def start(self):
        print(f"{self.brand} vehicle is starting...")

class Car(Vehicle):
    def __init__(self, brand, model):
        super().__init__(brand)
        self.model = model

    def start(self):
        super().start()
        print(f"{self.brand} {self.model} is ready to drive!")

v = Vehicle("GenericBrand")
v.start()
c = Car("Toyota", "Corolla")
c.start()

```

## کلاس انتظاعی-Abstract

```

class Person:
    def __init__(self, name, age):
        self.name = name
        self._age = age
    @property
    def age(self):
        return self._age
    @age.setter
    def age(self, value):
        if value < 0:
            raise ValueError("Age cannot be negative")
        self._age = value

    def introduce(self):
        print(f"My name is {self.name}, and I am {self.age} years old.")


class PoliceOfficer(Person):
    def __init__(self, name, age, badge_number):
        super().__init__(name, age)
        self.badge_number = badge_number

    def introduce(self):
        super().introduce()
        print(f"I am a police officer. My badge number is {self.badge_number}.")

    def arrest(self, suspect_name):
        print(f"Officer {self.name} is arresting {suspect_name}.")
officer = PoliceOfficer("John Doe", 35, "A1234")
officer.introduce()
officer.arrest("Mike Smith")

```

در مثال فوق دو کلاس داریم که متدهای خودشان را دارند اما هر دو از کلاس پرسون ارث بری می کنند.

## چند ریختی-Polymorphism

چند ریختی یعنی اشیا از کلاس های مختلف می توانند متدهای نام یکسان داشته باشند ولی رفتار متفاوتی را پیاده سازی کنند. پایتون بصورت داینامیک این موضوع را پشتیبانی می کند.

```
class Animal:  
    def speak(self):  
        return "Some generic animal sound"  
class Dog(Animal):  
    def speak(self):  
        return "Woof!"  
class Cat(Animal):  
    def speak(self):  
        return "Meow!"  
class Bird(Animal):  
    def speak(self):  
        return "Tweet!"  
  
def make_animal_speak(animal):  
    print(animal.speak())  
  
dog = Dog()  
cat = Cat()  
bird = Bird()  
  
make_animal_speak(dog)  
make_animal_speak(cat)  
make_animal_speak(bird)
```

ارث بری از کلاس های درونی پایتون

یادآوری: لیست توپل ست و دیکشنری کلاس های درونی پایتون هستند.

```

class CustomList(list):
    def __init__(self, *args):
        super().__init__(args)

    def __str__(self):
        return f"CustomList({super().__str__()})"

    def sum(self):
        return sum(self)

    def append(self, item):
        print(f"Adding {item} to the list.")
        super().append(item)

cl = CustomList(1, 2, 3)
print(cl)

cl.append(4)
print(cl.sum())

cl.append(5)
print(cl.sum())

```

در مثال فوق نشان دادیم کلاس کاستوم از لیست ارث بری کرده و متده اپند بازنویسی شده و قبل از افزودن چاپ می شود. متده جدیدی به نام سام اضافه شده است.

## ماژول ها

ماژول ها فایل هایی هستند که یکسری متغیر پایتون در آن وجود دارد.

```
def calc_text():
```

```
    pass
```

```
def calc_interest():
```

```
    pass
```

برای اینکه بخواهیم داده های یک فایل را به فایل دیگر ببریم ابتدا یک فایل جدید ایجاد کرده و سپس درونریزی می کنیم.

```
Import parsa
```

```
Parsa.calc_text(),calc_interest
```

در اینجا فایل پارسا به صورت آبجکت عمل می کند.

## Path

برای مسیر و مسیریابی به وسیله شی گرایانه در پایتون استفاده می شود. در مثال پایین دستورات مربوطه به مسیر یابی را آورده ایم:

```
from pathlib import path
path = path('فایل/فولدر')

print(path.exists())      # آیا این فایل وجود دارد؟
print(path.isfile())     # آیا یک فایل است؟
print(path.isdir())       # آیا یک پوشه است؟
print(path.name())        # اسم فایل را برمی گرداند
print(path.stem())        # برترمی گرداند py.پسوند#
print(path.suffix())      # بررمی گرداند .py.پسوند#
print(path.parent())       # فولدر پدر را به ما می دهد#
print(path.absolute())     # آدرس ابسولوت فایل را به ما می دهد#
```

```
path2 = path.with_name('file.txt')
print(path2)
```

## کار کردن با فرمت Json (java script object notation)

در پایتون یک ساختار متنی برای تبادل داده هاست که شبیه دیکشنری است. برای کار کردن با آن از مارژول جیسون استفاده می کنیم.

```

import json
person = {
    "name": "Alice",
    "age": 30,
    "is_employee": True,
    "skills": ["Python", "Django"]
}
json_string = json.dumps(person)
print(json_string)

python_dict = json.loads(json_string)
print(python_dict["name"])

```

نحوه خواندن و ذخیره کردن در جیسون نیز به شکل زیر می باشد:

```

# ذخیره به فایل
with open("person.json", "w") as f:
    json.dump(person, f)

# خواندن از فایل
with open("person.json", "r") as f:
    data = json.load(f)
    print(data["skills"])

```

## ماژول های زمان

در پایتون چند ماژول پر کاربرد برای تاریخ و زمان وجود دارد:

۱) کار با زمان های لحظه ای و تاخیر: time

۲) کار با تاریخ و زمان دقیق: datetime

۳) برای مناطق زمانی Zoneinfo

## مثال ۱

```
from datetime import datetime, timedelta

# زمان فعلی
now = datetime.now()
print("Now:", now)

# فرماتدهی به زمان
print("Formatted:", now.strftime("%Y-%m-%d %H:%M:%S"))

# ساخت یک زمان خاص
birthday = datetime(2000, 5, 15, 8, 30)
print("Birthday:", birthday)

# محاسبه تفاوت بین دو زمان
delta = now - birthday
print("Days since birthday:", delta.days)
print("Seconds since birthday:", delta.total_seconds())

# افزودن زمان
next_week = now + timedelta(days=7)
print("Next week:", next_week)

# مقایسه تاریخها
if now > birthday:
    print("You were born before now.")
```

مثال (۲)

```
from datetime import datetime
from zoneinfo import ZoneInfo
# زمان فعلی به وقت UTC
utc_time = datetime.now(tz=ZoneInfo("UTC"))
print("UTC Time:", utc_time)

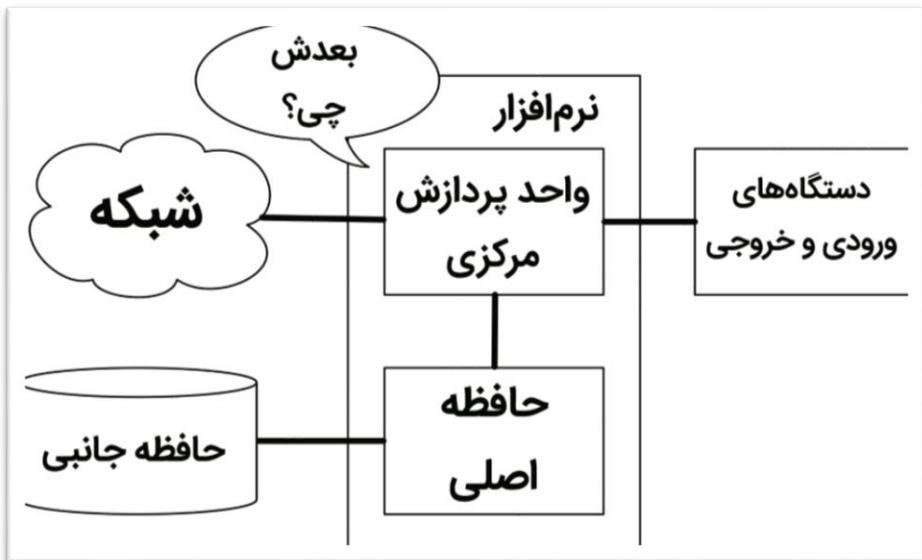
# تبدیل به منطقه زمانی دیگر
tehran_time = utc_time.astimezone(ZoneInfo("Asia/Tehran"))
newyork_time = utc_time.astimezone(ZoneInfo("America/New_York"))

print("Tehran Time:", tehran_time)
print("New York Time:", newyork_time)
```

## ۲-۸- کار با فایل‌ها و خطاهای مانندگاری

تا اینجا در خصوص نحوه نوشتن برنامه‌ها و ارتباط برقرار کردن با واحد پردازش مرکزی با استفاده از اجرای شرطی، توابع، و تکرارها آشنا شدیم. در خصوص چگونگی‌ساخت و استفاده از ساختار داده در حافظه‌ی اصلی نیز آموختیم. CPU و حافظه جایی‌اند که نرم افزار در آنها کار می‌کند و اجرا می‌شود. در اصل این منطقه همانجا بایست که "فکر کردن". رخ می‌دهد.

ولی مشکل اینجاست که می‌خواهیم بحث میان سخت افزارهای ما ادامه داشته باشد، اما همین که برق برود و سیستم خاموش شود، تمام آنچه در CPU و حافظه اصلی بوده پاک خواهد شد. برنامه‌های پایتونی ما تلاشهای تفریحی ناپایداری خواهند بود که از بین می‌روند.



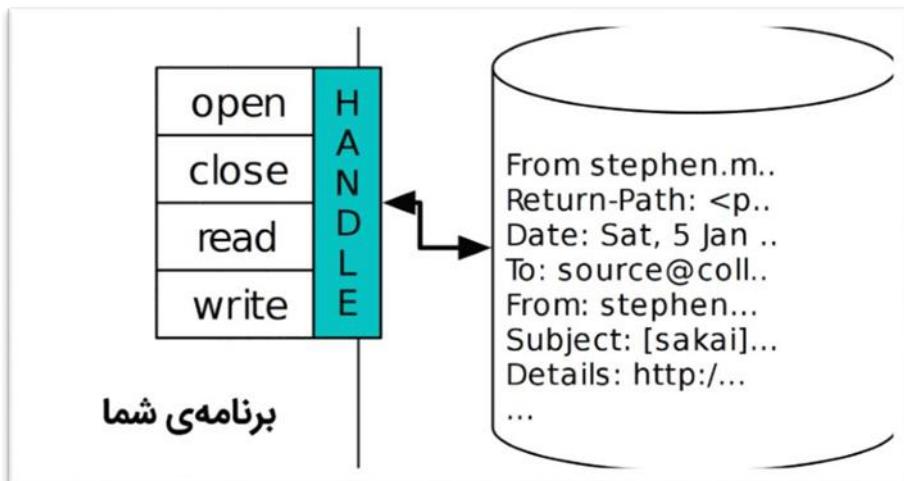
در این بخش با حافظه‌ی جانبی (یا فایلها) سر و کار داریم. اطلاعات در حافظه‌ی جانبی زمانیکه سیستم خاموش میشود، از بین نمیرود؛ یا مثلا برنامهای که نوشته ایم میتواند روی یک حافظه‌ی فلاش کپی شده، از سیستم جدا و سپس به یک سیستم دیگر منتقل و اجرا شود.

### باز کردن فایل‌ها

وقتی قرار است که فایلی را بخوانیم یا روی آن بنویسیم (مثلا فایلهای روی هارد دیسک شما)، ابتدا باید فایل را باز کنیم. باز کردن فایل، برقراری ارتباط با سیستم عامل شما به حساب میآید. چرا سیستم عامل؟ چون سیستم عامل است که میداند هر فایل‌در کجا ذخیره شده است. زمانیکه شما یک فایل را باز میکنید، از سیستم عامل میپرسید که آن فایل را از طریق نامش پیدا کرده و مطمئن شود که فایل اصلا وجود دارد. در این مثال، ما فایل mbox.t که در همان فolder جاری است را باز میکنیم؛ منظورمان از فolder جاری، همان مسیریست که پایتون در آن اجرا شده است.

```
>>> fhand = open('mbox.txt')
>>> print(fhand)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='cp1252'>
```

اگر عملیات open موفقیت آمیز باشد، سیستم عامل یک file handle یا «دستگیره ی فایل» باز میگرداند. دستگیره ی فایل، خود داده ای موجود در فایل نیست، اما در عوض یک «دستگیره» است که ما میتوانیم با استفاده از آن، داده ها را بخوانیم. اگر فایل درخواست شده وجود خارجی داشته باشد، شما یک دستگیره خواهید داشت و اگر مجوزهای لازم را در اختیار داشته باشید، فایل خوانده خواهد شد.



اگر فایل وجود خارجی نداشته باشد، open با شکست روبرو شده و شما را به یک تریسپک مهمان خواهد کرد. در نهایت شما دستگیره، برای دسترسی به محتویات فایل، نخواهید داشت

>>> fhand = open('stuff.txt') Traceback (most recent call last):

File "<stdin>", line 1, in <module> FileNotFoundError: [Errno 2] No such file or directory: 'stuff.txt'

### فایل‌های متنی و خطوط

یک فایل متنی را میتوان سلسله‌ای از خطوط درنظر گرفت. درست مثل سلسله ای از کاراکترها که در پایتون یک رشته خوانده میشود. نوشته زیر، یک فایل متنیست که فعالیتهای مرتبط با ایمیل را از افراد متفاوت جمع آوری میکند.

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008 Return-Path: <postmaster@collab.sakaiproject.org>

Date: Sat, 5 Jan 2008 09:12:18 -0500

To: source@collab.sakaiproject.org From: stephen.marquard@uct.ac.za

### خواندن فایل ها

ما از دستگیره‌ی فایل به عنوان یک توالی در حلقه‌ی for استفاده کرده‌ایم. حلقه‌ی for در مثال بالا، تعداد خطوط موجود در فایل را شمرده و نتیجه را چاپ می‌کند. ترجمه‌ی تحت الفظی حلقه‌ی for در به زبان آدمیزاد می‌شود: «برای هر خط در فایل که توسط دستگیره‌ی فایل ارائه شده است - یک عدد به متغیر count اضافه کن.» دلیل اینکه تابع open تمام فایل را نمی‌خواند این است که احتمال اینکه فایل بسیار بزرگ باشد و حاوی گیگابایت‌ها دیتا شود، وجود دارد. به این صورت گزاره

برای باز کردن هر فایل - از کوچک به بزرگ - مقدار یکسانی زمان نیاز دارد. گزاره‌ای که داده‌ی اصلی را از فایل می‌خواند، حلقه‌ی for است. زمانیکه ما از حلقه‌ی for به این صورت استفاده می‌کنیم، پایتون وظیفه‌ی بخش بخش کردن داده‌های داخل فایل را به خطوط جداگانه، با استفاده از کاراکتر خط جدید برعهده می‌گیرد. پایتون هر خط را از طریق خط جدید خوانده و کاراکتر خط جدید را به عنوان آخرین کاراکتر متغیر line برای هر تکرار حلقه‌ی for به حساب می‌آورد.

به این خاطر که حلقه‌ی for داده‌ها را به صورت یک خط در یک زمان می‌خواند، به راحتی می‌تواند که خطوط بیشمار در فایلهای بزرگ را خوانده و بشمارد، بدون اینکه حافظه‌ی اصلی تحت فشار حجم زیاد فایل قرار بگیرد. برنامه‌ی بالا می‌تواند که خطوط را در هر حجمی با استفاده‌ی حداقلی از حافظه‌ی اصلی بخواند چرا که برنامه‌ی هر خط را می‌خواند، می‌شمارد و سپس داده‌های آن را دور میریزد و به سراغ خط بعدی میرود. به عبارت ساده‌تر تنها برای همان خط در جریان اجرا، نیاز به حافظه‌ی اصلی وجود دارد و داده‌های قبلی، حافظه را اشغال نخواهند کرد. اما اگر میدانید فایلی که قرار است پردازش شود، نسبت به حجم حافظه اصلی، بسیار کوچک است، می‌توانید تمام فایل را با استفاده از متدهای read و دستگیره‌ی فایل در داخل یک متغیر قرار دهید. به این صورت برای تمام فایل روی حافظه اصلی فضا اشغال می‌شود:

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp)) 94626
```

```
>>> print(inp[:20]) From stephen.marquar
```

زمانیکه فایلی به این صورت خوانده میشود، تمام کاراکترهای خطوط و کاراکترهای خط جدید در یک رشته‌ی بزرگ قرار خواهند گرفت. در مثال‌ما، این رشته‌ی بزرگ در متغیر inp ذخیره میشود. در حالت تعاملی پایتون میتوانید با صادر کردن گزاره‌ی inp تمام کاراکترهای این رشته‌ی بزرگ را چاپ کنید. فراموش نکنید که این روش باستی تنها روی فایلهایی اجرا شود که مطمئن باشید با قرارگیری آنها روی حافظه‌ی اصلی مشکلی پیش نمی‌آید و به راحتی روی آن جا خواهند شد.

اگر فایل نسبت به حافظه‌ی اصلی خیلی بزرگ باشد، باستی با استفاده از حلقه‌های for یا while برنامه‌تان را طوری بنویسید که فایل را به قطعه‌های کوچکتر تقسیم کند.

### جستجو در یک فایل

وقتی که در بین داده‌های یک فایل شروع به جستجو میکنید، با احتمال زیاد با استفاده از یک الگوی خاص به دنبال خطوط ویژه‌ای برای پردازش خواهید گشت؛ خطوطی که شرایط به خصوصی را داشته باشند. همچنین میتوانید چند الگو را برای خواندن یک فایل با هم ترکیب کنید. برای این کار از متدهای رشته به منظور ساخت مکانیسم‌های ساده‌ای برای جستجو استفاده خواهیم کرد.

به عنوان مثال اگر ما بخواهیم که یک فایل را خوانده و سپس تنها خطوطی که با From: شروع شده‌اند را چاپ کنیم، میتوانیم از متدهای startswith برای پیدا کردن آن خطوط

بهره ببریم.

```
fhand = open('mbox-short.txt')
count = 0
for line in fhand:
    if line.startswith('From:'):
        print(line)

>>>From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
```

خروجی، عالی به نظر میرسد چرا که تمام خطوطی که با From: شروع شده اند در آن چاپ شده است. تنها مشکل وجود خطوط خالی بین آنهاست. این مشکل به خاطر کاراکتر نامرئی خط جدید/newline به وجود آمده است. دلیلش این است که هر خط با یک کاراکتر خط جدید تمام میشود، در نتیجه آن کاراکتر هم چاپ خواهد شد؛ و باعث میشود که در کنار چاپ جداگانه‌ی خطوط، یک خط خالی بین آنها ایجاد شود.

### انتخاب نام فایل را به کاربر بسپارید

اگر قرار باشد که هر بار برای پردازش یک فایل متفاوت، کد برنامه را عوض کنیم، اوضاع جالب نخواهد بود. بهتر است که از کاربر اسم فایلی که قرار است پردازش شود، پرسیده شده و سپس عملیات روی آن فایل صورت پذیرد. به این صورت کد ما میتواند روی فایلهای مختلف، بدون ایجاد تغییر در برنامه‌ی اصلی، عملیات انجام دهد.

این کار ساده است و از طرق خواندن فایل با اینبوت انجام می‌شود:

```
fname = input('Enter the file name: ')
fhand = open(fname)
count = 0
for line in fhand:
    if line.startswith('Subject:'):
        count = count + 1
print('There were', count, 'subject lines in', fname)
```

ما اسم را از کاربر می‌گیریم و سپس با استفاده از متغیری که در اختیار داریم در فایل باز می‌کنیم.

```
python search6.py
Enter the file name: mbox.txt
There were 1797 subject lines in mbox.txt
python search6.py
Enter the file name: mbox-short.txt
There were 27 subject lines in mbox-short.txt
```

## استفاده از try & except & open

اگر کاربر نامی را وارد برنامه کند که وجود خارجی نداشته باشد چه اتفاقی میافتد؟

```
python search6.py
Enter the file name: missing.txt Traceback (most recent call last):
File "search6.py", line 2, in <module> fhand = open(fname)
FileNotFoundException: [Errno 2] No such file or directory: 'missing.txt'
python search6.py
Enter the file name: na na boo boo Traceback (most recent call last):
File "search6.py", line 2, in <module> fhand = open(fname)
FileNotFoundException: [Errno 2] No such file or directory: 'na na boo boo'
```

واقعاً ممکن است کاربر کارهایی را انجام دهد که به نظر شما احتمالش وجودندارد. کارهایی که برنامه را با مشکل مواجه کند. حالا شاید با قصد، حتی قصد خرابکاری، این کار را انجام دهد. به عنوان یک اصل، بخشی از هر تیم توسعهٔ نرم افزار، افراد یا گروهی هستند که QA یا گروه «اطمینان از کیفیت» خوانده میشوند. کار این گروه یا شخص انجام چیزهای، حتی مسخره، برای از کار انداختن برنامه است. به عبارتی آنها به هر چیزی متول میشوند که برنامه با مشکل مواجه شود و کارش را درست انجام ندهد. تیم QA مسئول پیدا کردن عیبهای یک برنامه قبل از انتشار آن و رسیدن به دست کاربرنهایی است. کاربرنهایی هم همان کسی است که قرار است برنامه را بخرد یا حقوق ما را بپردازد. در نتیجه تیم QA بهترین رفیق یک برنامه نویس است.

حالا که مشکل در برنامه را کشف کردیم، بایستی که آن را به روش مناسبی با استفاده از try یا except فرض کنید که باز کردن فایل با open با مشکل مواجه شود با به دلیل ناهمانگی اسم فایل با فایلهای موجود در آن مسیر.

```

fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    exit()
count = 0
for line in fhand:
    if line.startswith('Subject:'):
        count = count + 1
print('There were', count, 'subject lines in', fname)

```

حفظات از فراخوانی `open` در برنامه، مثال خوبی از استفاده‌ی `try` و `except` در یک برنامه‌ی پایتونی است. ما عبارت Pythonic یا پایتونی را زمانیکه یک کاری را به روش پایتون انجام میدهیم استفاده میکنیم. به همین خاطر به مثال بالا «یک راه پایتونی برای باز کردن یک فایل «میگوییم. زمانیکه شما مهارت بیشتری در پایتون کسب کردید و با برنامه نویسی‌های دیگر بر سر راههای حل یک مساله صحبت کردید، میتوانید بگویید که این راه حل پایتونی تر است یا خیر. ولی چرا پایتونیتر باشد؟ به این خاطر که بخشی از برنامه‌نویسی هنر است و بخشی مهندسی؛ به عبارتی یک راه حل پایتونیتر راه حل مهندسی-هنری تری برای یک مساله است. قرار نیست که فقط برنامه‌ما کار کند، بلکه میخواهیم که راه حلمان زیبا و تحسین برانگیز باشد.

## نوشتن فایل‌ها

برای نوشتن یک فایل، بایستی که فایل را در مود `w` به عنوان پارامتر دوم باز کنید. بهتر است مثال زیر را برای درک بهتر ببینید:

```

>>> fout = open('output.txt', 'w')
>>> print(fout)
<_io.TextIOWrapper name='output.txt' mode='w' encoding='cp1252'>

```

اگر فایل از پیش موجود باشد، باز کردن فایل در مود `w` داده‌های پیشین را پاک کرده و از نو شروع میکند. به همین خاطر لازم است که مراقب باشید. اگر فایل از پیش موجود نباشد، فایل جدیدی ساخته خواهد شد.

متدهای write از آبجکت دستگیره‌ی فایل، داده‌ها را درون فایل قرار داده و تعداد کاراکترهای نوشته شده را برمی‌گرداند. حالت پیشفرض نوشتن، متن برای نوشتن (و خواندن) رشته‌ها است.

آبجکت فایل، جایی که قرار دارد را رهگیری می‌کند، بنابراین زمانیکه write را دو مرتبه فرآخوانی کنیم، داده‌های جدید به انتهای فایل اضافه می‌شوند.

با استی مطمئن باشیم که انتهای خطوط را به درستی مدیریت می‌کنیم. برای اینکار لازم است از کاراکتر خط‌جدید در پایان هر خط استفاده کنیم. گزاره‌ی print به صورت خودکار خطوط جدید را الحاق می‌کند ولی متدهای write این کار را انجام نمیدهد.

```
>>> line2 = 'the emblem of our land.\n'
```

```
>>> fout.write(line2) 24
```

زمانیکه کارتان تمام شد، فایل را بیندید و مطمئن شوید که آخرین بیت از داده‌ها روی دیسک به صورت فیزیکی نوشته شده است و با خاموش شدن سیستم از بین نخواهد رفت.

```
>>> fout.close()
```

همچنین ما می‌توانیم که فایلی را که برای خواندن باز کرده‌ایم، ببندیم ولی در این مورد (خواندن) اگر کمی سهل انگاری کنیم و تنها چند فایل را باز کرده باشیم مشکلی به وجود نمی‌آید. در اصل پایتون از بسته شدن فایلهای باز قبل از پایان برنامه مطمئن می‌شود. ولی وقتی در حال نوشتن فایلهای هستیم، باستی که صراحتاً به پایتون اعلام کنیم که فایل را بیند و چیزی را به دست تقدیر و شانس نسباریم.

## ۹-۲- معرفی چند کتابخانه مهم پایتون

1) Os: این کتابخانه این امکان را به ما می‌دهد که از طریق کد با سیستم عامل در ارتباط باشیم.

ابتدا یک فolder در دسکتاپ می‌سازیم سپس دو فolder در داخلش می‌سازیم و در فolder فایل ۱ سه فolder و یک فایل متنی دیگر می‌سازیم.

در زیر دستورات این کتابخانه را مشاهده می‌کنید.

```
import os

os.listdir() #همه فولدرهایی را که در مسیر داریم لیست می‌کند
os.listdir('c:/') #مسیر داده شده را لیست می‌کند
os.name #نوع سیستم عامل را نشان می‌کند
os.mkdir('file2') #فولدر جدیدی با نام file2 ایجاد می‌کند
os.rename('file2','file3') #فولدر را با نام file3 نامگذارد
os.chdir('../') #یک فولدر پایین‌تر را به عنوان مسیر معرفی می‌کند
os.rmdir('file3') #فولدر را حذف می‌کند
os.remove('example.txt') #فایل را حذف می‌کند
os.stat('smp.txt') #اطلاعات فایل را نشان می‌کند
```

برای شروع به نوشتن در ویرایشگر vim باید `s` را فشار داده تا به حالت INSERT بروید. پس از تکمیل فایل اسکریپت، باید کلید `esc` را فشار دهید و `:wq` را برای نوشتن تغییرات و خروج از ویرایشگر vim تایپ کنید.

دسترسی:

سه مدل دسترسی داریم:

۱- خواندن

۲- اجرا

۳- نوشتن

- 0 # عدم دسترسی
- 1 # اجرا
- 2 # فقط نوشتن
- 3 # نوشتن و اجرا
- 4 # فقط خواندن
- 5 # خواندن و اجرا
- 6 # خواندن و نوشتن
- 7 # خواندن و نوشتن و اجرا

یک مثال کامل از این کتابخانه می‌زنیم:

```

import os

os.system('fsutil volume diskfree c:> sutil.txt')
with open ('f sutil.txt','r') as file:
    content = file.read()
    print(content)
run_cmd('f sutil volume disk free c:')
with open('f sutil.txt','r') as file:
    content = file.read()
    print(content)
output_stream = os.open ('ping 8.8.8.8')
output = output_stream.read()
output_stream.close()
print(output)

```

این کتابخانه این امکان را به ما می دهد که با اج تی تی پی کار کنیم.  
Request(2)

```
import request as rq
```

```

r= rq.get('https://www.python.org/')
print(dir())

```

تمام آبجکت ها و متدها را نشان می دهد#  
 >>>  
 r.url # آدرس سایت را به ما می دهد  
 r.txt # کد را نشان می دهد  
 r.help # هر چیز که داخلش است را توضیح می دهد  
 r.content # ب اء، نمایش عکس

Numpy(3)

این کتابخانه مربوط به آرایه ها می باشد. آرایه ها از نظر سرعت بالاتر از لیست ها هستند.  
 آرایه ها عکس لیست ها عناصرشان هم نوع هستند و مانند لیست ها قایل پیمایش و دسترسی  
 و تغییر هستند.

```
import numpy as np

# Suppose these are daily page views for a week
page_views = np.array([120, 150, 130, 170, 160, 180, 200])

# Calculate basic statistics
average_views = np.mean(page_views)
std_views = np.std(page_views)
max_views = np.max(page_views)
min_views = np.min(page_views)

print("Average daily views:", average_views)
print("Standard deviation:", std_views)
print("Max views:", max_views)
print("Min views:", min_views)
```



# فصل سوم: آشنایی با خط فرمان لینوکس

## Bash

## ۱-۳ - تعریف bash

اسکریپت نویسی ابزار قدرتمندی برای خودکارسازی وظایف در یک سیستم لینوکس فراهم می‌کند. از کاربرد دستورات خروج از shell گرفته تا کنترل جریان یک اسکریپت با عبارات-if، elseif-else، اسکریپت‌های bash به شما این امکان را می‌دهند که از قدرت خط فرمان برای انجام عملیات پیچیده استفاده کنید.

با استفاده از مجوزهای فایل file permissions می‌توانید مطمئن شد که فقط یک کاربر معتبر می‌تواند به اطلاعات حساس دسترسی داشته باشد. با دستور bash read حتی می‌توان ورودی‌های کاربر را در میان یک اسکریپت جمع آوری نمود..

علاوه بر این، توانایی ذخیره مقادیر به عنوان متغیرهای رشته و دستکاری مقادیر رشته به اسکریپت bash سطحی از تطبیق پذیری قابل مقایسه با سایر زبان‌های برنامه نویسی می‌دهد. به دنبال خودکار کردن کارهای ساده یا نوشتن اسکریپت‌های پیچیده باشید، اسکریپت bash یک مهارت ضروری برای هر مدیر سرور لینوکس است. فایلی که حاوی مجموعه‌ای از دستورات اجرایی باشد، اسکریپت bash نامیده می‌شود که این فایل‌ها پسوند .sh دارند. یک نسخه بهبود یافته از bash است. این برنامه sh پیش فرض برای اکثر سیستم‌های شبه‌یونیکس است.

سایر Bash یک برنامه shell، یک مفسر خط فرمان bash و یک زبان برنامه نویسی با امکانات کامل برای نوشتن اسکریپت‌های شل است. در ادامه نحوه نوشتن اسکریپت‌های shell با استفاده از سینتکس bash را یاد خواهیم گرفت.

## ۲-۳ - اسکریپت نویسی

ابتدا یک شل اسکریپت در ترمینال ایجاد می‌کنیم. می‌توانید از دستور استفاده کنید. تمامی فایل‌های ذخیره شده در شل با پسوند touch script.sh خاتمه می‌یابد. در ابتدای هر اسکریپت (هنگام نوشتن اسکریپت) باید مشخص کنیم که با چه نوع شلی اجرا شود. خط اول اسکریپت به (هنگام نوشتن اسکریپت) باید مشخص کنیم که با چه نوع شلی اجرا شود. خط اول shebang line معروف است. در ادامه Bourne line برای شل‌های Bourne Again.Shell ..sh آمده است..sh و C shell

-SH – #!/bin/sh

-BASH – #!/bin/bash

-ZSH – #!/bin/zsh

فایل اسکریپت را باز کرده و ویرایش کنید

می توانید از هر ویرایشگر متنی برای باز کردن و ویرایش فایل اسکریپتی که ایجاد کردید استفاده کنید. برای مثال، می توانید از nano script.sh یا vim script.sh برای باز کردن فایل با از ویرایشگر vs code استفاده کنید.

در این مطلب نحوه نصب و استفاده از ویرایشگر متن nano به عنوان ویرایشگر متن محبوب در سیستم عامل لینوکس را بررسی می کنیم. ویرایشگر نانو دارای ویژگی های قدرتمند بسیاری است و به کاربر امکان ویرایش و ایجاد فایل های مختلف را در کامپیوتر یا سرور می دهد را بنویسیم. ابتدا باید خط فرمان

که به سیستم عامل می گوید از کدام برنامه فرمان شل برای اجرای خطوط فرمان زیر استفاده کند. سپس می توانیم اولین خط اسکریپت خود را برای خروجی پیامی مانند این بنویسیم:

```
echo "Hello world!"
```

حال باید مجوز ها را به فایل اسکریپت مان اضافه کنیم.

فایل را اجرا کنید. bash script.sh یا script.sh می توانید با دستور

### ۳-۳- مثال های اسکریپت

برای درک بهتر موضوع چند مثال برایتان آورده ام که گام به گام پیش می رویم  
مثال اول) استفاده از متغیر ها در اسکریپت

تعریف کنیم. name & place در این مثال می خوایهم متغیرمان را با دو نام

```
#!/bin/bash
```

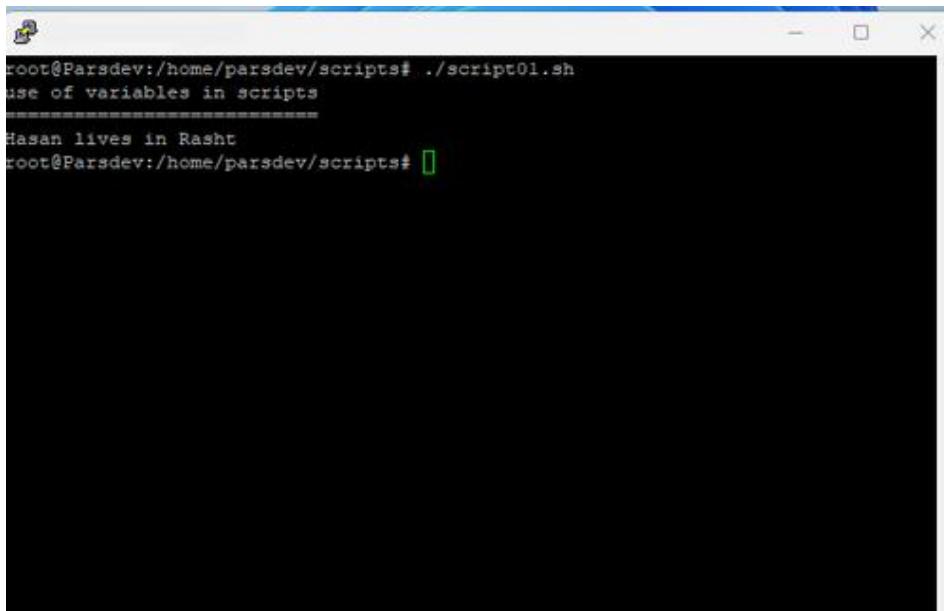
```
echo "use of variables in scripts"
```

```
echo "=====
```

```
name="Hasan"
```

```
place="Rasht"
```

```
echo "$name lives in $place"
```



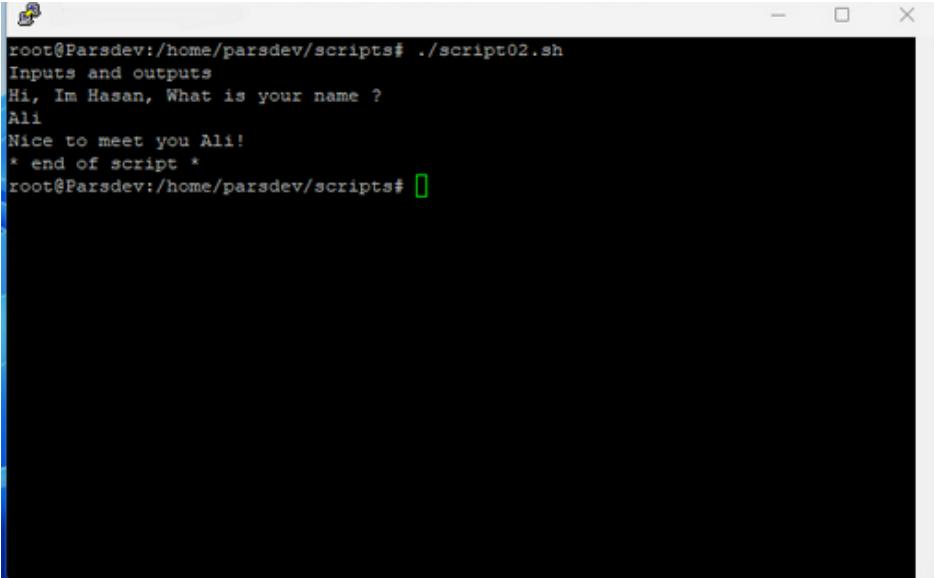
```
root@Parsdev:/home/parsdev/scripts# ./script01.sh
use of variables in scripts
=====
Hasan lives in Rasht
root@Parsdev:/home/parsdev/scripts#
```

### مثال دوم) ورودی ها و خروجی ها

می دانیم که می توان جریان های رشته را به ترمینال خروجی داد. علاوه بر این، می توان ورودی را هم از ترمینال دریافت کرد

از دستور echo برای خروجی و از دستور read برای ورودی گرفتن از کاربر استفاده کرد.

```
#!/bin/bash
echo "Inputs and outputs"
echo "Hi, Im Hasan, What is your name ?"
read name
echo "Nice to meet you $name!"
echo "* end of script *"
```



```
root@Parsdev:/home/parsdev/scripts# ./script02.sh
Inputs and outputs
Hi, Im Hasan, What is your name ?
Ali
Nice to meet you Ali!
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

مثال سوم) عبارات شرطی if

از این دستور برای موجود بودن فایل استفاده می کنیم.

```
#!/bin/bash

echo if then else statements

echo

file=parsdev.txt

if [ -e $file ]

then

echo "Yes, File exists!"

else

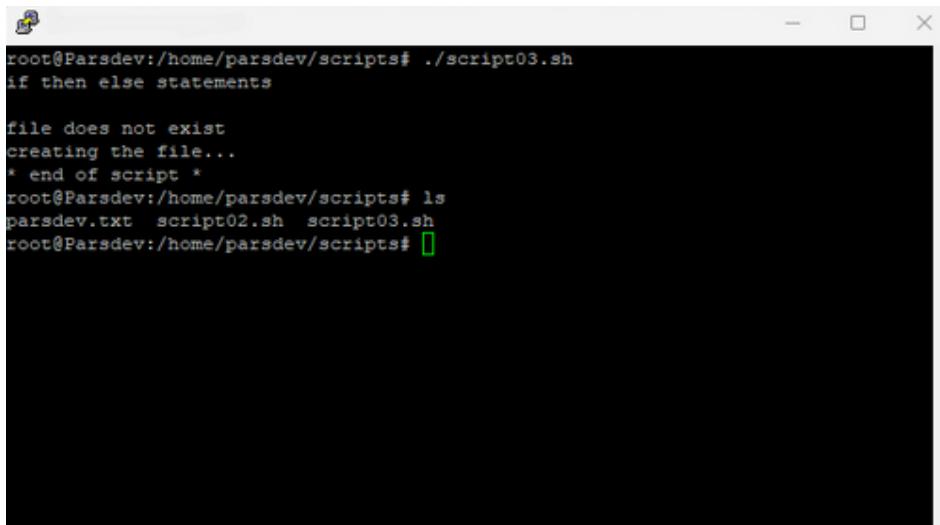
echo "file does not exist"

echo "creating the file..".

touch $file

fi

echo "* end of script *"
```



```
root@Parsdev:/home/parsdev/scripts# ./script03.sh
if then else statements

file does not exist
creating the file...
* end of script *
root@Parsdev:/home/parsdev/scripts# ls
parsdev.txt  script02.sh  script03.sh
root@Parsdev:/home/parsdev/scripts#
```

#### مثال چهارم) ساختار Case

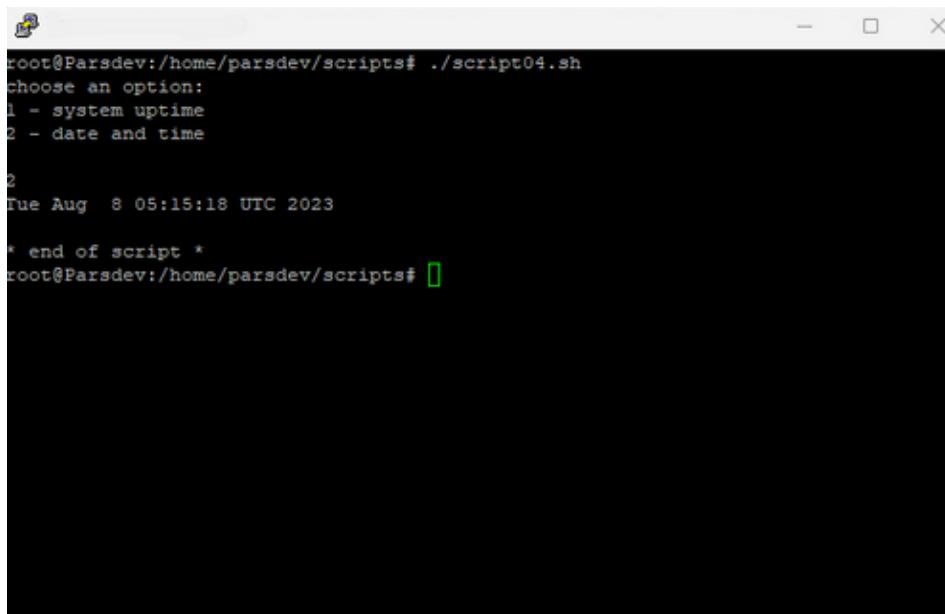
در Option های کاربر و ارائه شناسه ها هستیم و برای هر گزینه یک دستور انتخاب می کنیم  
این مثال در حال

```
#!/bin/bash

echo "choose an option:"
echo "1 - system uptime"
echo "2 - date and time"
echo
read choice

case $choice in
  1) uptime;;
  2) date;;
  *) invalid argument
esac

echo
echo "* end of script *"
```



```
root@Parsdev:/home/parsdev/scripts# ./script04.sh
choose an option:
1 - system uptime
2 - date and time

2
Tue Aug  8 05:15:18 UTC 2023

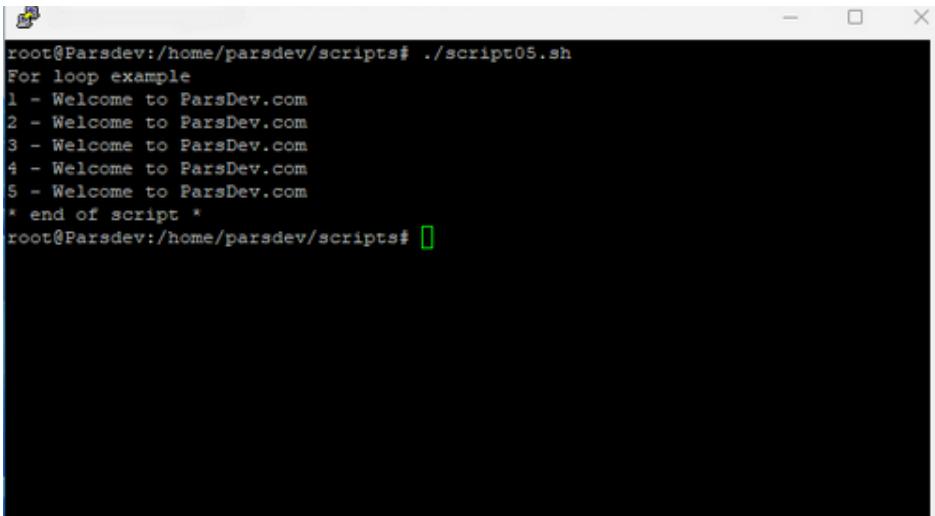
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

### مثال پنجم) حلقه For

از این دستور برای خروجی یک رشته معین با تعداد مشخص استفاده می کنیم.

tasksel install desktop gnome-desktop

```
#!/bin/bash
echo "For loop example"
for i in 1 2 3 4 5
do
echo "$i - Welcome to ParsDev.com"
done
echo "* end of script *"
```

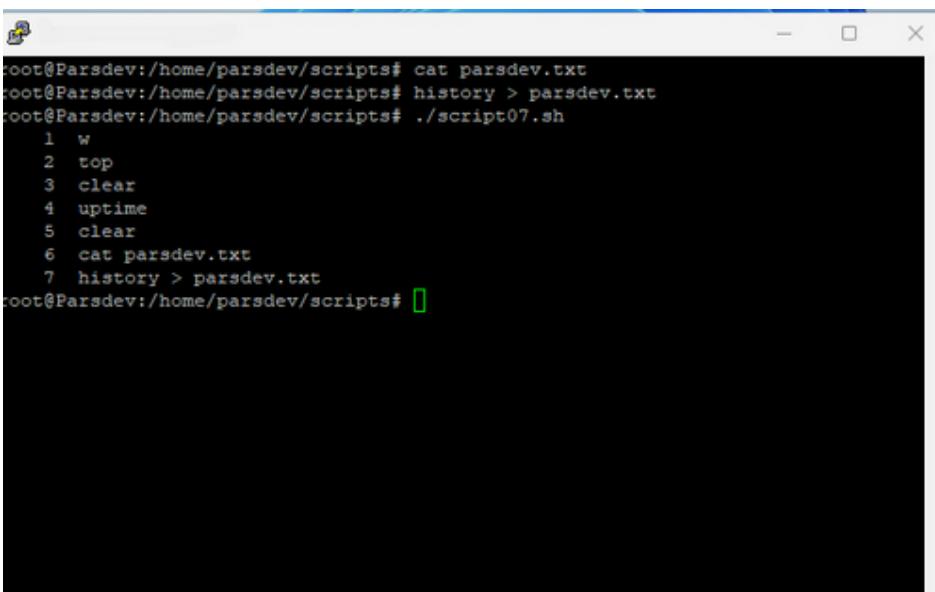
A screenshot of a terminal window titled 'root@Parsdev'. The command 'root@Parsdev:/home/parsdev/scripts# ./script05.sh' is run, displaying the output of a for loop that prints 'Welcome to ParsDev.com' five times, followed by an asterisk and the text 'end of script'.

```
root@Parsdev:/home/parsdev/scripts# ./script05.sh
For loop example
1 - Welcome to ParsDev.com
2 - Welcome to ParsDev.com
3 - Welcome to ParsDev.com
4 - Welcome to ParsDev.com
5 - Welcome to ParsDev.com
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

مثال ششم) دسترسی به داده های یک فایل

برای خواندن فایل ها از دستور Cat استفاده می کنیم.

```
#!/bin/bash
cat abc.txt
```

A screenshot of a terminal window titled 'root@Parsdev'. The user runs 'cat parsdev.txt' to check its contents, then 'history > parsdev.txt' to save the history to a file, and finally './script07.sh' which outputs the history from the file. The history shows various commands like w, top, clear, uptime, and cat parsdev.txt.

```
root@Parsdev:/home/parsdev/scripts# cat parsdev.txt
root@Parsdev:/home/parsdev/scripts# history > parsdev.txt
root@Parsdev:/home/parsdev/scripts# ./script07.sh
  1  w
  2  top
  3  clear
  4  uptime
  5  clear
  6  cat parsdev.txt
  7  history > parsdev.txt
root@Parsdev:/home/parsdev/scripts#
```

## مثال هفتم) بررسی اتصال سرور ریموت

```
#!/bin/bash
echo "Enter an ip address to ping test :"
read ip
echo "you gave - $ip"
echo "pinging &gt; +++++++"
ping -c2 $ip
if [ $? -eq 0 ]
then
echo
echo "====="
echo "Ping success!"
else
echo "-----"
echo "Ping not success!, try again.."
fi
echo
echo "* end of script *"
```

```
root@Parsdev:/home/parsdev/scripts# ./script08.sh
Enter an ip address to ping test :
8.8.8.8
you gave - 8.8.8.8
pinging > ++++++++
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=115 time=1.49 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=115 time=1.60 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.486/1.544/1.602/0.058 ms

=====
Ping success!

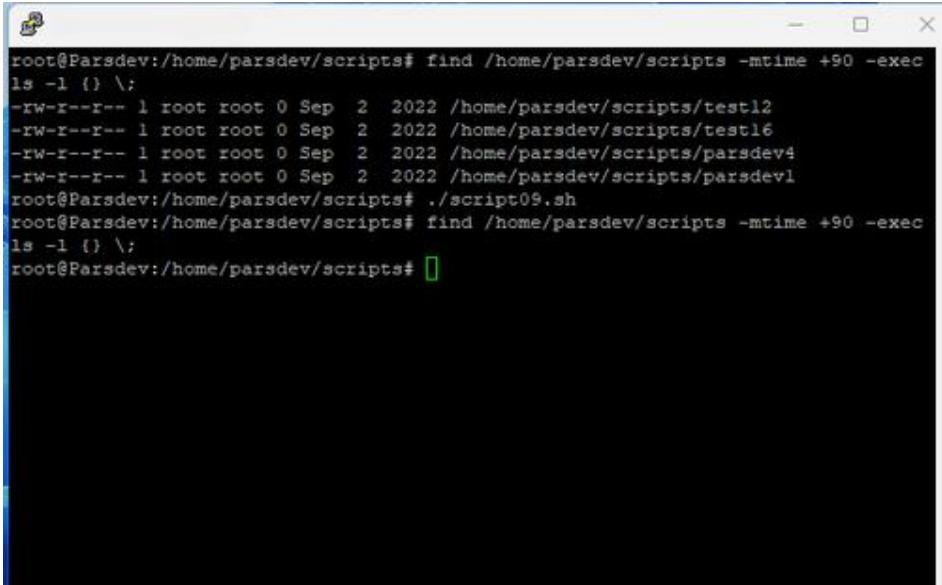
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

## مثال هشتم) حذف فایل‌های قدیمی

می‌توان یک اسکریپت برای حذف فایل‌های قدیمی نوشت، این کار را با تعیین قدمت یک اسکریپت و حذف آن از یک مکان خاص انجام. فرض کنید می‌خواهیم فایل‌های قدیمی‌تر از ۹۰ روز را حذف کنیم. اگر برای تست فایل‌های قدیمی‌تری ندارید، می‌توان آن‌ها را با دستور زیر ایجاد کرد.

```
touch -d "Tue, 21 March 2025 00:54:28" file1
</pre>
find /home/parsdev/scripts -mtime +90 -exec ls -l {} \;
```

```
#!/bin/bash
find /home/parsdev/scripts/ -mtime +90 -exec rm {} \;
</pre>
```



The screenshot shows a terminal window with the following command history:

```
root@Parsev:/home/parsdev/scripts# find /home/parsdev/scripts -mtime +90 -exec ls -l {} \;
-rw-r--r-- 1 root root 0 Sep  2 2022 /home/parsdev/scripts/test12
-rw-r--r-- 1 root root 0 Sep  2 2022 /home/parsdev/scripts/test16
-rw-r--r-- 1 root root 0 Sep  2 2022 /home/parsdev/scripts/parsdev4
-rw-r--r-- 1 root root 0 Sep  2 2022 /home/parsdev/scripts/parsdevl
root@Parsev:/home/parsdev/scripts# ./script09.sh
root@Parsev:/home/parsdev/scripts# find /home/parsdev/scripts -mtime +90 -exec ls -l {} \;
root@Parsev:/home/parsdev/scripts#
```

## مثال نهم) بکاپ فایل سیستم

با استفاده از tar یک نسخه بکاپ از دایرکتوری می‌گیریم.

```
#!/bin/bash
```

```
tar cvf /home/parsdev/scripts/scripts-backup.tar /home/parsdev/scripts/backup
gzip scripts-backup.tar
echo "process finished!"
```

The screenshot shows a terminal window with the following command history:

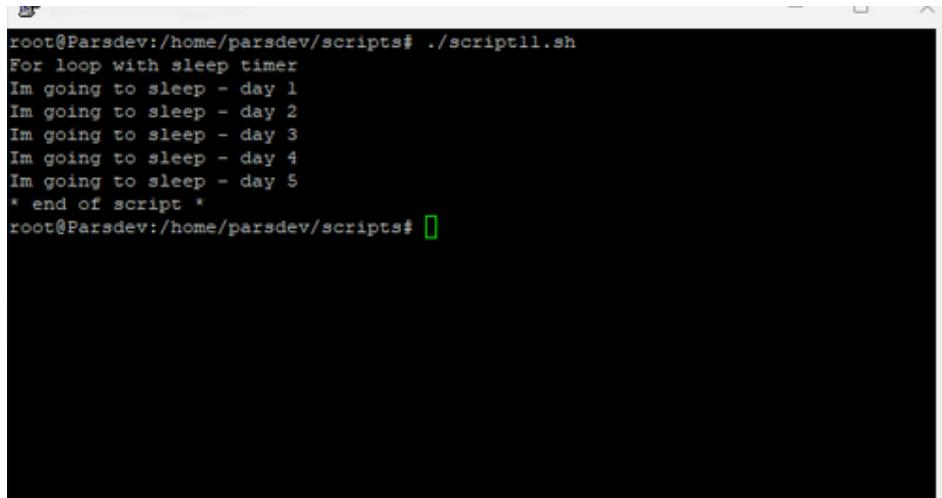
```
root@Parsev:/home/Parsev/scripts$ ls
backup      script02.sh  script04.sh  script06.sh  script08.sh  script10.sh
Parsev.txt  script03.sh  script05.sh  script07.sh  script09.sh
root@Parsev:/home/Parsev/scripts$ ./script10.sh
tar: Removing leading '/' from member names
/home/Parsev/scripts/backup/
process finished!
root@Parsev:/home/Parsev/scripts$ ls
backup      script03.sh  script06.sh  script09.sh
Parsev.txt  script04.sh  script07.sh  script10.sh
script02.sh  script05.sh  script08.sh  scripts-backup.tar.gz
root@Parsev:/home/Parsev/scripts$
```

The file `scripts-backup.tar.gz` is highlighted with a red box.

### Sleep (دهم)

از این دستور برای حالت خواب استفاده می کنیم. می توانیم زمان و نحوه عملکرد آن را نیز تعیین می کنیم.

```
#!/bin/bash
echo "For loop with sleep timer"
for i in 1 2 3 4 5
do
echo "Im going to sleep - day $i" && sleep 20
done
echo "* end of script *"
```

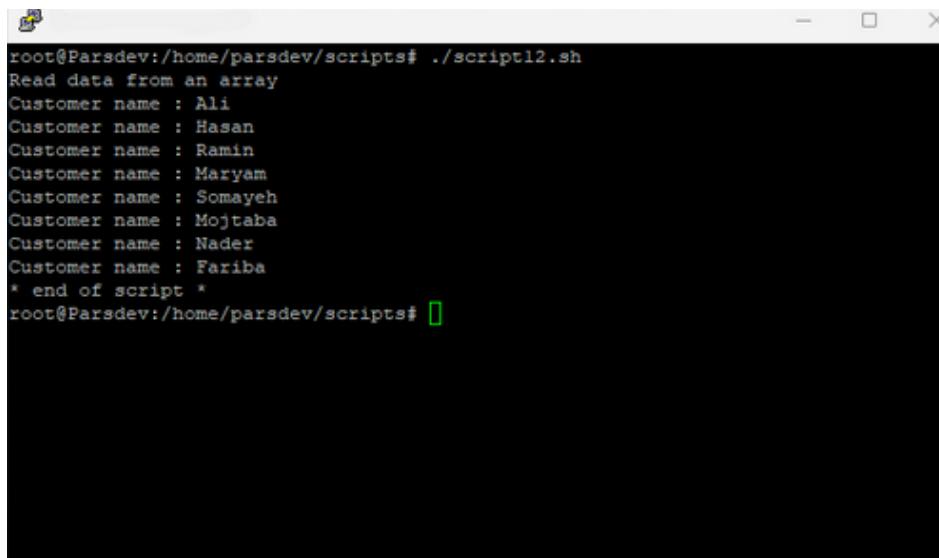


```
root@Parsdev:/home/parsdev/scripts# ./script11.sh
For loop with sleep timer
Im going to sleep - day 1
Im going to sleep - day 2
Im going to sleep - day 3
Im going to sleep - day 4
Im going to sleep - day 5
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

### مثال یازدهم) آرایه

از آزایه برای ذخیره اطلاعات مانند شاخص به روش های مختلف استفاده می کنیم. این آرایه ها برای ذخیره عناصر مختلف مانند رشته ها و اعداد خوب هستند. در این مثال از آرایه ای استفاده می کنیم که شامل مشتریان است و آنها را یک به یک با استفاده از حلقه می پردازیم.

```
#!/bin/bash
customers=(Ali Hasan Ramin Maryam Somayeh Mojtaba Nader Fariba)
echo "Read data from an array"
for i in "${customers[@]}"
do
echo "Customer name : $i"
done
echo "* end of script *"
```

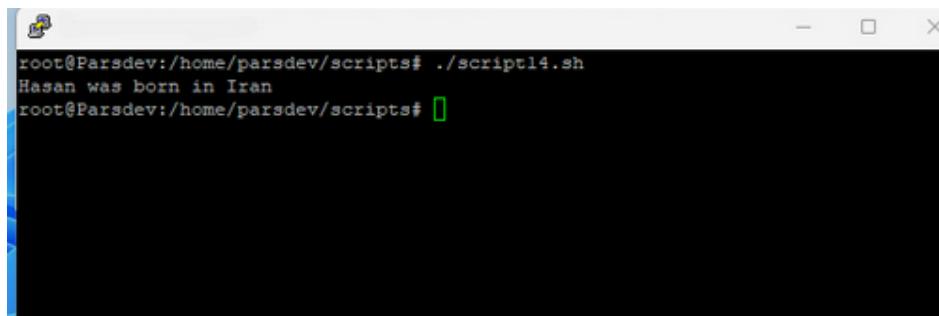


```
root@Parsdev:/home/parsdev/scripts# ./script12.sh
Read data from an array
Customer name : Ali
Customer name : Hasan
Customer name : Ramin
Customer name : Maryam
Customer name : Somayeh
Customer name : Mojtaba
Customer name : Nader
Customer name : Fariba
* end of script *
root@Parsdev:/home/parsdev/scripts#
```

### مثال دوازدهم) متصل کردن رشته‌ها

می‌توان یک رشته را به رشته دیگری که به عنوان رشته پیوسته شناخته می‌شود، اضافه نمود. ما از یک متغیر چهارم برای به هم پیوستن سه متغیری که رشته‌ها را ذخیره می‌کنند استفاده می‌نمائیم.

```
#!/bin/bash
beginning="Hasan "
middle="was born in"
end=" Iran"
apendedvalue="$beginning$middle$end"
echo "$apendedvalue"
```



```
root@Parsdev:/home/parsdev/scripts# ./script14.sh
Hasan was born in Iran
root@Parsdev:/home/parsdev/scripts#
```

## مثال سیزدهم) شمارش تعداد فایل ها

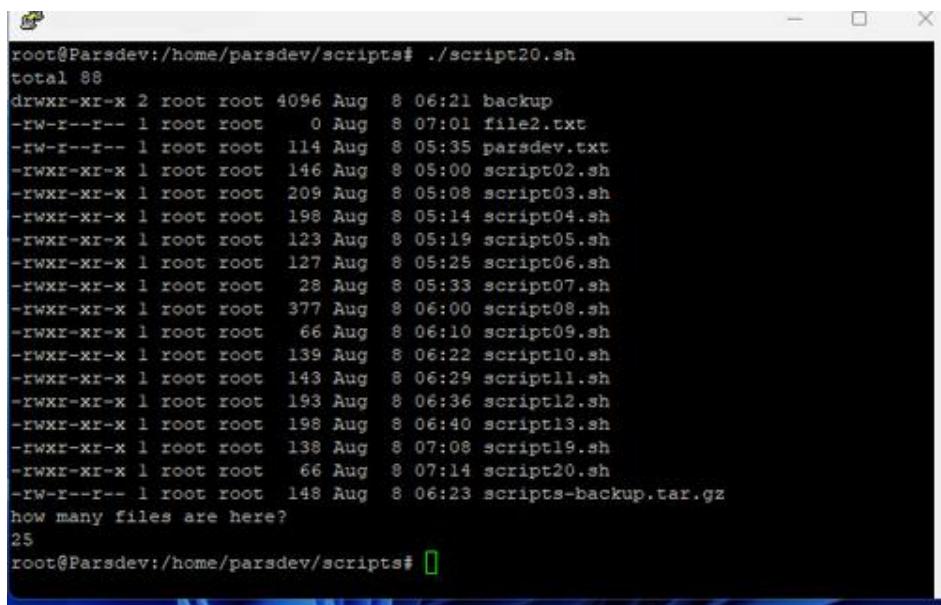
توانایی شمارش تعداد فایل های داخل یک دایرکتوری را نشان می دهد. این یک کار مهم برای هر کاربر لینوکس است، زیرا امکان می دهد محتويات یک دایرکتوری را نظارت و اطمینان حاصل کنید که فقط یک کاربر معتبر به فایل های داخل آن دسترسی دارد.

```
#!/bin/bash
```

```
ls -l
```

```
echo "how many files are here?"
```

```
ls -l * | wc -l
```



```
root@Parsdev:/home/parsdev/scripts# ./script20.sh
total 88
drwxr-xr-x 2 root root 4096 Aug  8 06:21 backup
-rw-r--r-- 1 root root     0 Aug  8 07:01 file2.txt
-rw-r--r-- 1 root root   114 Aug  8 05:35 parsdev.txt
-rwxr-xr-x 1 root root   146 Aug  8 05:00 script02.sh
-rwxr-xr-x 1 root root   209 Aug  8 05:08 script03.sh
-rwxr-xr-x 1 root root   198 Aug  8 05:14 script04.sh
-rwxr-xr-x 1 root root   123 Aug  8 05:19 script05.sh
-rwxr-xr-x 1 root root   127 Aug  8 05:25 script06.sh
-rwxr-xr-x 1 root root    28 Aug  8 05:33 script07.sh
-rwxr-xr-x 1 root root   377 Aug  8 06:00 script08.sh
-rwxr-xr-x 1 root root    66 Aug  8 06:10 script09.sh
-rwxr-xr-x 1 root root   139 Aug  8 06:22 script10.sh
-rwxr-xr-x 1 root root   143 Aug  8 06:29 script11.sh
-rwxr-xr-x 1 root root   193 Aug  8 06:36 script12.sh
-rwxr-xr-x 1 root root   198 Aug  8 06:40 script13.sh
-rwxr-xr-x 1 root root   138 Aug  8 07:08 script19.sh
-rwxr-xr-x 1 root root    66 Aug  8 07:14 script20.sh
-rw-r--r-- 1 root root   148 Aug  8 06:23 scripts-backup.tar.gz
how many files are here?
25
root@Parsdev:/home/parsdev/scripts#
```

## مثال چهاردهم) عملگرهای محاسباتی

عملگرهای جمع و ضرب تقسیم و تفریق مانند هم هستند:

```
#!/bin/bash
echo $ ((10+20))
*./script.sh
>>>30
```

lt-	کوچکتر
-le	کوچکتر مساوی
-ge	بزرگتر مساوی
-gt	بزرگتر
-eq	مساوی
-ne	نامساوی



## فصل چهارم: آشنایی با پایگاه‌های داده

## تعريف

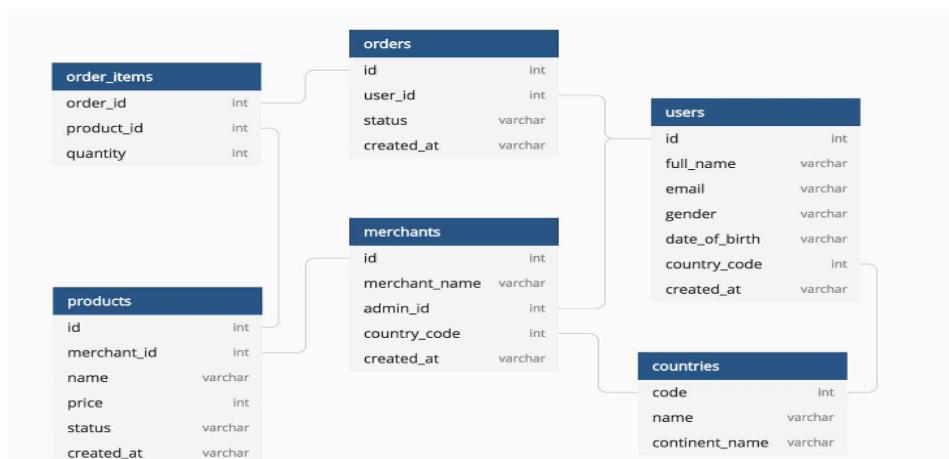
به مجموعه ای از داده ها که با ترتیب و فرمت خاص مدیریت شود دیتابیس می گویند. در این فصل ما به سه مدل از پرکاربردترین دیتابیس ها به صورت پروژه محور می پردازیم.

## My sql - ۴-۱

برای برنامه نویسی تحت وب که برای برنامه های کوچک و بزرگ بکار می رود یک دیتابیس رابطه ای



حال می خواهم یک شما میل کلی از جدول های دیتابیس در تصویر زیر به شما بدهم



برای دانلود به سایت Mysql.com مراجعه کنید.

و نسخه ی Mysql community server را دانلود کنید.

## محیط نرم افزاری

Mysql ----> schmas--> لیست دیتابیس ها را نشان می دهد

برای ایجاد جدول در دیتابیس به صورت زیر عمل می کنیم :

بر روی گزینه table کلیک راست کرده و اطلاعات زیر را وارد می کنیم

```
table name : users
column names: id - first_name - last_name - gender - email
Data type : PK - NN - AI
```

بر روی گزینه Apply کلیک می کنیم و جدول مان ایجاد می شود.

دو دایره سمت چپ جدول یکی اطلاعات جدول و دیگری امکان ویرایش را به ما می دهد.  
مربع کنار دایره ها مقادیر جدول را انتخاب می کند.

بر روی گزینه sql + دستورات به صورت پیش فرض انجام می شود.

حالا باید دستور Select را بزنیم:

Select \* from users

حالا گزینه Excute را میزنیم تا دستورات اجرا شود.

Select first\_name , last\_name , email,... from users

نویسی است. برای وارد کردن داده به جدول مان باید به همین روش عمل کنیم. Query زبان mysql زبان

Insert into users (first\_name , last\_name , ...) values ('parsa' , 'pakdel',...)

برای نوشتن کامنت از\*/ استفاده می کنیم.

در تصویر زیر پرکاربردترین Query های مربوطه را بررسی می کنیم

```

select * from users where is .....=..... #فیلتر
delete from users where .....=..... #حذف
update users set ..... = ..... #اصلاح
update users set .... where .... = .... #بروزرسانی
alter table users add age varchar(); #افزونه کردن فیلد
alter table users modify age INT; #تغییر دادن
select * from users order by ... #مرتب سازی
select * from users where gender = 'f' order #ترکیب کردن دستورات
select * from users where gender = 'f' limit 1 #محدود کردن دستورات

```

اکنون چند مثال Query برایتان میزنم:

کاربرانی را انتخاب کن بین 20 تا 30 سن دارند #کاربرانی که اسم کوچکشان با حرف شروع پی می شود#creat index lindex on users (location) #اگر چند فیلد جدول را بخواهیم با سرعت بالا جستجو کنیم #drop index lindex on users # برای حذف ایندکس مورد نظر

## Foreinkey

برای پیوند دادن داده های دو یا چند جدول به یکدیگر استفاده می شود.

اکنون یک جدول جدید ایجاد می کنیم.

```

Create table posts (
Id int auto_increment,
User_id int,
Title nvarchar(100),
Body text,
Publish_date datetime default current_timestamp,
Primary key (id),
Foreignkey (user_id) references users (id)
)

```

جدول ایجاد می شود حال آیکن چرخ دنده را می زنیم و چند داده به جدول مان اضافه می کنیم.

```

Insert into posts (userid , title , body) values
(1 , 'post one title' , this is post one body),
(2 , 'post two title' , this is post two body)

```

Forein key در user می نویسیم id باشد سپس انتخاب کرده و می بنیم post می نویسیم در همان id دقیقاً همان داشتیم و می بینیم باید دقت کنیم هر شده است.id که در

برای ادغام کردن اطلاعات دو جدول Join استفاده می کنیم.

Inner join <<<<<<<<<<<< فیلدهایی که مج هستند

Right join <<<<<<<<< فیلدهای سمت راست که مج نیستند

Left join <<<<<<<< فیلدهای سمت چپ که مج نیستند

می توانیم بجای اسم کامل جدول از حرف اولش استفاده کنیم.

select u.first\_name,u.last\_name,p.id,p.title\* from posts as p innerjoin users as u on u.id

= p.user\_id

استفاده می کنیم. Right و برای کاربر یوزر از left از inner استفاده می کنیم. posts چون بجای

ها را بیاورد. برای اینکه تمام comment اکنون می خواهیم جدول سوم را نیز اضافه کنیم. جدولی به نام

Select \* from comments as c right join posts as p on p.id = c.post\_id

برای دیدن ردیف های جدول:

Select count (\*) from posts

بیشترین و کمترین مجموع سن:

Select max/min (age) as max\_age from users

گروه بندی بر اساس جنسیت:

Select gender count (\*) from users groupby gender

برگرداندن صفر بجای null:

Select case when login\_count is null

Then else login\_count end as login count ,

Count(\*) from users groupby login\_count

انتخاب های تودر تو:

Select \* from users as u leftjoin(

Select user\_id , count (\*) as post\_count from posts groupby user\_id)  
as p on u.id = p.user.id ;

آموزش پایگاه داده mysql به پایان رسید. در بخش بعدی به طراحی آنلاین شاپ به کمک این دیتابیس می پردازیم.

```
from fastapi import FastAPI, Depends, HTTPException, UploadFile, File  
from sqlalchemy import create_engine, Column, Integer, String, ForeignKey  
from sqlalchemy.ext.declarative import declarative_base  
from sqlalchemy.orm import sessionmaker, relationship, Session  
from pydantic import BaseModel  
from typing import List  
import pymysql  
import os
```

```
# برای جایگزینی pymysql تنظیم  
pymysql.install_as_MySQLdb()
```

# تنظیمات پایگاه داده MySQL

DATABASE\_URL = "mysql+pymysql://root:1234@localhost/shopdb"

```
engine = create_engine(DATABASE_URL)  
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
Base = declarative_base()
```

# مدل های پایگاه داده

```
class UserDB(Base):
```

```

__tablename__ = "users"
id = Column(Integer, primary_key=True, index=True)
username = Column(String(50), unique=True, index=True)
password_hash = Column(String(100))

posts = relationship("PostDB", back_populates="owner")

```

```

class PostDB(Base):
    __tablename__ = "posts"
    id = Column(Integer, primary_key=True, index=True)
    caption = Column(String(255))
    image_url = Column(String(255))
    owner_id = Column(Integer, ForeignKey("users.id"))

    owner = relationship("UserDB", back_populates="posts")

```

# ایجاد جدول‌ها

```
Base.metadata.create_all(bind=engine)
```

# اسکیمای Pydantic

```

class PostBase(BaseModel):
    caption: str
    image_url: str

```

```

class PostCreate(PostBase):
    pass

```

```
class Post(PostBase):
```

```
id: int
owner_id: int
class Config:
    orm_mode = True

class UserBase(BaseModel):
    username: str

class UserCreate(UserBase):
    password: str

class User(UserBase):
    id: int
    posts: List[Post] = []
    class Config:
        orm_mode = True

# تابع اتصال به پایگاه داده
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# قاعده CRUD
def create_user(db: Session, user: UserCreate):
    db_user = UserDB(username=user.username, password_hash=user.password)
```

```
db.add(db_user)
db.commit()
db.refresh(db_user)
return db_user

def get_posts(db: Session, skip: int = 0, limit: int = 10):
    return db.query(PostDB).offset(skip).limit(limit).all()

def create_post(db: Session, post: PostCreate, owner_id: int):
    db_post = PostDB(**post.dict(), owner_id=owner_id)
    db.add(db_post)
    db.commit()
    db.refresh(db_post)
    return db_post

# اپلیکیشن FastAPI
app = FastAPI()

@app.post("/users/", response_model=User)
def api_create_user(user: UserCreate, db: Session = Depends(get_db)):
    existing = db.query(UserDB).filter(UserDB.username == user.username).first()
    if existing:
        raise HTTPException(status_code=400, detail="Username already registered")
    return create_user(db=db, user=user)

@app.post("/posts/", response_model=Post)
def api_create_post(post: PostCreate, db: Session = Depends(get_db), owner_id: int = 1):
```

```
return create_post(db=db, post=post, owner_id=owner_id)

@app.get("/posts/", response_model=List[Post])
def api_get_posts(skip: int = 0, limit: int = 10, db: Session = Depends(get_db)):
    return get_posts(db, skip=skip, limit=limit)

@app.post("/uploadfile/")
async def api_upload_file(file: UploadFile = File(...)):
    os.makedirs("uploads", exist_ok=True)
    file_path = f"uploads/{file.filename}"
    with open(file_path, "wb") as f:
        f.write(await file.read())
    return {"filename": file.filename}
```

اکنون می خواهیم پروژمان را به پایتون متصل کنیم.

```
from fastapi import FastAPI, UploadFile, File, Form, HTTPException
import pymysql
from typing import List
import os
```

```
# اتصال به پایگاهداده MySQL
def get_connection():
    return pymysql.connect(
        host="localhost",
        user="root",
        password="1234",
        database="shopdb",
        cursorclass=pymysql.cursors.DictCursor
    )
```

```
# ایجاد جدول‌ها ( فقط بار اول اجرا کن )
def create_tables():
    conn = get_connection()
    with conn:
        with conn.cursor() as cursor:
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS users (
                    id INT AUTO_INCREMENT PRIMARY KEY,
                    username VARCHAR(50) UNIQUE,
                    password_hash VARCHAR(100)
                )
                """
            )
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS posts (
                    id INT AUTO_INCREMENT PRIMARY KEY,
                    caption VARCHAR(255),
                    image_url VARCHAR(255),
                    owner_id INT,
                    FOREIGN KEY(owner_id) REFERENCES users(id)
                )
                """
            )
    print("Tables created.")
```

# اجرای فقط بار اول :

```
create_tables()
```

```
app = FastAPI()
```

```
@app.post("/users/")
def create_user(username: str = Form(...), password: str = Form(...)):
    conn = get_connection()
    with conn:
        with conn.cursor() as cursor:
            cursor.execute("SELECT * FROM users WHERE username=%s",
                           (username,))
            if cursor.fetchone():
                raise HTTPException(status_code=400, detail="Username already
exists")
            cursor.execute("INSERT INTO users (username, password_hash) VALUES
(%s, %s)", (username, password))
            conn.commit()
    return {"message": "User created successfully"}


@app.post("/posts/")
def create_post(caption: str = Form(...), image_url: str = Form(...), owner_id: int
= Form(...)):
    conn = get_connection()
    with conn:
        with conn.cursor() as cursor:
            cursor.execute("INSERT INTO posts (caption, image_url, owner_id)
VALUES (%s, %s, %s)", (caption, image_url, owner_id))
            conn.commit()
    return {"message": "Post created"}


@app.get("/posts/")
def get_posts():
    conn = get_connection()
```

with conn:

```
with conn.cursor() as cursor:
```

```
    cursor.execute("SELECT * FROM posts")
```

```
    posts = cursor.fetchall()
```

```
return posts
```

```
@app.post("/uploadfile/")
```

```
async def upload_file(file: UploadFile = File(...)):
```

```
    os.makedirs("uploads", exist_ok=True)
```

```
    file_path = f"uploads/{file.filename}"
```

```
    with open(file_path, "wb") as f:
```

```
        f.write(await file.read())
```

```
    return {"filename": file.filename}
```

## Sqllite3 - ۴-۲

### تعريف

این دیتابیس یک کتابخانه نرم افزاری سیستم مدیریت رابطه ای را ارائه می دهد و از نظر تنظیم مدیریت و منابع سبک است.

### ویژگی ها

:بر عکس My sql نیاز به سرور واسط نیست.

:به سیستم عامل خاصی احتیاج ندارد و همه را پشتیبانی می کند.

:به پیکربندی نیاز ندارد.

:یک عمل یا کامل انجام می گیرد یا کلا انجام نمی شود.

:۵ هر متاداده را در ستون خیره می کند.

:۶ هم زمان به چند پروژه پایگاه داده دسترسی دارد.

۷) قادر به ساخت پایگاه داده حافظه ای است که کار با آن سریع تر است.

را برای نمایش دادن گرافیکی داده ها دانلود کنیم dbbrowser می توانیم برنامه [از www.sqlitebrowser.org](http://www.sqlitebrowser.org)

اکنون مرحله به مرحله کار با این دیتابیس را پیش می بریم:

1) ایجاد اتصال – Create connection

```
Import sqlite3
```

```
Con = sqlite3.connect('my database.db')
```

```
Con.close()
```

2) ایجاد دیتابیس – Create db-

```
Import sqlite3
```

```
From sqlite3 import error
```

```
def sql_connection():
```

```
try:
```

```
    con = sqlite3.connect(':memory:')
```

```
    print('parsa')
```

```
except:
```

```
    print(error)
```

```
finally:
```

```
    con.close()
```

```
sql_connection()
```

3) شی مکان نما - cursor

```
Import sqlite3
```

```
Con = sqlite3.connect('my database.db')
```

```
Cursor obj = con.cursor()
```

اکنون با استفاده از () Excute() برای اجرای هر گونه نمایش داده استفاده می کنیم.

4) Create table

```

Import sqllite3
from sqllite3 import error
def sql_connection():
    try:
        con = sqllite3.connect('my database.db')
        return con
    except:
        print(error)
def sql_table(con):
    cursor obj = con.cursor()
    cursor obj. excute('Create table employees
Id integer primaray key , name text. salary Text,
deptamentvText, position Text , hireDate Text')
    con.commit()
con = sql_connection()
sql_table(con)

```

همیشه بعد از execute باید commit بیاوریم.

```

Insert(5
Import sqllite3
Con = sqllite3.connect ('my database.db')
def sql_insert (con. emplor):
    cursor obj = con.cursor()
    cursor obj.execute (INSERT INTO employees
Values (1,'jhon' , 700 , 'HR' , 'manager' , '2025-03-01'))
    Con.commit()
Sql_insert (con , emplor)

```

اگر بخواهیم تغییرات ایجاد کنیم باید در ورودیتابع اول قرار بدهیم و سپس آنرا تعریف کنیم  
و در تابع آخر بنویسیم

داده وارد کردن مهم است چون باید اطلاعات را یکجا نگهداری کنیم. و بعداً یکسری عملیات روی آن انجام می دهیم.

همچنین می توان مقادیر آرگومان را با علامت ? استفاده کنیم.

Import sqllite3

```
Con = sqllite3.connect ('my database.db')
```

```
def sql_insert(con,exist):
```

```
    cursor obj = con.cursor()
```

```
    cursor obj.execute ('insert into employee(id,name,salary,  
department,position) Values(?, ?, ?, ?, entities)
```

```
    con.commit()
```

```
    entities = (2, 'andrew',800,'IT','tech')
```

```
    sql_insert (con, entites)
```

update(6

```
import sqllite3
```

```
con = sqllite3.connect ('my database.db')
```

```
def sql_update (con):
```

```
    cursor obj = con. cursor()
```

```
    cursor obj.execute ('UPDATE employees
```

```
    SET name = 'parsa' where id = 2')
```

```
    Con. commit ()
```

```
Sql_update (con)
```

Select(7

برای انتخاب تمام ستون های داده از یک جدول از \* استفاده می کنیم. این دستور در روش اجرای شی مکان نما اجرا می شود.

```
Select column1 , column2 from tables_name
```

## Fetchall(7)

مکان نما برای ذخیره مقادیر در یک متغیر استفاده Fetchall و سپس select برای واکشی داده ها از یک دیتابیس از می شود.

```
Import sqllite3
con = sqllite3.connect ('my database.db')
def sql_fetch(con):
    cursor obj = con.cursor()
    cursor obj.execute ('SELECT * from employees')
    rows = cursor obj. fetchall ()
    for row in rows :
        print(row)
sql_fetch(con)
```

**Sqlalchemy -۴-۳**

کد بزنیم و بین پایتون و دیتابیس قرار می گیرد و کارهای SQL های پایتون است و دیگر لازم نیست با ORMyکی از دیتابیس را با کد پایتون به ما می دهد.

برای درک بهتر این موضوع یک پروژه سایت دانشگاهی را شروع به طراحی می کنیم:

```
from sqlalchemy import create_engine, Column, Integer, String, ForeignKey, Float
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker
```

```
Base = declarative_base()
```

```
class Department(Base):
    __tablename__ = 'departments'
    id = Column(Integer, primary_key=True)
```

```
name = Column(String, unique=True, nullable=False)
```

```
professors = relationship("Professor", back_populates="department")
```

```
students = relationship("Student", back_populates="department")
```

```
class Professor(Base):
```

```
    __tablename__ = 'professors'
```

```
    id = Column(Integer, primary_key=True)
```

```
    name = Column(String, nullable=False)
```

```
    department_id = Column(Integer, ForeignKey('departments.id'))
```

```
department = relationship("Department", back_populates="professors")
```

```
courses = relationship("Course", back_populates="professor")
```

```
class Student(Base):
```

```
    __tablename__ = 'students'
```

```
    id = Column(Integer, primary_key=True)
```

```
    name = Column(String, nullable=False)
```

```
    department_id = Column(Integer, ForeignKey('departments.id'))
```

```
department = relationship("Department", back_populates="students")
```

```
enrollments = relationship("Enrollment", back_populates="student")
```

```
grades = relationship("Grade", back_populates="student")
```

```
class Course(Base):
```

```
    __tablename__ = 'courses'
```

```
    id = Column(Integer, primary_key=True)
```

```
    name = Column(String, nullable=False)
```

```
    professor_id = Column(Integer, ForeignKey('professors.id'))
```

```
professor = relationship("Professor", back_populates="courses")
enrollments = relationship("Enrollment", back_populates="course")
grades = relationship("Grade", back_populates="course")

class Enrollment(Base):
    __tablename__ = 'enrollments'
    id = Column(Integer, primary_key=True)
    student_id = Column(Integer, ForeignKey('students.id'))
    course_id = Column(Integer, ForeignKey('courses.id'))

    student = relationship("Student", back_populates="enrollments")
    course = relationship("Course", back_populates="enrollments")

class Grade(Base):
    __tablename__ = 'grades'
    id = Column(Integer, primary_key=True)
    student_id = Column(Integer, ForeignKey('students.id'))
    course_id = Column(Integer, ForeignKey('courses.id'))
    grade = Column(Float, nullable=False)

    student = relationship("Student", back_populates="grades")
    course = relationship("Course", back_populates="grades")

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
```

```
password = Column(String, nullable=False)
role = Column(String, nullable=False) # 'student', 'professor', 'admin'
```

```
engine = create_engine('sqlite:///university.db')
Base.metadata.create_all(engine)
```

Run می کیم. تا اینجا جداولمان ساخته شد.

```
Session = sessionmaker(bind=engine)
session = Session()
```

امکام تعامل با داده (حذف یا بروزرسانی) با دستور sessionmaker اینجا با استفاده از

```
dep1 = Department(name='Computer Science')
dep2 = Department(name='Mathematics')
```

دو رشته تحصیلی ایجاد کردیم

```
prof1 = Professor(name='Dr. Smith', department=dep1)
```

یک استاد تعریف کردیم

```
stu1 = Student(name='Alice', department=dep1)
```

یک دانشجو تعریف کردیم

```
cour1 = Course(name='Data Structures', professor=prof1)
```

یک درس تعریف کردیم

```
enr1 = Enrollment(student=stu1, course=cour1)
```

ثبت نام دانشجو

```
grade1 = Grade(student=stu1, course=cour1, grade=95.0)
```

نمره دانشجو

```
user1 = User(name='Alice', email='alice@example.com',
password='hashed_password', role='student')
```

```
user2 = User(name='Dr. Smith', email='smith@example.com',
password='hashed_password', role='professor')
```

در اینجا کاربران سیستم را نمایش دادیم

```
session.add_all([dep1, dep2, prof1, stu1, cour1, enr1, grade1, user1, user2])  
session.commit()
```

تمام اشیایی که ساختیم به جلسه اضافه و در دیتابیس ذخیره می شود

```
print("Database initialized successfully!")
```

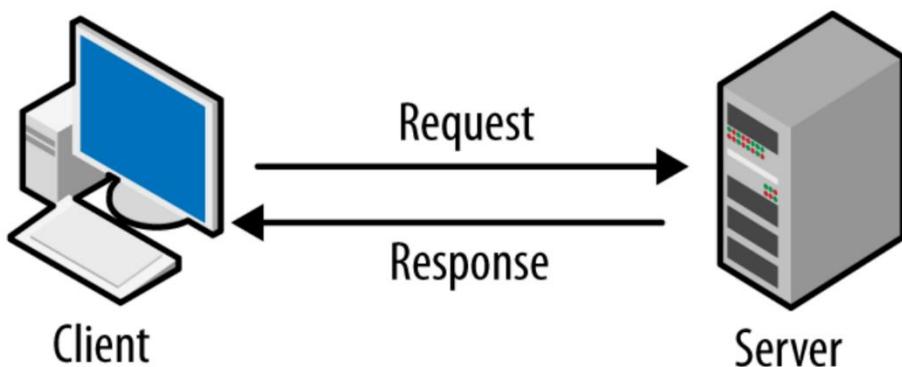
در نهایت موفقیت آمیز بودن عملیات چاپ می شود.



# فصل پنجم HTTP & SSH

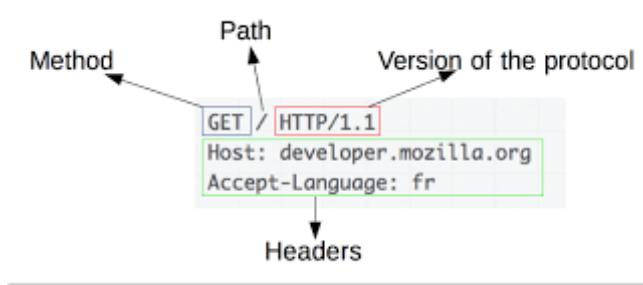
## ۱-۵ ساختار پروتکل HTTP

در بستر وب با پورت ۸۰ client و Server برای برقراری ارتباط بین یکسری قوانین که در محیط شبکه انجام شده تا دو سیستم در شبکه اطلاعات تبادل کنند: protocol درخواست برای نظرات RFC:



به در خواست هایی که از سمت کلاینت به سرور ارسال می شود. Request: از سرور به کلاینت ارسال می شود. Response: به پاسخ هایی که پس از دریافت آنون به بررسی ساختار آنها می پردازیم.

### ۱-۵-۱ ساختار Request



## ساختار Response(2)



اکنون به توضیح کامل هر یک از این ساختار ها می پردازیم:

:status\_code(a

پاسخ هایی هستند که سرور به در خواست مرورگر به کاربر ارائه می کند. این پاسخ ها وضعیت دسترسی به صفحات وب را مشخص می کند. در پایین به مهمترین این کد ها و وضعیت دسترسی شان اشاره می کنیم



## کدهای وضعیت ۵۰۰

کدهای وضعیت ۵۰۰ وقتی ایجاد می‌شوند که درخواستی از مرورگر به سرور ارسال شود اما سرور نتواند به آن پاسخ دهد. به این کدها اصطلاحاً خطاهای سمت سرور گفته می‌شود. مهم‌ترین کدهای وضعیت ۵۰۰ شامل موارد زیر است:

### کد ۵۰۰

این کد به خطا سرور داخلی معروف است و نشان می‌دهد سرور با موقعیتی مواجه شده که نمی‌داند چگونه آن را مدیریت کند. به عنوان یک وب مستر وقتی با این خطا مواجه شدید باید بدانید که مشکلی در مورد دسترسی به محتوای سایت وجود ندارد، بلکه مشکل از سرور سایت است. در این حالت نه ربات‌های خزنه گوگل و نه کاربران امکان دسترسی به صفحات سایت را ندارند و این می‌تواند به شدت بر جایگاه آن در موتورهای جستجو تأثیر بگذارد.

### کد ۵۰۱

این کد نشان می‌دهد درخواستی از سوی مرورگر به سرور سایت ارائه شده که برای سرور ناشناخته بوده و امکان پاسخگویی به آن را ندارد.

### کد ۵۰۲

این کد هم نشان می‌دهد سرور هنگام پاسخگویی به درخواست مرورگر، پاسخ نامعتبری دریافت کرده است.

### کد ۵۰۳

این کد وضعیت اعلام می‌کند که سرور برای پاسخگویی به درخواست مرورگر آماده نیست. این حالت معمولاً زمانی رخ می‌دهد که سرور برای تعمیر و نگهداری از دسترس خارج شده یا بار ترافیک زیادی به آن تحمیل شده است. از لحاظ سئو این خطا خیلی بد نیست، زیرا خبر از یک وقفه موقت در کار سرور می‌دهد و ربات‌های موتور جستجو متوجه می‌شوند که سرور برای مدت کوتاهی پاسخگو نیست، درنتیجه اتفاق بدی برای رتبه سایت در نتایج جستجو نمی‌افتد، مگر اینکه وقفه طولانی شود.

**کد ۵۰۴**

این کد نشان می‌دهد سرور نتوانسته در زمان مناسب به درخواست مرورگر پاسخ دهد.

**کد ۵۰۵**

این کد نشان می‌دهد قسمتی از پروتکل با که مرورگر کاربر از آن استفاده می‌کند با سرور سازگار نیست.

**کدهای وضعیت ۴۰۰**

این کدها بر خلاف سری قبل شامل خطاهای سمت کاربر بوده و در موقعی نمایش داده می‌شود که سایت یا صفحه درخواستی قابل دسترسی نیست. این دسته کدها اعلام می‌کنند درخواست مرورگر دریافت شده اما آدرس صفحه‌ای که درخواست شده، وجود ندارد. این خطاهای از طرف وبسایت ایجاد می‌شوند و معمولاً وقتی رخ می‌دهند که صفحه موردنظر کاربر در سایت وجود نداشته باشد یا مثلاً آدرس آن تغییر کرده است. در ادامه مهم‌ترین کدهای سری ۴۰۰ را بررسی می‌کنیم:

**کد ۴۰۰**

وقتی کد وضعیت ۴۰۰ اعلام شود بدین معنا است که سرور نتوانسته معنای درخواست مرورگر را درک کند، زیرا خطای در دستور آن وجود داشته است.

**کد ۴۰۱**

کد ۴۰۱ نشان می‌دهد که درخواست مرورگر توسط سرور پذیرفته نشده، زیرا برای این کار باید کاربر احراز هویت شود.

**کد ۴۰۳**

نمایش کد ۴۰۳ بدین معنا است که درخواست کاربر توسط سرور رد شده، زیرا اجازه دسترسی به محتوای درخواستی را نداشته است. تفاوت این کد با کد ۴۰۱ در این است که هویت کاربر تشخیص داده شده است. درواقع خطای ۴۰۱ زمانی رخ می‌دهد که هویت کاربر نامشخص بوده و سرور نمی‌تواند پاسخ درخواستی را به آن ارائه کند؛ اما خطای ۴۰۳ وقتی رخ می‌دهد

که هویت کاربر توسط سرور شناسایی شده اما کاربر اجازه دسترسی به محتوای درخواستی را ندارد. در این حالت درخواست کاربر به طور کلی رد می‌شود.

#### کد ۴۰۴

کد ۴۰۴ را احتمالاً در بین کدهای وضعیت زیاد دیده‌اید. کدی که اعلام می‌کند سرور صفحه‌ای را که باید برای پاسخ به درخواست کاربر نمایش دهد، پیدا نمی‌کند یا می‌خواهد وجود آن را از کاربری که هویتش نامشخص است، پنهان کند. البته این کد مشخص نمی‌کند، از دست رفتن صفحه به صورت موقتی اتفاق افتاده یا دائمی است.

کافی است یک آدرس اشتباه را در مرورگر وارد کنید و نتیجه را ببینید. در صورتی که در سایت شما هم صفحه ۴۰۴ یا زگردادن، کاربران سایت‌تان با چنین موقعیتی مواجه شده و معمولاً سایت شما را می‌بندند و می‌رونند. البته در هر سایتی ممکن است صفحاتی وجود داشته باشند که کد وضعیت ۴۰۴ را بازمی‌گردانند، از لحاظ سئو در صورتی که صفحه‌ای از سایت شما با خطای ۴۰۴ روبرو شده، باید آن را با ریدایرکت ۳۰۱ به یک صفحه دیگر که موضوعی مرتبط با آن دارد، تغییر مسیر دهید.

یکی از اقدامات اشتباهی که در مواجهه با این صفحات مرتکب می‌شوند، این است که آن را به صفحه اصلی سایت ریدایرکت می‌کنند. این کار اشتباه است، چراکه کاربر وارد شده به سایت را سردرگم کرده و باعث می‌شود کاربر سایت شما را ترک کند. بهترین کار ریدایرکت صفحه روی صفحه‌ای با موضوع مشابه است. برای مثال فرض کنید محصولی از سایت شما با عنوان «کیک رژیمی بدون شکر» دیگر وجود ندارد، می‌توانید صفحه این محصول را روی صفحه دسته‌بندی محصولات بدون شکر ریدایرکت کنید.

با این حال در برخی موارد هم استفاده از کد ۴۰۴ برای صفحه لازم است. برای مثال اگر یک محصول مشخص را به مدت طولانی موجود ندارید، بهتر است آن را ۴۰۴ کنید. این کار باعث می‌شود از خوش دوباره و ایندکس شدن صفحه جلوگیری کنید.

#### کد ۴۱۰

کد ۴۱۰ را باید دائمی‌تر از ۴۰۴ در نظر بگیرید. یعنی در این حالت صفحه موردنظر به طور کلی از سرور حذف شده و هیچ آدرس جایگزینی برای آن وجود ندارد. هر لینکی که از صفحات

سایت تان به یک صفحه 410 داده باشید، در واقع کاربران و خزنه‌های گوگل را به یک مقصد نامعتبر می‌فرستند، بنابراین بهتر است لینک‌هایی که به چنین صفحاتی داده شده را به طور کلی حذف کنید..

### کدهای وضعیت ۳۰۰

کدهای وضعیت سری ۳۰۰ شامل پیام تغییر مسیر است. این دسته کدها وقتی نمایش داده می‌شوند که هنگام درخواست یک صفحه به صفحه دیگری تغییر مسیر دهید. درواقع این کدها اعلام می‌کنند درخواست مروگر به درستی دریافت شده اما پاسخ آن در مسیر دیگری قرار دارد. این کدها نه تنها خطری ندارند، بلکه بخشی از کار شما به عنوان وبمستر، استفاده از ریدایرکت‌ها برای انتقال کاربر از برخی صفحات به صفحات دیگر است. برای مثال اگر نمی‌خواهید یک صفحه از سایت تان به هر دلیلی در معرض دید مخاطبان قرار گیرد، آن را به صفحه‌ای مشابه ریدایرکت می‌کنید

### کد ۳۰۰

این کد اعلام می‌کند درخواست ارائه شده به سرور چند پاسخ مختلف دارد؛ یعنی کاربر یا مروگر باید یکی از پاسخ‌ها را برای نمایش انتخاب کند.

### کد ۳۰۱

وقتی این کد نمایش داده می‌شود بدین معنا است که آدرس جدیدی به صفحه درخواست شده اختصاص یافته است، درنتیجه کاربر برای دیدن صفحه به آدرس جدید هدایت می‌شود. به لحاظ سئویی وقتی باید از این ریدایرکت استفاده کنید که صفحه موردنظر شما برای همیشه به آدرس جدید منتقل شده باشد. خوبی ریدایرکت ۳۰۱ این است که اعتبار محتوایی هر صفحه را به آدرس جدید آن انتقال می‌دهد.

### کد ۳۰۲

این کد که به ریدایرکت موقت معروف است، نشان‌دهنده انتقال موقت محتوای یک صفحه به آدرس جدید است. یعنی ممکن است پس از مدتی صفحه موردنظر به آدرس قبلی خود برگردد. از آنجا که در این حالت ریدایرکت به صورت موقت در نظر گرفته می‌شود، اعتبار سئویی صفحه به آدرس جدید منتقل نمی‌شود. بنابراین حتماً توجه داشته باشید، زمانی که قصد دارید صفحه

موردنظر را برای همیشه به آدرس جدید منتقل کنید، حتماً از ریدایرکت ۳۰۱ استفاده کرده و زمانی که نمی‌خواهید این کار را برای همیشه انجام دهید، حتماً از ریدایرکت ۳۰۲ استفاده کنید تا چار مشکلات سئوی نشوید.

## ۲۰۰ کدهای وضعیت

کدهای وضعیت سری ۲۰۰، کدهای شامل پیام موفقیت هستند. این دسته کدها اعلام می‌کنند که همه چیز به حالت طبیعی اجرا شده و پاسخ مورد انتظار برای مرورگر ارسال شده است. در این حالت سرور درخواست مرورگر را به خوبی دریافت کرده و پاسخ صحیح آن را بدون اختلال صادر کرده است. در نتیجه کاربر می‌تواند دقیقاً همان صفحه‌ای را که درخواست کرده، مشاهده کند. به عنوان یک وب مستر باید مطمئن شوید همه صفحات و منابع سایت شما، کدهای سری ۲۰۰ را در پاسخ به کاربران اعلام کرده و همه آنها به راحتی در دسترس کاربران قرار دارند. مهم‌ترین کدهای سری ۲۰۰ را در ادامه مرور می‌کنیم:

### ۲۰۱ کد

این کد اعلام می‌کند، درخواست موردنظر کاربر با موفقیت برآورده شده و در پاسخ آن یک چند منبع جدید به وجود آمده است.

### ۲۰۲ کد

کد ۲۰۲ هم نشانه‌ای از موفقیت درخواست کاربر است اما اعلام می‌کند درخواست کامل نشده است. یعنی در این حالت درخواست کاربر برای یک پردازش توسط سرور پذیرفته شده اما هنوز کامل نشده است. معمولاً این کد وقتی اعلام می‌شود که در میان روند پردازش، درخواست جدیدی ارائه شود.

### ۲۰۳ کد

این کد هم اعلام می‌کند که درخواست کاربر با موفقیت پاسخ داده شده است اما اطلاعات موجود در درخواست مرورگر با آنچه که باید باشد، کمی متفاوت بوده است.

**کد ۲۰۴**

کد ۲۰۴ اعلام می‌کند که سرور درخواست کاربر را به خوبی برآورده کرده است اما محتوای جدیدی در پاسخ به درخواست در دسترس نیست.

### **کدهای وضعیت ۱۰۰**

کدهای وضعیت سری ۱۰۰ شامل پاسخ اطلاعاتی است و اعلام می‌کند سرور در حال پردازش درخواست مرورگر است. این کدها کاملاً موقتی هستند و امکان ظاهر شدن آنها بسیار کم است. درواقع، کدهای وضعیت ۱۰۰ پاسخ نهایی یک پردازش نیستند و در صورتی که حین پردازش درخواستی به سرور ارسال شود، پاسخ‌هایی از سری ۱۰۰ داده می‌شود. البته این شرایط خیلی به ندرت اتفاق می‌افتد.

**کد ۱۰۰**

وقتی این کد به عنوان پاسخ سرور ارسال می‌شود، بدین معنا است که بخش اولیه درخواست کاربر توسط سرور دریافت شده و در حال اجرا است. در این حالت کاربر باید منتظر ادامه پردازش باشد یا اگر تمایلی به ادامه ندارد، درخواست خود را لغو کند.

**کد ۱۰۱**

کد ۱۰۱ نشان می‌دهد سرور درخواست ارائه شده را درک کرده و در حال تغییر پروتکل خود است.

**کد ۱۰۲**

کد ۱۰۲ اعلام می‌کند کل درخواست کاربر را پذیرفته و در حال پردازش آن است. از آنجا که پردازش درخواست به پایان نرسیده، هنوز پاسخی هم برای آن وجود ندارد.

### **Methods (b)**

GET(1): زمانی که می‌خواهیم به یک منبع دسترسی داشته باشیم.

POST(2): زمانی که می‌خواهیم یکسری اطلاعات به سرور ارسال کنیم.

PUT(3): زمانی که می‌خواهیم یکسری اطلاعات را بصورت کلی ویرایش کنیم.

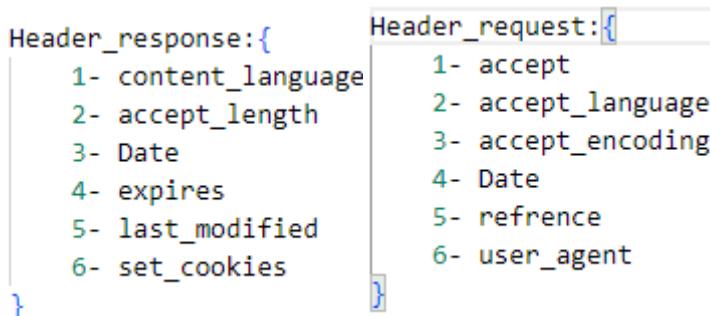
: زمانی که می خواهیم یکسری اطلاعات را بصورت جزئی ویرایش کنیم.

: زمانی که بخواهیم یکسری اطلاعات را حذف کنیم.

: زمانی که بخواهیم فقط به header دسترسی داشته باشیم.

### Headers (c

اطلاعات درباره وضعیت سرور و بستر وب و کلاینت می دهد. کلید و مقدار دارد و با علامت: نمایش داده میشود



### قابلیت های مهم

را باز کرده و هر تعداد درخواست و پاسخ را تبادل می کند TCP است که یکHeaderKeep\_alive(1) کبار

و در نهایت پس از پایان آنرا می بندد و سرعت بارگزاری را بالا می برد.

منتظر پاسخ از سرور نمی مانیم و هر تعداد در خواست به سرور بدهیم به ترتیب به ما پاسخ می دهد(2) Pipling

### نحوه کار

کلاینت به سرور درخواست یک منبعی را می دهد که اگر احراز هویت نیاز داشته باشد سرور به کلاینت یک پاسخ ۴۰۱ می دهد.

به مذاکره بر سر محتوا بین کلاینت و سرور Content negotiation می گویند.

دو نوع مذاکره داریم:

(1) Server: محور سرور تصمیم می‌گیرد کدام منبع را به عنوان پاسخ به کلاینت ارسال کند.

(2) Agent: محور کاری به هدرهای کلاینت ندارد بلکه یک لیست از پاسخ‌ها و لینک‌هایش دارد و آنرا به کلاینت می‌دهد و خود کلاینت تصمیم می‌گیرد چه چیزی را به عنوان پاسخ بگیرد.

Mime\_encoding: کلاینت و سرور می‌توانند یکسری فایل‌ها را با فرمات‌های مختلف بین خودشان به صورت بازبینی تبادل کنند. همچنانی یکسری هدرها را ست کرده و دارای کلید و مقدار هستند.

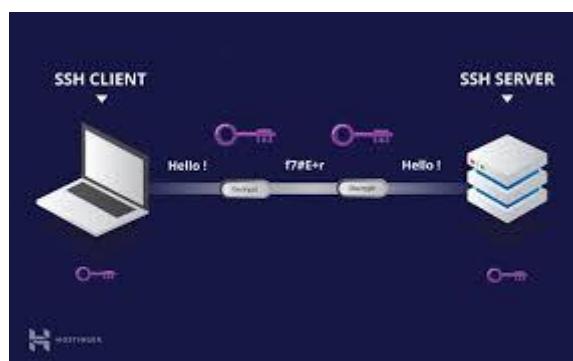
## ۲-۵- آشنایی با SSH و اتصال به سرور

این پروتکل این امکان را به ما می‌دهد که بتوانیم یک ارتباط امن بین دو سیستم به وسیله آی‌پی و پورت برقرار کنیم. ارتباط امن یعنی دیتا در مبدا رمزگذاری شده و در مقصد رمزگاری شود.

### نحوه احراز هویت بین کلاینت و سرور

(۱) الگوریتم متقارن: یک کلید برای رمزگذاری و یک کلید برای رمزگشایی. منظور از کلید یک رشته است.

(۲) الگوریتم نامتقارن: یک جفت کلید داریم که بصورت جداگانه برای رمزگشایی و رمزگاری استفاده می‌شود.



:درخواست یکسری فایل ها را می دهیم و به او می گوییم درباره ای منابعی که می خواهم تغییری ندهد و برود از سری های قبل بردارد و استفاده کند.

### Catch

اگر سرور اجازه کش به ما بدهد یک حالت زمانی است که چیزی را ست نمی کند پس می توانیم کش کنیم.

حالت بعد خصوصی است. یعنی فقط کاربر در خواست کننده می تواند کش کند و حالت آخر عمومی است و هر کس می تواند این محتوا را کش کند.

:سرور به کلاینت یک منبع می دهد و می گوید آخرین بار در یک زمانی یکسری تغییرات در آن اعمال کن.

:سرور به کلاینت یک منبع می دهد که انقضا دارد که به عنوان مقدار ست می کند و اگر انقضا بگذرد باید درخواست بروزرسانی بدهد.

### Ssh قابلیت های

۱) احراز هویت را بصورت نامتقارن انجام می دهد. پس ابتدا باید یک جفت کلید بسازیم.

۲) بطور پیش فرض از پورت ۲۲ استفاده می کند.

۳) این امکان را به ما می دهد زمانی که درخواست ارسال می کنیم یک کلید خصوصی در سیستم خودمان و یک کلید عمومی در سیستم مقصد ایجاد کنیم. زمانی که بخواهیم اتصال برقرار کنیم بوسیله خصوصی درخواست را تایید کرده و داده به مقصد می رسد و بوسیله کلید عمومی رمزگشایی می شود. اگر موفقیت آمیز بود به رمز عبور نیازی نیست.

```
Ssh _ keygen -l vf /.ssh/id_rsa.pub
```

```
cd.ssh ->ls-> id_rsa Id _rsa.pub
```

۴) از کاربردهای دیگر آن می توان به انتقال فایل ها و اجرای دستورات سمت سرور اشاره کرد. همچنین مدیریت شبکه ها امنیت پورت کنترل سطح دسترسی و مدیریت رمز عبورها.

## Ssh روش ارتباط در

(1) Slogin: برقراری ارتباط با سرورهای مبتنی بر سیستم عامل لینوکس.

(2) SSH: ارتباط با سرور و انتقال اطلاعات با کلاینت برای سرورها.

(3) Scp: برای تهیه کپی امن از داده ها.



# **فصل ششم**

# **Git & Github**

## ۶- سیستم کنترل نسخه

یک Git Version control system است. یعنی تغییرات فایل های ما را مدیریت و ردیابی می کند.

یک سرویس است که پروژه های گیت را در سرویس های گیت هاب قرار می دهد. GitHub می تواند در دو حالت عمومی یا خصوصی باشد.

یک ترمینال است که دستورات لینوکسی را اجرا می کند. Git bash: مخزن گرافیکی نمایش دستورات.

برای اضافه کردن گیت به پروژه به شیوه زیر عمل می کنیم :

cmd \_\_\_\_> cd desktop\_\_\_\_> cd git \_\_\_\_\_>git init (مخزن)

این است که اگر بیش از یک پروژه داشته باشیم محفظه هایی جدا (repository) دلیل ایجاد مخزن ایجاد می کند تا فایل ها قاطی نشوند.

git\_parsa(نام پوشه) \_\_\_\_\_> git \_status(وضعیت)

تا اینجا فایل ها قابل ردیابی نیستند.

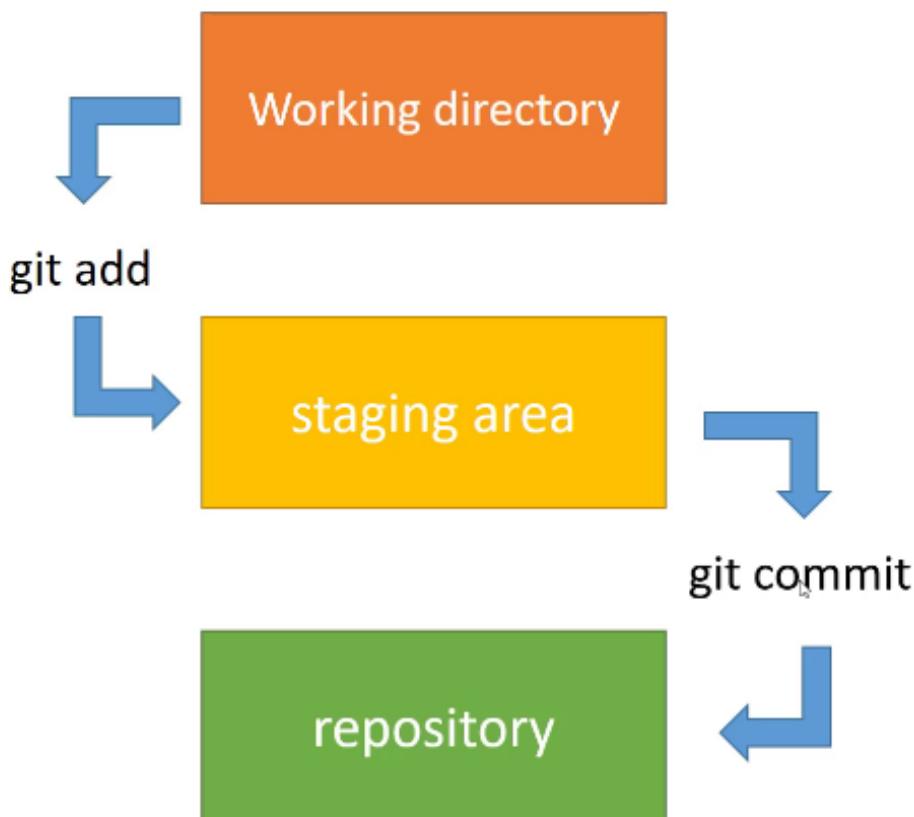
git\_parsa\_\_\_\_>git\_add file1.txt

new file : file1.txt

git\_parsa\_\_\_\_>git rm\_ \_ cached file1.txt

git\_parsa\_\_\_\_>git add.

حال تمام فایل های درون پوشه را به ما می دهد.



در ناحیه اول در همان پوشه پروژه خودمان هستیم.

در ناحیه دوم امکان این را داریم که با دستور گیت -ad یکسری فایل ها را کم یا زیاد کنیم.

در ناحیه سوم می توانیم با کامپیت کردن فایل ها را مدیریت کیم و تغییرات و باگ ها را بررسی کنیم.

`git_parsa__> git commit -m "initial project"`

خطا می دهد که نویسنده مشخص نیست پس باید دستورات زیر را انجام بدھیم:

`git_parsa__>git config __ global user.email "parsapakdel290@gmail`

`git_parsa__>git config __ global user.name "parsa pakdel"`

`git_parsa__>git commit -m "initial project"`

```
git_parsa__>git log
```

کامیت ایجاد می شود. حال باید تغییرات را اعمال کنیم

```
git_parsa__>git status
```

modified: file1.txt

```
git_parsa__>git add.
```

```
git_parsa__>git status
```

modified: file1.txt

```
git_parsa__>git commit -m "hello parsan"
```

```
git_parsa__>git log
```

تغییرات مدنظر اعمال می شود.

حال فرض کنید چند فایل را بخواهیم کامیت کنیم و فایل آخر را فراموش کردیم

```
git_parsa__>git add.
```

```
git_parsa__>git status
```

```
git_parsa__>git commit _ _amend
```

اکنون فایل آخر نیز کامیت شده است.

اگر را بزنیم از گیت لاغ خارج می شود.

اگر بخواهیم از فایلی صرف نظر کنیم به روش زیر عمل می کنیم :

```
git_parsa__> git status
```

```
git_parsa__>git ignore
```

```
git_parsa__> git add.
```

```
git_parsa__>git commit -m "add gitignore file"
```

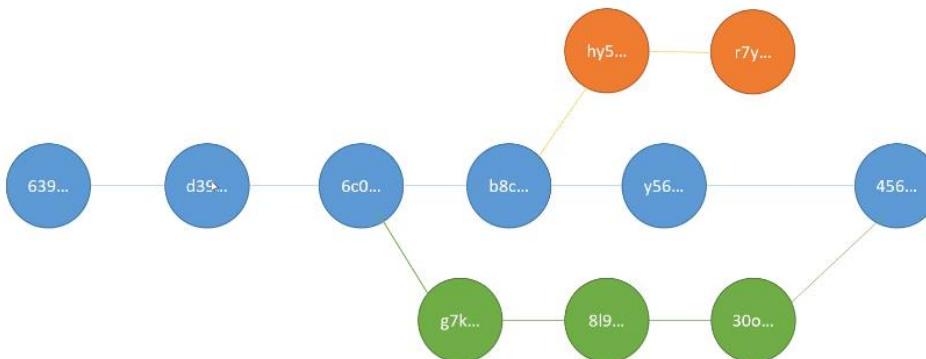
```
git_parsa__> git status
```

می توانید [www.gitignore.io](http://www.gitignore.io) در سایت ignore شده است. حال می بینیم که فایل

مان پروژه هایتان را ارسال کنید و خودش صرف نظر می کند.

## ۶-۲- دستورات اصلی گیت

:اگر بخواهیم یک شاخه از پروژه مان جدا کنیم و تغییرات مدنظرمان را اعمال کنیم.  
یک اشاره گر است که موقعیت جاری ما را نشان می دهد.



```
>git branch          >git branch bugfix
* master            >git branch
                    bugfix
                    *master
                    >git switch bugfix
                    >git branch
                    *bugfix
```

حال می خواهیم در فایل ۱ یا ۲ متنی اضافه کنیم ابتدا ذخیره کرده و ۲ تغییر را کامیت می کنیم:

git\_parsa\_\_> git add.

git\_parsa\_\_>git commit -m "commit 1 on bugfix"

حال می خواهیم به شاخه باگ فیکس سوییچ کنیم:

git switch master

تا اینجا همچنان تغییرات اعمال نشده چون در شاخه باگ فیکس جدا از شاخه اصلی مستر بودیم.

```
git switch bugfix
```

اکنون تغییرات اعمال شده است.

برای ایجاد شاخه جدید روی هر شاخه که بایستیم روی همان اعمال می شود.

```
git switch -c bugfix2
```

```
git branch
```

```
bugfix
```

```
*bugfix2
```

```
master
```

برای تغییر اسم شاخه:

```
git branch -m bugfix3
```

```
git branch
```

```
bugfix
```

```
*bugfix3
```

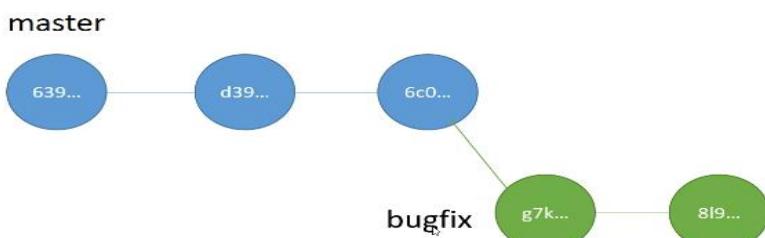
```
Master
```

برای حذف شاخه:

```
git switch master
```

```
git branch -d bugfix3
```

برای ادغام دو شاخه :



master



با توجه به تصویر فوق می خواهیم شاخه های مستر و باگ فیکس را با شاخه مستر ادغام کنیم.

cd Desktop\_\_> git pars\_\_>cls\_\_> git init \_\_>git status\_\_> git add.

git commit -m “initial project \_\_>git add.\_\_>git commit -m “commit 1 on master” \_\_>git log

تا اینجا دو کامیت روی مستر داریم

ما در روش برای تداخل داریم :

۱) بدون تداخل

۲) با تداخل

### روش اول

حال یک شاخه دیگر ایجاد می کنیم.

git switch -c bugfix\_\_ > git branch\_\_> git status

git add.\_\_> git commit -m “commit 1 on bugfix”

git add.\_\_> git commit -m “commit 2 on bugfix”

git branch\_\_> git switch master

فایل ها بدون تداخل ادغام شد. حال می توانیم متن هم اضافه کنیم.

git branch\_\_> git status\_\_> git add.\_\_> git commit -m “commit 2 on master”

git log\_\_> git switch -c bugfix2\_\_> git add.\_\_> git commit -m “commit 1 on bugfix” \_\_> git switch master \_\_>git add.\_\_>

\_\_> git commit -m “commit 3 on master” \_\_> git log\_\_>

git merge bugfix2

### روش دوم

git switch -c bugfix3 \_\_>git status\_\_>git commit -am “commit 1 on bugfix3”

git switch master\_\_> git merge bugfix3

git add.\_\_>git commit -m “fix conflicts”

فایل ها با تداخل ادغام شد.

## diff

(تغییرات بین شاخه ها و کامیت ها) working directory & staging area

1: دیدن تغییرات فایل های تغییر یافته git diff(1

2: دیدن تغییرات فایل های استیج شده git diff --staged(2

3: مقایسه بین دو کامیت git diff commit1 commit2(3

4: مقایسه بین دو شاخه git diff branch1 branch2(4

5: دیدن تغییرات یک فایل خاص git diff filename.py(5

6: مقایسه فایل در یک کامیت خاص با نسخه فعلی git diff commit\_id filename(6

7: نمایش فقط نام فایل های تغییر کرده git diff --name-only(7

8: نمایش خلاصه تغییرات (خطوط حذف یا اضافه شده) git diff --stat(8

## restore

1) تغییرات در working directory را از بین می برد.

2) فایل های استیج را بیرون می آورد.

3) تغییرات فایل را از یک کامیت موردنظر می گیرد.

git restore colors.txt

git add.\_\_> git restore --staged numbers.txt

git restore --source 986hwe colors.txt

## revert

زمانی که یک کامیت به فایل مان اضافه کردیم از ما می خواهد که تداخل را برطرف کنیم و ما را به کامیتی که خواسته بودیم بر می گرداند و یک کامیت جدید می دهد و چیزی را حذف نمی کند.

در بحث همکاری در گیت هاب نیز فرض کنیم یکسری از همکاران ما تعدادی کامیت دارند که اگر ما رست کنیم حذف می شود و زمان هدایت به گیت هاب دیگر همکاران آن کامیت ها را ندارند.

`git revert 8545`

`git add.`

`git revert --continue`

### stash

را پاک می کند working directory تغییراتی که برای کامیت آماده نشده را ذخیره می کنیم و در انتهای آن گوییم آن تغییرات را برگردان. کاربرد دیگر آن است که اگر در شاخه فرعی باشیم و بخواهیم با تداخل به مستر برویم از آن در شاخه فرعی استفاده می کنیم و سپس به مستر می رویم.

`git branch`\_\_>`git stash`\_> save working directory and index.

`git switch master`\_\_>`git commit -m "add pink"`

`git stash pop`\_\_>`git commit -m "add seven number"`

### apply

اگر بخواهیم تغییرات stash را بدون آنکه پاکش کنیم دریافت کنیم.

`git branch`

`git stash` \_\_>`git switch master` \_\_>`git stash apply`

`git commit -am "add number nine"`\_\_>`git swich odd-numbers`

`git stash apply`

### tag

نقاطی هستند که به یک کامیت خاص اشاره دارد و کاربرد آن ورژن بندی پروژه است.

برای ایجاد تگ یک پوشه جدید ایجاد کرده و یکسری کامیت ایجاد می کنیم.

`git tag <tag name>`

`git tag v2.0.0` \_\_>`git log --one line`

`git add.`\_\_>`git commit -m "add three"`

```
git tag -a v2.2.1  
git show v2.2.1  
git tag v1.0.2 dd732c2  
git tag -f v1.0.2 6nh2l5  
git tag -l "v2*"  
git checkout 68cf2e  
git tag -d v1.0.2  
git push origin master  
git push origin v1.0.2  
git push origin --tag
```

## ۶-۳- آشنایی با گیت‌هاب

### هاب در گیت‌هاب

هاب سیستم کنترل نسخه‌ای است که توسط اکثر توسعه‌دهندگان ترجیح داده می‌شود؛ زیرا نسبت به سایر سیستم‌ها برتری‌هایی دارد. مثلاً در ذخیره‌های تغییرات فایل‌ها بهتر عمل می‌کند. در بخش‌های بعدی این فصل به نحوه‌ی کار با گیت خواهیم پرداخت..

در دنیای رایانه، هاب به تجهیزات سخت‌افزاری گفته می‌شود که از آن برای اشتراک‌گذاری شبکه با گجت‌های مختلف استفاده می‌شود و در نتیجه هاب به نوعی شبکه را گسترش می‌دهد. هاب در گیت‌هاب نیز چنین مفهومی دارد. توسعه‌دهندگان پروژه‌های خود را در گیت‌هاب ذخیره می‌کنند و از این طریق به شبکه‌ی عظیم توسعه‌دهندگان دنیا وصل می‌شوند. در گیت‌هاب این امکان وجود دارد که پروژه‌ای را از مخزن توسعه‌دهنده به مخزن خود کپی و در آن تغییرات اعمال کنید و سپس درخواست اعمال تغییرات را به صاحب پروژه بفرستید تا در پروژه‌ی اصلی اعمال کند. امکان پرسش و پاسخ نیز در این شبکه‌ی گیت فراهم است.

حالا که با مفاهیم گیت و گیت‌هاب آشنا شدیم به نحوه‌ی کار با این سرویس‌ها می‌پردازیم اما ابتدا نیاز است تا در سرویس گیت‌هاب برای خود حساب کاربری ایجاد کنید.

## ساخت اکانت در گیت هاب

### مرحله اول) به سایت گیت هاب بروید

پس از ورود به سایت گیت هاب دکمه ساین آپ را بزنید.

### مرحله دوم ) مشخصات خود را وارد کنید

پس از ساین آپ کردن به صفحه‌ای هدایت می‌شود که باید مشخصات خود را وارد کنید و از شما خواسته می‌شود نام کاربری، آدرس ایمیل و رمز عبور خود را ارائه دهید. اطمینان حاصل کنید که یک نام کاربری منحصر به فرد انتخاب کرده تا آن را به راحتی به خاطر بسپارید. آدرس ایمیل شما باید آدرسی معتبر باشد زیرا گیت‌هاب برای فعال کردن حساب شما یک ایمیل تأیید برای شما ارسال خواهد کرد.

### مرحله سوم) آدرس ایمیل خود را وارد کنید

پس از وارد کردن مشخصات خود، گیت‌هاب برای تأیید حساب، ایمیلی برای شما ارسال می‌کند. صندوق ورودی خود را بررسی و روی لینک تأیید ارائه شده در ایمیل کلیک کنید. اگر ایمیل را دریافت نکردید، پوشه اسپم را بررسی کنید.

### مرحله چهارم) انتخاب طرح یا نقشه

هنگامی که آدرس ایمیل خود را تأیید کردید، از شما خواسته می‌شود یک طرح را انتخاب کنید. گیت‌هاب سه طرح را ارائه می‌دهد که شامل گزینه‌های رایگان، تیمی و سازمانی است. طرح رایگان برای توسعه‌دهندگانی مناسب است که می‌خواهند از گیت‌هاب برای پروژه‌های شخصی خود استفاده کنند. طرح تیمی نیز برای تیم‌هایی است که می‌خواهند در پروژه‌ها همکاری کنند، در حالی که طرح سازمانی برای سازمان‌های بزرگ با الزامات پیچیده است.

### مرحله پنجم) ساخت حساب کاربری خود را تکمیل کنید

پس از انتخاب یک طرح، به داشبورد حساب خود در گیت‌هاب منتقل خواهید شد. در اینجا، می‌توانید با افزودن عکس نمایه، توضیحاتی راجب خود و سایر جزئیات، حساب کاربری خود را سفارشی کنید. تکمیل نمایه شما به توسعه‌دهندگان دیگر کمک می‌کند شما را پیدا کنند و در پروژه‌ها با شما همکاری کنند.

اکنون مراحل ساخت اکانت در گیت‌هاب به پایان رسید و در ادامه به راحتی می‌توانید ایمیلی را که برای ساخت اکانت خود استفاده کردید، در پلتفرم گیت نیز وارد کنید.

# **فصل هفتم**

# **Html & Css**



دقت داشته باشید این فصل تخصصی مربوط به فرانت کار می باشد. اما یک بکند کار نیز باید با این مبحث آشنایی داشته باشد.

## ۷-۱- ساختار صفحات وب

### HTML آموزش مقدمات

مخفف عبارت html Hyper Text Markup Language می باشد که در ظاهر گستردگی ترین زبان استفاده شده در صفحات وب می باشد.

این آموزش برای آشنا کردن طراحان و توسعه دهندگان وب با نیاز برای درک HTML با رئیسیت کافی به همراه یک نظر کلی ساده و مثال های عملی، طراحی شده است. این آموزش به شما محتوای کافی برای شروع با HTML از چاچی که می توانید تخصصی با سطح بالاتر داشته باشید، ارائه می دهد.

اشارة به روشهای دارد که در آن صفحات وب (Dakimont های HTML) به هم لینک می شوند. بنابراین لینک موچود روی یک صفحه ای وب Hypertext نامیده می شود. همانطور که از اسم آن پیداست HTML یک Markup language ( زبان نشانه گذاری ) می باشد، که بدین معناست که شما می توانید از HTML برای نشانه گذاری متن Dakimont استفاده کنید با برچسب هایی که به مرورگر چگونگی ساخت آن را برای نمایش بیان می کنند. در واقع HTML به نیت تعریف ساختار Dakimont هایی مانند تیترها، پاراگراف ها، لیست ها و غیره توسعه پیدا کرد تا اشتراک گذاری اطلاعات علمی بین محققان را آسان تر کند. اکنون HTML به طور گستردگی مورد استفاده قرار می گیرد تا صفحات وب را به کمک برچسب های موچود در زبان HTML تنظیم کند.

### Dakimont پایه‌ی HTML

```
<!DOCTYPE html>
<html>
<head>
<title>This is document title</title>
</head>
```

```
<body>
<h1>This is a heading</h1>
<p>Document content goes here </p>
</body>
</html>
```

برای چک کردن نتیجه این کد HTML هم می توانید با استفاده از ویرایشگر مورد علاقه خود، آن را در یک فایل HTML و با نام test.htm ذخیره کنید. در انتها آن را با استفاده از یک مرورگر مانند اینترنت اکسپلورر یا گوگل کروم یا فایرفاکس و غیره باز کنید.

## ۲-۷- تگ‌های مهم Html

### تگ‌های HTML

همانطور که قبلاً گفته شد HTML یک زبان نشانه گذاری است و برای طراحی محتوا از برچسب‌های مختلفی استفاده می‌کند. این برچسب‌ها در داخل علامت <> قرار گرفته اند، به این شکل < نام برچسب >. به از تعداد کمی از برچسب‌ها، بسیاری از آنها برچسب متناظر بستن را نیز دارند. به عنوان مثال <html> دارای برچسب بستن </html> و <body> دارای برچسب بستن </body> می‌باشد.

Tag	Description
<!DOCTYPE...>	این برچسب نوع داکیومنت و وزن HTML را تعریف می‌کند
<html>	این برچسب داکیومنت کامل HTML را احاطه می‌کند و اساساً شامل تیتر داکیومنت می‌شود که توسط <head>...</head> نمایش داده می‌شود و بدنه داکیومنت نیز به وسیله برچسب‌های <body>...</body> نمایش داده می‌شوند.
<head>	این برچسب تیتر داکیومنتی را نشان می‌دهد که می‌تواند دیگر برچسب‌های HTML از مانند <title>, <link> را در خود نگه دارد.

<title>	برچسب <title> در داخل برچسب <head> استفاده می شود تا تیتر داکیومنت را ذکر کند.
<body>	ین برچسب بدنہ ی داکیومنتی را نشان می دهد که دیگر برچسب های HTML مانند <h1>, <div>, <p> را در خود دارد.
<h1>	ین برچسب تیتر را نمایش می دهد.
<p>	ین برچسب یک پاراگراف را نمایش می دهد.

برای یادگیری HTML لازم است برچسب های مختلفی را مطالعه کنید و درک کنید که در هنگام طراحی یک داکیومنت متنی چگونه رفتار می کنند. یادگیری HTML ساده است چرا که یوزرها باید مورد استفاده ای برچسب های مختلف را برای طراحی متن و یا تصاویر و ایجاد یک صفحه وب زیباتر یاد بگیرند.

#### <DOCTYPE>

برچسب اطلاعیه ی <!DOCTYPE> توسط مرورگرهای وب برای پی بردن به ورژن HTML استفاده شده در داکیومنت مورد استفاده قرار می گیرد. ورژن حال حاضر HTML ورژن 1 می باشد که از اطلاعیه ی زیر استفاده می کند

#### <!DOCTYPE html>

انواع دیگری اطلاعیه و شود دارند که می توانند در داکیومنت HTML، متناسب با ورژن HTML مورد استفاده قرار بگیرند. یزئیات بیشتری در این مورد در زمان بحث در مورد برچسب های دیگر HTML فرا خواهیم گرفت.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Heading Example </title>
```

```
</head>
```

```
<body>
```

```
<h1>This is heading 1 </h1>
```

```
<h2>This is heading 2 </h2>
<h3>This is heading 3 </h3>
<h4>This is heading 4 </h4>
<h5>This is heading 5 </h5>
<h6>This is heading 6 </h6>
</body>
</html>
```

این مثال نتیجه‌ی زیر را به دنبال دارد:

# This is heading 1

## This is heading 2

### This is heading 3

#### This is heading 4

##### This is heading 5

###### This is heading 6

## برچسب پاراگراف

برچسب `p` به روش طراحی متن شما در پاراگراف‌های مختلف اشاره دارد. هر پاراگراف متن باید بین برچسب بازکننده‌ی `<p>` و بستن `</p>` قرار بگیرد، همانطور که در مثال زیر مشاهده می‌کنید

```
<!DOCTYPE html>
<html>
<head>
<title>Paragraph Example </title>
</head>
<body>
<p>Here is a first paragraph of text. </p>
```

```
<p>Here is a second paragraph of text.</p>
<p>Here is a third paragraph of text.</p>
</body>
</html>
```

### برچسب شکست لینک

هر زمان که شما از `<br>` عنصر استفاده کنید، هر چیزی که آن را دنبال می کند از خط بعد شروع خواهد شد. این برچسب نمونه ای از یک عنصر empty می باشد، زمانی که لازم نیست برچسبی را باز کنید یا ببندید چرا که چیزی برای رفتن بین آنها و یو د ندارد.

در برچسب `<br>`، یک فضای خالی بین کاکترهای `br` و اسلش یلوی آن و یو دارد. اگر شما این فضا را حذف کنید، مروگرهای قدیمی تر در ایرانی خط شکست مشکل خواهند داشت، در حالیکه اگر اسلش را حذف کنید برچسب `<br>` باقیمانده در HTML معتبر نمی باشد.

```
<!DOCTYPE html>
<html>
<head>
<title>Line Break Example </title>
</head>
<body>
<p>
Hello <br />
You delivered your assignment on time. <br /> Thanks <br />
Mahnaz
</p>
</body>
</html>
```

Hello

You delivered your assignment ontime. Thanks

Mahnaz

### مرکز متن

می توانید با استفاده از برچسب <center> می توانید هر محتوایی را در مرکز صفحه یا در مرکز هر سلول از یک ۀدول قرار دهید.

```
<!DOCTYPE html>
<html>
<head>
<title>Centring Content Example </title>
</head>
<body>
<p>This text is not in the center.</p>
<center>
<p>This text is in the center.</p>
</center>
</body>
</html>
```

مثال بالا نتیجه ی زیر را تولید خواهد کرد

This text is not in the center.

This text is in the center.

### خطوط افقی

خطوط افقی برای بخش های شکست بصری یک داکیومنت استفاده می شوند. برچسب `hr` > خطی از موقعیت کنونی داکیومنت به حاشیه‌ی سمت راست ایجاد کرده و خط را طبق آن می‌شکند.

به عنوان مثال ممکن است تمایل داشته باشید بین دو پاراگراف خطی قرار دهید، همانطور که در مثال زیر ارائه شده است.

```
<!DOCTYPE html>
<html>
<head>
<title>Horizontal Line Example </title>
</head>
<body>
<p>This is paragraph one and should be on top </p>
<hr />
<p>This is paragraph two and should be at bottom </p>
</body>
</html>
```

این مثال نتیجه زیر را تولید خواهد کرد

This is paragraph one and should be on top

در عنصر `<hr>` یک فضای خالی بین کاراکترهای `hr` و اسلش مقابل آن و یود دارد. اگر این فضا را حذف کنید مورگرهای قدیمی تر در این خط افقی مشکل خواهند داشت. در حالیکه اگر اسلش مقابل آن را حذف کنید عنصر باقیمانده `<hr>` می‌باشد که در HTML فاقد اعتبار می‌باشد.

## حفظ طراحی

گاهی اوقات تمایل دارید که متن فرمت دقیق خود در HTML را دنبال کند، در این موارد می‌توانید از برچسب پریفرمت `<pre>` استفاده کنید. هر متن بین برچسب باز کننده‌ی `<pre>` و برچسب بستن `</pre>` طراحی متن منع را حفظ خواهد کرد.

```
<!DOCTYPE html>
<html>
<head>
<title>Preserve Formatting Example </title>
</head>
<body>
<pre>
function testFunction( strText ){ alert (strText)
}
</pre>
</body>
</html>
```

مثال بالا نتیجه زیر را می دهد

```
function testFunction( strText ){ alert
(strText)
```

## آموزش عناصر در HTML

یک عنصر HTML توسط یک برچسب شروع کننده تعریف می شود. اگر عنصر دارای محتوای دیگری باشد، با یک برچسب بسته کننده تمام می شود در حالیکه یک علامت لغوی نام عنصر یک علامت اسلش قرار گرفته است که می توانید در چندول زیر برخی از این برچسب ها را نشان می دهد.

Start Tag	Content	End Tag
<p>	این برچسب محتوای پاراگراف می باشد.	</p>
<h1>	این برچسب تیتر محتوا می باشد.	</h1>
<div>	این برچسب تقسیم محتوا می باشد.	</div>

بنابراین در اینجا <p>...</p> و <h1>...</h1> عناصر دیگری از HTML می باشد.

عناصری از HTML و void دارند که نیازی به بسته شدن ندارند مانند <img.../>, <hr />, <br />. این عناصر با عنوان void elements (عناصر خالی) شناخته می‌شوند. داکیومنت‌های HTML دارای درختی از این عناصر می‌باشند و مشخص می‌کنند که چگونه داکیومنت‌ها باید ساخته شوند و چه نوع محتوایی باید در چه بخشی از داکیومنت HTML قرار بگیرد.

### برچسب HTML در مقابل عنصر

یک عنصر HTML به وسیله‌ی یک برچسب شروع کننده تعریف می‌شود. اگر عنصر دارای محتوای دیگری باشد با یک برچسب بستن تمام می‌شود. برای مثال <p> برچسب شروع کننده‌ی یک پاراگراف می‌باشد و </p> برچسب بستن همان پاراگراف می‌باشد، اما <p>This is paragraph</p> عنصر یک پاراگراف می‌باشد.

### عناصر تو در توی HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Nested Elements Example </title>
</head>
<body>
<h1>This is <i>italic</i> heading</h1>
<p>This is <u>underlined</u> paragraph </p>
</body>
</html>
```

مثال بالا نتیجه زیر را دارد

This is *italic* heading

This is underlined paragraph

### ویژگی‌های زبان HTML

برخی از برچسب‌های HTML مانند برچسب‌های تیتر و برچسب‌های پاراگراف، و موارد استفاده‌ی آنها را مشاهده کردیم. تاکنون از آنها به ساده‌ترین شکل خود استفاده کرده‌ایم،

اما بیشتر برچسب های HTML می توانند ویژگی هایی داشته باشند که مقداری اطلاعات اضافه می باشد. یک attribute برای تعریف ویژگی های عنصر HTML استفاده می شود و در داخل برچسب بازگننده عنصر قرار می گیرد. همه ویژگی ها از دو بخش تشکیل شده اند و name value ویژگی مورد نظر شما برای تنظیم میباشد، به عنوان مثال عنصر پاراگراف p > در مثال ارائه شده دارای ویژگی می باشد که نام آن align می باشد و شما می توانید از آن برای تنظیم پاراگراف در صفحه استفاده کنید. همان است که شما می خواهید مقدار ویژگی تنظیم شود و همیشه در داخل گیومه قرار می دهید. مثال زیر سه مقدار ممکن از یک ویژگی تراز را نشان می دهد چپ، مرکز و راست.

```
<!DOCTYPE html>
<html>
<head>
<title>Align Attribute Example</title>
</head>
<body>
<p align="left">This is left aligned</p>
<p align="center">This is center aligned</p>
<p align="right">This is right aligned</p>
</body>
</html>
```

چهار ویژگی اصلی که می تواند در همه عناصر بکار گرفته شود:

id title class style

### ویژگی id

ویژگی id یک برچسب HTML می تواند برای تشخیص یک عنصر در یک صفحه ی HTML مورد استفاده قرار بگیرد. دو دلیل اصلی برای تمایل شما به استفاده از ویژگی id در یک عنصر و یود دارد. اگر یک عنصر یک ویژگی را به عنوان تشخیص دهنده ی منحصر به فرد استفاده می کند شناخت تنها آن عنصر و محتوای مربوط به آن ممکن می باشد. اگر دارای دو عنصر هم نام در یک صفحه ی وب می باشید، می توانید از عنصر id برای تشخیص این عناصر هم

نام استفاده کنید. در مورد طراحی صفحه در آموزش‌های مجزا بحث خواهیم کرد، اکنون اجازه بدھید از ویژگی id برای تشخیص عناصر بین دو پاراگراف استفاده کنیم، مانند مثال زیر:

```
<p id="html">This para explains what is HTML</p>
<p id="css">This para explains what is Cascading Style Sheet</p>
```

### ویژگی title

این ویژگی یک تیتر پیشنهادی برای عنصر ارائه می‌دهد. ترکیب مربوط به ویژگی title شبیه به ترکیب توضیح داده شده برای ویژگی id می‌باشد. رفتار این ویژگی بستگی به عنصری دارد که آن را حمل می‌کند، گرچه اغلب اوقات وقتی مکان نما روی عنصر قرار می‌گیرد یا عنصر در حال بارگذاری می‌باشد، با عنوان یک راهنمای ابزار (tooltip) نمایش داده می‌شود.

```
<!DOCTYPE html>
<html>
<head>
<title>The title Attribute Example</title>
</head>
<body>
<h3 title="Hello HTML!">Titled Heading Tag Example</h3>
</body>
</html>
```

### ویژگی Class

این ویژگی برای برقراری ارتباط بین یک عنصر با یک صفحه‌ی طراحی استفاده می‌شود و گروه عنصر را مشخص می‌کند. وقتی که را یاد بگیرید (CSS) در Cascading Style Sheet مورد این ویژگی بیشتر فراخواهید گرفت. اکنون تا همین حد کافیست. مقدار ویژگی نیز ممکن است لیستی از فضاهای مجزای نام‌های گروه باشد، برای مثال

```
class="className1 className2 className3"
```

### ویژگی style

این ویژگی به شما اجازه می‌دهد تا قوانین CSS را در داخل عنصر مشخص کنید.

```
<!DOCTYPE html>
<html>
<head>
<title>The style Attribute</title>
</head>
<body>
<p style="font-family:arial; color:#FF0000;">Some text...</p>
</body>
</html>
```

### دروني کردن ويزگي ها

سه ويزگي درونی و سود دارند که برای اکثر عناصر HTML استفاده می شوند

dir lang

xmllang

### dir ويزگي

اين ويزگي به شما اجازه می دهد تا مسیری را به مرورگر نشان دهيد که متن در آن باید  
کريان داشته باشد. ويزگي dir می تواند يكی از دو مقدار باشد، همانطور که در بدول زير  
نشان داده شده است.

Value	Meaning
ltr	از چپ به راست ( مقدار پيش فرض )
rtl	راست به چپ ( برای زبان هايى مانند هبرو یا عربي که از راست به چپ خوانده می شوند ).

(مثال)

```
<!DOCTYPE html>
<html dir="rtl">
<head>
<title>Display Directions</title>
```

```
</head>  
<body>  
This is how IE 5 renders right-to-left directed text.  
</body>  
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

This is how IE 5 renders right-to-left directed text.

وقتی که ویژگی dir در داخل برچسب <html> استفاده می‌شود، مشخص می‌کند که چگونه متن در کل داکیومنت نمایش داده می‌شود. وقتی در برچسب دیگری مورد استفاده قرار بگیرد، مسیر متن را برای محتوای مربوط به آن برچسب کنترل می‌کند.

ویژگی lang این ویژگی به شما کمک می‌کند تا زبان اصلی استفاده شده در یک داکیومنت را نشان دهید، اما این ویژگی فقط برای سازگاری معکوس با ورژن‌های قدیمی تر HTML در HTML حفظ شده است. این ویژگی به وسیله‌ی ویژگی xml:lang در داکیومنت‌های ئدید HTML ئایگزین شده است.

مقادیر ویژگی lang کدهای دو کاراکتری زبان استاندارد ISO-639 می‌باشند. Codes ISO 639 HTML Language را برای لیست کاملی از کدهای زبان چک کنید.

## مثال

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<title>English Language Page</title>  
</head>  
<body>  
This page is using English Language  
</body>  
</html>
```

## ویژگی xmlLang

این ویژگی ؎ایگزین XHTML برای ویژگی lang می باشد. مقدار ویژگی xmlLang باید یک کد ISO-639 می باشد، همانطور که در بخش قبل بیان شد.

ویژگی های عمومیدر اینجا ؎دولی را می بینید که برخی دیگر از ویژگی هایی که در برچسب HTML مفید می باشند، را نشان می دهد.

Attribute	Options	Function
align	راست، چپ، مرکز	برچسب ها را به طور افقی تنظیم می کند.
valign	بالا، وسط، پایین.	برچسب ها را به طور عمودی در یک عنصر HTML تنظیم می کند.
bgcolor	numeric, hexadecimal, RGB values	پشت یک عنصر یک رنگ پس زمینه قرار می دهد.
background	URL	پشت یک عنصر یک تصویر پس زمینه قرار می دهد.
id	User Defined	عنصری را برای استفاده با Cascading Style Sheets (CSS) نام می برد.
class	User Defined	عنصری را برای استفاده با Cascading Style Sheets طبقه بندی می کند.
width	Numeric Value	عرض ؎دول ها، تصاویر و یا سلول های ؎دول ها را مشخص می کند.
height	Numeric Value	طول ؎دول ها، تصاویر و یا سلول های ؎دول ها را مشخص می کند.
title	User Defined	تیتر عناصر را پاپ آپ می کند.

### ۳-۷- طراحی ظاهر با CSS

#### قالب بندی صفحات HTML

اگر با پردازشگر word کار می کنید باید با بولد کردن (bold)، ایتالیک کردن و آندرلاین کردن متن آشنا می شویم. این ها فقط سه گزینه از ده گزینه موقود برای چگونگی ظاهر شدن متن باشند.

بولد کردن متن

هر چیزی که بین عناصر **<...>** قرار می گیرد به صورت بولد ظاهر می شود، مانند مثال زیر

```
<!DOCTYPE html>
<html>
<head><title>Bold Text Example</title>
</head>
<body>
<p>The following word uses a <b>bold</b> typeface.</p>
</body>
</html>
```

مثال بالا نتیجه زیر را در بر خواهد داشت

The following word uses a bold typeface.

ایتالیک کردن متن

هر چیزی که بین عناصر *<...>* می باشد

قرار بگیرد به صورت ایتالیک ظاهر می شود، مانند مثال زیر

```
<!DOCTYPE html>
<html>
<head>
<title>Italic Text Example </title>
```

```
</head>
<body>
<p>The following word uses a <i>italicized</i> typeface.</p>
</body>
</html>
```

مثال بالا نتیجه زیر را تولید خواهد کرد

The following word uses a *italicized* typeface.

### آندرلاین کردن متن

هر چیزی که بین عناصر `<u>...</u>` قرار بگیرد به صورت آندرلاین ظاهر می شود، مانند  
مثال زیر

```
<!DOCTYPE html>
<html>
<head>
<title>Underlined Text Example</title>
</head>
<body>
<p>The following word uses a <u>underlined</u> typeface.</p>
</body>
</html>
```

مثال بالا نتیجه‌ی زیر را تولید خواهد کرد

The following word uses a underlined typeface.

### خط کشیدن روی متن

هر چیزی که بین گزینه‌های `<del>...</del>` قرار بگیرد، با یک `strikethrough` نمایش داده می شود که خط باریکی می باشد که روی متن کشیده می شود، همانطور که در مثال زیر مشاهده می کنید

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>Strike Text Example</title>
</head>
<body>
<p>The following word uses a <strike>strikethrough</strike> typeface.</p>
</body>
</html>
```

مثال بالا نتیجه زیر را تولید خواهد کرد

The following word uses a ~~strikethrough~~ typeface.

monospaced

محتوای عنصر `<tt>...</tt>` به فونت monospaced نوشته می شود. اکثر فونت ها با عنوان فونت هایی با عرض متغیر شناخته شده اند، زیرا حروف مختلف دارای عرضهای مختلف هستند (به عنوان مثال حرف m عریض تر از حرف I می باشد). به هر حال در فونت monospaced تمام حروف دارای عرض یکسان می باشند.

مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Monospaced Font Example</title>
</head>
<body>
<p>The following word uses a <tt>monospaced</tt> typeface.</p>
</body>
</html>
```

The following word uses a monospaced typeface.

متن (چاپ شده در بالا) **superscript**

محتوای عنصر `<sup>...</sup>` در بالا نوشته می شود، فونت استفاده شده برای آن همان فونت کarakترهای اطراف آن می باشد، اما به اندازه‌ی نصف یک کarakتر بالاتر از دیگر کarakترها نمایش داده می شود.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Superscript Text Example</title>
</head>
<body>
<p>The following word uses a <sup>superscript</sup> typeface.</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

The following word uses a <sup>superscript</sup> typeface.

### من (چاپ شده در زیر) **subscript**

محتوای عنصر `<sub>...</sub>` در زیر نوشته می شود. اندازه‌ی فونت استفاده شده برای آن به اندازه‌ی فونت کarakترهای اطراف می باشد اما به اندازه‌ی نصف ارتفاع یک کarakتر زیر کarakترهای دیگر نمایش داده می شود.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Subscript Text Example</title>
</head>
<body>
```

```
<p>The following word uses a <sub>subscript</sub> typeface.</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

The following word uses a <sub>subscript</sub> typeface.

### متن مندرج

هرچیزی که بین عنصر ... قرار بگیرد به عنوان متن مندرج نمایش داده می‌شود.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Inserted Text Example</title>
</head>
<body>
<p>I want to drink <del>cola</del> <ins>wine</ins></p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

I want to drink ~~cola~~ wine

متن حذف شده

هرچیزی که بین عنصر ~~...~~ ظاهر شود با عنوان یک متن حذف شده نمایش داده می‌شود.

### مثال

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Deleted Text Example</title>
</head>
<body>
<p>I want to drink <del>cola</del> <ins>wine</ins></p>
</body>
</html>
```

محتوای عنصر `<big>...</big>` اندازه فونت را بزرگتر از متن اطراف نشان می دهد. مانند مثال زیر

مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Larger Text Example</title>
</head>
<body>
<p>The following word uses a <big>big</big> typeface.</p>
</body>
</html>
```

این مثال نتیجه زیر را تولید می کند

The following word uses a big typeface.

متن کوچکتر

محتوای عنصر `<small>...</small>` متن را یک سایز کوچکتر از متن اطراف آن نشان می دهد، مانند مثال زیر

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Smaller Text Example</title>
</head>
<body>
<p>The following word uses a <small>small</small> typeface.</p>
</body>
</html>
```

این مثال نتیجه زیر را تولید خواهد کرد

The following word uses a small typeface.

### گروه بندی محتوا

عناصر `<div>` و `<span>` به شما اجازه می دهند تا برای ایجاد بخش ها و یا زیرمجموعه های یک صفحه، عناصر را گروه بندی کنید.

برای مثال ممکن است تمایل داشته باشید که تمام پاورقی ها را در یک صفحه در داخل عنصر `<div>` قرار دهید تا نشان دهید که تمام عناصر موضوع در آن عنصر مربوط به پاورقی می باشند. پس از آن ممکن است طرحی را به عنصر `<div>` ضمیمه کنید، طوریکه با استفاده از مجموعه ای از قوانین طراحی ظاهر شوند.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Div Tag Example</title>
</head>
<body>
<div id="menu" align="middle">
<a href="/index.htm">HOME</a> |
<a href="/about/contact_us.htm">CONTACT</a> |
<a href="/about/index.htm">ABOUT</a>
</div>
```

```
<div id="content" align="left" bgcolor="white">  
<h5>Content Articles</h5>  
<p>Actual content goes here</p>  
</div>  
</body>  
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

[HOME](#) | [CONTACT](#) | [ABOUT](#)

## CONTENT ARTICLES

Actual content goes here.....

از طرف دیگر عنصر `<span>` فقط می‌تواند برای گروه بندی داخل خطی عناصر استفاده شود.  
بنابراین اگر بخشی از جمله یا پاراگراف را دارید که می‌خواهید با هم در یک گروه قرار دهید،  
می‌توانید از عنصر `<span>` مانند زیر استفاده کنید

## مثال

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Span Tag Example</title>  
</head>  
<body>  
<p>This is the example of <span style="color:green">span tag</span> and the <span  
style="color:red">div tag</span> alongwith CSS</p>  
</body>  
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

This is the example of span tag and the div tag alongwith CSS

این برچسب ها عموماً با CSS استفاده می شوند تا به شما اجازه دهنند طرحی را به بخشی از یک صفحه ضمیمه کنید.

## آموزش متا تگ در HTML

علاوه بر مشخص کردن اطلاعات مهم به روش های مختلف در مورد یک داکیومنت، به شما اجازه می دهد تا متادیتا را نیز مشخص کنید. عناصر META می توانند برای وارد کردن چفت مقدار/نام استفاده شوند، این چفت ویژگی های داکیومنت HTML مانند نویسنده، تاریخ اتمام، لیست کلمات کلیدی، داکیومنت نویسنده و غیره را ارائه میدهد.

برچسب `<meta>` برای ارائه چنین اطلاعات اضافه ای استفاده می شود. این برچسب یک عنصر خالیست و دارای برچسب بستن نیست اما اطلاعاتی را با ویژگی آن در خود دارد. شما می توانید بر اساس اطلاعاتی که می خواهید در داکیومنت خود نگهداری کنید، یک برچسب یا بیشتر از یک برچسب متا وارد داکیومنت خود کنید. اما به طور کل برچسب های متا وضعیت ظاهری داکیومنت را تحت تاثیر قرار نمی دهند، بنابراین از لحاظ ظاهری استفاده کردن یا نکردن از آنها مشخص نمی شود.

### افزودن برچسب های متا به داکیومنت ها

شما می توانید با قرار دادن برچسب های `<meta>` در داخل تیتر داکیومنت که با برچسب های `<head>` مشخص می شوند، متابدیتا را به صفحات وب خود اضافه کنید. یک برچسب متا علاوه بر ویژگی های مرکزی، می تواند دارای ویژگی های زیر نیز باشد

Attribute	Description
Name	نام برای پراپرتی، می تواند هر چیزی از جمله کلمات کلیدی، توصیفات، نویسنده، تولید کننده و غیره باشد.
content	مقدار ویژگی را مشخص می کند.
Scheme	نموداری را برای توضیح مقدار ویژگی مشخص می کند. (همانطور که در محتوى ویژگی مشاهده کردید..)

http-equiv	برای تیترهای پیام پاسخ http استفاده می شود. برای مثال http-equiv می تواند برای تازه سازی صفحات یا تنظیم یک cookie استفاده شود. مقداری مانند نوع محتوا، اتمام، تازه سازی و تنظیم cookie.
------------	---

### مشخص کردن کلمات کلیدی

می توانید از برچسب <meta> برای مشخص کردن کلمات کلیدی مربوط به داکیومنت استفاده کنید، و پس از آن این کلمات توسط موتورهای بستجو استفاده می شوند، و صفحه ی وب شما را به هدف بستجو ایندکس می کنند.

در این مثال برچسب های متا، متادیتا و HTML را با عنوان کلمات کلیدی در مورد داکیومنت وارد می کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

این مثال نتیجه زیر را به دنبال دارد

Hello HTML5!

### توصیف داکیومنت

شما می توانید از برچسب <meta> برای توصیف داکیومنت استفاده کنید. این برچسب نیز می تواند توسط موتورهای مختلف بستجو مورد استفاده قرار بگیرد، در حالیکه صفحه ی وب شما را به هدف بستجو ایندکس می کند.

**مثال**

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

**بازبینی تاریخ داکیومنت**

شما می‌توانید از برچسب `<meta>` برای ارائه اطلاعات در مورد زمان آخرین آپدیت داکیومنت استفاده کنید. این اطلاعات می‌توانند توسط مرورگرهای مختلفی استفاده شوند، در حالیکه صفحه‌ی وب شما را تازه سازی می‌کنند.

**مثال**

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Tahlildadeh, 3/7/2014" />
</head>
<body>
<p>Hello HTML5!</p>
```

```
</body>  
</html>
```

## تازه سازی داکیومنت

یک برچسب `<meta>` می تواند برای مشخص کردن دوره ای که پس از آن صفحه ی وب شما به طور خودکار بازسازی می شود، استفاده شود.

اگر می خواهید صفحه ی وب شما پس از هر 1 ثانیه ریفرش شود، از ترکیب زیر استفاده کنید

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Meta Tags Example</title>  
<meta name="keywords" content="HTML, Meta Tags, Metadata" />  
<meta name="description" content="Learning about Meta Tags." />  
<meta name="revised" content="Tahlildadeh, 3/7/2014" />  
<meta http-equiv="refresh" content="5" />  
</head>  
<body>  
<p>Hello HTML5!</p>  
</body>  
</html>
```

## صفحه کردن Redirect

می توانید از برچسب `<meta>` برای Redirect کردن صفحه ی خود استفاده کنید، همچنین می توانید دوره ای را مشخص کنید که پس از آن صفحه به طور خودکار Redirect شود.

در این مثال صفحه ی ۶اری پس از 1 ثانیه به صفحه ی دیگر Redirect می شود. اگر می خواهید صفحه فورا Redirect شود، هیچ محتوایی برای آن مشخص نکنید.

```
<!DOCTYPE html>  
<html>
```

```
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Tahlildadeh, 3/7/2014" />
<meta http-equiv="refresh" content="5" url=http://www.tahlildadeh.com" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

## آموزش cookies

cookies داده هایی هستند که در یک فایل کوچک متن روی کامپیوتر شما ذخیره شده اند و بین مرورگر وب و سرور وب ردی بدل می شود تا مسیر اطلاعات مختلف را براساس نیاز برنامه ی وب شما حفظ کنند.

شما می توانید از برچسب `<meta>` برای ذخیره ی cookies در بخش کاربری استفاده کنید و پس از آن این اطلاعات می توانند توسط سرور وب استفاده شوند تا بازدیدکننده ی سایت را پیگیری کنند.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="revised" content="Tahlildadeh, 3/7/2014" />
```

```
<meta http-equiv="cookie" content="userid=xyz; expires=Wednesday, 08-Aug-15  
235959 GMT;" />  
</head>  
<body>  
<p>Hello HTML5!</p>  
</body>  
</html>
```

اگر تاریخ و زمان انقضا را مشخص نکرده اید، cookie یک session cookie می باشد و وقتی که یوزر از مرورگر خارج شود، پاک خواهد شد.

### تنظیم نام نگارنده

می توانید با استفاده از meta tag نام یک نگارنده را برای صفحه‌ی وب خود تنظیم کنید. یک مثال در این رابطه را در زیر مشاهده می کنید.

### مثال

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Meta Tags Example</title>  
<meta name="keywords" content="HTML, Meta Tags, Metadata" />  
<meta name="description" content="Learning about Meta Tags." />  
<meta name="author" content="Mahnaz Mohtashim" />  
</head>  
<body>  
<p>Hello HTML5!</p>  
</body>  
</html>
```

### تعیین تنظیم کاراکتر

می توانید از برچسب `<meta>` برای تعیین تنظیم کاراکتر مربوط به صفحه‌ی وب استفاده کنید.

## مثال

به طور پیش فرض مرورگرها و سرورهای وب از رمزگزاری ISO-8859-1 برای پردازش صفحات وب استفاده می کنند. در زیر مثالی را می بینید برای تنظیم رمزگزاری UTF-8

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="Mahnaz Mohtashim" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

برای ارائه کاراکترهای چینی سنتی به یک صفحه استاتیک، صفحه وب باید دارای یک برچسب `<meta>` باشد تا رمزگذاری Big5 را تنظیم کند.

```
<!DOCTYPE html>
<html>
<head>
<title>Meta Tags Example</title>
<meta name="keywords" content="HTML, Meta Tags, Metadata" />
<meta name="description" content="Learning about Meta Tags." />
<meta name="author" content="Mahnaz Mohtashim" />
```

```
<meta http-equiv="Content-Type" content="text/html; charset=Big5" />
</head>
<body>
<p>Hello HTML5!</p>
</body>
</html>
```

### آموزش استفاده از **comment** در html

کامنت قطعه ای از کد می باشد که توسط مرورگرها نادیده گرفته می شود. افرودن کامنت به کد HTML خود، تمرین خوبی است، به ویژه در داکیومنت های پیچیده برای نشان دادن بخش هایی از یک داکیومنت و یادداشت هایی دیگر به هر کس که کد را نگاه می کند. کامنت ها به شما و دیگران کمک می کنند تا کد خود را متوجه شوید و قابلیت خواندن آن را افزایش می دهد.

کامنت های HTML بین برچسب های ...--> واقع می شوند. بنابراین هر محتوایی که بین این برچسب ها واقع شود، مثل کامنت با آن رفتار خواهد شد و توسط مرورگرها به طور کامل نادیده گرفته خواهد شد.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<!-- Document Header Starts -->
<title>This is document title</title>
</head><!-- Document Header Ends -->
<body>
<p>Document content goes here </p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را بدون نمایش محتوای ارائه شده به عنوان بخشی از کامنت‌ها، به دنبال دارد.

Document content goes here.....

### کامنت‌های معتبر در مقابل کامنت‌های نامعتبر

کامنت‌ها تودرتو نمی‌شوند، یعنی اینکه یک کامنت نمی‌تواند در داخل کامنت دیگری قرار بگیرد. خط تیره‌ی دوتایی "--" ممکن نیست در داخل یک کامنت ظاهر شود، به از در مواردی که بخشی از برچسب closing باشد. شما باید مطمئن باشید که هیچ فضایی در ابتدای رشته‌ی کامنت و یو'd ندارد.

### مثال

در اینجا کامنت ارائه شده یک کامنت معتبر می‌باشد و توسط مرورگر پاک می‌شود.

```
<!DOCTYPE html>
<html>
<head>
<title>Valid Comment Example</title>
</head>
<body>
<!-- This is valid comment -->
<p>Document content goes here </p>
</body>
</html>
```

اما خط زیر یک کامنت معتبر نیست و توسط مرورگر نمایش داده خواهد شد. این به این خاطر است که فضایی بین حاشه‌ی چپ پرانتز و علامت تعجب و یو'd دارد.

### مثال

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Invalid Comment Example</title>
</head>
<body>
<!-- This is not a valid comment -->
<p>Document content goes here </p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را به دنبال خواهد داشت

```
<!-- This is not a valid comment --> Document content goes here.....
```

### کامنت‌های چندخطی

تاکنون فقط کامنت‌های تک خطی را مشاهده کردیم، اما HTML کامنت‌های چندخطی را نیز پشتیبانی می‌کند.

شما می‌توانید کامنت‌های چندخطی داشته باشید با استفاده از برچسب آغازگر `<!--!` و پایان دهنده‌ی `-->` که قبل از اولین خط و در پایان آخرین خط قرار دارد، همانطور که در مثال زیر می‌بینید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Multiline Comments</title>
</head>
<body>
<!--
```

This is a multiline comment and it can span through as many as lines you like.

```
-->
```

```
<p>Document content goes here </p>
</body>
```

```
</html>
```

این مثال نتیجه‌ی زیر را ایجاد خواهد کرد

Document content goes here.....

### کامنت‌های شرطی

کامنت‌های شرطی تنها در اینترنت اکسپلورر روی ویندوز کار می‌کنند، اما توسط مرورگرهای دیگر نادیده گرفته می‌شوند. این کامنت‌ها در اکسپلورر ۱ به بالاتر پشتیبانی می‌شوند و می‌توانید از آنها برای دادن دستورات شرطی به ورزش‌های مختلف IE استفاده کنید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Conditional Comments</title>
<!--[if IE 6]>
Special instructions for IE 6 here
<![endif]-->
</head>
<body>
<p>Document content goes here </p>
</body>
</html>
```

در ؟ایی که نیاز خواهید داشت یک صفحه‌ی طراحی متفاوت بر اساس ورزش‌های مختلف اینترنت اکسپلورر به کار بگیرید، به راه حلی نیاز خواهید داشت. در این موقع چنین کامنت‌های شرطی مفید خواهند بود.

### استفاده از برچسب کامنت

مرورگرهای کمی هستند که استفاده از برچسب `<comment>` را برای کامنت بخشی از کد HTML استفاده می‌کنند.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Using Comment Tag</title>
</head>
<body>
<p>This is <comment>not</comment> Internet Explorer.</p>
</body>
</html>
```

اگر از IE استفاده می کنید، نتیجه‌ی زیر حاصل خواهد شد

This is Internet Explorer.

اما اگر در حال استفاده از IE نمی باشد، نتیجه‌ی زیر حاصل می شود.

This is Internet Explorer.

## کد اسکریپت کامنت

گرچه شما اوا اسکریپت را با HTML یاد خواهید گرفت، اما در یک آموزش مجزا. در اینجا باید دقت کنید که اگر در حال استفاده از javascript یا java script یا vb script در کد HTML خود هستید، بنابراین توصیه می شود که آن کد اسکریپت را در داخل کامنت های مناسب HTML قرار دهید طوریکه مرورگرهای قدیمی بتوانند به درستی کار کنند.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Commenting Script Code</title>
<script>
<!--
```

```
document.write("Hello World!")  
//-->  
</script>  
</head>  
<body>  
<p>Hello , World!</p>  
</body>  
</html>
```

این مثال نتیجه‌ی زیر را در پی خواهد داشت

Hello World!

Hello , World!

### کامنت صفحات طراحی

گرچه شما HTML را در یک آموزش مجزا با صفحات طراحی فرا می‌گیرید، اما در اینجا به باید دقت داشته باشید که اگر از CSS در کد HTML خود استفاده می‌کنید، بنابراین توصیه می‌شود که کد صفحه‌ی طراحی را داخل کامنت‌های مناسب HTML قرار دهید، طوریکه مرورگرهای قدیمی بتوانند کار کنند.

### مثال

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Commenting Style Sheets</title>  
<style>  
<!--  
.example {  
border :1px solid #4a7d49;  
}
```

```
// -->  
</style>  
</head>  
<body>  
<div class="example">Hello , World!</div>  
</body>  
</html>
```

این مثال نتیجه‌ی زیر را تولید می‌کند

```
Hello, World!
```

## وارد کردن تصاویر در صفحات HTML

در این آموزش چگونگی استفاده از تصاویر در صفحات HTML را فرا می‌گیریم. زیبا سازی تصاویر و همچنین ترسیم بسیاری از مفاهیم پیچیده به یک روش ساده روی صفحه‌ی وب شما، بسیار مهم می‌باشد. این آموزش مراحل ساده‌ی استفاده از تصاویر در صفحات وب را به شما آموزش خواهد داد.

### وارد کردن تصویر

شما می‌توانید با استفاده از برچسب هر تصویری را وارد صفحه‌ی وب خود کنید. در زیر ترکیب ساده‌ی استفاده از این برچسب را می‌بینید.

```

```

برچسب یک برچسب خالی می‌باشد، یعنی تنها می‌تواند دارای لیستی از ویژگی‌ها باشد و دارای برچسب closing نمی‌باشد.

مثال برای امتحان کردن مثال، زیر اجا بهید فایل html خود یعنی test.htm و فایل تصویر خود را در یک مسیر قرار دهیم.

```
<!DOCTYPE html>  
<html>  
<head>  
<title>Using Image in Webpage</title>
```

```
</head>
<body>
<p>Simple Image Insert</p>

</body>
</html>
```

این مثال نتیجه‌ی زیر را به دنبال خواهد داشت

### Simple Image Insert

شما می‌توانید از فایل تصویر JPEG، GIF یا PNG متناسب با راحتی خود استفاده کنید، اما مطمئن شوید که در ویژگی src نام فایل تصویر را به درستی وارد کردید. نام تصویر همیشه یک مورد هوشمند می‌باشد. ویژگی alt یک ویژگی می‌باشد که اگر تصویر نمایش داده نشود، یک متن پایگزین را برای آن مشخص می‌کند.

### تنظیم موقعیت تصویر

معمولًا ما تمام تصاویر خود را در یک مسیر مجزا قرار می‌دهیم. بنابراین اجازه بدهید فایل مربوط به HTML را در مسیر اصلی نگه داشته و یک مسیر فرعی images داخل مسیر اصلی، پایی که تصویر test.png را نگهداری می‌کنیم، ایجاد کنیم. با فرض اینکه موقعیت تصویر ما "image/test.png" می‌باشد، مثال زیر را امتحان کنید مثلاً

```
<!DOCTYPE html>
<html>
<head>
<title>Using Image in Webpage</title>
</head>
<body>
<p>Simple Image Insert</p>

</body>
</html>
```

## Simple Image Insert

### تنظیم طول و عرض تصویر

می توانید طول و عرض تصویر را براساس نیاز خود و با استفاده از ویژگی های width و height تنظیم کنید. شما می توانید طول و عرض تصویر را به پیکسل یا براساس درصد اندازه واقعی آن تنظیم کنید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Width and Height</title>
</head>
<body>
<p>Setting image width and height</p>

</body>
</html>
```

Setting image width and height

### تنظیم حاشیه تصویر

به طور پیش فرض تصویر حاشیه ای در اطراف خود خواهد داشت، شما می توانید ضخامت این حاشیه را با استفاده از ویژگی border و به واحد پیکسل تنظیم کنید. ضخامت 1 یعنی هیچ حاشیه ای در اطراف تصویر و بود ندارد.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Border</title>
</head>
<body>
<p>Setting image Border</p>

</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد

Setting image Border

### تنظیم همترازی تصویر

به طور پیش فرض تصویر در سمت چپ صفحه تنظیم می‌شود، اما می‌توانید از ویژگی align برای تنظیم تصویر در سمت راست یا مرکز صفحه استفاده کنید.

مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Set Image Alignment</title>
</head>
<body>
<p>Setting image Alignment</p>

</body>
</html>
```

این مثال نتیجه‌ی زیر را به دنبال دارد

### Setting image Alignment

#### آموزش Table در HTML

جول‌های HTML به نویسنده‌گان وب اجازه می‌دهند تا داده‌هایی مانند متن، تصاویر، لینک‌ها، گدول‌های دیگر و غیره در ردیف‌ها و ستون‌ها تنظیم کنید. گدول‌های HTML با استفاده از برچسب `<table>` ایجاد می‌شوند که در آن برچسب `<tr>` برای ایجاد ردیف‌ها و برچسب `<td>` برای ایجاد داده‌های سلول‌ها استفاده می‌شوند.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Tables</title>
</head>
<body>
<table border="1">
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

در اینجا border یک ویژگی از برچسب <table> می باشد و برای قرار دادن حاشیه در پهنانی همه ی سلول هاستفاده می شود. اگر به حاشیه احتیاج نداشته باشید می توانید از border="0" استفاده کنید.

### تیتر جدول

تیتر جدول می تواند با استفاده از برچسب <th> تعریف شود. این برچسب برای یا یگرینی با برچسب <td> استفاده می شود که برچسب <td> برای نمایش داده ی حقیقی سلول استفاده می شود. طبیعتاً شما ردیف های بالای ۳دول را به عنوان تیتر قرار می دهید.

دو ویژگی به نام های Cellpadding و ۳ود دارند که برای تنظیم محیط سفید در سلول های ۳دول استفاده می شوند. ویژگی cellspacing عرض حاشیه را تعریف می کند، در حالیکه cellpadding فاصله ی بین حاشیه ی سلول و محتوا آن را تعریف می کند.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Cellpadding</title>
</head>
<body>
<table border="1" cellpadding="5" cellspacing="5">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
```

```
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
</body>
</html>
```

اگر بخواهید دو یا بیشتر از دو ستون را با هم تلفیق کنید از ویژگی colspan استفاده می کنید. به طور مشابه اگر بخواهید دو یا بیشتر از دو ردیف را در یک ردیف تلفیق کنید از rowspan استفاده کنید.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Colspan/Rowspan</title>
</head>
<body>
<table border="1">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
</body>
```

</html>

### پس زمینه جدول

می توانید به دو روش زیر برای چندول خود زمینه ای تنظیم کنید.

ویژگی `Bg color` می توانید رنگ زمینه را برای همه چندول و یا تنها برای یک سلول تنظیم کنید. ویژگی `background` می توانید یک تصویر را برای کل چندول یا تنها یک سلول تنظیم کنید.

همچنین می توانید با استفاده از ویژگی `bordercolor` رنگ حاشیه را نیز تنظیم کنید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Background</title>
</head>
<body>
<table border="1" bordercolor="green" bgcolor="yellow">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
</body>
</html>
```

در اینجا مثالی در رابطه با استفاده از ویژگی background می بینید.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Background</title>
</head>
<body>
<table border="1" bordercolor="green" background="/images/test.png">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td><td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
</body>
</html>
```

## طول و عرض جدول

شما می توانید طول و عرض جدولی را با استفاده از ویژگی های width و height تنظیم کنید. شما می توانید طول و عرض جدول را به پیکسل و یا متناسب با درصد صفحه ی یا ری تنظیم کنید.

## مثال

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>HTML Table Width/Height</title>
</head>
<body>
<table border="1" width="400" height="150">
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
</body>
</html>
```

## شرح جدول

برچسب `caption` یک توضیح یا یک تیتر برای جدول می باشد که در بالای جدول نمایش داده می شود. این برچسب در ورژن های قدید HTML/XHTML توصیه می شود.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table Caption</title>
</head>
<body>
<table border="1" width="100%">
```

```
<caption>This is the caption</caption>
<tr>
<td>row 1, column 1</td>
<td>row 1, columnn 2</td>
</tr>
<tr>
<td>row 2, column 1</td>
<td>row 2, columnn 2</td>
</tr>
</table>
</body>
</html>
```

### تیتر، بدن و پاورقی جدول

جدول ها می توانند به سه بخش تقسیم شوند تیتر، بدن و پاورقی جدول. عنوان و فوت (tfoot) در واقع شبیه به تیتر و پاورقی در یک فایل پردازش word می باشند که برای هر صفحه یکی می باشد، در حالیکه بدن همان نگه دارنده ای محتوای اصلی جدول می باشد.

سه عنصر برای مجزا کردن عنوان، بدن و فوت در یک جدول عبارتند از:

<thead> برای ایجاد یک تیتر مجزا

<tbody> برای نشان دادن بدن ای اصلی جدول

<tfoot> برای ایجاد یک پاورقی مجزا در یک جدول

یک جدول برای نشان دادن صفحات و گروه های مختلف یک داده، ممکن است دارای عناصر

<tbody> مختلفی باشد، اما ظاهر شدن برچسب های <thead> و <tfoot> قبل از <tbody> زیاد مهم نیست.

### مثال

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>HTML Table</title>
</head>
<body>
<table border="1" width="100%">
<thead>
<tr>
<td colspan="4">This is the head of the table</td>
</tr>
</thead>
<tfoot>
<tr>
<td colspan="4">This is the foot of the table</td>
</tr>
</tfoot>
<tbody>
<tr>
<td>Cell 1</td>
<td>Cell 2</td>
<td>Cell 3</td>
<td>Cell 4</td>
</tr>
</tbody>
</table>
</body>
</html>
```

شما می توانید از یک `td` دول در داخل `table` دولی دیگر استفاده کنید. نه تنها `td` ها، بلکه می توانید تمام برچسب ها را در داخل برچسب `td` استفاده کنید.

### مثال

در زیر مثالی را از استفاده `td` و برچسب های دیگر در داخل یک سلول مشاهده می کنید.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Table</title>
</head>
<body>
<table border="1" width="100%">
<tr>
<td>
<table border="1" width="100%">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
```

```
</td>
</tr>
</table>
</body>
</html>
```

## آموزش List

سه روش برای مشخص کردن لیست هایی از اطلاعات ارائه می دهد. تمام لیست ها باید شامل یک یا بیشتر از یک عنصر باشند. لیست ها ممکن است شامل موارد زیر شوند:

<ul>- لیست بدون ترتیب. این لیست آیتم ها را با استفاده از bullet های ساده لیست می کند.

<ol> - لیست منظم. این لیست از نمودارهای عددی مختلف برای لیست کردن آیتم های شما استفاده می کند.

<dl>- لیست تعریف. این لیست آیتم های شما را به همان روشهی که در دیکشنری منظم شده اند، منظم می کند.

## لیست های بدون ترتیب

لیست بدون ترتیب مجموعه ای از آیتم های مربوط به هم می باشد که هیچگونه نظم و ترتیب خاصی ندارند. این لیست با استفاده از برچسب <ul> در HTML ایجاد می شود. هر آیتم در لیست با یک bullet مشخص می شود.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
```

```
<ul>
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
</body>
</html>
```

### ویژگی type

می توانید از ویژگی type برای برچسب `<ul>` استفاده کنید تا نوع bullet خود را مشخص کنید، که به طور پیش فرض یک دیسک می باشد. در زیر گزینه های ممکن را مشاهده می کنید.

```
<ul type="square">
<ul type="disc">
<ul type="circle">
```

در زیر مثالی را میبینید که در آن از `<ul type="square">` استفاده می کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Unordered List</title>
</head>
<body>
<ul type="square">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ul>
```

```
</body>  
</html>
```

در زیر مثالی را می بینید که در آن از `<ul type="disc">` استفاده کرده ایم.

```
<!DOCTYPE html>  
<html>  
<head>  
<title>HTML Unordered List</title>  
</head>  
<body>  
<ul type="disc">  
<li>Beetroot</li>  
<li>Ginger</li>  
<li>Potato</li>  
<li>Radish</li>  
</ul>  
</body>  
</html>  
  
<ul type="circle">  
<li>Beetroot</li>  
<li>Ginger</li>  
<li>Potato</li>  
<li>Radish</li>  
</ul>  
</body>  
</html>
```

## لیست های منظم HTML

اگر تمایل دارید آیتم های خود را به ظای قرار دادن در یک لیست دارای bullet، در یک لیست عددگذاری شدهقرار دهید، می توانید از لیست منظم HTML استفاده کنید. این لیست با استفاده از بروچسب <ol> ایجاد میشود. شماره گذاری از یک شروع شده و برای هر لیست منظم عنصر بعدی با اضافه شدن یک عدد و به همراه بروچسب <li> اضافه می شود.

می توانید از ویژگی type برای بروچسب <ol> استفاده کنیم تا نوع شماره گذاریمورد نظر خود را مشخص کنید. به طور پیش فرض شماره گذاری به وسیله ای عدد انجام می شود. در زیر گزینه های ممکن را مشاهده می کنید.

```
<ol type="1">
- Default-Case Numerals.

<ol type="I">
- Upper-Case Numerals.

<ol type="i">
- Lower-Case Numerals.

<ol type="a">
- Lower-Case Letters.

<ol type="A"> - Upper-Case Letters.
```

در زیر مثالی را می بینید که در آن از <ol type="1"> استفاده کرده ایم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="1">
<li>Beetroot</li>
<li>Ginger</li>
```

```
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="I">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

در زیر مثالی را میبینید که در آن از <ol type="i"> استفاده کرده ایم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="i">
<li>Beetroot</li>
```

```
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

در زیر مثالی را می بینید که در آن از <ol type="A"> استفاده کرده ایم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="A">
<li>Beetroot</li>
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>

<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="a">
<li>Beetroot</li>
```

```
<li>Ginger</li>
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

### ویژگی start

شما می توانید از ویژگی start برای برچسب ol استفاده کنید تا نقطه‌ی شروع شماره گذاری خود را مشخص کنید. در زیر گزینه‌های ممکن را مشاهده می‌کنید.

```
<ol type="1" start="4">
- Numerals starts with 4.

<ol type="I" start="4">
- Numerals starts with IV.

<ol type="i" start="4">
- Numerals starts with iv.

<ol type="a" start="4">
- Letters starts with d.

<ol type="A" start="4"> - Letters starts with D.
```

در زیر مثالی را می‌بینید که در آن از <ol type="i" start="4" > استفاده می‌کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Ordered List</title>
</head>
<body>
<ol type="i" start="4">
<li>Beetroot</li>
<li>Ginger</li>
```

```
<li>Potato</li>
<li>Radish</li>
</ol>
</body>
</html>
```

## لیست های تعریف HTML

HTML و XHTML لیست هایی به نام لیست های تعریف را پشتیبانی می کنند که در این لیست ها ورودی ها مانند ترتیب لغات در دیکشنری قرار می گیرند. این لیست یک روش ایده آل برای ارائه یک فهرست از معانی یا لیستی از اصطلاحات یا لیستی از نام ها و مقادیر می باشد.

لیست تعریف از سه برچسب زیر استفاده می کند.

- شروع لیست را تعریف می کند.

- یک عبارت

- تعریف عبارت

- پایان لیست را تعریف می کند.

## مثال

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Definition List</title>
</head>
<body>
<dl>
<dt><b>HTML</b></dt>
<dd>This stands for Hyper Text Markup Language</dd>
<dt><b>HTTP</b></dt>
```

```
<dd>This stands for Hyper Text Transfer Protocol</dd>
</dl>
</body>
</html>
```

## لينك های متن HTML

يک صفحه‌ی وب می‌تواند لينك‌های متنوعی داشته باشد که شما را مستقیماً به صفحات دیگر یا حتی بخش‌هایی خاص از یک صفحه‌ی ارائه شده می‌برد. اين لينك‌ها هايپرلينك ناميده می‌شوند.

هايپرلينك‌ها به بازديدكنندگان اجازه می‌دهند تا با کلیک کردن روی لغات، اصطلاحات و تصاویر بين صفحات وب مسیريابی کنند. شما می‌توانيد هايپرلينك‌ها را روی صفحه‌وب با استفاده از متن یا تصاویر موجود ايجاد کنيد.

## لينك کردن داكيمونت‌ها

يک لينك با استفاده از برچسب `<a>` در HTML تعیین می‌شود. اين برچسب anchor tag ناميده ميشود و هر چيزی بين برچسب آغازين `<a>` و پياناني `</a>` بخشی از لينك می‌شود و يك يوزر می‌تواند آن بخش را کلیک کرده تا به داكيمونت لينك شده برسد. در زير ترکيب ساده‌ی استفاده از برچسب `<a>` را می‌بينيد.

```
<a href="Document URL"... attributes-list>Link Text</a>
```

## مثال

اجازه دهيد مثال زير را امتحان کنيم که <http://www.tahlildadeh.com> را در صفحه‌ی شما لينك می‌کند.

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
</head>
<body>
```

```
<p>Click following link</p>
<a href="http://www.tahlildadeh.com" target="_self"> tahlildadeh </a>
</body>
</html>
```

این مثال نتیجه زیر را تولید خواهد کرد که شما می توانید روی لینک تولیدشده tahlildadeh کلیک کنید تا به صفحه اصلی tahlildadeh برسید.

Click following link

[Tahlildadeh](http://www.tahlildadeh.com)

### ویژگی target

ما از ویژگی target در مثال قبلی خود استفاده کردیم. این ویژگی برای مشخص کردن موقعیتی که داکیومنت لینک شده باز می شود، مورد استفاده قرار می گیرد. در زیر گزینه های ممکن را مشاهده می کنید.

Option	Description
_blank	داکیومنت لینک شده را در یک پنجره یا تب پدید باز می کند.
_self	داکیومنت لینک شده را در همان چارچوب باز می کند.
_parent	داکیومنت لینک شده را در چارچوب اصلی باز می کند.
_top	داکیومنت لینک شده در کل بدن پنجره باز می شود.
Targetframe	داکیومنت لینک شده را در یک targetframe نام گذاری شده باز می کند.

### مثال

برای درک تفاوت اصلی در برخی گزینه های ارائه شده در ویژگی target مثال زیر را امتحان کنید.

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
```

```
<base href="http://www.tahlildadeh.com/">
</head>
<body>
<p>Click any of the following links</p>
<a href="http://www.tahlildadeh.com" target="_blank">Opens in New</a> |
<a href="http://www.tahlildadeh.com" target="_self">Opens in Self</a> |
<a href="http://www.tahlildadeh.com" target="_parent">Opens in Parent</a> |
<a href="http://www.tahlildadeh.com" target="_top">Opens in Body</a></body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد که می‌توانید روی لینک‌های مختلف کلیک کنید تا تفاوت بین گزینه‌های مختلف ارائه شده برای ویژگی target را درک کنید.

Click any of the following links

[Opens in New](#) | [Opens in Self](#) | [Opens in Parent](#) | [Opens in Body](#)

#### استفاده از مسیر پایه

وقتی که شما داکیومنت‌های HTML را مناسب با همان وب سایت لینک می‌کنید، ارائه یک URL کامل برای هر لینک ضروری نیست. اگر از برچسب <base> در تیتر داکیومنت HTML خود استفاده می‌کنید، می‌توانید از دست آن خلاص شوید. این برچسب برای ارائه یک مسیر پایه برای همه‌ی لینک‌ها استفاده می‌شود. بنابراین مرورگر شما مسیر ارائه شده‌ی مرتبط را به مسیر پایه پیوند خواهد داد و یک URL کامل ایجاد خواهد کرد.

مثال زیر از برچسب <base> برای مشخص کردن URL پایه استفاده می‌کند و پس از آن ما می‌توانیم به ؟ای ارائه‌ی URL کامل برای هر لینک از مسیرهای مرتبط ایتفاذه کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
<base href="http://www.tahlildadeh.com/">
</head>
```

```
<body>
<p>Click following link</p>
<a href="/html/index.htm" target="_blank">HTML tahlildadeh</a>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد که می‌توانید روی لینک تولید شده‌ی tahlildadeh HTML کلیک کنید تا به آموزش HTML برسید.

با عنوان `a` ای شده ارائه URL اکنون شود می‌گرفته نظر در `a href="http://www.tahlildadeh.com/html/index.htm"`<

Click following link HTML tahlildadeh

### لینک شدن به بخشی از صفحه

می‌توانید برای بخش خاصی از صفحه‌ی وب ارائه شده، با استفاده از ویژگی `name` یک لینک ایجاد کنید. این امر یک فرایند دو مرحله‌ای می‌باشد.

نخست اینکه در مکانی که می‌خواهید به داخل صفحه‌ی وب برسید یک لینک ایجاد کنید و آن را با استفاده از `برچسب <a>...</a>` نام‌گذاری کنید.

```
<h1>HTML Text Links <a name="top"></a></h1>
```

مرحله‌ی دوم ایجاد یک هایپر لینک می‌باشد برای لینک کردن داکیومنت و قرار دادن در مکانی که می‌خواهید برسید.

```
<a href="/html/html_text_links.htm#top">Go to the Top</a>
```

و این مثال لینک زیر را تولید خواهد کرد که می‌توانید در آن روی لینک تولید شده‌ی Top کلیک کنید تا به نقطه‌ی بالای آموزش HTML Text Link برسید.

Go to the Top

### تنظیم رنگ‌های لینک

شما می‌توانید رنگ لینک‌های خود، لینک‌های فعال و لینکهای مشاهده شده را با استفاده از ویژگی‌های `link` و `alink` و `vlink` از `برچسب <body>` تنظیم کنید.

## مثال

مثال زیر را در test.htm ذخیره کنید و سپس می توانید آن را در هر مرورگری باز کنید تا ببینید که چگونه ویژگی کنند می کار link, alink و vlink

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
<base href="http://www.tahlildadeh.com/">
</head>
<body alink="#54A250" link="#040404" vlink="#F40633">
<p>Click following link</p>
<a href="/html/index.htm" target="_blank">HTML tahlildadeh</a>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید می کند. فقط رنگ لینک را قبل از کلیک کردن روی آن چک کنید، سپس رنگ آن را در هنگام فعال کردن و بازدید آن چک کنید.

Click following link

[HTML tahlildadeh](/html/index.htm)

## دانلود کردن لینک‌ها

شما می توانید لینک متنی ایجاد کنید تا فایل‌های قابل دانلود ZIP, PDF و DOC خود را بسازید. این کار بسیار ساده می باشد، فقط کافیست یک URL کامل از فایل قابل دانلود ارائه بدهید.

```
<!DOCTYPE html>
<html>
<head>
<title>Hyperlink Example</title>
</head>
```

```
<a href="http://tahlildadeh.com/Files/Articles/04GL07.pdf" Download PDF File</a>
</body>
</html>
```

این مثال لینک زیر را تولید خواهد کرد و برای دانلود یک فایل استفاده می شود.

[Download PDF File](http://tahlildadeh.com/Files/Articles/04GL07.pdf)

## آموزش Image Link

تاکنون مشاهده کردیم که چگونه با استفاده از متن، یک لینک هایپرلینک استفاده کنیم و همچنین فرا گرفته ایم که چگونه از تصاویر در صفحات وب خود استفاده کنیم. اکنون فرا خواهیم گرفت که چگونه با استفاده از تصاویر هایپرلینک استفاده کنیم.

### مثال

استفاده از تصویر به عنوان هایپرلینک بسیار ساده می باشد. لازم است که یک تصویر را در داخل هایپرلینک در محل تصویر قرار دهیم، همانطور که در زیر نشان داده شده است.

```
<!DOCTYPE html>
<html>
<head>
<title>Image Hyperlink Example</title>
<head>
<body>
<p>Click following link</p>
<a href="http://www.tahlildadeh.com" target="_self">

<a>
<body>
<html>
```

این ساده ترین راه ایجاد هایپرلینک با استفاده از تصاویر می باشد.

## تصاویر حساس به ماوس

استانداردهای HTML و XHTML یک ویژگی ارائه می دهند که به شما اجازه می دهد لینک های مختلف را در داخل یک تصویر ا جرا کنید. شما می توانید براساس مختصات مختلف مو یک روی تصویر، لینک های مختلفی را روی یک تصویر مجزا ایجاد کنید. زمانی که لینک های متفاوت به مختصات متفاوت ضمیمه می شود، می توانیم برای باز کردن داکیومنت های تارگت روی بخش های مختلف تصویر کلیک کنیم. چنین تصاویری که به ماوس حساس می باشند، تصاویر نقشه نامیده می شوند. دو روش برای ایجاد چنین تصاویری و یک دارد.

- توسط ویژگی ismap از برچسب فعال می شود و دستیابی به یک سرور و پردازش برنامه های مربوط به تصویر نقشه لازم می باشد.

- با استفاده از ویژگی usemap از برچسب همراه با  و  ایجاد می شود.

### Server side image maps

در اینجا به سادگی تصویر خود را داخل یک هایپرلینک قرار داده و از ویژگی ismap استفاده کنید که آن را یک تصویر خاص می سازد و وقتی یوزر در بخشی از تصویر کلیک می کند، مرورگر مختصات اشاره گر ماوس را همراه با URL مشخص شده در برچسب  به سرور وب منتقل می کند. سرور از مختصات اشاره گر ماوس استفاده می کند تا تعیین کند کدام داکیومنت باید به مرورگر بازگردانده شود.

زمانی که ismap استفاده می شود، ویژگی href از برچسب  باید URL یک برنامه سرور را مانند یک cgi یا اسکریپت PHP و غیره در برداشته باشد، تا درخواست ورودی را براساس مختصات انتقال داده شده پردازش کند. مختصات موقعیت ماوس پیکسل های صفحه میباشند که از گوشه‌ی بالای سمت چپ تصویر شمرده می شوند و با (0,0) شروع می شوند. مختصات دنبال شده با یک علامت سوال، به انتهای URL اضافه می شوند.

مثال

```
<!DOCTYPE html>
<html>
<head>
<title>ISMAP Hyperlink Example</title>
</head>
<body>
<p>Click following link</p>
<p>Click following link</p>
<a href="Exp1.html" target="_self">

</a>
</body>
</html>
```

سپس مرورگر پارامترهای `ismap` زیر را به سرور می فرستد که می تواند توسط اسکریپت `ismap.cgi` یا فایل `map` پردازش شود و شما می توانید هر داکیومنتی را که دوست داشته باشید به این مختصات لینک کنید.

/cgi-bin/ismap.cgi?20,30

از این طریق شما می توانید لینک های مختلفی را به مختصات متفاوت یک تصویر اختصاص دهید، و وقتی این مختصات کلیک می شوند، می توانید داکیومنت های لینک شده می متناظر را باز کنید.

شما برنامه نویسی CGI را زمانی فراخواهید گرفت که برنامه نویسی perl را مطالعه کنید. می توانید اسکریپت خود را بنویسید تا این مختصات انتقال داده شده را با استفاده از PHP یا هر اسکریپت دیگری پردازش کنید. فعلاً اجازه بدھید روی یادگیری HTML تمرکز کنیم، می توانید این فصل را بعداً بازبینی کنید.

### Client side image maps

این تصاویر به وسیله ویژگی `usemap` از برچسب `<img>` فعال می شوند و به وسیله برچسب `<area>` و `<map>` تعریف می شوند.

نقشه‌ای که قرار است نقشه را طراحی کنده، به وسیله‌ی <img> به عنوان یک تصویر عادی وارد صفحه می‌شود، به اینکه این مورد ویژگی اضافه‌ای به نام usemap همرا خود دارد. مقدار ویژگی usemap مقداری می‌باشد که در یک برچسب <map> استفاده می‌شود تا برچسب‌های تصویر و نقشه را لینک کند. <map> همراه با برچسب‌های <area> همه مختصات تصویر و لینک‌های مربوطه را تعریف می‌کند.

برچسب <area> در داخل برچسب نقشه مختصات و شکل حاشیه‌های قابل کلیک در داخل تصویر را تعریف می‌کند. در اینجا مثالی از تصویر نقشه می‌بینید.

```
<!DOCTYPE html>
<html>
<head>
<title>USEMAP Hyperlink Example</title>
</head>
<body>

<map name="planetmap">
<area shape="rect" coords="0,0,40,126" href="02.jpg">
<area shape="circle" coords="90,58,20" href="04.jpg">
<area shape="circle" coords="124,78,20" href="05.jpg">
</map>
</body>
</html>
```

### سیستم مختصات

مقدار حقیقی مختصات کانلا به شکل در سوال وابسته است. در اینجا خلاصه‌ای می‌بینید که قرار است با مثال‌های مفصل دنبال شوند.

`rect=x1,y1,x2,y2`

$x1$  و  $y1$  مختصات گوشه‌ی بالای سمت چپ از مستطیل می‌باشد.  $x2$  و  $y2$  مختصات گوشه‌ی سمت راست پایین می‌باشند.

`circle=xc,yc,radius`

$xc$  و  $yc$  مختصات مرکز دایره و  $radius$  شعاع دایره می‌باشد. دایره‌ای به مرکز 200,50 با شعاع 21 دارای ویژگی "coords="200,50,25" خواهد بود.

`poly=x1,y1,x2,y2,x3,y3,...,xn,yn`

مختلف  $y-x$  رؤس چند ضلعی می‌باشند، با یک خط که از یک نقطه به نقطه‌ی دیگر کشیده شده. یک چندضلعی لوزی شکل با بالاترین راس آن در نقطه‌ی 21,21 و 01 پیکسل، در عریض ترین نقطه‌ی خود دارای ویژگی "coords="20,20,40,40,20,60,0,40" می‌باشد.

تمام مختصات مربوط به بالاترین گوشه‌ی سمت چپ تصویر می‌باشند. هر شکل دارای یک URL مربوطه می‌باشد. می‌توانید از هر نرم افزار تصویری برای دانستن مختصات موقعیت‌های مختلف استفاده کنید.

## لینک ایمیل HTML

قرار دادن یک لینک ایمیل HTML روی صفحه‌ی وب خود کار سختی نیست، اما این کار ممکن است باعث بروز مشکل اسپیم‌های غیرضروری در اکانت ایمیل شما شود. افرادی هستند که می‌توانند برنامه‌هایی را برای جمع آوری چنین ایمیل‌هایی اجرا کنند و سپس آنها را با راه‌های مختلف برای اسپیم کردن استفاده کنند.

می‌توانید از گزینه‌ی دیگری استفاده کنید تا افراد به راحتی بتوانند به شما ایمیل ارسال کنند. یک گزینه‌ی می‌تواند استفاده از فرم‌های HTML باشد برای جمع آوری داده‌های یوزر و سپس استفاده از اسکریپت PHP یا CGI برای ارسال ایمیل.

برای یک مثال ساده، فرم Contact Us را چک کنید. با استفاده از این فرم یک فیدبک از یوزر گرفته و سپس از یک برنامه‌ی CGI استفاده می‌کنیم که در حال جمع آوری این اطلاعات و ارسال ایمیل به یک email ID ارائه شده می‌باشد.

## برچسب HTML Email

برچسب <a> در HTML برای مشخص کردن یک آدرس ایمیل و ارسال ایمیل، گزینه هایی را به شما ارائه می دهد. در حالیکه از برچسب <a> به عنوان یک email tag استفاده می کنید، از href email address همراه با ویژگی href نیز استفاده خواهد کرد. در زیر ترکیب استفاده از mailto را به یاری استفاده از http می بینید.

<a href="mailto:abc@example.com">Send Email</a>

این کد لینک زیر را تولید خواهد کرد که می توانید به آن ایمیل ارسال کنید.

Send Email

اکنون اگر یوزر روی این لینک کلیک کند، آغاز به کار خواهد کرد (مانند express outlook و غیره). که روی کامپیوتر شما نصب شده است. ریسک دیگری در استفاده از این گزینه برای ارسال ایمیل و گود دارد، زیرا اگر یوزر email client نصب شده روی کامپیوتر خود نداشته باشد، ارسال ایمیل ممکن نخواهد بود.

### تنظیمات پیش فرض

شما می توانید یک موضوع ایمیل و یک بدنه ای ایمیل پیش فرض همراه با آدرس ایمیل خود ایجاد کنید. در زیر مثال استفاده از بدنه و موضوع پیش فرض را مشاهده می کنید.

<a href="mailto:abc@example.com?subject=Feedback&body=Message"> Send Feedback

</a>

این کد نیز لینک زیر را تولید خواهد کرد که می توانید ایمیل ارسال کنید.

Send Feedback

### آموزش HTML Layout

برای یک صفحه ای وب بسیار مهم است، چرا که دید بهتری نسبت به وب سایت شما ارائه می دهد. طراحی Layout خوب با منظره و حس خوب برای یک وب سایت زمان فوق العاده زیادی می گیرد. این روزها همه ای وب سایت های مدرن از چارچوب هایی بر اساس CSS استفاده می کنند تا به وب سایت های پاسخگو و دینامیک پیروز شوند، اما کار یک Layout در صفحه ای وب شما، صرفا از HTML و ویژگی های ان استفاده می کند.

## استفاده از چند دل ها - HTML Layout

ساده ترین و محبوب ترین راه برای ایجاد Layout ها، استفاده از برچسب <table> در HTML می باشد. این چند دل ها در ردیف ها و ستون ها منظم می شوند که شما می توانید از این ردیف ها و ستون ها به هر طریقی که می خواهید استفاده کنید.

### مثال

برای مثال، نمونه زیر از طریق استفاده ای یک چند دل با سه ردیف و دو ستون به دست می آید، اما عنوان و پاورقی ستون هر دو ستون را با استفاده از ویژگی colspan احاطه می کند.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layout using Tables</title>
</head>
<body>
<table width="100%" border="0">
<tr>
<td colspan="2" bgcolor="#b5dcb3">
<h1>This is Web Page Main title</h1>
</td>
</tr>
<tr valign="top">
<td bgcolor="#aaaa" width="50">
<b>Main Menu</b><br /> HTML<br />
PHP<br /> PERL...
</td>
```

```
<td bgcolor="#eeee" width="100" height="200"> Technical and Managerial Tutorials
</td>
</tr>
<tr>
<td colspan="2" bgcolor="#b5dcb3">
<center>
Copyright © 2007 Tahlildadeh.com
</center>
</td>
</tr>
</table>
</body>
</html>
```

این مثال نتیجه زیر را تولید خواهد کرد.

This is Web Page Main Title	
Main Menu	Technical and Managerial Tutorials
HTML PHP PERL...	
Copyright © 2007 Tahlildadeh.com	

### چند ستونی - استفاده از چند Layout

شما می توانید صفحه‌ی وب خود را طوری طراحی کنید تا محتوای وب خود را در چند صفحه قرار دهید. می توانید محتوای خود را در ستون وسط قرار دهید و از ستون سمت چپ برای قرار دادن منو و از ستون سمت راست برای تبلیغ یا موارد دیگر استفاده کنید. این بسیار شبیه آنچه می باشد که ما در tahlildadeh.com داشتیم.

در اینجا مثالی را از ایجاد یک Layout سه ستونی می بینید.

```
<!DOCTYPE html>
<html>
```

```

<head>
<title>Three Column HTML Layout</title>
</head>
<body>
<table width="100%" border="0">
<tr valign="top">
<td bgcolor="#aaaa" width="20%">
<b>Main Menu</b><br /> HTML<br />
PHP<br /> PERL...
</td>
<td bgcolor="#b5dcb3" height="200" width="60%"> Technical and Managerial
Tutorials
</td>
<td bgcolor="#aaaa" width="20%">
<b>Right Menu</b><br /> HTML<br />
PHP<br />
PERL...
</td>
</tr>
</table>
</body>
</html>

```

این مثال نتیجه زیر را تولید خواهد کرد.

Main Menu HTML PHP PERL...	Technical and Managerial Tutorials	Right Menu HTML PHP PERL...
----------------------------------	------------------------------------	-----------------------------------

Span و DIV های HTML – استفاده از Layout

عنصر `<div>` یک عنصر block level می باشد که برای گروه بندی عناصر HTML استفاده می شود. در حالیکه این عنصر یک عنصر block level است، عنصر `<span>` برای گروه بندی عناصر در یک سطح درون خطی استفاده می شوند.

گرچه می توانیم با چندول های Layout های بسیار زیبایی به دست آوریم، اما چندول ها در واقع به عنوان ابزار Layout طراحی نشده اند، و بیشتر برای نمایش داده های چندولی استفاده می شوند.

این مثال از CSS استفاده می کند، بنابراین قبل از درک این مثال، بهتر است در ک بهتری از چگونگی کار CSS داشته باشید.

در اینجا سعی می کنیم با استفاده از برچسب `<div>` همراه با CSS همان نتیجه ای را به دست آوریم که هنگام استفاده از برچسب `<table>` در مثال قبل به دست آوردیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Layouts using DIV, SPAN</title>
</head>
<body>
<div style="width:100%">
<div style="background-color:#b5dcb3; width:100%">
<h1>This is Web Page Main title</h1>
</div>
<div style="background-color:#aaa; height:200px; width:100px; float:left;">
<div><b>Main Menu</b></div> HTML<br />
PHP<br /> PERL...
</div>
<div style="background-color:#eee; height:200px; width:350px; float:left;">
<p>Technical and Managerial Tutorials</p>
</div>
```

```
<div style="background-color:#aaa; height:200px;width:100px;float:right;">
<div><b>Right Menu</b></div> HTML<br />
PHP<br /> PERL...
</div>
<div style="background-color:#b5dcb3;clear:both">
<center>
Copyright © 2007 Tahlildadeh.com
</center>
</div>
</div>
</body>
</html>
```

شما می توانید با استفاده از DIV و SPAN به همراه CSS Layout های بهتری طراحی کنید.  
برای درک بیشتر از CSS لطفا به CSS Tutorial مرا عه کنید.

## آموزش Iframes

شما می توانید یک frame درون خطی را با استفاده از برچسب <iframe> مربوط به تعريف کنید. این برچسب به برچسب <frameset> ارتباطی ندارد، در عوض می تواند در هر یکی در داکیومنت شما ظاهر شود.

برچسب <iframe> یک محدوده ی مستطیلی را در داخل داکیومنت تعريف می کند که در آن مرورگر می تواند یک داکیومنت مجزا را ارائه دهد، مانند نوارهای اسکرول و حاشیه ها.

ویژگی src برای مشخص کردن URL مربوط به داکیومنتی استفاده می شود که حاوی frame درون خطی می باشد.

### مثال

در زیر مثالی را می بینید که چگونگی استفاده از <iframe> را توضیح می دهد.

```
<!DOCTYPE html>
```

```
<html>
```

```

<head>
<title>HTML Iframes</title>
</head>
<body>
<p>Document content goes here...</p>
<iframe src="/html/menu.htm" width="555" height="200">Sorry your browser does
not support inline frames.
</iframe>
<p>Document content also go here...</p>
</body>
</html>

```

### ویژگی های برچسب <iframe>

بسیاری از ویژگی های برچسب <iframe> شامل نام، گروه، حاشیه، id، طول حاشیه، عرض حاشیه، نام، اسکرول کردن، روش و تیتر، درست مانند ویژگی های متناظر با <frame> رفتار می کنند.

Attribute	Description
src	این ویژگی برای نام گذاری فایلی استفاده می شود که باید در frame بارگذاری شود. مقدار آن می تواند هر URL باشد. برای مثال فایل HTML موکود در مسیر html src="/html/top_frame.htm" بارگذاری خواهد کرد.
name	این ویژگی به شما اجازه می دهد تا یک frame را نامگذاری کنید. این ویژگی نشان می دهد که یک داکیومنت در کدام frame باید بارگذاری شود. هنگامی که می خواهید لینک هایی را در یک ایجاد کنید که صفحاتی را در یک دیگر بارگذاری می کند، که در این مورد دو مین frame برای تشخیص خود به عنوان هدف لینک به نام نیاز دارد، در اینجا این ویژگی بسیار مهم است..

frameborder	این ویژگی مشخص می کند که آیا حاشیه های frame نشان داده شوند یا خیر. این ویژگی مقدار داده شده به ویژگی frameborder روی برچسب <frameset> را می گیرد، البته اگر مقداری مشخص شده باشد، این مقدار می تواند 1 (بله) و یا 0 (خیر) باشد.
marginwidth	این ویژگی به شما اجازه می دهد تا عرض فضای بین حاشیه های چپ و راست frame و محتوای آن را مشخص کنید. مقدار به پیکسل داده می شود. برای مثال "marginwidth="10"
marginheight	این ویژگی به شما اجازه می دهد تا طول فضای بین بالا و پایین حاشیه frame و محتوای آن را مشخص کنید. مقدار به پیکسل داده می شود. برای مثال "marginheight="10"
noresize	به طور پیش فرض هر frame را با کلیک کردن و درگ کردن روی حاشیه های آن می توانید تغییر اندازه دهید. ویژگی noresize مانع تغییر اندازه frame می شود. برای مثال "noresize="noresize"
scrolling	این ویژگی ظاهر نارهای اسکرول نمایان شده در frame را کنترل می کند. این ویژگی مقادیر "auto", "yes", "no" یا "no" را می گیرد. برای مثال "scrolling="no" به معنای نبودن نوارهای اسکرول می باشد.
longdesc	این ویژگی به شما اجازه می دهد تا یک لینک به صفحه ای حاوی یک توصیف طولانی از "longdesc="framedescription.htm" مثال برای بدھید ارائه، محتوای frame،

### html در background

به طور پیش فرض رنگ زمینه ای صفحه ای وب شما سفید می باشد. ممکن است این زمینه را دوست نداشته باشد، اما نگرانی و چوبد ندارد. HTML دو روش مناسب زیر را ارائه می دهد تا زمینه ای صفحه وب خود را به دلخواه بیارایید.

زمینه ای HTML با رنگ ها.

زمینه‌ی HTML با تصاویر.

اکنون اجازه بدھید هر دو روش را یکی یکی و با استفاده از مثال‌های مناسب بررسی کنیم.

زمینه‌ی HTML با رنگ‌ها

ویژگی bgcolor برای کنترل زمینه‌ی یک عنصر HTML، به ویژه بدن‌ی صفحه و زمینه‌ی دول، استفاده می‌شود. در زیر ترکیب استفاده از bgcolor را با هر برچسب HTML می‌بینید.

```
<tagnname bgcolor="color_value">...
```

این color- value می‌تواند به هر کدام از فرمات‌های زیر ارائه شود.

```
<!-- Format 1 - Use color name -->
```

```
<table bgcolor="lime">
```

```
<!-- Format 2 - Use hex value -->
```

```
<table bgcolor="#f1f1f1">
```

```
<!-- Format 3 - Use color value in RGB terms -->
```

```
<table bgcolor="rgb(0,0,120)">
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>HTML Background Colors</title>
```

```
</head>
```

```
<body>
```

```
<!-- Format 1 - Use color name -->
```

```
<table bgcolor="yellow" width="100%">
```

```
<tr>
```

```
<td>
```

This background is yellow

```
</td>
</tr>
</table>
<!-- Format 2 - Use hex value -->
<table bgcolor="#6666FF" width="100%">
<tr>
<td>
This background is sky blue
</td>
</tr>
</table>
<!-- Format 3 - Use color value in RGB terms -->
<table bgcolor="rgb(255,0,255)" width="100%">
<tr>
<td>
This background is green
</td>
</tr>
</table>
</body>
</html>
```

### زمینه HTML با تصاویر

ویژگی background همچنین می تواند برای کنترل زمینه ی یک عنصر HTML، به ویژه بدنه ی صفحه و زمینه های چاله، استفاده شود. شما می توانید یک تصویر را به عنوان زمینه ی صفحه و یا چاله HTML خود استفاده کنید. در زیر ترکیب استفاده از ویژگی background را با هر عنصر HTML می بینید.

ویژگی background به عنوان ویژگی خوبی تلقی نمی شود و توصیه می شود از style sheet برای تنظیم زمینه استفاده کنید.

<tagname background="Image URL" ...>

متداول ترین فرمت های مورد استفاده ای تصویر عبارتند از JPEG و PNG.

در اینجا مثال هایی را از تنظیم تصویر به عنوان زمینه ای ۳دول مشاهده می کنید.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>HTML Background Images</title>
```

```
</head>
```

```
<body>
```

```
<!-- Set table background -->
```

```
<table background="/images/html.jpg" width="100%" height="400">
```

```
<tr>
```

```
<td>
```

This background is filled up with HTML image.

```
</td>
```

```
</tr>
```

```
</table>
```

```
</body>
```

```
</html>
```

## زمینه های طرح دار و شفاف

شما ممکن است الگوها و زمینه های شفاف بسیاری را در وب سایت های مختلف دیده باشید. این امر به راحتی و با استفاده از تصاویر طرح دار و شفاف در زمینه قابل دستیابی می باشد. توصیه می شود در هنگام ایجاد تصاویر طرح دار یا شفاف GIF یا PNG، از کوچکترین ابعاد ممکن آنها استفاده کنید، حتی به کوچکی 1x1 برای ۳لوگیری از بارگذاری آهسته.

```
<!DOCTYPE html>
```

```
<html>
<head>
```

### مثال

در اینجا مثالی از تنظیم یک الگوی زمینه برای یک جدول را مشاهده می کنید

```
<title>HTML Background Images</title>
</head>
<body>
<!-- Set a table background using pattern -->
<table background="/images/pattern1.jpg" width="20%" height="100">
<tr>
<td>
This background is filled up with a pattern image.
</td>
</tr>
</table>
<!-- Another example on table background using pattern -->
<table background="/images/pattern2.jpeg" width="20%" height="100">
<tr>
<td>
This background is filled up with a pattern image.
</td>
</tr>
</table>
</body>
</html>
```

رنگ ها برای دادن یک ظاهر و احساس خوب به وب سایت شما بسیار مهم می باشند. شما می توانید رنگ ها را روی لایه ای صفحه با استفاده از برچسب <body> مشخص کنید، یا می توانید رنگ ها را برای برچسب های مجزا با استفاده از ویژگی bgcolor مشخص کنید.

برچسب <body> دارای ویژگی های زیر می باشد که می تواند برای تنظیم رنگ های مختلف استفاده شود.

bgcolor رنگی را برای زمینه ای صفحه تنظیم می کند.  
text رنگی را برای متن تنظیم می کند.

alink رنگی را برای لینک های فعال یا انتخاب شده تنظیم می کند.

link رنگی را برای متن لینک شده تنظیم می کند.

رنگی را برای لینک های بازدید شده تنظیم می کند- یعنی برای متن لینک شده که روی آن کلیک کرده اید.

## روش های کدگذاری رنگ

در زیر سه روش متفاوت برای تنظیم رنگ در صفحه ای وب خود مشاهده می کنید  
- می توانید نام رنگ ها را به طور مستقیم تعیین کنید، به عنوان مثال سبز، قرمز، آبی و غیره.  
- یک کد شش رقمی که نشان دهنده مقدار آبی، قرمز و سبز سازنده ای رنگ، می باشد.

-Color decimal or percentage values- این مقدار با استفاده از ویژگی rgb() تعیین می شود.

اکنون این روش های رنگ گذاری را یکی یکی بررسی خواهیم کرد.

W3C می توانید برای تنظیم رنگ متن یا زمینه به طور مستقیم از نام رنگ استفاده کنید. لیستی از 11 رنگ پایه را دارد که توسط یک اعتبارسنج HTML ارزیابی می شود، اما نام بیشتر از 211 رنگ مختلف و سود دارد که توسط مرورگرهای مهم پشتیبانی می شوند.

در اینجا نام 11 رنگ استاندارد W3C را مشاهده می کنید، پیشنهاد می شود که از آنها استفاده کنید.

Black	Gray	Silver	White
Yellow	Lime	Aqua	Fuchsia
Red	Green	Blue	Purple
Maroon	Olive	Navy	Teal

### مثال

در اینجا مثال هایی را میبینید از تنظیم زمینه‌ی یک برچسب HTML به وسیله‌ی نام رنگ.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by Name</title>
</head>
<body text="blue" bgcolor="green">
<p>Use different color names for for body and table and see the result.</p>
<table bgcolor="black">
<tr>
<td>
<font color="white">This text will appear white on black background.</font>
</td>
</tr>
</table>
</body>
</html>
```

رنگ های HTML – کدهای

هگزادسیمال یک نمایش 1 رقمی از یک رنگ می باشد. دو رقم اول یعنی RR نمایانگر رنگ قرمز (Red) می باشند، دو رقم بعدی (GG) مقدار سبز (green) را نشان می دهند و آخرین دو رقم (BB) نیز مقدار آبی (blue) را نشان می دهند.

هر کد هگزادسیمال به وسیلهٔ یک علامت # دنبال می شود. در ادامه رنگ‌های مورد استفاده در نشانه گذاری هگزادسیمال را مشاهده می کنید.

Color	Color HEX
Black	#000000
Red	#FF0000
Green	#00FF00
Blue	#0000FF
Yellow	#FFFF00
Cyan	#00FFFFFF
Magenta	#FF00FF
Grey	#C0C0C0
White	#FFFFFF

در زیر مثال هایی را می بینید از تنظیم زمینهٔ یک برچسب HTML به وسیلهٔ کد رنگ در هگزادسیمال.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by Hex</title>
</head>
<body text="#0000FF" bgcolor="#00FF00">
<p>Use different color hexa for body and table and see the result.</p>
<table bgcolor="#000000">
<tr>
```

```
<td>
<font color="#FFFFFF">This text will appear white on black background.</font>
</td>
</tr>
</table>
</body>
</html><
```

### رنگ های HTML - مقادیر RGB

مقدار این رنگ با استفاده از ویژگی (rgb) مشخص می شود. این ویژگی سه مقدار می گیرد که هر کدام برای سبز، قرمز و آبی می باشد. مقدار می تواند عددی بین ۱ و ۲۱۱ یا یک درصد باشد.

تو چه همه می مرور گرها ویژگی (rgb) را پشتیبانی نمی کنند، بنابراین توصیه می شود از آن استفاده نکنید.

در زیر لیستی از رنگ ها با مقادیر RGB را مشاهده می کنید.

Color	Color RGB
	rgb(0,0,0)
	rgb(255,0,0)
	rgb(0,255,0)
	rgb(0,0,255)
	rgb(255,255,0)
	rgb(0,255,255)
	rgb(255,0,255)
	rgb(192,192,192)
	rgb(255,255,255)

در اینجا مثال هایی را مشاهده می کنید از تنظیم زمینه‌ی یک برجسب HTML به وسیله‌ی برجسب() با کد رنگ.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Colors by RGB code</title>
</head>
<body text="rgb(0,0,255)" bgcolor="rgb(0,255,0)">
<p>Use different color code for body and table and see the result.</p>
<table bgcolor="rgb(0,0,0)">
<tr>
<td>
<font color="rgb(255,255,255)">This text will appear white on black
background.</font>
</td>
</tr>
</table>
</body>
</html><
```

در زیر لیستی از 211 رنگ را مشاهده می کنید که تصور می شود سالم ترین و مستقل ترین رنگ‌های کامپیوتر باشند. این رنگ‌ها از کد هگزای 000000 تا FFFFFF متفاوت می باشند و توسط همه‌ی کامپیوترهای دارای پالت رنگ 211 تایی پشتیبانی می شوند.

000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF

00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF

FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF

### فونت‌ها

فونت‌ها نقش مهمی در زیبا ساختن و خواناتر کردن وب سایت بازی می‌کنند. ظاهرو رنگ فونت به طور کامل بستگی به کامپیوتر و مرورگری دارد که استفاده می‌شود، اما شما می‌توانید از برچسب `<font>` در HTML برای افزودن استایل، سایز و رنگ به متن خود در وب سایت استفاده کنید. می‌توانید از یک برچسب `<basefont>` برای تنظیم تمام متن خود به اندازه، ظاهر و رنگ یکسان استفاده کنید.

برچسب فونت دارای سه ویژگی به نام‌های `size` و `color` و `face` می‌باشد که فونت شکا را به دلخواه در می‌آورد. برای تغییر هر کدام از ویژگی‌های فونت در هر زمانی در صفحه‌ی وب خود، به سادگی از برچسب `<font>` استفاده کنید. متنی که دنبال می‌کند، تغییر یافته باقی می‌ماند تا زمانی که شما آن را با `</font>` برچسب ببندید. شما می‌توانید یکی از ویژگی‌ها یا همه‌ی ویژگی‌های داخل برچسب `<font>` را تغییر دهید.

برچسب‌های `font` و `basefont` استفاده نمی‌شوند و احتمال می‌رود که در ورژن‌های بعدی HTML حذف شوند. بنابراین نباید مورد استفاده قرار بگیرند، پیشنهاد می‌شود که برای اجرای فونت‌های خود از استایل‌های CSS استفاده کنید. اما برای رسیدن به هدف این فصل برچسب‌های `font` و `basefont` را با مزئیات توضیح می‌دهد.

### تنظیم اندازه فونت

شما می‌توانید با استفاده از ویژگی `size` اندازه فونت محتوا را تنظیم کنید. دامنه‌ی مقادیر قابل قبول از 1 (کوچکترین) تا 7 (بزرگترین) می‌باشد. اندازه‌ی فونت پیش‌فرض 3 می‌باشد.

### مثال

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Setting Font Size</title>
</head>
<body>
<font size="1">Font size="1"</font><br />
<font size="2">Font size="2"</font><br />
<font size="3">Font size="3"</font><br />
<font size="4">Font size="4"</font><br />
<font size="5">Font size="5"</font><br />
<font size="6">Font size="6"</font><br />
<font size="7">Font size="7"</font>
</body>
</html>
```

### اندازه فونت مربوط

شما می توانید مشخص کنید چه تعداد از فونت ها بزرگتر و چه تعداد کوچکتر از اندازه **ی** فونت حاضر باشند. می توانید آن را مانند **<font size="-n">** or **<font size="+n">** مشخص کنید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Relative Font Size</title>
</head>
<body>
<font size="-1">Font size="-1"</font><br />
<font size="+1">Font size="+1"</font><br />
<font size="+2">Font size="+2"</font><br />
```

```
<font size="+3">Font size="+3"</font><br />
<font size="+4">Font size="+4"</font>
</body>
</html>
```

### تنظیم ظاهر فونت

شما می توانید با استفاده از ویژگی face ظاهر فونت را تنظیم کنید، اما باید بدانید که اگر کاربر بازدیدکننده ای صفحه، فونت را نصب نکرده باشد، قادر به دیدن آن نخواهد بود. در عوض کاربر ظاهر فونت پیش فرض را می بیند که برای کامپیوترش مناسب می باشد.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Font Face</title>
</head>
<body>
<font face="Times New Roman" size="5">Times New Roman</font><br />
<font face="Verdana" size="5">Verdana</font><br />
<font face="Comic sans MS" size="5">Comic Sans MS</font><br />
<font face="WildWest" size="5">WildWest</font><br />
<font face="Bedrock" size="5">Bedrock</font><br />
</body>
</html>
```

### تعیین ظاهر فونت چاکر

یک بازدید کننده فقط قادر خواهد بود فونت شما را ببیند، اگر آن را نصب شده روی کامپیوتر خود داشته باشد. بنابراین امکان تعیین دو یا بیشتر از دو ظاهر چاکر با ارائه ای نام های مربوط به فونت ها وجود دارد.

```
<font face="arial,helvetica">
```

<font face="Lucida Calligraphy,Comic Sans MS,Lucida Console">

اگر هیچکدام از فونت های ارائه شده نصب نشده باشند، بنابراین فونت پیش فرض Roman نمایش داده خواهد شد.

### تنظیم رنگ فونت

شما می توانید با استفاده از ویژگی color هر رنگی را برای فونت تنظیم کنید. شما می توانید رنگ مورد نظر خود را یا با استفاده از نام رنگ و یا با استفاده از کد هگزادسیمال برای آن رنگ تعیین کنید.

### مثال

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Font Color</title>
</head>
<body>
<font color="#FF00FF">This text is in pink</font><br />
<font color="red">This text is red</font>
</body>
</html>
```

### عنصر <basefont>

انتظار می رود که عنصر <basefont> یک اندازه، رنگ و ظاهر پیش فرض برای هر بخشی از داکیومنت که در برچسب <font> قرار نمی گیرند، تنظیم کند. شما می توانید از عناصر <basefont> استفاده کنید تا تنظیمات <font> را انجام دهید.

برچسب <basefont> همچنین ویژگی های رنگ، اندازه و ظاهر را می گیرد و با دادن مقدار بیشتر از 1+ برای فونت های بزرگتر و کمتر از 2- برای فونت های کوچکتر، تنظیمات فونت مربوطه را پشتیبانی می کند.

**مثال**

```
<!DOCTYPE html>
<html>
<head>
<title>Setting Basefont Color</title>
</head>
<body>
<basefont face="arial, verdana, sans-serif" size="2" color="#ff0000">
<p>This is the page's default font.</p>
<h2>Example of the <basefont> Element</h2>
<p>
<font size="+2" color="darkgray">
This is darkgray text with two sizes larger
</font>
</p>
<p>
<font face="courier" size="-1" color="#000000">
It is a courier font, a size smaller and black in color.
</font>
</p>
</body>
</html>
```

**frm های**

وقتی که می خواهید داده هایی را از سایت بازدید کننده بمع آوری کنید، به فرم های HTML احتیاج خواهید داشت. به عنوان مثال در هنگام ثبت کاربر، اطلاعاتی مانند نام، آدرس ایمیل و کارت اعتباری و غیره را بمع آوری می کنید. یک فرم داده ها را از بازدید کننده ی سایت می گیرد و سپس آن را به یک برنامه ی پایانی مانند CGI، اسکریپت ASP یا اسکریپت PHP

باز می گرداند. برنامه‌ی پایانی فرایندهای مورد نیاز را بر اساس کار تعریف شده در برنامه، روی داده‌ی منتقل شده انجام می‌دهد.

عناصر متفاوتی برای فرم در دسترس می‌باشند، مانند فیلدات متن، فیلدات `textarea`، منوهای رو به پایین، دکمه‌های رادیو، چک باکس‌ها و غیره.

برچسب `<form>` مربوط به HTML، برای ایجاد یک فرم استفاده می‌شود و دارای ترکیب زیر می‌باشد.

`<form action="Script URL" method="GET|POST">` form elements like `input`, `textarea` etc.

`</form>`

### ویژگی‌های فرم

علاوه بر ویژگی‌های متداول، در زیر لیستی از متداول‌ترین ویژگی‌های مربوط به فرم را مشاهده می‌کنید.

Attribute	Description
<code>action</code>	اسکریپت backend برای پردازش داده‌ی انتقال شده‌ی شما آماده می‌باشد.
<code>method</code>	روشی برای بارگذاری داده. متداول‌ترین روش‌های مورد استفاده GET و POST می‌باشند.
<code>target</code>	پنجره‌ی هدف یا چارچوب را، یا بی که نتیجه‌ی اسکریپت نمایش داده خواهد شد، مشخص می‌کند. این ویژگی مقادیر <code>_blank</code> , <code>_self</code> , <code>_parent</code> و غیره را به خود می‌گیرد.

enctype	<p>You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are</p> <p>شما می توانید از ویژگی enctype برای مشخص کردن چگونگی برنامه نویسی توسط مرورگر قبل از ارسال به سرور، استفاده کنید. مقدار ممکن عبارتند از</p> <ul style="list-style-type: none"> <li>- application/x-www-form-urlencoded</li> </ul> <p>که است استانداردی روش این بیشتر فرم ها در سناریو های ساده استفاده می کنند.</p> <p>- این روش زمانی استفاده می شود که شما می خواهید داده های دوتایی را به شکل فایل هایی مانند تصویر بارگذاری کنید.</p>
---------	---

## کنترل های فرم

انواع مختلفی از کنترل های فرم وجود دارند که می توانید برای چم آوری داده با استفاده از فرم HTML از آنها استفاده کنید.

### کنترل های ورودی متن

#### کنترل های چک باکس

کنترل های رادیو باکس کنترل های انتخاب باکس های انتخاب فایل کنترل های مخفی دکمه های قابل کلیک شدن ثبت و تنظیم مجدد

### کنترل های ورودی متن

سه نوع ورودی متن و یو دارد که در فرم ها استفاده می شوند.

کنترل های ورودی متن تک خطی این کنترل برای آیتم هایی استفاده می شود که فقط یک خط از ورودی کاربر را لازم دارد، مانند باکس های بستجو و نام ها. این کنترل ها با استفاده از برجسب <input type="text"> مربوط به HTML ایجاد می شوند.

کنترل ورودی گذر واژه این نیز یک متن ورودی تک خطی می باشد، اما به محض اینکه کاربر کاراکتر را وارد می کند، این کنترل آن را می پوشاند.

کنترل های متن ورودی چند خطی زمانی استفاده می شود که لازم است کاربر **جزئیاتی** را وارد کند که بیشتر از یک **نمایه** می باشند. کنترل های ورودی چند خطی با استفاده از **برچسب <textarea>** استفاده می شوند.

### کنترل های ورودی تک خطی

این کنترل ها برای آیتم هایی استفاده می شوند که کاربر فقط یک خط ورودی احتیاج دارد، مانند باکس های جستجو و نام ها و با استفاده از **برچسب <input>** مربوط به HTML ایجاد می شوند.

در اینجا مثال پایه ای می بینید از یک ورودی تک خطی که برای گرفتن نام کوچک و نام خانوادگی به کار می رود.

```
<!DOCTYPE html>
<html>
<head>
<title>Text Input Control</title>
</head>
<body>
<form>
First name <input type="text" name="first_name" />
<br>
Last name <input type="text" name="last_name" />
</form>
</body>
</html>
```

### ویژگی ها

در زیر لیستی از ویژگی های **برچسب <input>** را برای ایجاد فیلد متن می بینید.

Attribute	Description
<b>type</b>	نوع کنترل ورودی را نشان می دهد و برای کنترل متن ورودی برای <b>text</b>

	تنظیم خواهد شد.
<b>name</b>	برای دادن نام به کنترلی استفاده می شود که قرار است برای تشخیص به سرور ارسال شود و مقدار بگیرد.
<b>value</b>	می تواند برای ارائه ی یک مقدار اصلی در داخل کنترل استفاده شود.
<b>size</b>	ا چازه می دهد تا عرض کنترل متن ورودی را متناسب با کاراکترها تعیین کنید.
<b>maxlength</b>	ا چازه می دهد تا حداقل تعداد کاراکترهایی را که یک کاربر می تواند در یک تکست باکس وارد کند، مشخص کنید.

### کنترل های پسورد ورودی

این کنترل نیز یک کنترل تک خطی می باشد، اما به محض اینکه کاربر کاراکترها را وارد می کند، آنها را می پوشاند. این ها نیز با استفاده از برچسب  مربوط به HTML ایجاد می شوند، اما نوع ویژگی با عنوان password تنظیم میشود.

### مثال

در اینجا مثالی از ورودی تک خطی پسورد می بینید که برای گرفتن پسورد کاربر استفاده می شود.

```
<!DOCTYPE html>
<html>
<head>
<title>Password Input Control</title>
</head>
<body>
<form>
User ID <input type="text" name="user_id" />
<br>
Password <input type="password" name="password" />
```

```
</form>
</body>
</html>
```

## ویژگی ها

در زیر لیست مربوط به ویژگی های برچسب `<input>` را برای ایجاد فیلد پسورد می بینید.

Attribute	Description
<code>type</code>	نوع کنترل ورودی را نشان می دهد و برای کنترل ورودی پسورد، برای <code>password</code> تنظیم خواهد شد.
<code>name</code>	برای نام گذاری کنترل استفاده می شود که برای تشخیص و گرفتن مقدار به سرور فرستاده می شود.
<code>value</code>	برای ارائه می یک مقدار اولیه در داخل کنترل استفاده می شود.
<code>size</code>	ا یازه می دهد تا عرض کنترل متن ورودی را با کاراکترها تعیین کنید.
<code>maxlength</code>	ا یازه می دهد تا حداقل تعداد کاراکترهایی را که یک کاربر می تواند در یک تکست باکس وارد کند، تعیین کنید.

## کنترل های متن ورودی چندخطی

زمانی استفاده می شود که یک کاربر باید چند سیگنال را ئارد کند که بیشتر از یک کلمه می باشند. کنترل های ورودی چند خطی با استفاده از برچسب `<textarea>` ایجاد میشوند.

### مثال

در اینجا مثالی را می بینید از یک ورودی متن چند خطی که برای ارائه می توصیفات آیتم استفاده می شود.

```
<!DOCTYPE html>
<html>
<head>
<title>Multiple-Line Input Control</title>
```

```

</head>
<body>
<form>
Description <br/>
<textarea rows="5" cols="50" name="description"> Enter description here...
</textarea>
</form>
</body>
</html>

```

### ویژگی ها

در زیر لیستی از ویژگی های برچسب <textarea> ارائه شده اند.

Attribute	Description
name	برای نامگذاری کنترلی استفاده می شود که به سرور ارسال می شود تا تشخیص داده شده و مقدار بگیرد.
rows	تعداد ردیف های area box مربوط به متن را نشان می دهد.
cols	تعداد ستون های area box مربوط به متن را نشان می دهد.

### کنترل چک باکس

چک باکس ها زمانی استفاده می شوند که بیشتر از یک گزینه قرار است انتخاب شود. آنها نیز با استفاده از برچسب <input> ایجاد می شوند، اما نوع ویژگی به checkbox تنظیم می شود.

مثال

در اینجا مثالی از کد HTML را مشاهده می کنید برای یک فرم با دو چک باکس.

```
<!DOCTYPE html>
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
</form>
</body>
</html>
```

در زیر لیستی از ویژگی های برچسب <checkbox> را می بینید.

Attribute	Description
type	نوع کنترل ورودی را نشان می دهد و برای کنترل ورودی چک باکس با checkbox کنترل خواهد شد.
name	برای نامگذاری کنترلی استفاده می شود که برای تشخیص و گرفتن مقدار به سرور ارسال می شود.
value	مقداری که اگر چک باکس انتخاب شود، استفاده خواهد شد.
checked	اگر بخواهید آن را به طور پیش فرض انتخاب کنید، با checked تنظیم می شود.

کنترل دکمه رادیو

دکمه های رادیو زمانی استفاده میشوند که بین چندین گزینه تنها یک گزینه باید انتخاب شود. این کنترل ها نیز با بروچسب  ایجاد میشوند، اما نوع ویژگی با radio تنظیم می شود.

### مثال

در اینجا مثالی از کد HTML می بینید برای یک فرم با دو دکمه رادیو.

```
<!DOCTYPE html>
<html>
<head>
<title>Checkbox Control</title>
</head>
<body>
<form>
<input type="radio" name="maths" value="on"> Maths
<input type="radio" name="physics" value="on"> Physics
</form>
</body>
</html>
```

### ویژگی ها

در زیر لیستی از ویژگی های دکمه رادیو را می بینید.

Attribute	Description
type	نوع کنترل ورودی را نشان می دهد و برای کنترل ورودی چک باکس با عنوان radio تنظیم می شود.
name	برای نامگذاری کنترلی استفاده می شود که برای تشخیص و گرفتن مقدار به سرور ارسال می شود.
value	مقداری که radio box انتخاب شود، استفاده خواهد شد.

checked	اگر می خواهید آن را به عنوان پیش فرض استفاده کنید، به checked تنظیم کنید.
---------	---

### Select Box

یک منوی رو به پایین نیز نامیده می شود، گزینه ای را برای ارائه ی گزینه های مختلف به شکل یک لیست رو به پایین ارائه می دهد، که کاربر می تواند از آن یک یا بیشتر از یک گزینه را انتخاب کند.

#### مثال

در اینجا مثالی از کد HTML برای یک فرم با یک چعبه ای رو به پایین می بینید.

```
<!DOCTYPE html>
<html>
<head>
<title>Select Box Control</title>
</head>
<body>
<form>
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
</form>
</body>
</html>
```

در زیر لیستی از ویژگی های برچسب <select> را می بینید.

Attribute	Description
-----------	-------------

name	برای نامگذاری کنترلی استفاده می شود که برای تشخیص و گرفتن مقدار به سرور انتقال می شود.
size	برای نمایش یک لیست باکس اسکرولینک استفاده می شود.
multiple	اگر روی multiple تنظیم شده باشد، به کاربر اجازه می دهد تا چند آیتم را از منو انتخاب کند.

در زیر لیستی از ویژگی های مهم برچسب <option> ارائه شده است.

Attribute	Description
value	اگر در سلکت باکس یک گزینه انتخاب شده باشد، این مقدار استفاده خواهد شد.
selected	مشخص می کند که این گزینه روزمان بارگذاری صفحه، باید گزینه انتخابی باشد.
label	یک روش یا گزینه برای برچسب دار کردن گزینه ها.

### فایل آپلود باکس

اگر می خواهید به کاربر اجازه دهید تا فایلی را روی وب سایت شما آپلود کند، به یک کنترل فایل آپلود احتیاج خواهید داشت، که Select Box نیز نامیده می شود. این ویژگی با استفاده از برچسب <input> نیز ایجاد می شود، اما نوع ویژگی با file تنظیم می شود.

### مثال

در اینجا مثالی را می بینید از کد HTML برای یک فرم با یک فایل آپلود باکس.

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
```

```
<form>
<input type="file" name="fileupload" accept="image/*" />
</form>
</body>
</html>
```

## ویژگی ها

در زیر لیستی از ویژگی های فایل آپلود باکس را می بینید.

Attribute	Description
<b>name</b>	برای نامگذاری کنترل استفاده می شود که برای تشخیص و گرفتن مقدار به سرور انتقال می شود.
<b>accept</b>	نوع فایل هایی را که سرور می پذیرد، تعیین می کند.

## کنترل های دکمه

راه های مختلفی برای ایجاد دکمه های قابل کلیک شدن و بود دارد. شما با استفاده از بروچسب <input> و با تنظیم نوع ویژگی آن به button دکمه های قابل کلیک شدن ایجاد کنید. نوع ویژگی می تواند مقادیر زیر را بگیرد.

Type	Description
submit	دکمه ای را ایجاد می کند که به طور خودکار یک فرم را می پذیرد.
reset	دکمه ای را ایجاد می کند که به طور خودکار کنترل های فرم را به مقادیر اولیه خود بازمی گرداند.
button	دکمه ای را ایجاد می کند که برای اجرای اسکریپت کاربر، زمانی که کاربر آن دکمه را کلیک می کند، استفاده می شود.
image	یک دکمه ای قابل کلیک شدن ایجاد می گند، اما ما می توانیم از یک تصویر به عنوان زمینه ای دکمه استفاده کنیم.

مثال

در اینجا مثالی از کد HTML برای یک فرم با سه نوع دکمه را می بینید.

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
<body>
<form>
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset" value="Reset" />
<input type="button" name="ok" value="OK" />
<input type="image" name="imagebutton" src="/html/images/logo.png" />
</form>
</body>
</html>
```

### کنترل های مخفی شده ی فرم

کنترل های مخفی شده فرم برای پنهان کردن داده در داخل صفحه ای استفاده می شوند که بعدا می توانند به سرور ارسال شوند. این کنترل در داخل کد مخفی شده و روی صفحه ی حقیقی ظاهر نمی شود. برای مثال فرم مخفی شده ی زیر برای حفظ شماره ی صفحه ی `next page` کلیک کند، مقدار کنترل مخفی شده به سرور وب ارسال شده و در آنجا تصمیم خواهد گرفت که بر اساس انتقال صفحه ی `باری`، `کدام` صفحه نمایش داده خواهد شد.

```
<!DOCTYPE html>
<html>
<head>
<title>File Upload Box</title>
</head>
```

```
<body>  
<form>  
<p>This is page 10</p>
```

مثال

در اینجا مثالی از کد HTML برای نمایش کاربرد کنترل مخفی شده می بینید.

```
<input type="hidden" name="pagename" value="10" />  
<input type="submit" name="submit" value="Submit" />  
<input type="reset" name="reset" value="Reset" />  
</form>  
</body>  
</html>
```

## آموزش چند رسانه ای در HTML

گاهی اوقات تمایل دارید که موسیقی یا ویدئو به صفحه‌ی وب خود اضافه کنید. ساده ترین راه برای افزودن صدا یا ویدئو به وب سایت، وارد کردن برچسب خاص HTML به نام `<embed>` می باشد. این برچسب باعث می شود که مرورگر به طور خودکار کنترل هایی را برای مولتی مедیا وارد کند که مرورگر ارائه شده برچسب `<embed>` و نوع میدیای ارائه شده را پشتیبانی می کند.

همچنین می توانید یک برچسب `<noembed>` برای مرورگرهایی وارد کنید که برچسب `<embed>` را نمی شناسند. به عنوان مثال می توانید از `<embed>` برای نمایش یک فیلم به `<noembed>` انتخاب خود استفاده کنید و اگر مرورگر `<embed>` را پشتیبانی نمی کند، از برای نمایش یک تصویر JPG مجزا استفاده کنید.

مثال

در اینجا مثال ساده ای می بینید از افرای یک فایل پاسازی شده midi.

```
<!DOCTYPE html>  
<html>  
<head>
```

```
<title>HTML embed Tag </title>
<head>
<body>
<embed src="/html/yourfile.mid" width="100%" height="60">
<noembed></noembed>
</embed>
</body>
</html>
```

### ویژگی های برچسب <embed>

در زیر لیستی از ویژگی های مهمی را می بینید که توسط برچسب <embed> استفاده می شود.

Attribute	Description
align	تعیین می کند که چگونه یک آبجکت را تنظیم کرد، که می تواند هم در مرکز، هم در راست و یا چپ تنظیم شود.
autostart	این ویژگی Boolean نشان می دهد که آیا میدیا باید به طور خودکار شروع به کار کند. شما می توانید آن را با true یا false تنظیم کنید.
loop	تعیین می کند که آیا صدا متداولاً پشت سر هم تکرار شود (لوب را روی تنظیم کنید)، یا چند دفعه باید تکرار شود (یک مقدار مثبت) و یا اصلاً تکرار نشود(false).
playcount	تعداد دفعاتی را که یک صدا باید تکرار شود تعیین می کند. اگر در حال استفاده از IE هستید، این گزئیه پایگرینی برای loop می باشد.
hidden	مشخص می کند که آیا آبجکت مولتی میدیا باید روی صفحه نمایش داده شود. مقدار false یعنی نه و مقدار true یعنی بله.
width	عرض آبجکت به پیکسل.
height	عرض آبجکت به پیکسل.

name	نامی که برای اشاره به آبجکت استفاده می شود.
src	URL مربوط به آبجکت که باید ئاسازی شود.
volume	میزان صدا را کنترل می کند که می تواند از 1 (صدا قطع شده) تا 111 (آخرین حد صدا) باشد.

## embed در داخل برچسب MOV و AVi flash movies

### انواع ویدیوهای پشتیبانی شده

می توانید از انواع مدیاهای مختلفی مانند فایل های استفاده کنید.

فایل های swf – فایل هایی هستند که با برنامه‌ی macromedia's flash تولید می شوند.

فایل های wmv – انواع فایل های تصویری ویندوز مایکروسافت می باشند.

فایل های mov - فرمت Quick time movie در اپل.

فایل های mpeg – فایل های تصویری هستند که توسط گروه تخصصی تصاویر متحرک (Moving

Pictures Expert Group) ایجاد شوند می ایجاد.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML embed Tag</title>
</head>
<body>
<embed src="/html/yourfile.swf" width="200" height="200">
<noembed></noembed>
</embed>
</body>
</html>
```

## صدای زمینه

شما می توانید از برچسب `<bgsound>` در html برای صدا در زمینه‌ی صفحه‌ی وب خود استفاده کنید. این برچسب فقط توسط اینترنت اکسپلورر پشتیبانی می‌شود و دیگر مرورگرهای آن را نادیده می‌گیرند. زمانی که سند اصلی در ابتدا توسط کاربر دانلود شده و نمایش داده می‌شود، این برچسب یک فایل صدا را دانلود کرده و آرا می‌کند. همچنین هر وقت کاربر مرورگر را تازه سازی کند، صدای زمینه نیز دوباره آرا خواهد شد.

این برچسب دارای فقط دو ویژگی می‌باشد، `src` و `loop`، که همانطور که در بالا توضیح داده شد هر دوی این ویژگی‌ها دارای یک معنا می‌باشند.

در اینجا مثال ساده‌ای از آرای فایل کوچک midi را می‌بینید.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML embed Tag</title>
<head>
<body>
<bgsound src="/html/yourfile.mid">
<noembed></noembed>
<bgsound>
<body>
<html>
```

این مثال یک صفحه‌ی خالی تولید خواهد کرد. این برچسب هیچ مولفه‌ای را آرا نمی‌کند و مخفی باقی می‌ماند.

اینترنت اکسپلورر نیز تنها سه فرمت صدا را آرا می‌کند wav فرمت داخلی کامپیوترها، au فرمت داخلی برای کارابزارهای Unix و MIDI یک کد برنامه گذاری پهانی برای موسیقی.

برچسب آبجکت در HTML

عنصر `<object>` را معرفی می کند که یک راه حل چند منظوره برای وارد کردن آبجکت عمومی ارائه می دهد. عنصر `<object>` به نویسنده `HTML` اجازه می دهد تا هر چیز لازم را با یک آبجکت برای ارائه ی آن توسط یک کاربر، تعیین کند.

در اینجا چند مثال در این رابطه می بینید.

### مثال ۱

شما می توانید یک سند `HTML` را در خود سند `HTML` ا چرا کنید.

```
<object data="data/test.htm" type="text/html" width="300" height="200"> alt <a href="data/test.htm">test.htm<a>
```

```
</object>
```

در اینجا اگر مرورگر برچسب `object` را پشتیبانی نکند، ویژگی `alt` وارد تصویر می شود.

### مثال ۲

شما می توانید یک سند `PDF` را در یک سند `HTML` ا چرا کنید.

```
<object data="data/test.pdf" type="application/pdf" width="300" height="200"> alt <a href="data/test.pdf">test.htm<a>
```

```
</object>
```

شما می توانید با استفاده از برچسب `<param>` برخی پارامترهای متناسب با سند را مشخص کنید. در اینجا مثالی از یک فایل `wav` را می بینید.

```
<object data="data/test.wav" type="audio/x-wav" width="200" height="20">
```

```
<param name="src" value="data/test.wav">
```

```
<param name="autoplay" value="false">
```

```
<param name="autoStart" value="0">
```

```
alt <a href="data/test.wav">test.wav<a>
```

```
</object>
```

### مثال ۰

شما می توانید یک سند `flash` مانند زیر ا چرا کنید.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
id="penguin" codebase="someplace/swflash.cab" width="200" height="300">
<param name="movie" value="flash/penguin.swf" />
<param name="quality" value="high" />

</object>
```

### مثال ۱

شما می توانید یک سند HTML وارد سند java applet کنید.

```
<object classid="clsid:8ad9c840-044e-11d1-b3e9-00805f499d93" width="200"
height="200">
<param name="code" value="applet.class">
</object>
```

ویژگی classid تعیین می کند که چه نسخه ای از java plug-in استفاده شود. شما می توانید از ویژگی انتخابی codebase برای مشخص کردن چگونگی دانلود JRE استفاده کنید.

## آموزش HTML Marquee

در HTML یک قطعه اسکرولینگ می باشد که یا به صورت افقی در عرض متن و یا به صورت عمودی در پایین صفحه ی وب شما نمایش داده می شود، بستگی به تنظیمات دارد و توسط برچسب <Fmarquees> مربوط به HTML ایجاد می شود.

توسط مرورگرهای زیادی پشتیبانی نشود، بنابراین HTML در تو ۶ ممکن است برچسب <marquees> توصیه می شود که به این برچسب تکیه نکنید، در عوض می توانید از CSS و javascript برای ایجاد چنین تاثیراتی استفاده کنید.

ترکیب ساده برای استفاده از برچسب <marquees> در HTML مانند زیر می باشد

```
<marquee attribute_name="attribute_value"      more attributes>
One or more lines or text message or image
</marquee>
```

## ویژگی های برچسب <marquee>

در زیر لیست مهمی از ویژگی های برچسب <marquee> را می توانید مشاهده کنید.

Attribute	Description
width	این ویژگی عرض marquee را مشخص می کند. می تواند مقداری مانند 11 یا 21 درصد باشد.
height	این ویژگی طول marquee را مشخص می کند. می تواند مقداری مانند 11 یا 21 درصد باشد.
direction	این ویژگی مسیری را که marquee باید در آن اسکرول شود، تعیین می کند. می تواند مقداری مانند up, down, left و right داشته باشد.
behavior	این ویژگی نوع اسکرول marquee را تعیین می کند. می تواند مقداری side و alternate یا scroll باشد داشته باشد.
scrolldelay	این ویژگی میزان تاخیر بین دو پرش را تعیین می کند. می تواند مقداری مانند 11 داشته باشد.
scrollamount	سرعت متن marquee را تعیین می کند. می تواند مقداری مانند 11 داشته باشد.
loop	تعداد دفعات loop را تعیین می کند. مقدار پیش فرض INFINITE می باشد که به این معناست که در marquee به طور پایان ناپذیری loop انجام می شود.
bgcolor	این ویژگی رنگ زمینه را به شکل نام رنگ یا مقدار شش تایی رنگ مشخص می کند.
hspace	این ویژگی فضای افقی اطراف marque را تعیین می کند. می تواند مقداری مانند 11 یا 21 درصد داشته باشد.

vspace	این ویژگی فضای عمودی اطراف marquee را تعیین می کند. می تواند مقداری مانند 11 یا 21 درصد داشته باشد.
--------	---

در ادامه چند مثال از استفاده‌ی برچسب marquee می بینید.

### مثال ۱

```
<!DOCTYPE html>
<html>
<head>
<title>HTML marquee Tag</title>
</head>
<body>
<marquee>This is basic example of marquee</marquee>
</body>
</html>
```

### مثال ۲

```
<!DOCTYPE html>
<html>
<head>
<title>HTML marquee Tag</title>
</head>
<body>
<marquee width="50%">This example will take only 50% width</marquee>
</body>
</html>
<!DOCTYPE html>
<html>
<head>
```

```
<title>HTML marquee Tag</title>
</head>
<body>
<marquee direction="right">This text will scroll from left to right</marquee>
</body>
</html>
```

مثال (۳)

مثال \*

```
<!DOCTYPE html>
<html>
<head>
<title>HTML marquee Tag</title>
</head>
<body>
<marquee direction="up">This text will scroll from bottom to up</marquee>
</body>
</html>
```

## سربگ HTML

یاد گرفتیم که یک نمونه داکیومنت HTML دارای ساختار زیر می باشد.

Document declaration tag

```
<html>
```

```
<head>
```

Document header related tags

```
<head>
```

```
<body>
```

Document body related tags

```
<body>
```

```
<html>
```

این فصل بزئیات بیشتری در مورد بخش سربرگ در HTML ارائه می دهد که به وسیلهٔ  
برچسب `<head>` نمایش داده می شود. برچسب `<head>` حاوی برچسب‌های مهمی می باشد  
که عبارتند از `<title>`, `<meta>`,

`<noscript>` tags. و `<link>`, `<base>`, `<style>`, `<script>`,

برچسب `<title>` در HTML

این برچسب برای تعیین تیتر داکیومنت HTML استفاده می شود. در زیر مثالی می بینید از  
ارائهٔ این تیتر به داکیومنت HTML.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>HTML Title Tag Example</title>
```

```
<head>
```

```
<body>
```

```
<p>Hello, World!</p>
```

```
<body>
```

```
<html>
```

این مثال نتیجهٔ این زیر را تولید خواهد کرد.

Hello, World!

### برچسب `<meta>` در HTML

این برچسب برای ارائهٔ متأذیتا در مورد داکیومنت HTML استفاده می شود که اطلاعاتی  
از قبیل انقضا صفحه، گردآورندهٔ صفحه، لیست کلمات کلیدی، توصیف صفحه و غیره ارائه  
می دهد.

در ادامه استفاده های مهم برچسب `<meta>` در داکیومنت HTML ارائه شده اند.

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>HTML Meta Tag Example</title>
<!-- Provide list of keywords -->
<meta name="keywords" content="C, C++, Java, PHP, Perl, Python">
<!-- Provide description of the page -->
<meta name="description" content="Simply Easy Learning by Tutorials Point">
<!-- Author information -->
<meta name="author" content="Tutorials Point">
<!-- Page content type -->
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<!-- Page refreshing delay -->
<meta http-equiv="refresh" content="30">
<!-- Page expiry -->
<meta http-equiv="expires" content="Wed, 21 June 2006 142527 GMT">
<!-- Tag to tell robots not to index the content of a page -->
<meta name="robots" content="noindex,nofollow">
<head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید می‌کند.

Hello, World!

## برچسب <base> در HTML

این برچسب برای تعیین URL پایه برای همه‌ی URL‌های وابسته در صفحه استفاده می‌شود، که به این معناست که همه‌ی URL‌های دیگر هنگامی که برای آیتم ارائه شده قرار می‌گیرند، دیگر URL‌ها به URL پایه زنجیر خواهند شد.

به عنوان مثال تمام صفحات و تصاویر ارائه شده، پس از پیشوند دار کردن URL های ارائه شده با URL پایه مسیر prefixing ستجو خواهند شد.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Base Tag Example</title>
<base href="http://www.tahlildadeh.com/" />
<head>
<body>

<a href="/html/index.htm" title="HTML Tutorial" />HTML Tutorial</a>
<body>
<html>
```

### برچسب `HTML <link>` در

این برچسب ارتباط بین داکیومنت ااری و منبع خارجی را تعیین می کند. در ادامه مثالی را می بینید از لینک یک

یک با css مسیر در و دموی خارجی sheet style فایل web root

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>HTML link Tag Example</title>
<base href="http://www.tahlildadeh.com/" />
<link rel="stylesheet" type="text/css" href="/css/style.css">
<head>
<body>
<p>Hello, World!</p>
```

```
<body>  
<html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد.

Hello, World

### برچسب **HTML** در **<style>**

این برچسب برای تعیین style sheet برای داکیومنت ئاری HTML استفاده می‌شود. در ادامه مثالی را می‌بینید از تعریف برخی قوانین style sheet در داخل برچسب **<style>**.

```
<!DOCTYPE html>  
<html>  
<head>  
<title>HTML style Tag Example</title>  
<base href="http://www.tahlildadeh.com/" />  
<style type="text/css">  
.myclass {  
background-color: #aaa; padding :10px;  
}  
<style>  
<head>  
<body>  
<p class="myclass">Hello, World!</p>  
<body>  
<html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد.

Hello, World

### برچسب **HTML** در **<script>**

این برجسب برای وارد کردن فایل اسکریپت خارجی و یا تعریف فایل اسکریپت داخلی برای داکیومنت HTML استفاده می شود. در زیر مثالی را می بینید که در آن از چاوا اسکریپت برای تعریف عملکرد ساده چاوا اسکریپت استفاده می کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML script Tag Example</title>
<base href="http://www.tahlildadeh.com/" />
<script type="text/javascript"> function Hello() {
alert("Hello, World");
}
<script>
<head>
<body>
<input type="button" onclick="Hello();" name="ok" value="OK" />
</body>
</html>
```

نشان داده می شوند، یا شاید چگونه بیان می شوند. از زمانی که کنسرسیوم در سال 1990 تاسیس شد، W3C به طور فعالانه ای در استفاده از Style sheet روی وب پیشرفت کرده است.

CSS چاگزین های ساده و موثری را برای تعیین ویژگی های مختلف برجسب های HTML ارائه می دهد. با استفاده از CSS شما می توانید تعدادی از ویژگی های طراحی را برای یک HTML ارائه شده تعیین کنید. هر ویژگی دارای نام مقدار میباشد که به وسیله ی علامت () از هم جدا شده اند. اطلاعیه ی مربوط به هر ویژگی نیز توسط علامت نقطه ویرگول (;) از یکدیگر جدا شده اند.

## مثال

ابتدا اجازه بدهید که مثالی از داکیومنت HTML را مورد بررسی قرار دهیم که برای تعیین رنگ و اندازه‌ی فونت از برچسب `<font>` استفاده می‌کند.

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>HTML CSS</title>
</head>
<body>
<p><font color="green" size="5">Hello, World!</font></p>
</body>
</html>
```

می‌توانیم با استفاده از style sheet مثال بالا را مانند زیر بازنویسی کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML CSS</title>
</head>
<body>
<p style="color:green;font-size:24px;">Hello, World!</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد.

Hello, World!

شما می‌توانید به سه روش از CSS در داکیومنت HTML خود استفاده کنید.

خاری - قوانین CSS Style sheet را در یک فایل CSS مجزا تعریف می کند، و سپس آن فایل را با استفاده از برچسب <link> در HTML وارد داکیومنت HTML شما می کند.

داخلی - قوانین CSS Style sheet را در با استفاده از برچسب <style> در بخش سربرگ داکیومنت HTML تعریف می کند.

Inline style sheet - قوانین CSS را مستقیماً و به همراه عناصر HTML با استفاده از ویژگی style تعریف می کند.

ا) ازه بدھید هر سه مورد را یکی یکی و با استفاده از مثال های مناسب بررسی کنیم.

### خاری Style sheet

اگر می خواهید از style sheet خود در صفحات مختلف استفاده کنید، توصیه می شود که یک style sheet متداول در یک فایل مجزا تعریف کنید. یک فایل cascading style sheet دارای ضمیمه هایی مانند CSS می باشد و با استفاده از برچسب <link> وارد فایل های HTML خواهد شد.

### مثال

تو چه داشته باشید که ما یک فایل style.css به نام style.css که دارای قوانین زیر می بشد، تعریف می کنیم.

```
.red{
color :red;
}

.thick{
font-size:20px;
}

.green{ Color:green;
}
```

در اینجا سه قانون CSS را تعریف کردیم که برای سه گروه متفاوت تعریف شده در برچسب های HTML مناسب می باشند. پیشنهاد می کنم در مورد چگونگی تعریف این قوانین خود

را اذیت نکنید، زیرا هنگام مطالعه‌ی CSS آنها را فرا خواهید گرفت. اکنون از اینجا بدهید از فایل CSS خارجی بالا در داکیومنت HTML زیر استفاده کنیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML External CSS</title>
<link rel="stylesheet" type="text/css" href="/html/style.css">
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد.

This is red This is thick

This is green

This is thick and green

### داخلی Style sheet

اگر می‌خواهید قوانین style sheet را برای یک داکیومنت مجزا به کار ببرید، فقط آن موقع است که می‌توانید این قوانین را با استفاده از بروچسب <style> وارد بخش سربرگ داکیومنت HTML کنید.

قوانین تعریف شده در style sheet داخلی، قوانین تعریف شده در فایل CSS خارجی را نیز در بر می‌گیرد.

## مثال

اجازه بدهید مثال بالا را یک بار دیگر بازنویسی کنیم، اما در اینجا قوانین style sheet را در همان داکیومنت HTML و با استفاده از برچسب <style> می نویسیم.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Internal CSS</title>
<style type="text/css">
.red {
Color: red;
}
.thick {
font-size :20px;
}
.green {
Color: green;
}
</style>
</head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body>
</html>
```

این مثال نتیجه‌ی زیر را تولید خواهد کرد.

This is red

This is thick This is green

This is thick and green InlineStyleSheet

شما می توانید با استفاده از ویژگی style از برچسب مربوطه، قوانین style sheet را مستقیما برای هر عنصر HTML به کار ببرید. این امر فقط زمانی می تواند انجام شود که شما علاقمند به ایجاد تغییرات خاص در هر کدام از عناصر HTML می باشید.

قوانین تعريف شده با عنصر درون خطی، قوانین تعريف شده در یک فایل CSS خارجی و نیز قوانین تعريف شده در یک عنصر <style></style> را در برمی گیرد.

اگر بدهید مثال بالا را یک بار دیگر بازنویسی کنیم، اما این بار قوانین style sheet را همراه با قوانین HTML و با استفاده از ویژگی style در عناصر خواهیم نوشت.

```
<!DOCTYPE html>
<html>
<head>
<title>HTML Inline CSS</title>
</head>
<body>
<p style="color:red;">This is red</p>
<p style="font-size:20px;">This is thick</p>
<p style="color:green;">This is green</p>
<p style="color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

# فصل هشتم: فریم ورک قدرتمند توسعه وب جنگو

## چرا جنگو؟

امروزه اکثر کسب و کارهای نوین، خدمات خود را به صورت آنلاین و از طریق وب ارائه می‌دهند؛ از بانک‌ها و فروشگاه‌های اینترنتی گرفته، تا آموزشگاه‌ها و مراجع معتبر علمی. یکی از مهم‌ترین مزایای تحت وب بودن سرویس‌ها، در دسترس بودن آن‌ها در هر مکان و در هر زمانی است. همچنین، می‌توان از طریق هر دستگاهی (نظیر موبایل، رایانه، ساعت هوشمند و...) از آن‌ها استفاده کرد.

جنگو یکی از پر استفاده‌ترین تکنولوژی‌های توسعه‌ی وب بر پایه‌ی پایتون است به طوری که در تمام فایل‌ها حتی در تنظیمات پروژه و ساخت مدل‌های داده‌ها از پایتون استفاده می‌کند. این چارچوب نرم‌افزاری (*framework*) از معماری سه‌لایه‌ی MTV استفاده می‌کند. از جنگو در ساخت بسیاری از وب‌سایت‌های مشهور مانند اینستاگرام، یوتیوب، اسپاتیفای و... می‌باشد.

در معماری MTV با قسمت‌های سروکار داشتیم؛ ولی در معماری MVC، این قسمت‌ها ب تبدیل می‌شود که Model وظیفه‌ی ارتباط با پایگاه داده، View وظیفه‌ی ارتباط با کاربر و Controller نیز وظیفه‌ی کنترل و نحوه‌ی ارتباط بین این دو View و Model را برعهده دارد.

به عبارت ساده‌تر، معماری MVC همان معماری MTV است با این تفاوت که بخش Template به تبدیل شده است.. View و View به تبدیل شده است.. Controller..

نگران اصطلاحات به کاررفته نباشید؛ زیرا در مطالب پیش رو با تمام موارد ذکر شده به صورت مفصل آشنا خواهید شد.

## مزایا جنگو

۱) ساخت راحت و سریع برنامه‌های تحت وب که پیچیدگی و ارتباط زیادی با پایگاه داده دارند؛

۲) آماده بودن بسیاری از امکانات، مانند احراز هویت، مدیریت سایت و کار با کوکی و سشن.

۳) توانایی توسعه‌ی پروژه‌های بزرگ با آن (**Scalable**) و پیاده‌سازی وب‌سایت‌های کوچک تا کسب و کارهای بسیار بزرگ با آن.

در این فصل کتاب شما با مفاهیم زیر بطور کامل آشنا خواهید شد:

- ۱) مقدمه‌ای بر جنگو: آشنایی با نحوه نصب، ساخت پروژه و ابتدایی ترین مفاهیم جنگو.
- ۲) مدل‌ها: در این فصل با مفهوم مدل، نحوه ساختن مدل‌ها و استفاده از آن‌ها آشنا می‌شوید. مدل‌ها مهم‌ترین ساختارداده‌های جنگو هستند که امکان ارتباط با پایگاهداده را برای ما فراهم می‌کنند.
- ۳) ساختن و مدیریت صفحات (ویوها): در این فصل نیز چگونگی ایجاد توابع و رویه‌های مختلف که کاربران می‌توانند در سایت ما از آن‌ها استفاده کنند را فرا خواهید گرفت.
- ۴) ارتباط با پایگاه داده: در این فصل با جنگو *ORM* آشنا خواهید شد و یاد خواهید گرفت که چگونه بدون استفاده مستقیم از پایگاه داده، اطلاعات مورد نیاز را از آن استخراج کنید.
- ۵) مدیریت ریکوئست‌ها: در این فصل به طور عمیق‌تر با نحوه مدیریت ریکوئست‌هایی که به سمت سرویس جنگو می‌آید آشنا می‌شوید و با استفاده از پایگاه داده به آن‌ها جواب می‌دهید.
- ۶) ایجاد صفحات پویا (تمپلیت‌ها): در این فصل با نمایش مناسب اطلاعات در صفحات وب به شکلی که قابلیت استفاده‌ی راحت از آن‌ها برای کاربران موجود باشد آشنا خواهید شد.
- ۷) فرم‌ها: در این فصل یاد خواهید گرفت که نحوه کار با داده‌هایی که کاربران در قالب فرم برای سرور می‌فرستند، به چه صورت است.
- ۸) شی‌گرایی در مدیریت صفحات (ویوها): در این فصل مطالب پیشرفته‌تری را در مورد ویوهای جنگو یاد می‌گیرید که مبتنی بر شی‌گرایی می‌باشند و کارایی و تمیزی کد را ارتقاء می‌بخشد.
- ۹) ادمین و یوزر: یکی از امکانات بسیار پرکاربردی که جنگو در اختیار توسعه‌دهندگان قرار داده یک بخش ادمین آمده است، که خیلی از امکانات رایج و موردنیاز هر سایتی از یک پنل ادمین را فراهم می‌کند. در این فصل با بخش ادمین جنگو آشنا می‌شوید. سیستم حساب کاربری و احراز هویت جنگو نیز مطلب دیگری است که در این فصل می‌آموزید.

## مباحث پیشرفته جنگو

در این بخش با مباحثی آشنا می‌شوید که در پروژه‌های ساده استفاده نمی‌شوند؛ اما مواردی هستند که در پروژه‌های بزرگ ضروری هستند و بدون آن‌ها ساختن یک پروژه بزرگ غیرممکن می‌شود. در این بخش Case Study‌های مختلفی هم از کاربرد جنگو در صنعت آورده شده تا با کمک آن‌ها آمادگی حداکثری برای استخدام را پیدا کنید. مباحث این بخش به شرح زیر است:

(۱) جنگو REST : معماری سرویس‌گرا نوعی معماری است که در فرایند توسعه یک نرم‌افزار تحت وب، به ما اجازه می‌دهد تا تیم توسعه قسمت سرور (بک‌اند) و تیم توسعه‌دهنده سمت کاربر (فرانت‌اند) بصورت مجزا بر روی پروژه کار کنند و حداکثر بازدهی را داشته باشند. در این فصل با کتابخانه *Django rest framework* آشنا می‌شویم که در جنگو امکان استفاده از این معماری را برای ما بسیار راحت کرده است و به ما کمک می‌کند تا بتوانیم به صورت مستقل بر بخش بک‌اند تمرکز کنیم.

(۲) تست‌نویسی: بسیاری از پروژه‌های بزرگ، به تست کردن کد قبل از ریلیز نهایی آن، توجه زیادی دارند و بدون نوشتن تست توسعه دادن یک پروژه بزرگ تقریباً غیرممکن است. در این فصل با نحوه تست یک پروژه جنگو و امکاناتی که جنگو در زمینه‌ی تست در اختیار ما قرار می‌دهد آشنا می‌شوید.

(۳) مباحث ویژه: در این فصل با مفاهیمی از جنگو آشنا می‌شوید که به نظر کوچک هستند ولی استفاده از آن‌ها کار شما را خیلی راحت می‌کند. این فصل شامل توضیحات کوهیزن و کاپلینگ، میدلور، مایگریشن و سیگنال است

(۴) اجرای Async: در این فصل با ویژگی خاصی از جنگو که که تنها در جنگو ۳ موجود است، آشنا خواهید شد.

(۵) دیپلوی پروژه: نهایی ترین کاری که برای یک پروژه وب می‌توان در نظر گرفت، قرار دادن آن بر روی یک هاست است که هر کسی که به اینترنت دسترسی داشته باشد، بتواند از پروژه وب ما استفاده کند. در این فصل با مفاهیم و چگونگی انجام یک دیپلوی مقدماتی را یاد خواهید گرفت.

## نصب جنگو

برای نصب جنگو باید از دستور زیر در ترمینال وی اس کد استفاده کنید:

```
pip install django>=4.1,<5 $
```

### ۱-۸ - مقدمه

این بخش شامل موارد زیر است:

- چرا وب فریم ورک ها: توضیحاتی درباره مزایای و ب فریم ورک ها
- معرفی محیط های مجازی: آشنایی با محیط های مجازی در پایتون و پکیج *virtualenv*
- نصب جنگو: نحوه راه اندازی جنگو و ساختن اولین پروژه
- جنگو و پایگاه داده: آشنایی با مفهوم پایگاه داده و معرفی موارد پشتیبانی شده در جنگو
- فایل های اولیه پروژه جنگو: آشنایی با فایل های ابتدایی یک پروژه جنگو
- وب سرورها: بیان مفاهیمی کلی از وب سرورها
- فایل تنظیمات: آشنایی با بعضی از تنظیمات مهم یک پروژه جنگو
- ریکوئست **HTTP**: آشنایی با مفهوم ریکوئست *client-server* و معماری **HTTP**
- اپ های جنگو: آشنایی با مفهوم اپ در جنگو و نحوه ساختن آن ها

## چرا وب فریم ورک ها

فرض کنید می خواهیم نرم افزاری بسازیم که از طریق وب قابل دسترسی باشد و بتوان با آن کار کرد. در این صورت علاوه بر بخش اصلی برنامه، به انبوهی از قابلیت های اضافی نیاز است. به عنوان مثال باید کدی نوشته شود که بتواند با پایگاه داده کار کند، همچنین کدی نیاز است که امنیت برنامه می را تضمین کند و ...

این قابلیت های اضافی، در همه وب اپلیکیشن ها نیاز هستند و جامعه برنامه نویسان نیازمند به راه کاری واحد برای حذف دوباره کاری و تکرار این قابلیت ها در هر پروژه بود. و ب فریم ورک (*Web framework*), ابزاری است که در توسعه و راه اندازی وب اپلیکیشن ها به توسعه دهنده گان،

در همین مسیر کمک می‌کند به طوری که میزان قابل توجهی زمان و هزینه‌ی توسعه را کاهش می‌دهد. در گذشته، تمامی وب‌اپ‌ها از ابتدا توسط شخص توسعه‌دهنده پیاده‌سازی می‌شد. از معایب این روش می‌توان به زمان زیاد صرف کدنویسی و باگ‌های زیاد در امنیت اشاره کرد.

اما ظهور وب‌فریمورک‌ها پاسخی شد برای تمام معایب ذکر شده:

- اکنون بخش‌های مشترک وب‌اپ‌ها بصورت پکیج‌های آماده در هنگام توسعه در دسترس توسعه‌دهنده‌گان هستند.
- با وجود فریمورک‌ها، ساختار پروژه حتی برای توسعه‌دهنده‌گانی که در آینده به پروژه می‌پیوندد قابل درک‌تر خواهد بود.
- در فریمورک‌ها تا میزان خوبی برای پیش‌گیری از به وجود آمدن باگ‌ها، راه حل‌های مطمئن در نظر گرفته شده‌است.

احتمالاً تا الان به اهمیت وجود فریمورک‌ها در دنیای وب پی برده‌اید. اگر هم هنوز قانع نشده‌اید، مثال زیر را بادقت دنبال کنید. فرض کنید می‌خواهید قسمتی از کدتان را که مربوط به اتصال به پایگاه داده است، کمی عوض کنید. برای این کار باید در هزاران خط کد از پروژه‌تان نیز این تغییر را اعمال کنید. این بار این فکر به سرتان می‌زنند که از توابع کمکی‌ای که خودتان به صورت جداگانه در فایلی نوشته‌اید، استفاده کنید تا دچار عذاب تغییر هزاران خط کد نشوید!

این کار به تدریج باعث بزرگ و بزرگ‌تر شدن فایل موردنظر شده و هر چه جلوتر می‌روید به توابع کمکی بیشتری نیاز پیدا می‌کنید. اگرچه شما نیز با این کار تقریباً در حال پیاده‌سازی یک فریمورک شخصی هستید، استفاده از یک فریمورک که چندصد نفر طی چند سال بر قسمت‌های مختلف آن کار و تحت شرایط مختلفی آن را تست کرده‌اند تنها راه نجات‌تان از گردداب بی‌نهایت توابع کمکی خواهد بود!.

پس در واقع وب‌فریمورک مجموعه‌ای از کتابخانه‌های (Library) که نیاز برنامه‌نویس را برای انجام کارهای تکراری به حداقل می‌رساند.

## معرفی محیط مجازی

این اتفاق برای توسعه‌دهندگان پایتون زیاد می‌افتد که همزمان در یک سیستم، بر روی چند پروژه‌ی پایتون کار کنند که هر یک از این پروژه‌ها، پکیج‌های موردنیاز خاص خود را داشته باشند. حتی گاهی دو پروژه با پکیج‌های موردنیاز یکسان، ممکن است از ورژن‌های متفاوتی از این پکیج‌ها استفاده کنند.

امروزه برای حل این مشکلات (*Conflicts*) راه‌های متنوعی وجود دارد، اما خود پایتون کتابخانه‌هایی دارد که این کار را با ایجاد یک محیط مجازی (*Virtual environment*) او ایزوله برای ما انجام می‌دهد. با کمک این کتابخانه می‌توان از پکیج‌های یکسان با ورژن‌های مختلف در پروژه‌های مجزا استفاده کرد. که یکی `virtualenv`

```
pip install virtualenv
```

```
python3 -m virtualenv parsa_env
```

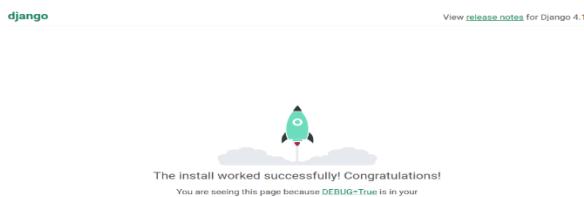
```
.\parsa_env\Scripts\activate.bat
```

اکنون می‌توانید اولین پروژه جنگو خود را بسازید

```
django-admin startproject first_project
```

```
python manage.py runserver
```

پس از اجرای این دستور اصطلاحاً پروژه شما ران می‌شود و با لینکی که در اختیارتان می‌دهد می‌توانید وارد نتیجه زیر را مشاهده کنید.



## جنگو و پایگاه های داده

به طور مفصل در فصل چهارم این کتاب با پایگاه های داده آشنا شدیم اما پایگاهدادهای که جنگو به صورت پیشفرض بر روی آن تنظیم شده، *SQLite* نام دارد. *SQLite* یک پایگاهداده ساده و سریع است که استفاده از آن در محیطهای پروداکشن به هیچ عنوان توصیه نمی شود ولی برای کاربردهای آموزشی ما عملکرد خوبی دارد. اکنون می خواهیم این پایگاهداده را برای پروژه ای که در قسمت قبل ساخته ایم، ایجاد کنیم..

ابتدا وارد پوشه‌ی پروژه شوید و در جایی که فایل *manage.py* قرار دارد، خط فرمان *(Command line)* سیستم خود را باز کنید. اکنون دستور زیر را وارد کنید.

```
python manage.py migrate
```

خروجی شما باید به شکل زیر باشد:

```
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

اکنون اگر محتويات پوشه‌ی پروژه خود را ببینید، فایلی جدید با اسم *db.sqlite3* را مشاهده می کنید که همان پایگاهداده پیشفرضی گفته شده است. در واقع پس از اجرای این فرمان، جنگو بررسی می کند که آیا قبل از آن را برای شما ایجاد کرده است یا خیر. در صورتی که قبل از این کار را برایتان کرده باشد دیگر این کار را تکرار نمی کند و از همان پایگاهداده قبلی شما

استفاده خواهد کرد. همه‌ی این نکات کوچک، هوشمندی و سادگی جنگو را به ما نشان می‌دهند.

فایل‌های اولیه پروژه جنگو به صورت زیر است :

root

```

├── first_project
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

```

## ۲-۸- داکر

با وجود تمامی پیشرفت‌ها در برنامه نویسی مدرن، پیکربندی اصولی یک محیط توسعه محلی همچنان یک چالش بزرگ است. در یک پروژه فاکتورهای متعددی از قبیل کامپیوترهای مختلف، سیستم عامل‌های مختلف، نسخه‌های گوناگون جنگو، موارد مختلفی از یک محیط مجازی و غیره وجود دارد. اما چالش زمانی بزرگتر می‌شود که باید در یک محیط تیمی کار کنیم که همه افراد به پیکربندی یکسانی [از محیط توسعه] نیاز دارند.

راه حلی بنام داکر در سالهای اخیر پدیدار شده است. اگر چه عمری از پیدایش آن نمی‌گذرد، اما بسرعت تبدیل به گزینه‌ی اصلی برای توسعه دهنده‌گانی شد که در سطح تولید کار می‌کنند. سرانجام، با استفاده از داکر می‌توان یک محیط توسعه‌ی محلی قابل اعتماد و منسجمی بوجود آورد که از ورژن مدنظر پایتون و نصب جنگو گرفته تا سرویس‌های مازادی در کنار آنها نظیر پایگاه داده‌ها را شامل می‌شود. این به این معناست که دیگر مهم نیست شما از چه سیستم عاملی (مک، لینوکس، ویندوز) استفاده می‌کنید، چرا که همه چیز در خود داکر در حال اجراست. همچنین داکر همکاری در محیط‌های تیمی را رفته رفته آسان‌تر می‌کند. آن زمان که از فایل‌های طولانی و قدیمی README برای افزودن یه محیط توسعه جدید در پروژه‌های گروهی استفاده می‌کردیم دیگر گذشته است. در عوض داکر این امکان را میدهد که به

سادگی دو فایل Dockerfile و docker-compose.yml را به اشتراک بگذارید و توسعه دهنده میتواند اطمینان داشته باشد که محیط توسعه محلی او همانند سایر اعضای تیم است.

داکر یک فناوری کامل نیست و نسبتاً نوپا است که زیر ساخت های آن در حال توسعه هستند. اما میتوان این نوید را داد که یک محیط توسعه سازگار و قابل اشتراک است که قادر است بصورت محلی روی هر رایانه یا سرویس اجرا شود که همین موضوع آن را به انتخابی مناسب تبدیل میکند. در این فصل، کمی بیشتر درباره داکر و داکرایز کردن (dockerize) اولین پروژه جنگو می آموزیم.

## داکر چیست

داکر ابزاری است که سیستم عامل نصبی روی سیستم شما را بوسیله کانتینر های لینوکسی از سایر اجزا مجزا میکند. این روشی برای مجازی سازی میباشد.

ریشه مجازی سازی به ابتدای علوم کامپیوتر برمیگردد، زمانی که ابر کامپیوتر ها رایج بودند. این سوال بوجود آمد که "چند برنامه نویس چگونه میتوانند از یک رایانه بطور همزمان استفاده کنند؟" پاسخ، مجازی سازی بود؛ بالاخص ماشین های مجازی که نسخه کاملی از کامپیوتر ها به همراه سیستم عامل آنها بود.

اگر شما یک فضای ابری در سرویس های ارائه دهنده مثل آمازون تهیه کنید، این چنین نیست که یک سخت افزار اختصاصی به شما ارائه دهنده. در عوض شما هستید که یک سرور فیزیکی را با سایر مشتریان به اشتراک میگذارید. اما چونکه مشتریان ماشین های مجازی خود را روی سرور آمازون اجرا میکنند، بنظر میرسد هر کس برای خودش یک سرور اختصاصی دارد. این فناوری امکان افزودن یا حذف سرور از فضای ابری را ممکن میسازد. این فناوری تا حدی بسیار زیادی توسط نرم افزار ها پشتیبانی میشوند و سخت افزار ها بطور کامل در این تغییرات دخیل نیستند.

و اما! نقطه ضعف ماشین های مجازی چیست؟ اندازه و سرعت دو تا از چالش های هستند. یک سیستم عامل معمولی به راحتی میتواند ۷۰۰ مگابایت حجم داشته باشد. بنابراین اگر یک سرور فیزیکی از سه ماشین مجازی (۳×۷۰۰) ۲،۱ گیگابایت از فضای دیسک بعلاوه ی نیاز های سی پی یو. منابع حافظه اشغال میشود.

خوب راه حل چیست؟ از داکر استفاده کنید. ایده اصلی این است که اکثر رایانه ها از یک سیستم عامل لینوکس استفاده میکنند. حال اگر مجازی سازی را از لایه های بالایی لینوکس شروع کنیم چه میشود؟ (منظور این است که از هسته اصلی لینوکس شروع نکنیم) آیا حجم کمتر و سرعت بیشتری ارائه نمیشود. راه حلی برای تکرار عملکرد های مشابه در پروژه است؟ پاسخ بله است. راه حل این چالش ها کانتینر های لینوکسی هستند که در سالهای اخیر بسیار محبوب شده اند. برای برنامه نویس ها بویژه وب اپلیکیشن ها (همچون جنگو) ماشین های مجازی و کانتینر های لینوکسی منابعی بیش از آنچه که نیاز است ارائه میدهند. این اساساً همان چیزی است که داکر ارائه میدهد: راهی برای پیاده سازی کانتینر های لینوکسی.

بهترین تشیبیهی که میتوانیم اینجا بکار ببریم، محله و آپارتمان هستند. فرض کنید ماشین مجازی همان محله است. در هر محله ساختمان های مجزا از هم با زیر ساخت های خاص خود وجود دارد. از جمله لوله کشی، سیستم گرمایش، حمام و آشپزخانه. کانتینر های ماشین مجازی همان ساختمان ها هستند که در لوله کشی و سیستم گرمایش مشترک هستند، اما ظرفیت هر کدام از این بخش ها در ساختمان های مختلف بسته به نیاز مالک ساختمان و تعداد خانوار ها متفاوت است.

### کانتینرها در مقایسه با محیط های مجازی

شما بایستی از قبل بعنوان یک برنامه نویس پایتون با مفهوم محیط های مجازی که راهی برای ایزوله کردن پکیج های پایتونی هستند آشنا باشید. جا داره از این محیط مجازی یه تشکری کنیم (با وجود محیط مجازی در یک کامپیوتر میتوانیم بصورت محلی چند پروژه را اجرا کنیم. مثلاً فرض کنید یک پروژه از پایتون نسخه ۳,۴ و جنگو ۱,۱۱ استفاده میکند، در حالی که پروژه دیگر از پایتون ۳,۸ و جنگو ۳,۱ بهره گرفته است. با ایجاد یک محیط مجازی اختصاصی برای هر یک از این دو پروژه میتوان پکیج های متفاوت را در حین آلوده نکردن سیستم کامپیوتری [که بخار نصب نسخه های مختلف از یک پکیج ایجاد میشوند] مدیریت کرد).

اما اساساً همه اینها یک کار انجام میدهند. مهم ترین فرق بین محیط های مجازی و داکر این است که، محیط های مجازی فقط میتوانند از پکیج های پایتونی پشتیبانی کنند. مثلاً قابلیت نصب برنامه هایی که پایتونی نیستند (مثل PostgreSQL یا MySQL ندارند. چرا که این

برنامه ها بايستی در سیستم اصلی کامپیوتر شما بصورت محلی نصب باشند. به بیانی دیگر، محیط های مجازی فقط و فقط به اشاره به پایتون و هر آنچه که از جنس پایتون است دارد و خود به تنهایی شامل این موارد نیست. کانتینر های لینوکسی یک قدم فرا تر رفته، نه فقط بخش های مربوط به پایتون را بلکه کل سیستم عامل و هر چیزی که در آن نصب است را تفکیک میکند. مثلا هم میتوان پایتون و موارد مربوط به نوع دیتا بیس [از قبیل MySQL] را در داکر نصب و اجرا کنیم.

داکر به خودی خود موضوعی پیچیده است و ما در این کتاب عمیقاً آن را بررسی نمیکنیم. درک مفاهیم اولیه و نکات کلیدی آن مهم است.

### نصب داکر

برای نصب داکر به سایت داکرهاب مراجعه کنید. وقتی که داکر نصب شد دستور زیر را وارد می کنیم:

```
$ docker --version
# Docker version 19.03.12, build 48a66213f
```

بعضی اوقات داکر از یک ابزار جانی به اسم Docker Compose برای کمک به اجرای خودکار دستورات استفاده میشود. Docker Compose را برای مک و ویندوز میتوانید دانلود کنید اما اگر از لینوکس استفاده میکنید، بايستی به صورت دستی آنرا نصب کنید. این کار را میتوانید با اجرای دستور sudo pip install docker-compose پس اینکه نصب داکر تمام شد انجام دهید.

داکر یک Image مخصوص به خود را دارد که به عنوان اولین اجرا می تواند مفید باشد.

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f
2a5301c00
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
```

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

<https://hub.docker.com/>

For more examples and ideas, visit:

<https://docs.docker.com/get-started/>

دستور docker info این امکان را به ما می دهد تا به هر آنچه در داکر است جستجو کنیم.

```
$ docker info
```

Client:

Debug Mode: false

Server:

Containers: 1

Running: 0

Paused: 0

Stopped: 1

Images: 1

تمامی این خطوط نشان می دهد که داکر با موفقیت نصب شده است.

حالا یک پروژه جنگو به اسم Hello world ایجاد می کنیم.

روی سیستم شما اجرا می‌شود و آن را به داکر منتقل می‌کنیم و می‌بینیم که چطور همه چیز با هم درست کار می‌کند

اولین قدم انتخاب یه مکان برای قرار دادن کد ها می‌باشد. این قسمت می‌تواند هر قسمی از سیستم شما باشد.

```
$ cd Desktop
$ mkdir code && cd code
$ mkdir hello && cd hello
$ pipenv install django~=3.1.0
$ pipenv shell
(hello) $
(hello) $ django-admin startproject config.
(hello) $ python3 manage.py migrate
(hello) $ python3 manage.py runserver
```

## اپ pages

هم اکنون ما یک صفحه خالی در دسکتاپ و یک اپ اختصاصی Pages. تعریف می‌کنیم

```
# config/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages', # new
]
```

حال مسیر Url. را برایش تعریف می‌کنیم

```
# config/urls.py
```

```
from django.contrib import admin
from django.urls import path, include # new

urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('pages.urls')), # new
]
```

در این مرحله بجای تنظیم یک قالب می توانیم یک پیام هارد کد در در فایل ویو که خروجی رشته را می دهد انجام دهیم.

```
# pages/views.py
from django.http import HttpResponse
```

```
def home_page_view(request):
    return HttpResponse('Hello, World!')
```

ما اخرين مرحله يك فایل `home_page_view` در اپ `pages`.py ایجاد کردیم و آنرا `urls.py` متصل کردیم  
`(hello) $ touch pages/urls.py`

```
# pages/urls.py
from django.urls import path
from .views import home_page_view
```

```
urlpatterns = [
    path("", home_page_view, name='home')
]
```

صفحه را دوباره راه اندازی می کنیم.

```
(hello) $ python manage.py runserver
```

اکنون اگر مرورگر خود را رفرش کنید پیام مورد نظر برای شما ارسال می شود  
 سرور را متوقف کرده و از محیط مجازی خارج می شویم `Control+c` حال به سراغ داکر می رویم. با زدن

```
(hello) $ exit  
$  
$ python manage.py runserver  
File "./manage.py", line 14  
    ) from exc  
    ^  
SyntaxError: invalid syntax
```

### ایمیج ها، کانتینر ها و میزبانی داکر

اجرا می شود. شامل دستور عمل های یک ایمیج میباشد. یک کانتینر به عنوان مثال ۰ ایمیج داکر اجرا میشود. به مثال آپارتمان بر میگردیم، ایمیج یک طرح یا مجموعه ای از طرح های آپارتمان است. کانتینر هم ساختمان واقعی و کاملا ساخته شده است.

سومین مفهوم، میزبانی داکر است که سیستم عاملی اساسی است. حتی ممکن هست شما چند کانتینر توسط میزبان داکر اجرا کنید. وقتی میخواهیم توسط داکر کد بزنیم یا روندی را انجام دهیم، یعنی آنها توسط میزبانی داکر اجرا میشوند.

```
$ touch Dockerfile  
# Pull base image  
FROM python:3.8  
# Set environment variables  
ENV PYTHONDONTWRITEBYTECODE 1  
ENV PYTHONUNBUFFERED 1  
# Set work directory  
WORKDIR /code  
# Install dependencies  
COPY Pipfile Pipfile.lock /code/  
RUN pip install pipenv && pipenv install --system  
# Copy project  
COPY . /code/
```

همچنین Dockerfile از بالا به پایین وقتی ایمیج ساخته میشود خوانده میشوند. اولین قسمت باشد FROM دستور تا ایمیج اصلی را برای ایمیج ما فراخوانی کند. این بار، Python 3.8.

سپس از دستور ENV برای تعریف دو محیط استفاده میکنیم.

- **PYTHONUNBUFFERED** اطمینان می دهد که خروجی کنسول ما آشنا به نظر می رسد و توسط داکر بافر نمی شود، که ما نمی خواهیم.

- **PYTHONDONTWRITEBYTECODE** به این معنی که پایتون سعی نمی کند فایل های `crypt` بینویسد که ما نیز انجام می دهیم میل ندارد.

مرحله بعد استفاده از **WORKDIR** برای تنظیم مسیر پیش فرض فهرست کار در ایمیج خود به نام `code` که کد استفاده می کنیم جایی که ما کد خود را ذخیره می کنیم اگر این کار را نکردیم، هر بار می خواستیم دستورات را اجرا کنیم کانتینر ها باید در مسیری طولانی تایپ کنیم. در عوض، داکر فرض میکند که ما میخواهیم تمام دستورات را داخل پوشه اجرا کنیم.

برای وابستگی ها، از Pipenv استفاده میکنیم. پس دو فایل `Pipfile` و `Pipfile.lock` را داخل پوشه `code` داخل داکر کپی میکنیم.

ارزش داره که توضیح بدیم که چرا Pipenv `Pipfile.lock` را درست میکند. مفهومون فایل `lock` هنوز یونیک نیست داخل پایتون و Pipenv. در واقع در پکیج منیجر ها قابل مشاهده است برای زبان های محبوب. مثل `Gemfile.lock` و `yarn.lock` و `Composer.lock`. اولین و محبوب ترین پروژه ای بود که این ها را داخل پایتون قرار داد.

مزایای فایل `lock` این هست که منجر به نصب کامل میشوند. مهم نیست چند بار پکیج را نصب میکنید، همیشه نتیجه یکسان خواهد بود. بدون این فایل ها وابستگی ها و ترتیب ها اینطور نیست. اعضای تیمی که لیست یکسانی از بسته های نرم افزاری را نصب می کنند ممکن است ساختار کمی متفاوت داشته باشند.

وقتی که از داکر استفاده میکنیم، چه در سیستم شخصی و چه در جای دیگه، و بروزسازی بسته های نرم افزاری، احتمال درگیری Pipfile.lock وجود می آید. در فصل بعد این موضوع را بررسی میکنیم.

برگردیم، ما از دستور RUN استفاده کردیم که اول Pipenv را نصب کنیم تا پکیج های لیست شده داخل Pipfile.lock را نصب کند. فعلا فقط جنگو، خیلی مهم هست که از استفاده کنید تا یک محیط مجازی را درست کند. اما ولی از داکر استفاده میکنیم نیازی به محیط مجازی نیست.

در این حالت، کانتینر داکر ما محیط مجازی و موارد دیگر ما میباشد. بنابر این از system-- استفاده میشود تا مطمئن شویم که بسته های ما در تمام سیستم نصب میشود. برای آخرین مرحله ما پوشه code خود را به پوشه code داکر انتقال میدهیم. ابتدا Pipfile و Pipfile.lock و بعد بقیه را؟

دلیل این است که ایمیج بر اساس دستور العمل های بالا به پایین ایجاد می شوند، بنابر این ما می خواهیم چیزهایی که اغلب تغییر می کنند مانند کد محلی ما آخرین باشند. به این ترتیب ما فقط باید آن قسمت از ایمیج را هنگام تغییر ایجاد کنیم، نه اینکه هر بار که تغییری رخ می دهد همه چیز را دوباره نصب کنیم. و از آنجا که بسته های نرم افزاری موجود در Pipfile ما به ندرت تغییر می کنند، منطقی است که آنها را کپی کرده و زودتر نصب کنیم.

دستور العمل های ایمیج ما اکنون انجام شده است، بنابر این اجازه دهید image را با استفاده از دستور docker build بسازیم. نقطه، .. نشان می دهد که فهرست فعلی محل اجرای دستور است. در اینجا خروجی زیادی وجود خواهد داشت. به همین دلیل فقط چند خط آن را وارد کردیم.

```
$ docker build.
```

Sending build context to Docker daemon

Step 1/7 : FROM python:3.8

3.8: Pulling from library/python

...

Successfully built 8d85b5d5f5f6

فایل ما شامل کدهای زیر خواهد شد

version: '3.8'

services:

web:

build:

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

-./code

ports:

- 8000:8000

در خط بالا ما ورژن داکر را مشخص میکنیم که در حال حاضر ۳,۸ میباشد. با ورژن حال پایتون که ۳,۸ هست اشتباه نگیرید. کاملاً تصادفی است!

سپس اطلاعات کانتینر را مشخص میکنیم که در داکر هاست اجرا میشود. امکان اجرای چند کانتینر نیز وجود دارد اما ولی فعلاً یک کانتینر را توضیح میدهیم. مشخص میکنیم که کانتینر چطور ساخته، به پوشه فعلی، نگاه کن برای Dockerfile. بعد داخل کانتینر وب سرور را اجرا کن.

ظرفیت پایه به صورت خودکار فایل داکر را توسط کامپیوتر ما همگام می سازد. این به این معنی است که ما نیاز نداریم هر زمان که یک فایل را تغییر می دهیم image، را دوباره بسازیم. در آخر ما پورت خروجی را در داکر مشخص میکنیم که پورت 8000 خواهد بود که همان پورت دیفالت جنگو میباشد.

اگر اولین بار است که از docker استفاده می کنید، به احتمال زیاد الان سردرگم شده اید. اما نگران نباشید. ما چندین ایمیج داکر و کانتینر ها (containers) در طول فصل های این کتاب خواهیم ساخت که با تمرین کردن مهارت لازم را به دست می آورید. Dockerfile و docker-compose.yml مشابه در پروژه ها استفاده.

قدم آخر هم اجرای دستور docker-compose up این دستور باعث یک خروجی طولانی دیگر در کامند لاین می باشد..می

```
$ docker-compose up
Creating network "hello_default" with the default driver
Building web
Step 1/7 : FROM python:3.8
...
Creating hello_web_1... done
Attaching to hello_web_1
web_1 | Watching for file changes with StatReloader
web_1 | Performing system checks...
web_1 |
web_1 | System check identified no issues (0 silenced).
web_1 | August 03, 2020 - 19:28:08
web_1 | Django version 3.1, using settings 'config.settings'
web_1 | Starting development server at http://0.0.0.0:8000/
web_1 | Quit the server with CONTROL-C.
```

ب داخل مرورگر وارد Hello, World. جنگو هم اکنون با داکر اجرا میشود. ما در یک محیط مجازی کار نمیکنیم، ما حتی دستور runserver را هم اجرا نکردیم. تمام کارهای پروژه روی یک وب سرور مستقل داکر اجرا میشود. موفقیت را نشان می دهد.

```
. $ docker-compose down
Removing hello_web_1... done
Removing network hello_default
```

### نتیجه‌گیری

داکر یه محیط کاملا مستقل شامل تمام چیزایی هست که ما برای توسعه لوکال نیاز داریم. وب سرویس، دیتابیس و حتی بیشتر. الگو کلی یک پروژه جنگو یکسان میباشد.

## postgre sql -۸-۳

یکی از تفاوت های اصلی بین یک اپلیکیشن ابتدایی و یک اپلیکیشن آماده به کار (production-ready) جنگو، در دیتابیس های آنان است. جنگو بدلیل راحت بودن، سریع بودن و SQLite بودن file-based، از این دیتابیس بصورت پیشفرض برای توسعه ای محلی (local development) بهره می برد و آن را به گزینه ای مناسبی تبدیل می کند. علاوه بر آن، این دیتابیس نیاز به هیچ گونه نصب و پیکربندی ندارد.

هرچند این دیتابیس معایب خاص خود را دارد. بطور کلی، SQLite دیتابیس مناسبی برای سایت های رده بالا و حرفه ای نیست. اما برای پیاده کردن ایده های اولیه این دیتابیس می تواند پاسخ گو باشد. بطور کلی SQLite بصورت خیلی کم و محدود برای پروژه های بزرگ مورد استفاده قرار می گیرد.

جنگو چهار دیتابیس را پشتیبانی می کند: Oracle، PostgreSQL، MySQL و SQLite. ما در این کتاب از PostgreSQL بدلیل معروف بودن آن استفاده خواهیم کرد. زیبایی Django ORM در آن است که اگر حتی از Oracle یا MySQL به جای PostgreSQL استفاده کنیم، تفاوتی در کد ما ایجاد نمی کند. Django ORM کار ترجمه کد از زبان برنامه نویسی پایتون، به زبان دیتابیس ها را به راحتی مدیریت می کند و این بسیار شگفت انگیز است.

چالشی که این سه دیتابیس برای ما ایجاد می کنند این است که اگر بخواهید یک production environment را بر روی کامپیوتر خود (local computer) ایجاد کنید؛ باید هر کدام از این دیتابیس ها را نصب و بصورت محلی (local) آن ها را اجرا نمایید و ما دقیقاً می خواهیم همین عمل را انجام دهیم. در حالی که جنگو جزیات سوییچ بین دیتابیس ها را مدیریت می کند اما زمانی که شما از SQLite برای توسعه ای محلی (local development) و از دیتابیسی دیگر برای محصول نهایی استفاده می کنید؛ ممکن است به باگ های خیلی کوچک و اجتناب ناپذیر برخورد کنید که پیدا کردن و رفع آن ها ممکن است شما را به دردسر بیندازد. بنابراین استفاده از یک دیتابیس هم برای توسعه ای محلی و هم برای محصول نهایی راه حل بهتری است.

در این فصل ابتدا می یک پروژه ای جنگو را با استفاده از دیتابیس SQLite توسعه می دهیم و سپس بر روی مباحث Docker و PostgreSQL سوییچ می کنیم.

```
$ cd ..  
$ mkdir postgresql && cd postgresql  
$ pipenv install django==2.2.7  
$ pipenv shell  
(postgresql) $ django-admin startproject postgresql_project.  
(postgresql) $ python manage.py migrate  
(postgresql) $ python manage.py runserver  
(postgresql) $ ls  
Pipfile  Pipfile.lock  db.sqlite3  manage.py  postgresql_project.  
# Pull base image  
FROM python:3.7  
  
# Set environment variables  
ENV PYTHONDONTWRITEBYTECODE 1  
ENV PYTHONUNBUFFERED 1  
  
# Set work directory  
WORKDIR /code  
  
# Install dependencies  
COPY Pipfile Pipfile.lock /code/  
RUN pip install pipenv && pipenv install --system  
# Copy project  
COPY . /code/  
version: '3.7'  
  
services:  
web:
```

build:.

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

-./code

ports:

- 8000:8000

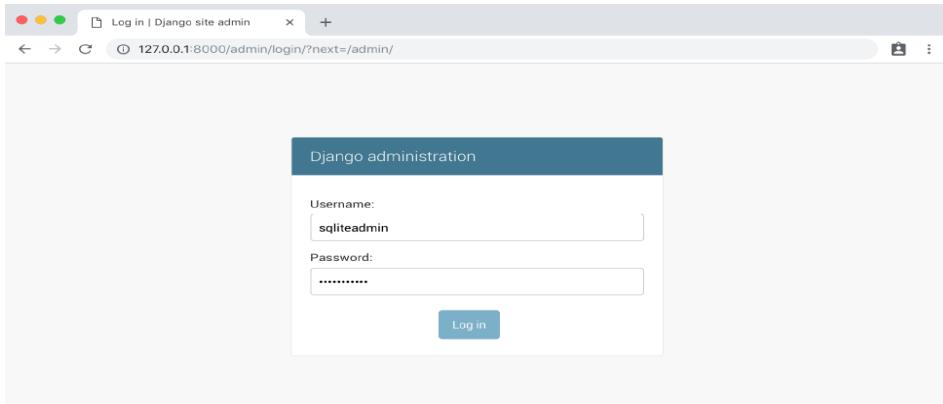
حال تفکیک شده، container را در پس زمینه اجرا می کند. این بدان معنی است که ما می توانیم تنها از یک command line استفاده کنیم، بدون این که نیازی به command line دیگری باشد. این مهم باعث می شود که وقت ما بیهوده صرف سوییچ کردن بین command line ها نشود. از طرفی دیگر بدی این روش این است که اگر اروری بوجود آید؛ این ارور همیشه نمایش داده نمی شود. بنابراین اگر، در برخی موارد، صفحه i نمایش کامپیوتر شما با این کتاب همخوانی نداشت؛ با تایپ کردن logs docker-compose خروجی فعلی را چاپ کنید و خطاهای ارور را برطرف نمایید.

به احتمال زیاد شما با پیام Warning: Image for service web was built because it did not already exist در زیر command مواجه خواهید شد. داکر بصورت اتوماتیک یک image درون container ساخته است. همانطور که در ادامه i این کتاب خواهیم دید؛ اضافه کردن فلک--build، زمانی که پکیج های نرم افزار (software package) آپدیت هستند، لازم است. چرا که داکر، بصورت پیش فرض، بدنبال کپی حافظه i محلی (local cached copy) نرم افزار می گردد و از آن برای ارتقا عملکرد استفاده می کند.

برای این که اطمینان پیدا کنیم که همه چیز بدرستی کار می کند به آدرس <http://127.0.0.1:8000/> در درون مرورگرتان برگردید؛ صفحه را ریفرش کنید تا دوباره با صفحه i خوش آمد گویی جنگو مواجه شوید.

\$ docker-compose exec web python manage.py createsuperuser

پس از اینکار به قسمت ادمین در آدرس <http://127.0.0.1:8000/admin> بروید و سپس وارد شوید



سپس به صفحه ادمین هدایت می شوید.

کلیک کنید به صفحه یوزرها هدایت می شوید و آنجا می توانید بگویید تنها یک یوزر ساخته شود! اگر به دکمه Users

- تا به اینجا ما در حال آپدیت کردن دیتابیس db.sqlite3 در داکر بودیم.
- حال زمان آن رسیده برای ادامه پروژه sql سوییچ کنیم. این کار سه مرحله دارد :
- به سمت
  - وفق دهنده ی دیتابیس (PostgreSQL) Psycopg2 (database adapter) را نصب کنید. با این کار می توانید با PostgreSQL ارتباط برقرار کند.
  - کانفیگ DATABASE را در فایل settings.py آپدیت کنید.
  - PostgreSQL را بصورت محلی اجرا نمایید.

حالا اجرای docker-compose down را با استفاده از docker container متوقف کنید.

```
$ docker-compose down
Stopping postgresql_web_1... done
Removing postgresql_web_1... done
Removing network postgresql_default
```

سپس در درون فایل docker-compose.yml یک سرویس به نام db اضافه نمایید. این بدان معنی است که دو سرویس کاملاً جداگانه وجود خواهد داشت، هر کدام دارای یک container که در درون Docker host اجرا می شوند: سرویس web برای سرور محلی Django و سرویس PostgreSQL برای دیتابیس db

PostgreSQL دارای آخرین ورژن است؛ یعنی ورژن ۱۱. اگر ورژن خاصی را مشخص نکنیم و به جای آن تنها از postgres استفاده کنیم؛ آخرین نسخه ی PostgreSQL دانلود خواهد شد. بنابراین در آینده ممکن است PostgreSQL نسخه ی ۱۲ برای شما نصب گردد که پیش نیاز های دیگری خواهد داشت.

در آخر، خط depends-on را به سرویس web اضافه خواهیم کرد؛ چرا که این سرویس بنا بر نوع دیتابیس اجرا می گردد. این بدان معنی است که db قبل از web شروع بکار خواهد کرد.

```
version: '3.7'
```

services:

web:

build:..

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

-./code

ports:

- 8000:8000

depends\_on:

- db

db:

image: postgres:11

حالا -d docker-compose up را اجرا نمایید. با این کار، docker image ما خواهد rebuild شد و دو container را راه اندازی می کند. یکی PostgreSQL را در db اجرا می کند و دیگری جنگو web سرور را اجرا می کند. در اینجا مهم است به این نکته دقت کنیم که production database هایی نظیر PostgreSQL file-based نیستند. این دیتابیس کاملاً در سرویس db اجرا شده و بصورت موقت می باشد. زمانی که docker-compose down را اجرا می کنیم، تمام دیتای درون آن از بین می رود. این کاملاً بر خلاف کد ما در درون web container می باشد که دارای یک سوار بر سینک محلی (sync local) و کد داکر است. در فصل بعدی یاد خواهیم گرفت که چگونه volumes mount را برای سرویس db اضافه نماییم که اطلاعات دیتابیس را مقاوم نگه دارد.

```
# postgresql_project/settings.py
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

}

جنگو بصورت پیش فرض از sqlite3 به عنوان موتور دیتابیس (database engine) استفاده می کند. آن را db.sqlite3 نام گذاری کنید و در BASE\_DIR قرار دهید؛ که این دایرکتوری دایرکتوری پروژه می باشد.

از آن جایی که ساختار دایرکتوری بگونه ای است که باعث سردرگمی می شود، ضمن این نکته لازم است که دایرکتوری پروژه جایی است که فایل های postgresql\_project, Pipfile.lock و Pipfile در آن واقع هستند.

```
(postgresql) $ ls
```

```
Dockerfile Pipfile.lock docker-compose.yml postgresql_project
```

```
Pipfile db.sqlite3 manage.py
```

برای این که به PostgreSQL سوییچ کنیم، ما کانفیگ ENGINE را آپدیت خواهیم کرد. داده های PostgreSQL موارد HOST، USER، NAME، PASSWORD را از شما می خواهد.

برای راحتی عبارت postgresql را برای سه مورد اول وارد نمایید؛ که اسم service set در فایل docker-compose.yml می باشد را رو بروی HOST وارد نمایید. در آخر PORT را ۵۴۳۲ قرار دهید؛ که port پیش فرض PostgreSQL می باشد.

```
# postgresql_project/settings.py
```

```
DATABASES = {
```

```
    'default': {
```

```
        'ENGINE': 'django.db.backends.postgresql',
```

```
        'NAME': 'postgres',
```

```
        'USER': 'postgres',
```

```
        'PASSWORD': 'postgres',
```

```
        'HOST': 'db',
```

```
        'PORT': 5432
```

```
}
```

}

Postgre sql دیتابیسی است که از آن می توان تقریبا در هر زبان برنامه نویسی استفاده کرد. اما به این مورد فکر کنید که چگونه یک زبان برنامه نویسی می تواند به دیتابیس متصل گردد.

جواب database adaptor است. دقیقا کاری است که Psycopg انجام می دهد. Psycopg معروف ترین و پرطرفدار ترین database adaptor در python می باشد. اگر مایل هستید که اطلاعات بیشتری را در مورد کارکرد Psycopg بدانید، می توانید توضیحات بیشتری را در سایت رسمی دنبال نمایید.

Psycopg را می توان با Pipenv نصب کرد. در command line دستور زیر را وارد نمایید. با این کار Psycopg در درون Docker host نصب می گردد.

```
$ docker-compose exec web pipenv install psycopg2-binary==2.8.3
```

ممکن است بپرسید که چرا Psycopg باید در Docker نصب گردد؟ جواب کوتاه این است که نصب پکیج های نرم افزاری در درون Docker و rebuild کردن image از اول باعث می شود که از اختلالات احتمالی ناشی از Pipfile.lock در امان بمانیم.

تولید Pipfile.lock شدیدا به نوع سیستم عامل (os) مورد استفاده بستگی دارد. ما کل سیستم عامل را در درون Docker تعریف می کنیم؛ به علاوه می Python 3.7. Psycopg2 اگر شما environment می باشد؛ Pipfile.lock منتج شده بصورت دیگری خواهد شد. بعد از آن volumes بر روی فایل docker-compose.yml سوار می گردد؛ که بصورت اتوماتیک فایل های سیستمی Docker و فایل های محلی را سینک می کند. این کار باعث می گردد که Pipfile.lock محلی ورژن را در درون Docker باز نویسی کند. بعد از آن، docker container سعی در اجرا کردن فایل Pipfile.lock می کند!

یک راه برای جلوگیری کردن از این مشکل، نصب کردن پکیج های نرم افزاری در درون Docker به جای نصب محلی می باشد.

اگر شما صفحه را ریفرش نمایید، دوباره با خطای روبرو خواهید شد. اجازه دهید log ها را چک کنیم.

```
$docker-compose logs
```

دقیقا مشابه قبلی است. چرا این اتفاق رخ داد؟ Docker بصورت اتوماتیک image ها را cache می کند. مگر این که به دلایل عملکردی، مورد تغییر کند. ما این را می خواهیم که همراه با Pipfile و Pipfile.lock بصورت اتوماتیک image مان rebuild شوند. اما بدلیل این که خط آخر COPY./code/Dockerfile می باشد؛ فقط فایل ها کپی می شوند. بنابراین در ساختار داخلی خودش rebuild نمی کند. مگر این که ما این کار را بصورت دستی و با اضافه کردن فلگ --build انجام دهیم.

به عنوان یادآوری: زمانی که می خواهید یک پکیج نرم افزاری را اضافه کنید؛ ابتدا آن را در Docker نصب نمایید؛ Container ها را متوقف و بصورت دستی image را rebuild کنید. سپس Container را مجددا راه اندازی نمایید. در طول این کتاب، این چرخه را بار ها و بار ها تکرار خواهیم کرد

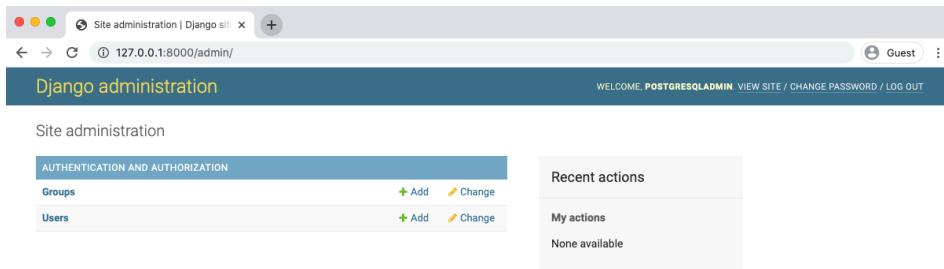
```
$ docker-compose down
$ docker-compose up -d --build
```

اگر صفحه را مجددا ریفرش کنید، جنگو به صورت موقتی آمیز به جنگو و دیتابیس متصل شده است.

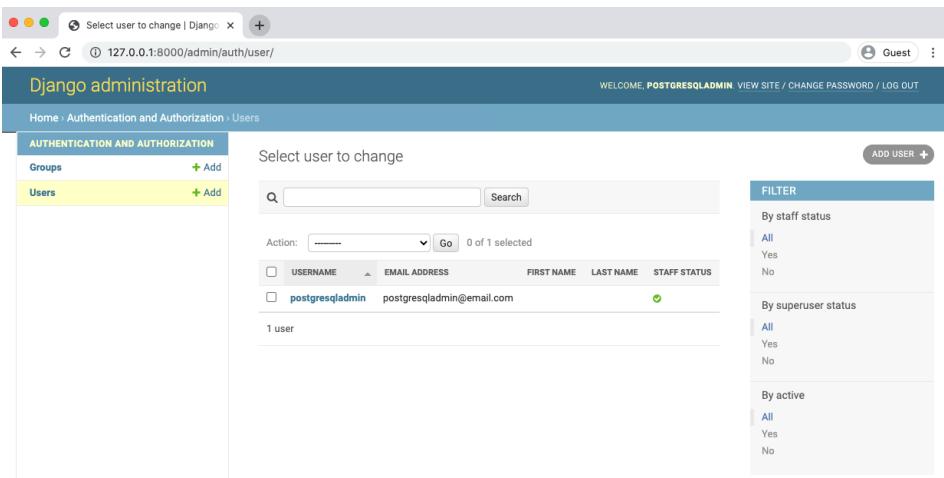
با این حال از آنجا که ما از PostgreSQL استفاده می کنیم نه SQLite دیتابیس ما خالی است. اگر نگاه کنید گزارش های فعلی را دوباره با تایپ کردن docker-compose logs مشاهده خواهید کرد. شکایاتی مانند "شما ۱۸ مهاجرت (migrations) های اعمال نشده دارید."

برای تقویت این نکته به admin در <http://127.0.0.1:8000/admin/> مراجعه کنید و وارد شوید. آیا حساب کاربری superuser ما از sqliteadmin testpass123 کار می کند ؟

پاسخ منفی است ! ما در admin/ ProgrammingError میبینیم. برای حل کردن این مشکل ، ما میتوانیم هم مهاجرت بکنیم و هم یک کاربر superuser در داخل داکر بسازیم که به دیتابیس PostgreSQL دسترسی خواهد داشت.



در گوشہ سمت راست بالا نشان می دهد Postgre admin وارد شدیم.



توجه کنید که برنامه در حال اجرا را با دستور docker-compose down متوقف کنید.

```
$ docker-compose down
```

## نتیجه‌گیری

هدف اصلی این بخش ارتباط جنگو و داکرba Postgre بود.

نکته کلیدی این است که با داکر دیگر نیازی به حضور در یک محیط مجازی محلی نداریم.

داکر محیط مجازی ما و... و همچنین دیتابیس ما و در صورت تمایل بیشتر. هاست داکر اساساً جایگزین سیستم عامل ما شده است و در داخل آن میتوانیم چندین ظرف (containers) را اجرا کنیم ، مانند برنامه وب و دیتابیس ما ، که همه می توانند جدا شوند و جداگانه اجرا شوند.

## ۴-۸- پروژه کتابفروشی آنلاین

وقتشه که پروژه اصلی کتاب رو شروع کنیم، یک کتاب فروشی آنلاین. در این فصل یه پروژه جدید شروع میکنیم، بر میگردیم به داکر، یه مدل سفارشی میسازیم و اولین آزمایشات خود را پیاده میکنیم.

شروع کنیم با ساخت یک پروژه جنگو جدید با Pipenv و بعد کار با داکر به نظر میرسه هنوز توی پوشه postgresql باشید از فصل قبل، پس داخل کامند لاین دستور cd .. /را وارد کنید تا وارد پوشه دلخواه code شوید. (فرض میکنیم که از مک استفاده میکنید). یه پوشه به اسم book میسازیم و داخل آن جنگو را نصب میکنیم. همچنین از PostgreSQL هم استفاده میکنیم. پس میتوانیم psycopg2 نصب کنیم. این فقط تنها کار پس از ساخت ما هست که در اینده پکیج های خود را روی خود داکر نصب میکنیم. در آخر از pipenv shell استفاده کنید تا وارد محیط مجازی جدید شوید.

```
$ cd...
$ mkdir book && cd book
$ pipenv install django~=3.1.0 psycopg2-binary==2.8.5
$ pipenv shell
```

اسم اولین پروژه رو config میزاریم. این قسمت یادتون نره. اگر در آخر دستور باشد، جنگو پوشه ای را درست میکند که به آن نیاز نداریم. بعد از دستور runserver استفاده کنید تا پروژه لوکال جنگو را اجرا کنیم و از از صحت این که همه چیز درست کار میکند اطمینان حاصل کنید.

(books) \$ django-admin startproject config.

(books) \$ python manage.py runserver

حال می توانیم پروژه خود را به داکر منتقل کنیم.

```
# Pull base image
FROM python:3.8
# Set environment variables
ENV PYTHON_DONTWRITEBYTECODE 1
```

```
ENV PYTHONUNBUFFERED 1
# Set work directory
WORKDIR /code
# Install dependencies
COPY Pipfile Pipfile.lock /code/
RUN pip install pipenv && pipenv install --system
# Copy project
COPY . /code/
```

پیمانه های داکر، ذاتا موقتی هستند. یعنی تا زمانی وجود دارند که اجرا شده باشند و تمامی داده هایی که در خود جا داده اند با توقف پیمانه پاک میشوند. برای داده های مانا (داده هایی که میخواهیم دائمی باشند) (در اینجا از volume استفاده میکنیم.

دانستینم که در وب سرویس های آنلاین یک مخزن داریم که پروژه محلی ما را به پیمانه های در حال اجرا پیوند میدهد و بالعکس. اما برای دیتابیس PostgreSQL یک مخزن اختصاصی نداریم که داده هایمان را در آن دسته بندی کنیم. بنابراین با توقف پیمانه هر اطلاعاتی که در آن است از دست میرود. راه حل اضافه کردن یک volume برای دیتابیس میباشد. ما اینکار را در سرویس دیتابیس با مشخص کردن یک محل و هر volume که خارج از پیمانه قرار دارد انجام میدهیم.

این توضیحات احتمالاً کمی گیج کننده باشد و نیاز به توضیح بیشتری دارد که خارج از اهداف مرکز این کتاب نیست. فقط در همین حد بدانید که پیمانه های داکر داده های مانا را ذخیره نمیکنند، بنابراین هر چیزی مثل سورس کد و اطلاعات پایگاه داده ای که میخواهیم ماندگار باشد، بایستی یک volume اختصاصی داشته باشد در غیر این صورت هر زمان که پیمانه متوقف شود از دست میرود.

```
. version: '3.8'
services:
  web:
    build: .
    command: python /code/manage.py runserver 0.0.0.0:8000
```

volumes:

-./code

ports:

- 8000:8000

depends\_on:

- db

db:

image: postgres:11

volumes:

- postgres\_data:/var/lib/postgresql/data/

environment:

- "POSTGRES\_HOST\_AUTH\_METHOD=trust"

### **volumes**

حال زمان آن است یک یوزر سفارشی که داکیومنت رسمی جنگو بسیار بر آن تاکید دارد ایجاد کنیم. چرا؟ چون شما احتیاج دارید بعضی از موقع در یوزر پیش فرض پروژه خود تغییراتی بوجود بیاورید [باصطلاح آنرا سفارشی کنید]. اگر در اولین دستور migrate که اجرا کردید، یوزر سفارشی را نساخته اید و استارت نزدید باید بگوییم سخت در اشتباهید ((: چون که user رابطه تنگاتنگی با سایر بخش های پروژه‌ی جنگو دارد. سفارشی کردن یوزر در میانه مسیر پروژه چالش برانگیز است. بهتر است در ابتدای استارت پروژه یوزر را سفارشی کنید.).

یک مسئله گیج کننده برای اکثر مردم این است که مدل یوزر سفارشی فقط در جنگو ۱,۵ اضافه شده است. تا قبل از آن روش پیشنهادی برای سفارشی کردن این بود که یک فیلد یک به یک (OneToOneField) برای یوزر ایجاد میکردند که به آن اغلب مدل پروفایل میگفتند. معمولاً این ساختار در پروژه های قدیمی قابل مشاهده است ولی امروزه استفاده از یوزر سفارشی یک روش فرآگیرتر است. هر چند برای یوزر سفارشی هم مانند سایر موارد در جنگو روش های پیاده سازی مختلفی وجود دارد: یا میتوان از AbstractUser که تمامی فیلد های مربوط به یوزر پیش فرض و سطح دسترسی ها را دارد استفاده کرد یا اینکه از که شامل موارد دقیق تر و انعطاف پذیر تر است AbstractBaseUser

- ساخت مدل CustomUser

- config/setting.py

- سفارشی کردن UserChangeForm و UserCreationForm

- اضافه کرد یوزر سفارشی ساخته شده به admin.py

```
# accounts/models.py
```

```
from django.contrib.auth.models import AbstractUser
```

```
from django.db import models
```

```
class CustomUser(AbstractUser):
```

```
    pass
```

```
# config/settings.py
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
# Local
```

```
    'accounts', # new
```

```
]
```

```
...
```

```
AUTH_USER_MODEL = 'accounts.CustomUser' # new
```

یک یوزرمدل میتواند توسط ادمین جنگو ساخته و ویرایش شود. بنابراین نیاز است در فرم های پیش فرض استفاده شود.`custom user` از

```
from django.contrib import admin
from django.contrib.auth import get_user_model
from django.contrib.auth.admin import UserAdmin

from forms import CustomUserCreationForm, CustomUserChangeForm

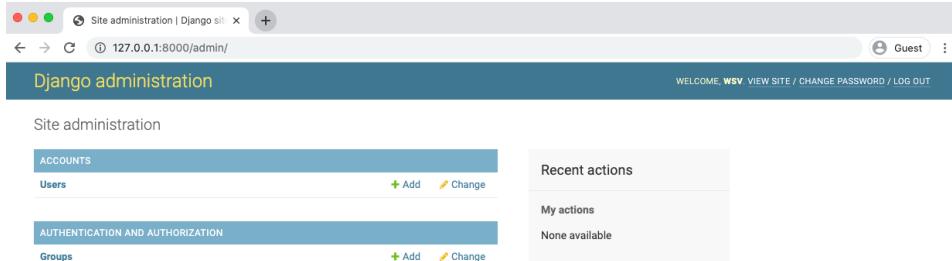
CustomUser = get_user_model()

class CustomUserAdmin(UserAdmin):
    add_form = CustomUserCreationForm
    form = CustomUserChangeForm
    model = CustomUser
    list_display = ['email', 'username']

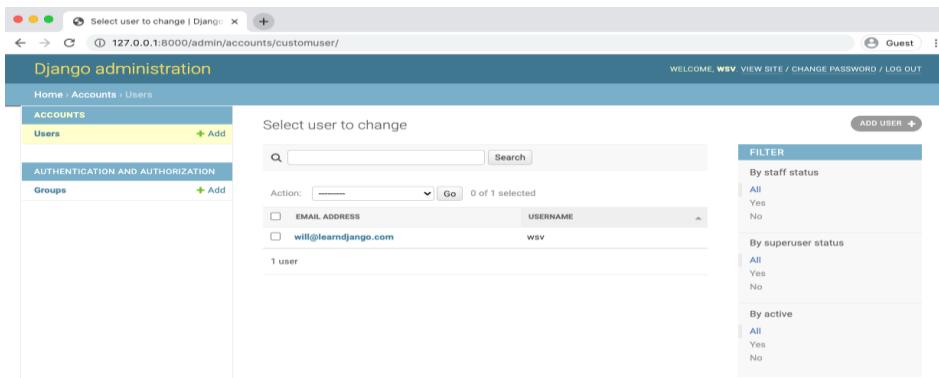
admin.site.register(CustomUser, CustomUserAdmin)
```

یک راه خوب برای تأیید اینکه مدل کاربر سفارشی ما به درستی راه اندازی و اجرا می شود، ایجاد یک ابرکاربر است که از طریق آن بتوانیم به داشبورد ادمین دسترسی داشته باشیم.

```
$ docker-compose exec web python manage.py createsuperuser
```



همچنین با رفتن به زبانه کاربران میتوانید ایمیل و شناسه ابرکاربر خود را مشاهده کند.



حال پس از اضافه شدن توابع جدید به پروژه موقع تست آن است. چه شما یک برنامه نویس انفرادی باشید و چه عضو یک تیم باشید، تست پروژه در هر حالتی یکی از قسمت های مهم پروژه است. در این باره جیکوب، بنیانگذار جنگو میگوید: " کد بدون تست، در هنگام طراحی شکسته است". به این معنی که این دو بدون هم بی معنا هستند.

دو نوع اصلی تست وجود دارد:

- تست واحد که ساده، سریع و به صورت اختصاصی برای تست عملکرد یک قسمت از کد است

- تست های یکپارچه که حجمیم، با سرعت کم و برای تست کلی برنامه یا یکسری از عملکردهای خاص مانند پرداخت و... است.

شما باید در هنگام نوشتن تست تعداد زیادی تست واحد و تعداد محدودی تست یکپارچه داشته باشید.

زبان برنامه نویسی پایتون شامل کتابخانه های زیادی برای تست واحد است، همچنین جنگو نیز کتابخانه های زیادی برای تست اتوماتیک برنامه در اختیار شما قرار می دهد. بر این اساس بهانه ای برای نوشتن تست های فراوان برای یک برنامه وجود ندارد و همانطور که گفته شد این تست ها در زمان شما برای دیباگ کردن برنامه صرفه جویی میکنند.

برای نوشتن تست واحد در جنگو از اکستنشنی به نام TestCase استفاده میکنیم. در حال حاضر در اپلیکیشن ما فایلی به نام `test.py` وجود دارد که به صورت خودکار هنگام ایجاد برنامه

با دستور startapp ساخته شد؛ این فایل در حال حاضر یک فایل خالی است و ما شروع به تغییر دادن آن میکنیم.

در این قسمت نام هر متدهای باشد با test شروع شود تا بتواند به عنوان یک تست واحد در جنگو اجرا شود. این روش نام گذاری همچنین کمک میکند که شناسایی تست ها ساده تر باشد زیرا در جنگو حدود صدها و شاید هزاران متده وجود داشته باشد

```
# accounts/tests.py

from django.contrib.auth import get_user_model
from django.test import TestCase
class CustomUserTests(TestCase):

    def test_create_user(self):
        User = get_user_model()
        user = User.objects.create_user(
            username='will',
            email='will@email.com',
            password='testpass123'
        )
        self.assertEqual(user.username, 'will')
        self.assertEqual(user.email, 'will@email.com')
        self.assertTrue(user.is_active)
        self.assertFalse(user.is_staff)
        self.assertFalse(user.is_superuser)

    def test_create_superuser(self):
        User = get_user_model()
        admin_user = User.objects.create_superuser(
            username='superadmin',
            email='superadmin@email.com',
            password='testpass123'
        )
```

```
self.assertEqual(admin_user.username, 'superadmin')
self.assertEqual(admin_user.email, 'superadmin@email.com')
self.assertTrue(admin_user.is_active)
self.assertTrue(admin_user.is_staff)
self.assertTrue(admin_user.is_superuser)
```

در ابتدا هر دو کتابخانه `get_user_model` و `TestCase` را قبل از ایجاد کلاس `CustomUserTests` فراخوتنی میکنیم. در این کلاس دوتابع تست مختلف خواهیم داشت. `test_create_user` برای تایید کاربر جدید است، که در این تابع در ابتدا ما یک شی از مدل `create_user` کاربر میسازیم و سپس با استفاده از متدهای `create_user` و `is_superuser` کاربری با دسترسی های مورد نیاز تعريف میکنیم.

برای تابع `test_create_superuser` روند بالا را با یک تفاوت جزئی تکرار میکنیم. در اینجا برای ساخت کاربر از تابع `create_user` به جای تابع `create_superuser` استفاده میشود. تفاوت این دو کاربر در این است که در ابر کاربر هر دو متغیر `is_staff` و `is_superuser` مقدار `True` دارند. میگیرند.

برای ران کردن تست ها با داکر میتوان از دستور `docker-compose exec` استفاده کرد. یا به طور سنتی `python manage.py test` اجرا کنیم.

### Command Line

```
$ docker-compose exec web python manage.py test
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
..
```

---

```
Ran 2 tests in 0.268s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

## نتیجه گیری

پروژه کتابفروشی ما هم اکنون با داکر و دیتابیس در حال اجرا است و یک مدل کابر سفارشی نیز راه اندازی کردیم.

## Pages app -۸-۵

باید یک صفحه‌ی اصلی برای پروژه‌ی جدید ایجاد کنیم. برای الان این صفحه یک صفحه استاتیک خواهد بود به این معنی که با پایگاه داده به هیچ عنوان در ارتباط نیست. بعده این صفحه یک صفحه دینامیک خواهد شد که کتاب‌های فروشی را نمایش خواهد داد. استفاده از چندین صفحه استاتیک امری رایج است، مثل صفحه درباره ما، پس باید صفحات اختصاصی مربوط به هر کدام را ایجاد کنیم.

```
. $ docker-compose exec web python manage.py startapp pages
```

```
. # config/settings.py
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
        'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
    'django.contrib.staticfiles',
```

```
# Local
```

```
    'accounts',
```

```
    'pages', # new
```

```
]
```

```
TEMPLATES = [
```

```
{
```

```
...
```

```
'DIRS': [str(BASE_DIR.joinpath('templates'))], # new
```

```
...
```

```
}
```

```
]
```

حالا باید پوشه تمپلیت را ایجاد کنیم.

```
$ mkdir templates
```

```
$ touch templates/_base.html
```

```
$ touch templates/home.html
```

در فایل base ، ما کد های پایه ای را قرار دادیم و tag block را برای عنوان و محتوا در نظر گرفته ایم block tag .ها به عنوان یک option تگ ها اضافه می شوند که از دوباره نویسی جلوگیری می کنند. به عنوان مثال، تگی که از استفاده می کند، محتوای ان خیلی راحت تر به روز رسانی می شود.

چرا ما از اسم content برای محتوای اصلی پروژه خود استفاده میکنیم؟ این اسم، هر اسمی می تواند باشد مثل main یا هر اسم مرتبط دیگری، اما استفاده از نام content در دنیای django رایج تر می باشد. آیا می توان از نام دیگه ای استفاده کرد؟ قطعاً بله.

```
<!-- templates/_base.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>{ % block title % }Bookstore{ % endblock title % }</title>
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

```
{ % block content % }
```

```
{ % endblock content % }
```

```
</div>
```

```
</body>
```

```
</html>content.
```

هر صفحه‌ی وب در پروژه‌های django نیاز به دو فایل urls.py و views.py دارد. برای مقدماتی مهم نمی‌باشد که به چه ترتیبی عمل بشود، در اینجا ما نیاز به ۳ فایل یا گاهی اوقات به ۴ فایل، (برای پایگاه داده) models.py داریم. به طور کلی من ترجیح می‌دهم که با urls شروع کنم و روی این فایل کار کنم، اما هیچ وقت یه راه مستقیم برای کار روی این فایل‌ها و ارتباط دهی به آن‌ها وجود ندارد.

باید با فایل urls.py شروع کنیم برای آدرس دهی مناسب به صفحات سایتمان در app زمانیکه ما در پروژه‌ای بر روی صفحه‌ی اصلی کار می‌کنیم، نیاز نیست که هیچ چیزی را به url نسبت بدهیم و ان را در یک ''قرار می‌دهیم. ما همچنان در خط دوم وارد کردیم.

```
# config/urls.py
from django.contrib import admin
from django.urls import path, include # new
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", include('pages.urls')), # new
]
```

ما تقریبا تمام مراحل را انجام دادیم. اگر شما به صفحه‌ی اصلی رجوع کنید به خطاب خواهید خورد. اما چرا این اتفاق افتاد؟ زمانیکه شما بایک خطاب مواجه می‌شوید باید در خط فرمان با استفاده از دستور logs ، آن را بررسی کنید.

پس دستور docker-compose logs : no را تایپ کنید که با این عبارت مواجه می‌شوید :  
اتفاقی که افتاده این است که django module named pages.urls .  
تواند فایل settings.py را به روز رسانی کند. برای اینکه فایل بروزرسانی شود settings.py به روزرسانی شود باید از دستور زیر استفاده کرد.

از دستور زیر استفاده می‌کنیم.

```
$ dcker-compose down
$ docker-compose up -d
```

الان زمانه تست است. برای صفحه‌ی اصلی ما می‌توانیم از Django TestCase خود استفاده کنیم که یک حالت خاص از Testcase می‌باشد که طراحی شده است برای تست مواردی که با پایگاه داده در ارتباط نمی‌باشند.

تست‌ها در ابتدا غافلگیر کننده هستند اما به سرعت خسته کننده می‌شوند. شما از ساختارها و تکنیک‌های مشابه بارها و بارها استفاده خواهید کرد. در editor خود قسمت pages/tests.py را به روز رسانی کنید. ما ابتدا با تست template شروع می‌کنیم.

```
# pages/tests.py
from django.test import SimpleTestCase
from django.urls import reverse

class HomepageTests(SimpleTestCase):
    def test_homepage_status_code(self):
        response = self.client.get('/')
        self.assertEqual(response.status_code, 200)
    def test_homepage_url_name(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)
```

در بالا ما ابتدا SimpleTestCase و reverse را که برای تست URLs ها استفاده می‌شود، وارد کرده‌ایم. بعد از آن یک کلاس به نام HomepageTests ایجاد کردیم که از simpleTestCase انسحاب گرفته و همچنین در داخل آن متدهایی برای هر تست نوشته شده است.

توجه کنید که ما در ابتدای هر آرگومان، self را وارد کرده‌ایم. که یک python convention می‌باشد.

بهترین روش این است که نام‌های تست‌ها خود را به نحوی انتخاب کنید که توصیف کنندهٔ درستی از تست‌های شما باشد اما توجه کنید که هر متد باید شروع به تست شود فقط با خود تست django.

دو تست موجود در اینجا هردو وضعیت کد HTTP برای صفحه‌ی اصلی را که باید برابر ۲۰۰ باشد (صفحه موجود است) را چک می‌کنند. این دو متدهای چیزی در رابطه با محتوای صفحه به ما نخواهد گفت. برای `test_homepageview_status_code` یک متغیر به نام `response` مانند `assertEqual` باشد. ایجاد کردیم که به صفحه‌ی اصلی (/) دسترسی دارد و با استفاده از `assertEqual` وضعیت کد را با ۲۰۰ مقایسه کرده‌ایم.

الگوریتم مشابهی را برای `test_homepage_url_name` استفاده می‌کنیم با این تفاوت که از `reverse` برای صدازن صفحه‌ی اصلی یا همان `home` استفاده می‌کنیم.. توجه داشته باشید که روشی که از `reverse` استفاده می‌کنیم بهترین روش می‌باشد. حتی اگر بعداً نام صفحه‌ی اصلی را تغییر بدھیم دوباره می‌توانیم به صفحه‌ی اصلی با نام جدید دسترسی داشته باشیم.

```
$ docker-compose exec web python manage.py test
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
..
```

```
Ran 4 tests in 0.277s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

```
# pages/tests.py
from django.test import SimpleTestCase
from django.urls import reverse

class HomepageTests(SimpleTestCase):
    def test_homepage_status_code(self):
        response = self.client.get('/')

```

```
self.assertEqual(response.status_code, 200)

def test_homepage_url_name(self):
    response = self.client.get(reverse('home'))
    self.assertEqual(response.status_code, 200)

def test_homepage_template(self): # new
    response = self.client.get('/')
    self.assertTemplateUsed(response, 'home.html')

مجددا متغیر response را تعریف کردیم و دوباره تست می کنیم.
```

```
$ docker-compose exec web python manage.py test pages
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
...
```

---

```
Ran 3 tests in 0.023s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

وجه کردید که این بار خط فرمان پیغام متفاوتی را نشان می دهد؟ ما اسم pages را به انتهای کد اضافه کردیم پس فقط تست های موجود در همین App اجرا خواهند شد. در پروژه های کوچک تر مشکلی نیست که کل تست ها را اجرا کنید اما در پروژه های بزرگ تر، بهتر است که هر تست را در همان App تست کنید. این باعث سریع تر شدن کار خواهد شد.

برای این قسمت باید تست کنیم که صفحه‌ی اصلی از HTML درستی استفاده می کند و همچنین تست کنیم که محتویات HTML نادرست در صفحه‌ی اصلی وجود نداشته باشد. خوب است که همیشه تست هایی که انتظار داریم قبول بشوند و ان هایی که قرار است رد بشوند را با هم تست کنیم...

```
# pages/tests.py
from django.test import SimpleTestCase
```

```

from django.urls import reverse, resolve
from.views import HomePageView

class HomepageTests(SimpleTestCase):
    def test_homepage_status_code(self):
        response = self.client.get('/')
        self.assertEqual(response.status_code, 200)

    def test_homepage_url_name(self):
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 200)

    def test_homepage_template(self):
        response = self.client.get('/')
        self.assertTemplateUsed(response, 'home.html')

    def test_homepage_contains_correct_html(self):
        response = self.client.get('/')
        self.assertContains(response, 'Homepage')

    def test_homepage_does_not_contain_incorrect_html(self):
        response = self.client.get('/')
        self.assertNotContains(
            response, 'Hi there! I should not be on the page.')

```

تست را اجرا کنید.

توجه کرده اید که در قسمت تست ها ما داریم یک قسمت را بارها و بارها تکرار می کنیم؟ برای هر تست ما داریم متغیر response را هر دفعه ایجاد می کنیم. به نظر کار بیهوده و

مستعد خطا است این روش. بهتر است که روشی را پیدا کنیم که هر دفعه کدی را تکرار نکنیم.

از انجایی که تست های واحد از بالا به پایین اجرا می شوند، می توانیم یک متده است setup اضافه کنیم که قبیل اجرای تست ها اجرا خواهد شد. قبل از هر تست self.response را در صفحه اصلی اجرا خواهد کرد. بنابراین نیازی نیست که متغیر response را برای هر تست ایجاد کنیم. و همچنین به این معنی است که دیگر می توانیم test\_homepage\_url\_name را حذف کنیم به خاطر اینکه برای هر تست، ان را با استفاده از reverse صدا خواهیم کرد.

```
# pages/tests.py
from django.test import SimpleTestCase
from django.urls import reverse

class HomepageTests(SimpleTestCase): # new

    def setUp(self):
        url = reverse('home')
        self.response = self.client.get(url)

    def test_homepage_status_code(self):
        self.assertEqual(self.response.status_code, 200)

    def test_homepage_template(self):
        self.assertTemplateUsed(self.response, 'home.html')

    def test_homepage_contains_correct_html(self):
        self.assertContains(self.response, 'Homepage')

    def test_homepage_does_not_contain_incorrect_html(self):
        self.assertNotContains(
```

self.response, 'Hi there! I should not be on the page.')  
 این آخرین تست ما خواهد بود که نتیجه عالی به همراه دارد.

```
$ docker-compose exec web python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

.....
```

---

Ran 7 tests in 0.282s

OK

Destroying test database for alias 'default'...

### نتیجه گیری

ما قالب های خود را پیکر بندی کرده ایم و صفحه‌ی اصلی را به پروژه‌ی خود اضافه کردیم. ما همچنین برای هر یکی از موارد یک تست ایجاد کرده ایم که شامل کارهایی است که در این فصل انجام دادیم. بعضی از توسعه دهنده‌گان ترجیح می‌دهند از روشی استفاده کنند که Test\_Driven نامیده می‌شود که به صورتی است که ابتدا همه‌ی کد‌ها نوشته می‌شود بعد تست‌ها نوشته می‌شوند. من ترجیح می‌دهم که هر قسمتی که نوشته شد تست همان هم نوشته شود و تست شود. در پروژه‌هایی که سنگین هستند تقریباً محال است به خاطر سپردن تمامی مراحل. همچنین در یک کار تیمی که چندین نفر بر روی یک پروژه کار می‌کنند، مشکل است که فهمید چه کدی و توسط چه کسی بعداً با خطا مواجه شده است.

### ۶- ثبت‌نام مقدماتی کاربر

یک ویژگی مهم در وب سایت‌های پویا، بحث ثبت‌نام کاربران است. این مسئله در پروژه فروشگاه کتاب مورد نظر ما نیز وجود دارد. در این فصل به پیاده‌سازی موارد ورود به سایت، خروج از سایت و ثبت‌نام در سایت می‌پردازیم. از آنجایی که جنگو ویوها و url لازم برای دو مورد ورود و خروج از سایت را در اختیار ما قرار می‌دهد، پیاده‌سازی دو مورد اول نسبتاً راحت است اما پیاده‌سازی مورد ثبت‌نام به دلیل اینکه هیچ راه حل داخلی از پیش تعریف شده‌ای برای آن در جنگو وجود ندارد، چالش برانگیز است

## اپ Auth

بیایید با پیاده سازی دو مورد ورود به سایت و خروج از سایت با استفاده از سیستم احراز هویت auth app جنگو شروع کنیم. جنگو، ویو ها و url های مهمی را در اختیارمان قرار میدهد و این یعنی ما فقط به یک تمپلیت ای برای بروزرسانی کارهایی که میباشد انجام دهیم نیاز داریم. این کار زمان زیادی را برای ما به عنوان یک توسعه دهنده ذخیره کرده و تضمین می کند که مرتكب اشتباهی نخواهیم شد. زیرا کد اصلی قبل از توسعه میلیونها توسعه دهنده مورد آزمایش و بهره برداری قرار گرفته است.

با این حال، این سادگی در پیاده سازی موجب می شود افرادی که در جنگو تازه کار هستند دچار حس "جادویی بودن" شوند. ما برخی از این موارد را در کتاب خود Django for Beginners، پوشش داده ایم اما با این وجود سرعت خود را کاهش نداده و کد منبع اصلی نگاه نکرده ایم. مقصود برای یک تازه کار این بود که به صورت گسترده توضیح داده و نشان دهیم که "چگونه" به درستی ثبت نام کاربر را پیاده سازی نماییم اما این واقعاً به قیمت فرو رفتن در اینکه "چرا" از آن کد استفاده کرده ایم تمام شد.

از آنجایی که این یک کتاب بسیار پیشرفته است ما برای فهم بهتر کد منبع عمیق تر خواهیم شد. رویکردی که در اینجا مطرح می شود می تواند برای کشف سایر عملکردهای داخلی جنگو هم به تنها یی مورد استفاده قرار گیرد.

```
# config/settings.py

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth', # Yoohoo!!!!
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Local
    'accounts',
```

```
'pages',
]
```

به منظور استفاده از auth app داخلی جنگو، ما باید آن را به فایل config/urls.py اضافه کنیم. ساده ترین روش این است که فایل accounts به عنوان پیشوند قرار بگیرد چون عموماً در جامعه جنگو از این حالت استفاده می‌شود. یک خط را در زیر تغییر دهید. توجه کنید که هر URL-admin، user URLs.py افزایش یابد، افزودن کامنت برای هر بخشی از management، local apps,... چقدر طول فایل فایل افزایشی باشد.

```
# config/urls.py
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    # Django admin
    path('admin/', admin.site.urls),

    # User management
    path('accounts/', include('django.contrib.auth.urls')), # new

    # Local apps
    path("", include('pages.urls')),
]
```

حال ببینیم چه چیزهایی Auth app اضافه شده است.

```
accounts/login/ [name='login']
accounts/logout/ [name='logout']
accounts/password_change/ [name='password_change']
accounts/password_change/done/ [name='password_change_done']
accounts/password_reset/ [name='password_reset']
accounts/password_reset/done/ [name='password_reset_done']
```

```
accounts/reset/<uidb64>/<token>/ [name='password_reset_confirm']
accounts/reset/done/ [name='password_reset_complete']
```

مرحله بعدی چیست؟ باید هم پیج خود را به روز کنیم تا اگر کاربری لگین نمود یا نه به ما اطلاع دهد که در حال حاضر فقط از طریق ادمین امکانپذیر است.

اینجا کد جدیدی برای فایل templates/home.html وجود دارد که از تگ های if/else موجود در template.engine جنگو برای منطق اصلی استفاده می کند.

```
<!-- templates/home.html -->
{% extends '_base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}

<h1>Homepage</h1>

{% if user.is_authenticated %}

    Hi {{ user.email }}!

{% else %}

    <p>You are not logged in</p>
    <a href="{% url 'login' %}">Log In</a>

{% endif %}

{% endblock content %}
```

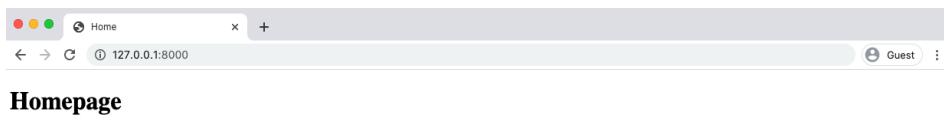
اگر کاربر وارد سیستم شده باشد(تایید شده باشد)، ما پیامی که شامل "سلام" و آدرس ایمیل است را به او نشان می دهیم. هر دوی اینها متغیرهایی هستند که ما می توانیم با engine جنگو از طریق دو براکت باز {} و بسته {} استفاده کنیم.

دیفالت یوزر(کاربر پیش فرض)، شامل فیلدهای متعددی از جمله email و is\_authenticated می باشد. که به آن ها ارجاع داده شده است.

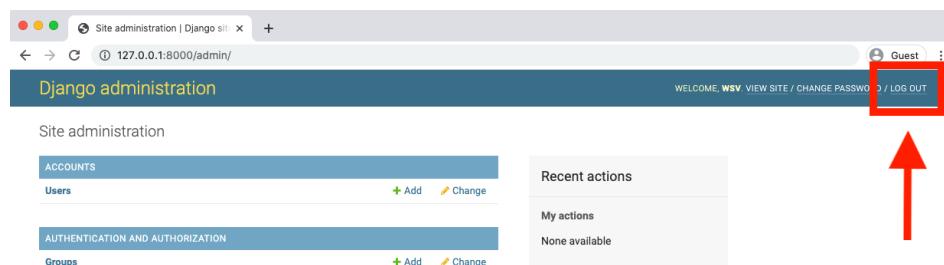
و login و logout اسامی URL هستند. تگ url این معناست که اگر ما نام URL را مشخص نمودیم، لینک به طور خودکار به مسیر آن URL متصل شود. به عنوان مثال در فصل قبلی ما نام URL صفحه homepage خودمان را به home تنظیم نمودیم. بنابراین یک لینک به

به فرمت `url 'home' %} خواهد بود. در ادامه بیشتر به این موضوع می‌پردازیم.`

اگر حالا به `http://127.0.0.1:8000/` نگاهی بیاندازید به احتمال زیاد آدرس ایمیل حساب کاربری ابرکاربر (superuser) شما را نشان خواهد داد چون قبل از لایگین شدن از آن استفاده کرده بودیم.



در بخش `admin` در `http://127.0.0.1:8000/admin`، اگر روی دکمه "log out" که در گوشه بالا سمت راست قرار دارد کلیک کنید، می‌توانیم استفاده کنیم. `admin` خارج شویم.



ممکن است بتوانید این مراحل را از طریق خواندن دکیومنت رسمی به تنها یی کنار هم قرار دهید اما رویکرد عمیق و بهتر این است که یاد بگیرید چگونه کد منبع جنگو را خودتان بخوانید.

یک سوال این است که، چطور `user` و متغیرهای مرتبط با آن در تمپلیت ظاهر شده اند؟ پاسخ این است که جنگو مفهومی به نام `template context` دارد و به این معناست که هر تمپلیتی با داده هایی از فایل `views.py` مربوطه بارگذاری شده است. ما میتوانیم از یوزر درون `User` کلاس برای دسترسی به ویژگی های `User` استفاده کنیم. به عبارت دیگر، جنگو این را به طور خودکار برایما محیا می سازد.

حال برای این که بدانیم یک یوزر وارد سایت شده است یا نه، به `user.is_authenticated` می کنیم و پس از آن می توانیم از `True` برگردانده شده و ما می توانیم کارهایی مانند نمایش ایمیل کاربر را اجرا کنیم. یا اگر هیچ کاربری وارد نشده باشد، نتیجه `False` خواهد بود.

در ادامه ما `url name login` را داریم. این اسم از کجا می آید؟ البته که پاسخ خود جنگو است! بیایید قطعه کد `{% url 'login' %}` را بخش به بخش باز کنیم.

در ابتدا ما از `url template tag` استفاده کرده ایم که آرگومان اول آن یک named URL pattern می باشد. این یک بخش دلخواه است که به عنوان تمرین به تمامی مسیرهای URL خود اضافه نموده ایم. بنابراین باید یک نام `login` اضافی به URL شده باشد!

دو راه وجود دارد که ما باید با آن ها آشنا باشیم. به عبارت دیگر اگر من به شما نمی گفتم که می خواهیم از `{% url 'login' %}` استفاده کنیم، چطور متوجه آن می شدیم؟

در ابتدا به دокумент رسمی نگاه کنید. من به شخصه اغلب اوقات از قابلیت جستجو استفاده می کنم و چیزی مثل "login" را تایپ نموده و جستجو می کنم تا زمانی که توصیفی از `login` ابابام. چیزی که ما می خواهیم authentication views نام دارد و الگوهای URL مربوطه را برایمان لیست می کند.

`accounts/login/ [name='login']`

`accounts/logout/ [name='logout']`

`accounts/password_change/ [name='password_change']`

`accounts/password_change/done/ [name='password_change_done']`

`accounts/password_reset/ [name='password_reset']`

`accounts/password_reset/done/ [name='password_reset_done']`

`accounts/reset/<uidb64>/<token>/ [name='password_reset_confirm']`

`accounts/reset/done/ [name='password_reset_complete']`

جنگو یک خطای `TemplateDoesNotExist` برایمان دارد. به نظر می رسد انتظار یک تمپلیت `register/login.html` در `in` `accounts` نباشد. به علاوه می توانیم دокумент جنگو را نگاه کنیم و ببینیم که `template_name` خواسته شده، آن لوکیشن را دارد.

اما بیایید واقعاً مطمئن شویم و سورس کد را بررسی کنیم. بنابراین اینجا می‌توانیم هر جادویی را از بین ببریم. به هر حال این فقط جنگو است.

اگر به فایل `auth/views.py` برگردیم می‌توانیم در خط ۴۷ ببینیم که برای `LoginView` نام تمپلیت 'registration/login.html' است. بنابراین اگر می‌خواستیم مسیر پیش فرض را تغییر دهیم، می‌توانستیم، اما این به معنای `override` کردن `LoginView` است که بیش از حد به نظر می‌رسد. بیایید فقط از چیزی که جنگو در اینجا به ما می‌گوید استفاده کنیم.

یک فolder `registration` جدید در مسیر تمپلیت‌های موجود، ایجاد کرده و سپس فایل `login.html` را هم در آن قرار دهید..

```
$ mkdir templates/registration
$ touch templates/registration/login.html
<!-- templates/registration/login.html -->
{ % extends '_base.html' %

{ % block title % }Log In{ % endblock title % }

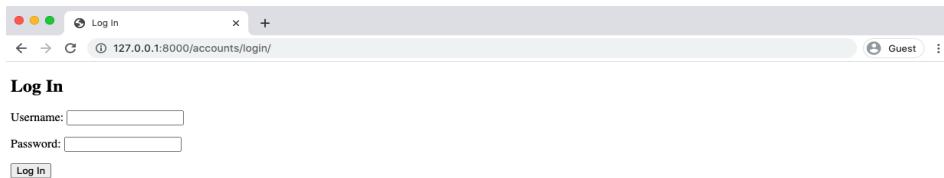
{ % block content % }
<h2>Log In</h2>
<form method="post">
{ % csrf_token %
{ { form.as_p }}
<button type="submit">Log In</button>
</form>
{ % endblock content % }
```

شما همیشه باید در هر فرم قابل ارسالی حفاظت CSRF protection را اضافه کنید. در غیراینصورت یک سایت آلوده و مخرب می‌تواند لینک را تغییر داده و به سایت و کاربر حمله نماید. جنگو دارای میان افزار CSRF است که این مسئله را برایمان مدیریت می‌کند. تنها

کاری که باید انجام دهیم این است که تگ های `{% csrf_token %}` را به ابتدای فرم اضافه کنیم.

در مرحله بعد می توانیم ظاهر محتویات فرم را کنترل کنیم. در حال حاضر ما از `form` استفاده می کنیم تا هر فیلد `p` پاراگراف `p` نمایش داده شود.

با این توضیحات، بگذارید بررسی کنیم که آیا تمپلیت جدیدمان به درستی کار می کند یا خیر. وب پیج را در `http://127.0.0.1:8000/accounts/login/` را رفرش کنید.



به انتها فایل Setting این خط را اضافه کنید.

```
# config/settings.py
LOGIN_REDIRECT_URL = 'home'
```

حالا تلاش کنید تا دوباره در `http://127.0.0.1:8000/accounts/login/` لاگین نمایید. پس با موفقیت، کاربر را به صفحه `home` پیج و خوشامد گویی به اکانتی که با آن لاگین نموده اید هدایت می شوید.

حالا بیایید گزینه `log out` را هم به `home` پیج خود اضافه کنیم. چون فقط یک ابرکاربر یا سوپریوزر به ادمین دسترسی خود داشت. چطور این کار را انجام دهیم؟

اگر به `auth` ویوهای بالا نگاهی بیاندازید می بینیم که ویژگی لاغ اوت از `LogoutView` استفاده می کند، و `name` آن `logout` است. و ما می توانیم با یک تمپلیت تگ به آن به عنوان `logout` بنگریم.

اما اگر بخواهیم، می توانیم با استفاده از `LOGOUT_REDIRECT_URL` که می تواند به انتهای فایل `config/settings.py` اضافه شود، خودمان این را تنظیم کنیم. بیایید این کار را انجام دهیم تا کاربری که لاغ اوت نموده به `home` پیج هدایت شود.

```
# config/settings.py
LOGIN_REDIRECT_URL = 'home'
```

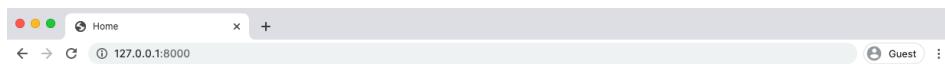
```
LOGOUT_REDIRECT_URL = 'home' # new
```

```
<!-- templates/home.html -->
{% extends '_base.html' %}

{% block title %}Home{% endblock title %}

{% block content %}
<h1>Homepage</h1>
{% if user.is_authenticated %}
    Hi {{ user.email }}!
    <p><a href="{% url 'logout' %}">Log Out</a></p>
{% else %}
    <p>You are not logged in</p>
    <a href="{% url 'login' %}">Log In</a>
{% endif %}
{% endblock content %}
```

هم پیج را رفرش کنیم لوگ اوت ظاهر می شود و اگر روی آن کلیک کنیم لوگ این می شود.



## Homepage

You are not logged in

[Log In](#)

پیاده سازی یک صفحه ثبت نام برای ثبت نام کاربر کاملا بر عهده خودمان هست. برای هر صفحه جدیدی یکسری گام های استاندارد وجود دارد:

- یک فایل accounts.urls.py در سطح اپ ایجاد نمایید

- روزرسانی config.urls.py را به اپ اشاره کند

- یک ویویی به نام SignupPageView اضافه کنید

- یک تمپلیتی به نام signup.html ایجاد کنید

- صفحه home.html را به روز رسانی کنید تا صفحه ثبت نام را نمایش دهد

```
$ touch accounts/urls.py
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    # Django admin
    path('admin/', admin.site.urls),

    # User management
    path('accounts/', include('django.contrib.auth.urls')),

    # Local apps
    path('accounts/', include('accounts.urls')), # new
    path("", include('pages.urls')),
]
```

لا ویوی SignupPageView را می سازیم. که به CustomUserCreationForm ارجاع می دهد و یک success\_url دارد که به صفحه لاگین اشاره می کند، یعنی بعد از اینکه فرم ارسال شد، یوزر به آن جا هدایت خواهد شد. نیز template-name signup.html است.

```
<!-- templates/home.html -->
```

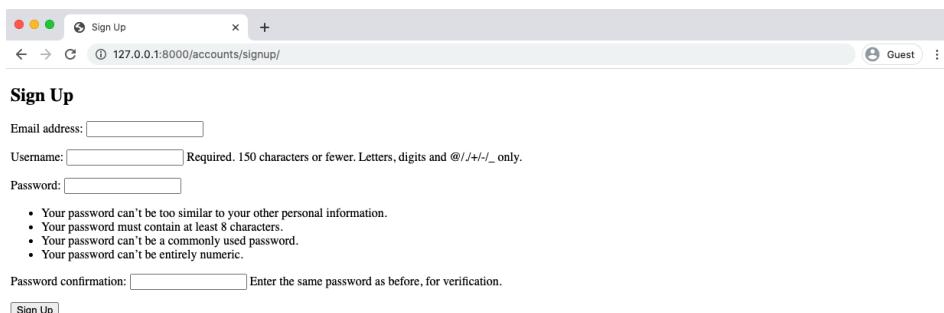
```
{% extends '_base.html' %}
```

```
{% block title %}Home{% endblock title %}
```

```
{% block content %}
```

```
<h1>Homepage</h1>
{ % if user.is_authenticated % }
    Hi {{ user.email }}!
    <p><a href="{% url 'logout' %}">Log Out</a></p>
{ % else % }
    <p>You are not logged in</p>
    <a href="{% url 'login' %}">Log In</a>
    <a href="{% url 'signup' %}">Sign Up</a>
{ % endif % }
{ % endblock content %}
```

هدايت خواهد كرد. <http://127.0.0.1:8000/accounts/signup>



از آن جایی که لاگین و لاگ اوت فیچرهای built-in در جنگو هستند نیازی به تست ندارند.  
اما ما باید ویژگی sign up را تست نماییم!

بیایید ابتدا یک متد setUp که صفحه مان را لود کند ایجاد کنیم. سپس متد test\_signup\_template را برای تست status code و تمپلیت مورد استفاده مشابه با روشی که در فصل قبلی برای هوم پیج انجام داده ایم، ایجاد می کنیم.

accounts/tests.py را با تغییرات زیر به روز کنید.

```
# accounts/tests.py
from django.contrib.auth import get_user_model
from django.test import TestCase
from django.urls import reverse, resolve # new
```

```
from.forms import CustomUserCreationForm # new
from.views import SignupPageView # new

class CustomUserTests(TestCase):
    class SignupPageTests(TestCase):

        def setUp(self):
            url = reverse('signup')
            self.response = self.client.get(url)

        def test_signup_template(self):
            self.assertEqual(self.response.status_code, 200)
            self.assertTemplateUsed(self.response, 'registration/signup.html')
            self.assertContains(self.response, 'Sign Up')
            self.assertNotContains(
                self.response, 'Hi there! I should not be on the page.')

        def test_signup_form(self): # new
            form = self.response.context.get('form')
            self.assertIsInstance(form, CustomUserCreationForm)
            self.assertContains(self.response, 'csrfmiddlewaretoken')

        def test_signup_view(self): # new
            view = resolve('/accounts/signup/')
            self.assertEqual(
                view.func.__name__,
                SignupPageView.as_view().__name__
            )
```

حال تست هایمان را با موفقیت اجرا می کنیم.

```
$ docker-compose exec web python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
```

.....

Ran 10 tests in 0.328s

OK

Destroying test database for alias 'default'...

## ۸- ثبت‌نام پیشرفته کاربر

در این مرحله ما کاربر استاندارد جنگو را داریم اما اغلب اوقات این تنها نقطه شروع پروژه‌های حرفه‌ای می‌باشد نظرتان راجع به سفارشی کردن چیزها چیست؟ مثلاً الگوی نام کاربری/ایمیل/رمز عبور پیش فرض جنگو تا حدودی قدمت دارد. برای ثبت نام و ورود به سیستم نیاز به ایمیل/رمز عبور بسیار متداول است در واقع هر قسمت از جریان و روند احراز هویت فرم‌ها ایمیل‌ها و صفحات در صورت تمایل قابلیت سفارشی سازی دارد.

یکی دیگر از عوامل مهم در بسیاری از پروژه‌ها احراز هویت با شبکه‌های اجتماعی می‌باشد که از طریق یک سرویس ثالث مانند گوگل فیسبوک و... انجام می‌گیرد.

ما می‌توانیم برای انجام این کار (احراز هویت با شبکه‌های اجتماعی) آن را از ابتدا خودمان پیاده‌سازی کنیم اما خطراتی وجود دارد: ثبت نام کاربر دارای یک محدوده‌ی پیچیده و اجزای متحرک بسیاری است و یک ناحیه‌ای است که ما واقعاً نمی‌خواهیم باعث به وجود آمدن اشتباه امنیتی شویم.

به این علت بسیاری از توسعه‌دهندگان جنگو از پکیج محبوب ثالث [django-allauth](#) استفاده می‌کنند. اضافه کردن هر پکیج ثالث باید با کمی احتیاط همراه باشد چون شما در حال اضافه کردن وابستگی دیگری به دسته‌ی فنی خود هستید. بسیار مهم باشد که مطمئن شوید هر پکیجی که از آن استفاده می‌کنید هم آپدیت باشد و هم به خوبی تست شده باشد. خوشبختانه

پکیج django-allauth هر دوی این ویژگی‌ها را دارد. این پکیج با مقداری جادو تمام این نگرانی‌ها را برطرف می‌کند و سفارشی سازی بسیار آسان می‌شود.

```
$ docker-compose exec web pipenv install django-allauth==0.42.0
```

```
$ docker-compose down
```

```
$ docker-compose up -d --build
```

وب سایت ما همچنان مانند قبل عمل می‌کند زیرا ما به طور واضح به جنگو در مورد این بسته جدید (django-allauth) نگفته ایم. برای انجام این کار، ما باید پیکربندی sites INSTALLED\_APPS را در تنظیمات خود به روز کنیم. اضافه کردن فریمورک allauth و ویژگی‌های آکانت allauth.account اختیاری می‌باشد.

فریمورک sites جنگو یک ویژگی قدرتمند هست که اجازه می‌دهد یک پروژه جنگو توسط چند سایت کنترل شود. با توجه به اینکه ما تنها یک سایت در پروژه خود داریم، ID را ۱ تعیین خواهیم کرد. اگر سایت دومی را اضافه کنیم، دارای ۲ ID خواهد بود، سایت سوم دارای ۳ ID، و به همین ترتیب خواهد بود.

```
# config/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites', # new
    # Third-party
    'crispy_forms',
    'allauth', # new
    'allauth.account', # new
    # Local
    'accounts',
```

```
'pages',
]

# django-allauth config
SITE_ID = 1 # new
```

فایل setting.py که توسط جنگو ساخته شده برای هر پروژه جدیدی شامل یک سری تنظیمات می باشد که به وضوح می توان آن را در فایل setting.py دید علاوه بر این ها یک سری تنظیمات نیز وجود دارد که قابل مشاهده در فایل setting.py نیستند. شاید در ابتدا گیج کننده باشد تمامی این تنظیمات در این قسمت داکیومنت قابل مشاهده می باشد.

برای مثال می توان به AUTHENTICATION\_BACKENDS اشاره کرد. در پشت صحنه جنگو مقدار این متغیر را برابر با 'django.contrib.auth.backends.ModelBackend' قرار می دهد. که برای اعتبارسنجی (authenticate) استفاده می شود. ما میتوانیم این خط را به فایل اضافه کنیم اما رفتار جنگو در خصوص اعتبارسنجی تغییری نخواهد کرد.

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)
```

```
# config/settings.py
# django-allauth config
SITE_ID = 1
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend', # new
)
```

پیکربندی دیگری که به طور ضمنی در setting.py تنظیم شده است و ما آن را نمی بینیم EMAIL BACKED می باشد. به طور پیش فرض جنگو به دنبال تنظیمات SMTP server برای ارسال ایمیل می باشد.

django-allauth به محض این که ثبت نام کاربر موفقیت آموز بود یک ایمیل می فرستد که ما میتوانیم این مورد را سفارشی (customize) کنیم، اما از آنجا که هنوز SMTP server به درستی پیکربندی نشده است، منجر به خطا می شود.

فعلا برای حل این مورد جنگو خروجی هر ایمیل را به کنسول خط فرمان ارسال کند. بنابراین ما می توانیم تنظیمات پیش فرض جنگو را که بر روی smtp می باشد را به کنسول تغییر دهیم این پیکربندی جزو پیکربندی های ضمنی می باشد و در ابتدای پروژه ما آن را نمی بینیم. این تکه کد را به انتهای فایل setting.py خود اضافه کنید.

یک تغییر کوچک دیگری نیز لازم است که باید آن را انجام دهیم. اگر شما مجدداً به صفحه configuration در داکیومنت جنگو نگاهی بیندازید یک سری تنظیمات برای وجود دارد که به صورت پیش فرض در مسیر صفحه REDIRECT\_LOGOUT\_ACCOUNT اصلی در root url قرار دارد.

در فایل settings.py فعلی ما دو خط زیر برای ریدایرکت کردن به صفحه اصلی از طریق پارامتر name در url (که مقدار آن برابر با home است) می باشد.

موضوع این است که متغیر ACCOUNT\_LOGOUT\_REDIRECT پکیج django-allauth را بازنویسی (override) می کند. متغیر داخلی (built-in) LOGOUT\_REDIRECT\_URL را اشاره کنند تغییرات زیاد واضح به هر حال وقتی که هر دوی این متغیر ها به صفحه home اشاره کنند تغییرات زیاد واضح نخواهد بود. شاید در آینده برای اپلیکیشن خودمان همواره نخواهیم که کاربر پس از log out به صفحه اصلی سایت ریدایرکت شود. در هر صورت ما باید به طور واضح مشخص کنیم که کاربر پس از لاگین به کدام صفحه ریدایرکت شود.

همچنین می توان دو خط مربوط به ریدایرکت را در زیر قسمت پیکربندی پکیج django-allauth انتقال داد. در این زمان پیکربندی مربوط به پکیج django-allauth ما باید چیزی شبیه به کد زیر باشد.

```
# config/settings.py
# django-allauth config
LOGIN_REDIRECT_URL = 'home'
ACCOUNT_LOGOUT_REDIRECT = 'home' # new
```

```
SITE_ID = 1
```

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
)
```

```
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

ما تغییرات بسیار زیادی را در فایل config/setting.py خود انجام داده ایم بنابراین نیاز هست که ما یک را بزنیم تا تغییرات در دیتابیس هم اعمال شود..

```
$ docker-compose exec web python manage.py migrate
```

همچنین ما نیاز داریم که url های از پیش ساخته (built-in) مربوط به پکیج django-allauth را به url های اپ خود منتقل کنیم. به هرحال ما از تمپلیت و روت(routing) های خود پکیج-django استفاده میکنیم. url های مربوط به ثبت نام در اپ accounts را پاک کنیم.

```
# config/urls.py
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    # Django admin
    path('admin/', admin.site.urls),
    # User management
    path('accounts/', include('allauth.urls')), # new
    # Local apps
    path("", include('pages.urls')),
]
```

اپ views.py و gurls.py در این مرحله ما می توانیم فایل های که صرفا برای ثبت نام نوشتهیم را پاک کنیم و دیگر مورد استفاده قرار نمیگیرند.

```
$ mkdir templates/account
```

```
$ mv templates/registration/login.html templates/account/login.html
```

```
$ mv templates/registration/signup.html templates/account/signup.html
```

```
. $ rm -r templates/registration
```

در دستور بالا rm به معنای حذف کردن و -r به معنای بازگشتی (recursive) بودن دستور حذف می باشد که نیاز می باشد تمام فایل ها و دایرکتوری های مسیر registration پاک شود. اگر میخواهید اطلاعات بیشتری راجب کامند rm دستور man rm میتوانید اطلاعات بیشتری در این رابطه پیدا کنید.

آخرین قدم آپدیت لینک های url در templates/home.html و templates/\_base.html برای استفاده از نام های url پکیج django-allauth به جای نام های url جنگو می باشد. برای این کار ما پیشوند \_account اضافه میکنیم. به این ترتیب نام های signup login logout به اسمی account\_logout account\_login account\_signup به ترتیب تغییر خواهد کرد.

```
<!-- templates/_base.html -->  
...  
<nav class="my-2 my-md-0 mr-md-3">  
  <a class="p-2 text-dark" href="{% url 'about' %}">About</a>  
  {% if user.is_authenticated %}  
    <a class="p-2 text-dark" href="{% url 'account_logout' %}">Log Out</a>  
  {% else %}  
    <a class="p-2 text-dark" href="{% url 'account_login' %}">Log In</a>  
    <a class="btn btn-outline-primary" href="{% url 'account_signup' %}">Sign  
      Up</a>  
  {% endif %}  
</nav>  
...
```

اکنون صفحه لاگین ظاهر جدیدی به خود گرفته است.

ه چک باکس Remember Me توجه کنید. این گزینه اولین پیکربندی از پیکربندی های زیادی است که پکیج django-allauth در اختیار ما قرار می دهد. به طور پیش فرض از کاربر پرسیده نمی شود که اطلاعات در session ذخیره شود تا در موارد بعدی نیاز به لاگین مجدد نباشد. این گزینه می تواند False باشد تا اطلاعات ذخیره نشود همچنین می تواند true باشد تا اطلاعات در session ذخیره شود. در این قسمت ما true را انتخاب میکنیم همانطور که لاگین کار می کند. کار django میکند

حال می خواهیم صفحه لوگ اوت را طراحی کنیم.

. \$ touch templates/account/logout.html

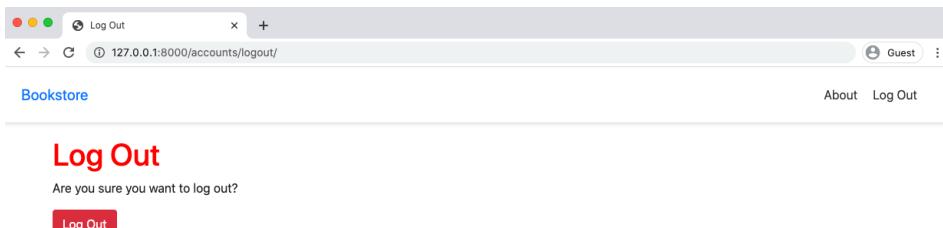
```
<!-- templates/account/logout.html -->
{% extends '_base.html' %}
{% load crispy_forms_tags %}

{% block title %}Log Out{% endblock %}

{% block content %}
<h1>Log Out</h1>
```

```
<p>Are you sure you want to log out?</p>
<form method="post" action="{% url 'account_logout' %}">
    {% csrf_token %}
    {{ form|crispy }}
    <button class="btn btn-danger" type="submit">Log Out</button>
</form>
{% endblock content %}
```

ادامه می دهیم و صفحه را رفرش می کنیم.



یاد می آورید که ما چگونه تنظیمات مربوط به ایمیل را انجام دادیم که خروجی در کنسول نمایش داده شود؟ به محض ثبت نام پکیج django-allauth به صورت اتومات یک ایمیل را می فرستد که از طریق Docker-compose logs میتوانیم آن ها را ببینیم.

```
$ docker-compose logs
```

...

```
web_1 | Content-Type: text/plain; charset="utf-8"
```

```
web_1 | MIME-Version: 1.0
```

```
web_1 | Content-Transfer-Encoding: 7bit
```

```
web_1 | Subject: [example.com] Please Confirm Your E-mail Address
```

```
web_1 | From: webmaster@localhost
```

```
web_1 | To: testuser@email.com
```

```
web_1 | Date: Mon, 03 Aug 2020 14:04:15 -0000
```

```
web_1 | Message-ID:
```

```
<155266195771.15.17095643701553564393@cdab877c4af3>
```

```
web_1 |
```

web\_1 | Hello from example.com!

web\_1 |

web\_1 | You are receiving this e-mail because user testuser1 has given yours as an e-mail address to connect their account.

web\_1 |

web\_1 | To confirm this is correct, go to <http://127.0.0.1:8000/accounts/confirm-email/MQ:1h4oIn:GYETeK5dRCIGjcgA8NbuOoyvafA/>

web\_1 |

web\_1 | Thank you from example.com!

web\_1 | example.com

web\_1 | -----

...

به عنوان superuser به صفحه ای ادمین در آدرس <http://127.0.0.1:8000/admin> لایگین کنید. همانطور که می بینید یک سری تغییرات را پکیج انجام داده است..

EMAIL ADDRESS	USERNAME
testuser@testuser.com	testuser
testuser1@testuser1.com	testuser1
will@learndjango.com	wsv

فایل تست را آپدیت می کنیم. اولین مشکل این است که url ثبت نام درست نمی باشد و به جای آن باید از account\_signup که پکیج django-allauth در اختیار ما قرار می دهد استفاده کنیم. چطور نام urls را بفهمیم؟ من در داخل [source code](#) پکیج آن ها را پیدا کرده ام.

صفحه html ثبت نام هم در مسیر account/signup.html می باشد و ما دیگر از استفاده نمی کنیم بنابراین می توانیم تست مربوط به آن را پاک کنیم همینطور SignupPageView و CustomUserCreationForm که در بالای صفحه ایمپورت شده اند را نیز پاک می کنیم.

```
# accounts/tests.py
from django.contrib.auth import get_user_model
from django.test import TestCase
from django.urls import reverse, resolve
class CustomUserTests(TestCase):
...
class SignupTests(TestCase): # new
    username = 'newuser'
    email = 'newuser@email.com'

    def setUp(self):
        url = reverse('account_signup')
        self.response = self.client.get(url)

    def test_signup_template(self):
        self.assertEqual(self.response.status_code, 200)
        self.assertTemplateUsed(self.response, 'account/signup.html')
        self.assertContains(self.response, 'Sign Up')
        self.assertNotContains(self.response, 'Hi there! I should not be on the page.')

    def test_signup_form(self):
        new_user = get_user_model().objects.create_user(
            self.username, self.email)
        self.assertEqual(get_user_model().objects.all().count(), 1)
```

```
self.assertEqual(get_user_model().objects.all()[0].username, self.username)
self.assertEqual(get_user_model().objects.all()[0].email, self.email)
```

فایل تست را ران می کنیم.

```
$ docker-compose exec web python manage.py test
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
.....
```

---

```
Ran 14 tests in 0.410s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

## جمع بندی

در حال حاضر ما یک جریان ثبت نام کاربر را در وبسایت خودمان داریم که به این مورد احراز هویت با شبکه های اجتماعی به سرعت می تواند اضافه شود. در فصل بعد ما متغیر های محیطی را به پروژه خودمان برای امنیت و انعطاف بیشتر وب اپلیکیشن اضافه می کنیم.

## ۸-۸- متغیرهای اینوایرنمنت

یک پروژه، در زمان اجرا برخلاف هارد کدینگ داخل کدبیس متغیرهایی هستند که می توانند در اینوایرنمنت در نظر گرفته شوند و امنیت را بالا می بردند.

چرا امنیت بیشتر؟ زیرا می توانیم اطلاعات محرمانه (اطلاعات پایگاه داده، چون استفاده از ورژن کنترل (version control system) مانند git) به این معناست که فقط یک commit بد برای افزودن credentials برای همیشه باقی می ماند. این بدان معناست که هر کسی به کدبیس دسترسی داشته باشد کنترل کامل به روی پروژه دارد که این بسیار خطروناک است. بهتر است دسترسی افراد به برنامه محدود شود و متغیرهای محیطی راهی برای این کار ارائه می کنند.

مزیت دوم این است که متغیرهای محیطی، جایه جایی بین محیط های داخلی و عملی کد را آسان تر می کند. همانطور که خواهیم دید تنظیماتی وجود دارد که جنگو به طور پیش فرض از

آن برای توسعه آسان‌تر استفاده می‌کند. اما زمانی که همان پروژه برای محیط عملی آماده می‌شود باید تغییراتی را اعمال کرد.

در پایتون روش‌های مختلفی برای کار با متغیرهای محیطی وجود دارد اما برای این پروژه از environs استفاده می‌کنیم که شامل گزینه مخصوص برای جنگو است که پکیج‌های اضافی را همراه خود نصب می‌کند که در تنظیمات به ما کمک می‌کند..

```
$ docker-compose exec web pipenv install 'environs[django]==8.0.0'
```

```
$ docker-compose down
```

```
$ docker-compose up -d --build
```

در فایل config/settings.py سه خط imports برای اضافه کردن در بالای فایل در زیر وجود دارد.

```
# config/settings.py
from pathlib import Path
from environs import Env # new
env = Env() # new
env.read_env() # new
```

حالا همه چیز آماده است.

برای اولین environment variable خود SECRET\_KEY را تنظیم می‌کنیم، رشته‌ای که به طور تصادفی تولید شده و برای cryptographic signing استفاده می‌شود و هر زمان که دستور SECRET\_KEY اجرا شود ایجاد می‌شود. بسیار مهم است که SECRET\_KEY مخفی نگه داشته شود. در فایل config/settings.py من مقدار زیر را دارد:

```
. # config/settings.py
SECRET_KEY=')_s#exg*#w+-_
xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8*n'
```

توجه داشته باشید که نقل قول‌های اطراف SECRET\_KEY باعث می‌شود که تبدیل به رشته پایتونی شود. در واقع آنها بخشی از SECRET\_KEY نیستند که به آسانی اشتباه گرفته می‌شوند.

دو گام برای جابه‌جایی environment variables وجود دارد:

را به فایل docker-compose.yml اضافه کنید.

config/settings.py - را برای اشاره به متغیر بروز کنید.

در فایل docker-compose.yml web service environment در زیر اضافه کنید. این متغیری خواهد بود که آن را با مقدار DJANGO\_SECRET\_KEY با نام web در environment اضافه کنید. فایل آپدیت شده به این شکل است:

```
# config/settings.py
version: '3.8'
services:
  web:
    build: .
    command: python /code/manage.py runserver 0.0.0.0:8000
    volumes:
      - ./cod
    ports:
      - 8000:8000
    depends_on:
      - db
  environment:
    - "DJANGO_SECRET_KEY=_s#exg*w+#+xt=vu8b010%a&p@4edwyj0=(nqq90b9a8*n"
db:
  image: postgres:11
  volumes:
    - postgres_data:/var/lib/postgresql/data/
  environment:
    - "POSTGRES_HOST_AUTH_METHOD=trust"
```

volumes:

postgres\_data:

توجه کنید اگر SECRET\_KEY شما دارای علامت دلار \$ باشد باید یک علامت دلار دیگر اضافه کنید \$\$ در غیر این قدم دوم آپدیت کردن تنظیمات SECRET\_KEY در config/settings.py صورت با ارور مواجه می شوید!

```
# config/settings.py
```

```
SECRET_KEY = env("DJANGO_SECRET_KEY")
```

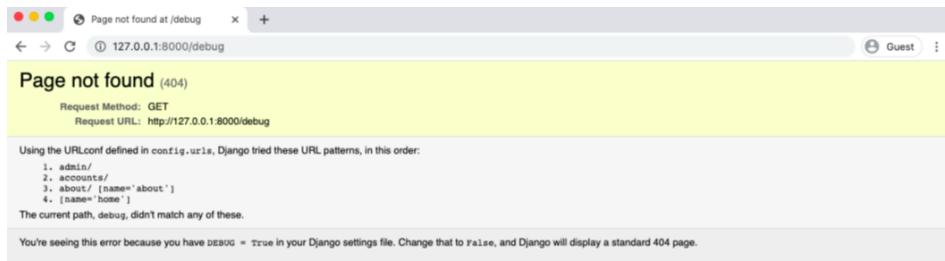
اگر وبسایت را رفرش کنید، خواهید دید همه چیز مانند قبل کار می کنند که همان چیزیست که می خواهیم. اگر به دلایلی SECRET\_KEY به درستی اجرا نشده بود، اروری خواهیم دید با عنوان، جنگو برای کار کردن به آن نیاز دارد.

خوانندگان ریزبین ممکن است متوجه شوند، با اینکه از environment variable استفاده می کنیم، مقدار SECRET\_KEY در سورس کد هنوز قابل مشاهده است. همانطور که صرفاً به docker-compose.yml منتقل شد. درست است! با این حال، وقتی وبسایت خود برای فاز نهایی آماده می کنیم فایل جداگانه‌ای را برای اهداف عملی ایجاد خواهیم کرد(docker-compose-production.yml) و از طریق فایل env آن را در محیط عملی بارگذاری خواهیم کرد که توسط گیت ردیابی نمی شود.

هر چند، در حال حاضر هدف این فصل استفاده از variables environment به صورت لوکال و برای مقادیری است که باید به طور حتم مخفی باشند و یا در محیط عملی تغییر کنند.

همانطور که چک لیست دیپلوی جنگو اشاره می کند، تنظیماتی وجود دارد که باید قبل از دیپلوی امن سایت آپدیت شوند. اصلی ترین بخش‌ها DEBUG و ALLOWED HOSTS هستند.

وقتی DEBUG روی True تنظیم شده باشد، جنگو پیامی طولانی و با جزئیات از باگ را در موقع وقوع یک ارور نشان می دهد. برای مثال از صفحه‌ای که وجود ندارد دیدن کنید، مانند .debug/



#### Debug Page

این برای اهداف ما به عنوان توسعه‌دهنده عالی است، اما همچنین یک نقشه راه برای یک هکر در محیط عملی است. وقتی DEBUG روی False تنظیم باشد، لازم است ALLOWED\_HOSTS را تغییر دهید که هاست و دامنه‌های خاصی را که می‌توانند به وبسایت دسترسی پیدا کنند را کنترل می‌کند. دو پورت محلی (localhost و 127.0.0.1) و همچنین herokuapp.com را اضافه خواهیم کرد که توسط Heroku برای وبسایت ما استفاده می‌شود.

فایل config/settings.py را با دو خط جدید آپدیت می‌کنیم:

```
# config/settings.py
DEBUG = False # new
ALLOWED_HOSTS = ['herokuapp.com', 'localhost', '127.0.0.1'] # new
```

صفحه وب را رفرش می‌کنیم

این همان رفتاری است که از سایت می‌خواهیم: بدون اطلاعات، فقط یک پیام عمومی. زمانی که وبسایت را دیپلوا می‌کنیم، از راهی بخصوص برای جابه‌جایی بین تنظیمات استفاده می‌کنیم، اما حال DEBUG را به متیر محیطی DJANGO\_DEBUG تغییر دهید.

```
# config/settings.py
DEBUG = env.bool("DJANGO_DEBUG")
```

سپس بروزرسانی docker-compose.yml را انجام دهید تا DJANGO\_DEBUG روی True تنظیم شود.

```
version: '3.8'
```

services:

web:

build::

command: python /code/manage.py runserver 0.0.0.0:8000

volumes:

-./code

ports:

- 8000:8000

depends\_on:

- db

environment:

- "DJANGO\_SECRET\_KEY=)\*\_s#exg\*#w+#+xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8\*n"  
- "DJANGO\_DEBUG=True"

db:

image: postgres:11

volumes:

- postgres\_data:/var/lib/postgresql/data/

environment:

- "POSTGRES\_HOST\_AUTH\_METHOD=trust"

volumes:

postgres\_data:

پس از تغییرات وبسایت را رفرش کنید و مانند قبل کار خواهد کرد

وقتی قبلتر True را نصب کردیم، "Django goodies" شامل پکیج djdatabase-url بود که تمام پیکربندی‌های مورد نیاز دیتابیس را شامل می‌شود، SQLite، PostgreSQL یا این بعدا در محیط عملی مفید خواهد بود.

حال می‌توانیم آن را با یک مقدار پیش‌فرض به صورت لوکال از PostgreSQL استفاده کنیم. پیکربندی DATABASES موجود را با موارد زیر آپدیت کنید:

```
# config/settings.py

DATABASES = {
    "default": env dj_db_url("DATABASE_URL",
        default="postgres://postgres@db/postgres")
}
```

هنگام دیپلوی، متغیر محیطی DATABASE\_URL توسط Heroku ایجاد می‌شود.

وبسایت را رفرش کنید تا از کارکرد درست همه‌چیز اطمینان حاصل کنید.

## ۸-۹-بخش نهم

در این فصل ما به طور کامل ایمیل را پیکربندی می‌کنیم و تغییر رمز عبور و بازنشانی رمز عبور را اضافه می‌کنیم. در حال حاضر ایمیل‌ها به طور واقعی به کاربران ارسال نمی‌شوند. آن‌ها به سادگی در کنسول خط فرمان در خروجی نشان داده می‌شوند. ما آن را با ثبت نام برای یک سرویس ایمیل ثالث، گرفتن یک API Key و به روزرسانی settings.py تغییر خواهیم داد. جنگو به بقیه‌اش نظارت می‌کند.

تا کنون تمام کارهای ما - مدل کاربر سفارشی، اپ pages، فایل‌های استاتیک، احراز هویت با gallauth و متغیرهای محیطی - می‌توانستند در هر پروژه جدیدی اعمال شوند. بعد از این فصل ما ساخت خود سایت B را برخلاف مراحل بنیادین آغاز خواهیم کرد.

بیایید برای یک اکانت کاربری جدید ثبت نام کنیم تا جریان کنونی ثبت نام کاربر را مرور کنیم. سپس ما آن را سفارشی می‌کنیم. اطمینان حاصل کنید که خارج شده اید و دوباره به صفحه ثبت نام بروید. من انتخاب کرده ام که از [testuser3@email.com] به عنوان نام کاربری و testpass123 به عنوان پسورد استفاده کنم



پس از ارائه این فصل، ما به صفحه خانه هدایت می‌شویم که یک عبارت خوش‌آمدگویی سفارشی نشان می‌دهد و یک ایمیل از طریق کنسول خط فرمان به ما ارسال می‌شود. شما می‌توانید این را با بررسی لگ‌های docker-compose logs بینید. برای سفارشی سازی این ایمیل ما ابتدا نیاز داریم که قالب‌های موجود را پیدا کنیم. به سورس کد django-allauth در گیت‌هاب بروید و در نتیجه می‌بینیم که از دو فایل استفاده شده است: یک فایل برای موضوع ایمیل، email\_confirmation\_subject.txt، و یک فایل برای بدنی ایمیل به نام email\_confirmation\_message.txt. برای هر دو فایل، ما ابتدا همان ساختار django-allauth را دوباره‌سازی می‌کنیم که به معنای ساخت یک فolder email داخل templates/account است و سپس آن دو فایل را override می‌کنیم

```
$ mkdir templates/account/email
```

```
$ touch templates/account/email/email_confirmation_subject.txt
```

```
$ touch templates/account/email/email_confirmation_message.txt
```

حال به سراغ خود پیام ایمیل تاییدیه می‌رویم:

```
{% load account %}{% user_display user as user_display %}{% load i18n %}\n\n{# autoescape off %}{% blocktrans with site_name=current_site.name%\nsite_domain=current_site.domain %}\n
```

Hello from {{ site\_name }}!

You're receiving this e-mail because user {{ user\_display }} has given yours as an e-mail address to connect their account.

To confirm this is correct, go to {{ activate\_url }}

```
{% endblocktrans %}{% endautoescape %}
```

```
{% blocktrans with site_name=current_site.name\
site_domain=current_site.domain %}

Thank you from {{ site_name }}!

{{ site_domain }}{% endblocktrans %}
```

توجه داشته باشید که بکالسلاش‌ها \ برای قالب‌بندی اضافه شده‌اند و داخل کد خام ضروری نیستند. به عبارت دیگر، شما می‌توانید بر حسب نیاز آن‌ها را از این کد - و هر مثال کد دیگری - حذف کنید.

شما احتمالاً متوجه شده‌اید که ایمیل پیش‌فرضی که ارسال شد به سایت ما با آدرس `example.com` ارجاع داده است که اینجا با {{ site\_name }} نشان داده شده است. این از کجا می‌آید؟ پاسخ در قسمت `sites` پنل ادمین جنگو است، که توسط `django-allauth` استفاده می‌شود. پس به پنل ادمین با آدرس <http://127.0.0.1:8000/admin/> بروید و در صفحه‌ی خانه روی لینک بزنید. `Sites` کلیک کنید.

The screenshot shows the Django Admin interface with the title "Change site". In the main form, the "Domain name" field contains "djangobookstore.com" and the "Display name" field contains "Django Bookstore". At the bottom, there are three buttons: "Delete" (red), "Save and add another" (blue), "Save and continue editing" (blue), and a large blue "SAVE" button. On the left, a sidebar lists "ACCOUNTS", "AUTHENTICATION AND AUTHORIZATION", and "SITES". The "SITES" section is currently selected and highlighted in yellow. The URL in the browser is `127.0.0.1:8000/admin/sites/site/1/change/`.

طمئن شوید که از سایت خارج شده اید و دوباره به صفحه ثبت‌نام بروید و یک کاربر جدید بسازید. من برای راحتی از استفاده کرده‌ام. `testuser4@email.com` استفاده کرده‌ام.

لاگین کنید و بعد از هدایت شدن به صفحه خانه، خط فرمان را بررسی کنید.

...

`web_1 | Content-Transfer-Encoding: 7bit`

`web_1 | Subject: [Django Bookstore] Confirm Your Sign Up`

`web_1 | From: admin@djangobookstore.com`

web\_1 | To: testuser4@email.com

web\_1 | Date: Mon, 03 Aug 2020 18:34:50 -0000

web\_1 | Message-ID:  
<156312929025.27.2332096239397833769@87d045aff8f7>

web\_1 |

web\_1 | Hi from Django Bookstore!

web\_1 |

web\_1 | You're receiving this e-mail because user testuser4 has given yours\  
as an e-mail address to connect their account.

web\_1 |

web\_1 | To confirm this is correct, go to <http://127.0.0.1:8000/accounts/confirm-email/NA:1hmjKk:6MiDB5XoLW3HAhePuZ5WucR0Fiw/>

web\_1 |

web\_1 | Thank you from Django Bookstore!

web\_1 | djangobookstore.com

روی لینک یکتاوی داخل ایمیل که به صفحه‌ی تایید ایمیل هدایت می‌کند کلیک کنید



### Confirm E-mail Address

Please confirm that [testuser4@email.com](mailto:testuser4@email.com) is an e-mail address for user testuser4.

خیلی جذاب نیست. بیایید آن را به روزرسانی کنیم تا با ظاهر بقیه‌ی سایت ما هماهنگ باشد.  
جستجوی دوباره در سورس [django-allauth](#) در گیت‌هاب مشخص می‌کند که نام و موقعیت این فایل [templates/account/email\\_confirm.html](#) است. پس بیایید قالب خودمان را ایجاد کنیم.

<!-- templates/account/email\_confirm.html -->

{% extends '\_base.html' %}

```

{ % load i18n % }

{ % load account % }

{ % block head_title % }{ % trans "Confirm E-mail Address" % }{ % endblock % }

{ % block content % }

<h1>{ % trans "Confirm E-mail Address" % }</h1>

{ % if confirmation % }

{ % user_display confirmation.email_address.user as user_display % }

<p>{ % blocktrans with confirmation.email_address.email as email % }Please
confirm

that <a href="mailto:{ { email } }">{ { email } }</a> is an e-mail address for user
{ { user_display } }.{ % endblocktrans % }</p>

<form method="post" action="{ % url 'account_confirm_email' confirmation.key
% }">

{ % csrf_token % }

<button class="btn btn-primary" type="submit">{ % trans 'Confirm' % }</button>
</form>139

Chapter 9: Email

{ % else % }

{ % url 'account_email' as email_url % }

<p>{ % blocktrans % }This e-mail confirmation link expired or is invalid. Please
<a href="{ { email_url } }">issue a new e-mail confirmation request</a>.<
{ % endblocktrans % }</p>

{ % endif % }

{ % endblock % }

```

صفحه را بروزرسانی می کنیم و تغییرات را می بینیم.

ایمیل‌هایی که تاکنون پیکربندی کرده‌ایم به طور کلی تحت نام «ایمیل‌های تراکنشی» ارجاع داده می‌شوند، زیرا آن‌ها بر اساس یک نوع فعالیت کاربر اتفاق می‌افتد. این در مقابل «ایمیل‌های بازاریابی» است که به طور مثال، همچون یک خبرنامه‌ی ماهانه‌اند. تعداد بسیار زیادی ارائه‌دهنده ایمیل‌های تراکنشی وجود دارد، مثل SendGrid، MailGun، Amazon's

برای جنگو تفاوتی نمی‌کند که از کدام ارائه‌دهنده استفاده شود، Simple Email Service. مراحل انجام کار برای همه‌ی آن‌ها مشابه است و بسیاری از آن‌ها امکان عضویت رایگان دارند. پس از اینکه در سرویس ایمیلی که انتخاب کردید ثبت‌نام انجام دادید، اغلب بین استفاده از Web API یا SMTP می‌توانید انتخاب کنید. پیکربندی SMTP آسان‌تر است، اما قابلیت‌های پیکربندی بیشتری دارد و قدرتمندتر است. با SMTP شروع کنید و کار خودتان را از آن‌جا پیش ببرید: پیکربندی‌های ایمیل می‌توانند به خودی خود کاملاً پیچیده باشند. پس از دریافت نام کاربری و رمز عبور از یک ارائه‌دهنده ایمیل، یکسری تنظیمات دقیق به جنگو اجازه می‌دهد تا برای ارسال ایمیل از آن‌ها استفاده کند. اولین قدم به روزرسانی تنظیمات EMAIL\_BACKEND است، که در انتهای فایل config/settings.py قرار دارد

چون ما قبل آن را آپدیت کرده‌ایم.

```
# config/settings.py
```

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend' # new
EMAIL_HOST_USER      ،   EMAIL_HOST    و سپس برای پیکربندی
EMAIL_HOST_PASSWORD و EMAIL_PORT    EMAIL_USE_TLS
EMAIL_HOST_PASSWORD به عنوان متغیرهای محلی بر اساس دستورالعمل‌های ارائه‌دهنده سرویس ایمیل شما. در سورس کد رسمی، EMAIL_BACKEND همان console باقی خواهد ماند، اما مراحل قبلی چگونگی اضافه کردن یک سرویس ایمیل بودند. اگر هنگام پیکربندی مناسب ایمیل نامید شدید، شما تنها نیستید! حداقل جنگو این کار را برای ما بسیار بسیار آسان‌تر کرده است.
```

## Books -۱۰-اپ

در این فصل می‌خواهیم یک اپلیکیشن تحت عنوان Books بسازیم که قرار است یک page شامل همه کتاب‌های در دسترس و به صورت اختصاصی یک page برای هر کتاب را به ما نمایش دهد. در حین انجام این پروژه روش‌های مختلف نوشتتن URL از جمله با استفاده از id، سپس با استفاده از slug و در نهایت با استفاده از UUID را نیز بررسی خواهیم کرد.

برای شروع ابتدا یک اپلیکیشن می‌سازیم.

```
$ docker-compose exec web python manage.py startapp books
```

برای اینکه اطمینان حاصل کنید جنگو اپلیکیشن جدیدی که ساخته‌اید را می‌شناسد text editor خود را باز کنید سپس به config/settings.py رفته و نام اپلیکیشن را در INSTALLED\_APPS [ ] اضافه کنید:

```
# config/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    # Third-party
    'allauth',
    'allauth.account',
    'crispy_forms',
    # Local
    'accounts',
    'pages',
    'books', # new
]
```

به طور کلی برای هر صفحه که قرار است توسط اپلیکیشن نمایش داده شود نیاز به مدل (model)، view و template خواهیم داشت. اینکه از کدام یک شروع به نوشتن برنامه کنیم در شرایط مختلف می‌تواند متفاوت باشد اما معمولاً شروع از model، به دلیل اینکه ساختار کلی برنامه را مشخص می‌کند می‌تواند یک گزینه‌ی مناسب باشد. قبل از نوشتن کدهای مربوط به مدل باید به این فکر کنیم که مدل ما قرار است شامل چه فیلدهایی باشد. برای اینکه از ساده‌ترین حالت شروع کنیم فیلدهای مدل را title، author و price در نظر می‌گیریم.

به رفته و مدل جدید Book را به صورت زیر اضافه کنید.

```
# books/models.py
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=200)
    price = models.DecimalField(max_digits=6, decimal_places=2)

    def __str__(self):
        return self.title
```

در قسمت بالایی از models را ایمپورت کردیم و در ادامه مدل Book را ساختیم که از کلاس models.Model ارث بری می‌کند. به این ترتیب ما در کلاس Book به هر چیزی که در django.db.models.Model وجود دارد دسترسی خواهیم داشت در عین حال می‌توانیم فیلدها و متدهای مورد نظر خود را نیز به آن اضافه کنیم.

برای فیلدهای title و author تعداد کاراکترها را به ۲۰۰ عدد محدود کردیم و برای price نیز از استفاده DecimalField کردیم که هنگام کار کردن با اعداد و ارقام واحد پول گزینه‌ی بسیار مناسبی است.

در انتهای یک متد \_\_str\_\_ نوشتیم که نحوه‌ی نمایش نمونه‌های object (این مدل در shell Admin و جنگو را مشخص می‌کند).

مدل ما آماده است. لازم است که یک migration record نیز از آن ایجاد کنیم

```
$ docker-compose exec web python manage.py makemigrations books
```

Migrations for 'books':

```
books/migrations/0001_initial.py
```

- Create model Book \$docker-compose exec web python manage.py migrate

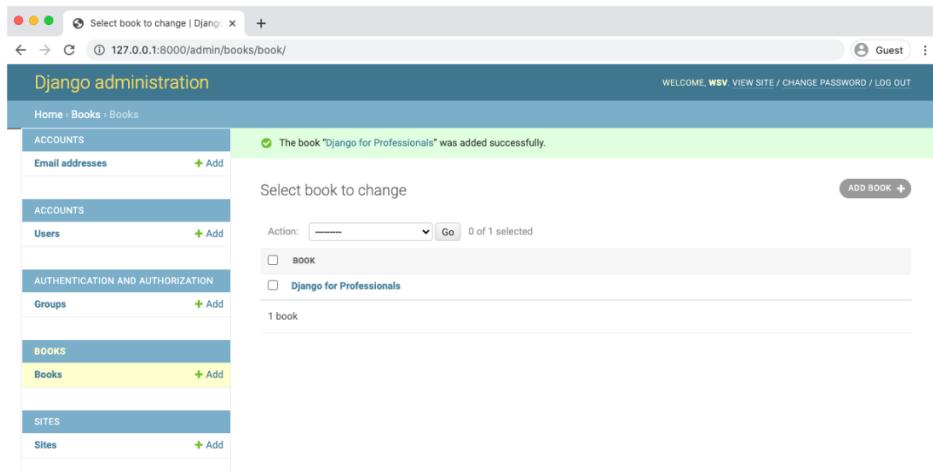
ما به ابزاری نیاز داریم که به کمک آن به داده‌های وبسایت خود دسترسی داشته باشیم. پنل ادمین جنگو کاملاً مناسب با این نیاز طراحی شده است. همچنین فراموش نکنید که فایل books/admin.py را به صورت زیر تغییر دهید در غیر اینصورت اپلیکیشن در بخش ادمین ظاهر نمی‌شود. بعد از سال‌ها کار با جنگو من تقریباً همیشه این مرحله رو فراموش می‌کنم.

```
# books/admin.py
```

```
from django.contrib import admin
from .models import Book
admin.site.register(Book)
```

بسیار خوب بباید یک نمونه از Book تحت عنوان ”Django for Professionals“ ایجاد کنیم. روی دکمه ای add(+) کلیک کنید. قسمت title را ”Django for Professionals“، قسمت author را ”William S.Vincent“ و price را ”\$39.00“ قرار دهید. نیازی به نوشتن علامت دلار در قسمت price نیست، در ادامه می‌توانیم در بخش template آن را به price اضافه کنیم.

بعد از کلیک روی دکمه ای ”Save“ به صفحه ای اصلی اپلیکیشن Books هدایت می‌شویم که title کتاب‌های اضافه شده را به ما نمایش می‌دهد.



باید فایل books/admin.py را مجدداً تغییر دهیم و مشخص کنیم کدام فیلدها نمایش داده شود.

```
# books/admin.py
from django.contrib import admin
from .models import Book

class BookAdmin(admin.ModelAdmin):
    list_display = ("title", "author", "price",)
```

admin.site.register(Book, BookAdmin)

به این ترتیب مدل خود را در دیتابیس ایجاد کردیم. در قدم بعد برای اینکه بتوانیم اطلاعاتی را نمایش دهیم باید URLها، Viewها و Template های لازم را ایجاد کنیم. اینکه از کدامیک شروع کنیم همیشه می تواند سؤال برانگیز باشد. شخصاً اکثر اوقات با URL ها شروع کرده، سپس به سراغ View ها رفته و در نهایت Template ها را می نویسم.

```
# config/urls.py
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    # Django admin
    path('admin/', admin.site.urls),

    # User management
    path('accounts/', include('allauth.urls')),

    # Local apps
    path("", include('pages.urls')),
    path('books/', include('books.urls')), # new
]
```

برای ایجاد ListView از یک BookListView به نام Generic Class-Based View استفاده می‌کنیم که به صورت پیش‌فرض در خود جنگو وجود دارد و برای مواردی از این دست ساخته شده است. تنها کاری که باید انجام دهیم این است که مدل و مسیر template مربوطه را مشخص کنیم.

```
# books/views.py
from django.views.generic import ListView
from .models import Book
```

```
class BookListView(ListView):
    model = Book
    template_name = 'books/book_list.html'
```

ایجاد کردن یک پوشه جداگانه در پوشه templates مخصوص فایل‌های html هر اپلیکیشن، یک کار کاملاً اختیاری است اما انجام دادن این کار زمانی که تعداد فایل‌های html یا تعداد

اپلیکیشن ها زیاد می شود بسیار مفید و کمک کننده خواهد بود. پس ما نیز یک پوشه به نام درست می کنیم و فایل book\_list.html را در این پوشه ایجاد می کنیم.

```
<!-- templates/books/book_list.html -->
```

```
{% extends '_base.html' %}
```

```
{% block title %}Books{% endblock title %}
```

```
{% block content %}
```

```
{% for book in object_list %}
```

```
<div>
```

```
    <h2><a href="">{{ book.title }}</a></h2>
```

```
</div>
```

```
{% endfor %}
```

.{% endblock content %}.

در قسمت بالایی مشخص شده که این فایل درواقع از فایل base.html توسعه پیدا کرده است و به این ترتیب کدهای مورد نظر خود را می توانیم در block content در قرار دهیم. برای اینکه کتابها را نمایش دهیم، با استفاده از زبان Django template جنگو (Django template language) یک حلقه `for` ساده روی لیست کتابها می نویسیم. توجه داریم که درواقع شامل همه کتابهای موجود در view که قبل تر نوشتیم است و به کمک ListView ایجاد شده است. در آخرین مرحله لازم است که یک بار container های خود را down و مجدداً up کنیم تا تغییرات ایجاد شده با ریلود شدن فایل settings.py اعمال شود. در غیر اینصورت با یک صفحه error مواجه خواهیم شد و در قسمت logs پیام "ModulNotFoundError: No module named 'books.url'" را دریافت خواهیم کرد.

```
$ docker-compose down
```

```
$ docker-compose up -d
```

حالا با مراجعه به آدرس <http://127.0.0.1:8000/books> صفحه‌ی کتاب‌ها هدایت می‌شویم.



همانطور که دیدیم به صورت پیش‌فرض کلاس ListView از متغیری به نام object\_list به عنوان context گرفته است برای template مربوطه ارسال شود استفاده می‌کند. اگرچه این روش کاملاً درست کار می‌کند اما روش بهتر و اصطلاحاً friendly تر استفاده از یک متغیر با نامی مرتبط است و این کار به کمک context\_object\_name، یک attribute از کلاس ListView انجام پذیر است.

```
# books/views.py
from django.views.generic import ListView
from .models import Book
```

```
class BookListView(ListView):
    model = Book
    context_object_name = 'book_list' # new
    template_name = 'books/book_list.html'
```

```
<!-- templates/books/book_list.html -->
{ % extends '_base.html' % }
```

```
{ % block title % }Books{ % endblock title % }
```

```
{ % block content % }
```

```
{% for book in book_list %}

<div>
    <h2><a href="">{{ book.title }}</a></h2>
</div>

{% endfor %}

%}endblock content{.%
```

صفحه را refresh کنید و همه چیز مثل قبل کار می کند. این تکنیک به خصوص زمانی که پروژه ها بزرگ‌تر می‌شوند و developer های بیشتری روی آن کار می‌کنند مؤثر تر و کمک کننده تر خواهد بود. به طور مثال برای یک frontend developer حدس زدن اینکه متغیر object\_list مربوط به چه قسمتی است بسیار سخت تر خواهد بود در مقایسه با زمانی که از یک نام مرتبط استفاده شده باشد

حالا می‌توانیم به کمک یک DetailView Generic Class-Based View دیگر به نام DetailView یک صفحه اختصاصی برای هر کتاب ایجاد کنیم.

مشابه قبل که page مربوط به نمایش همه کتاب‌ها را ایجاد کردیم، در اینجا نیز ابتدا URL را می‌نویسیم. مشابه زیر در خط دوم import BookDetailView را می‌کنیم و مسیر url را مربوط به هر کتاب قرار می‌دهیم که به عنوان یک عدد صحیح به صورت primary\_key int:pk نوشته می‌شود.

```
# books/urls.py
from django.urls import path
from .views import BookListView, BookDetailView # new
urlpatterns = [
    path("", BookListView.as_view(), name='book_list'),
    path('<int:pk>/', BookDetailView.as_view(), name='book_detail'), # new
]
```

برای هر مدلی که ما در دیتابیس ایجاد می‌کنیم جنگو به صورت خودکار یک فیلد دیگر به Book ایجاد می‌کند. پس در حالی که برای مدل auto-incrementing primary key

فقط فیلد های title و author و body را نوشتهیم، جنگو در پشت صحنه یک فیلد دیگر با نام `id` بنه عنوان `primary_key` به مدل Book اضافه کرد.

دسترسی به این فیلد از طریق هر کدام از نام های id یا pk امکان پذیر است.

فایل template را جوری ایجاد کنید که همه ی فیلد های کتاب مربوطه را نشان دهد. همچنین می توانیم title کتاب را در tag قرار دهیم تا نام کتاب در تب مرورگر نیز نمایش داده شود.

```
<!-- templates/books/book_detail.html -->
{ % extends '_base.html' % }

{ % block title % } {{ object.title }} { % endblock title % }

{ % block content % }
<div class="book-detail">
<h2><a href="">{{ object.title }}</a></h2>
<p>Author: {{ object.author }}</p>
<p>Price: {{ object.price }}</p>
</div>
{ % endblock content %}.
```

در یک پروژه استفاده از primary key(PK) یا ID می تواند کمی گیج کننده باشد به خصوص که رفتار) یک DetailView از جنگو که قبل تر با آن کار کردیم) با این دو به نحوی است که کاملاً می توانند به جای یکدیگر استفاده شوند به طوری که مشابه یا برابر با یکدیگر تلقی شوند. با این حال یک تفاوت کوچک و البته مهم میان این دو وجود دارد. در واقع یک فیلد از مدل است که توسط خود جنگو به صورت اتوماتیک به فیلد های دیگر مدل اضافه می شود و به صورت اتوماتیک نیز مقدار آن افزایش پیدا می کند. به طور مثال برای مدل Book اولین کتاب دارای id یک، دومین کتاب دارای id دو و به همین ترتیب برای نمونه های بعدی

اضافه می شود. به صورت پیش فرض فیلد id یک مدل، همان primary key یا pk آن مدل نیز خواهد بود.

هرچند primary key یا pk یک مدل لزوماً id آن مدل نیست و این امکان وجود دارد تا primary key یک مدل را به صورت دستی تغییر دهیم. به طور مثال می توانیم pk را برابر با فیلد دیگری مثل object\_id قرار دهیم که البته بسته به نیاز ما می تواند متفاوت باشد. همچنین زبان پایتون یک متدهای built-in (id) دارد که ممکن است با فیلد id اشتباہ گرفته شود و خطای را باگ هایی را برای ما ایجاد کند.

به طور خلاصه pk به فیلد primary key یک مدل اشاره می کند که به صورت پیش فرض همان فیلد id مدل خواهد بود. در قسمت بعد می خواهیم فیلد id مدل خود را تغییر می دهیم.

اگر به خاطر داشته باشید هنگام نوشت URL مربوط به DetailView از فیلد pk استفاده کردیم که راهی آسان و سریع است و ما را به چیزی که می خواهیم می رساند. اما این سبک نوشت URL مناسب پروژه هایی که قرار است در دنیای واقعی استفاده شوند نیست. همانطور که قبل تر گفته شد pk به صورت پیش فرض برابر با فیلد id مدل ما است. در این شرایط یک هکر با دیدن URL، می تواند متوجه تعداد رکورد های موجود در دیتابیس ما شود و یا با استفاده از id که در URL قابل مشاهده است یک حمله (هک) به وبسایت ما انجام دهد. همچنین نوشت URL به این شکل، در شرایطی که چندین front-end داشته باشیم می تواند مشکلات synchronization را نیز ایجاد کند.

دو روش جایگزین به جای استفاده مستقیم از pk وجود دارد. روش اول استفاده از slug است. درواقع یک label کوتاه است که معمولاً در URLs استفاده می شود. به طور مثال زمانی که در مدل Book نمونه ای با عنوان Django for Professionals slug اضافه کردیم، می توانست Book مدل این عنوان را در جنگو این امکان وجود دارد که فیلدی به صورت SlugField نیز تعریف کنیم. مقدار دهنده به فیلد slug می تواند به صورت دستی و با توجه به مقدار یک فیلد دیگر (به طور مثال در مدل Book با توجه به فیلد title) یا اینکه به صورت اتوماتیک هنگام save شدن object از مدل مربوطه (فرآخونی متدهای save) باشد. چالش اصلی هنگام استفاده از slug ها، هندل کردن slug های مشابه برای object های متفاوت است که می توانیم با اضافه کردن رشته ای از اعداد و حروف تصادفی

در انتهای slug این مشکل احتمالی را برطرف کنیم. توجه داریم که حتی با استفاده از نیز مشکلات synchronization همچنان باقی می‌ماند.

روش دوم و بهتر استفاده از UUID (Universally Unique Identifier) است که جنگو هم‌اکنون به کمک یک فیلد اختصاصی به نام UUIDField از UUID نیز پشتیبانی می‌کند.

بسیار خوب بباید یک فیلد UUID را به مدل خود اضافه کنیم و سپس URL را نیز با استفاده از همین فیلد به روز رسانی کنیم.

ابتدا در بالا import uuid را کنید و بعد از آن فیلد id را به عنوان primary key مدل و به صورت یک UUIDField به روز رسانی کنید. همچنین از uuid4 برای encryption استفاده کنیم. به این ترتیب می‌توانیم با DetailView که در حالت عادی به یکی از فیلدهای pk یا slug نیاز دارد، کار کنیم. در صورتی که بخواهیم DetailView با فیلد UUID دیگری به غیر از pk یا slug کار کند باید اصلاحات بیشتری هنگام نوشتن View مربوطه انجام دهیم.

```
# books/models.py
import uuid # new
from django.db import models
from django.urls import reverse
class Book(models.Model):
    id = models.UUIDField( # new
        primary_key=True,
        default=uuid.uuid4,
        editable=False)
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=200)
    price = models.DecimalField(max_digits=6, decimal_places=2)
    def __str__(self):
        return self.title
    def get_absolute_url(self):
        return reverse('book_detail', args=[str(self.id)])
```

در URL path BookDetailView با جای int uuid، ایجاد قرار دهید.

مشابه با migration file ها ساده‌ترین روش پاک کردن volume و ایجاد مجدد آن است. باید به این موضوع توجه داشته باشیم که انجام دادن کارهایی از این دست برای پروژه هایی که در ابتدای راه و در مراحل development است کاملاً عادی است اما برای پروژه هایی که از مراحل ابتدایی گذشته اند و یا اینکه در حالت production هستند باید از روش‌های دیگر و پیچیده‌تری استفاده کرد.

دستورات زیر شامل پاک کردن volume مربوط به دیتابیس، up کردن container های web و db، ایجاد کردن migration file های جدید برای اپلیکیشن book، اعمال کردن migartion و ها به دیتابیس با دستور migrate و ایجاد مجدد یک superuser می‌باشد.

```
$ docker volume rm books_postgres_data
$ docker-compose up -d
$ docker-compose exec web python manage.py makemigrations books
$ docker-compose exec web python manage.py migrate
$ docker-compose exec web python manage.py createsuperuser
```

بسیار خوب بیایید لینک مربوط به page همه‌ی کتاب‌ها را در navbar نیز اضافه کنیم. می‌توانیم این کار را به کمک tag url template book\_list و نام URL که بود انجام دهیم.

```
<!--templates/_base.html -->
<nav class="my-2 my-md-0 mr-md-3">
    <a class="p-2 text-dark" href="{% url 'book_list' %}">Books</a>
    >    <a class="p-2 text-dark" href="{% url 'about' %}">About</a>
```

ر این قسمت می‌خواهیم برای view model و view خود تست بنویسیم. باید اطمینان حاصل کنیم که مدل ما و متدهایی که برای آن نوشتمیم (شامل متدهای str به درستی کار می‌کنند. همچنین ListView و DetailView را نیز تست خواهیم کرد.. یک نمونه تست نویسی می‌تواند مشابه زیر باشد که در فایل books/tests.py نوشته می‌شود.

```
# books/tests.py
from django.test import TestCase
from django.urls import reverse
from.models import Book

class BookTests(TestCase):

    def setUp(self):
        self.book = Book.objects.create(
            title='Harry Potter',
            author='JK Rowling',
            price='25.00',
        )

    def test_book_listing(self):
        self.assertEqual(f'{self.book.title}', 'Harry Potter')
        self.assertEqual(f'{self.book.author}', 'JK Rowling')
        self.assertEqual(f'{self.book.price}', '25.00')

    def test_book_list_view(self):
        response = self.client.get(reverse('book_list'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, 'Harry Potter')
        self.assertTemplateUsed(response, 'books/book_list.html')

    def test_book_detail_view(self):
```

```
response = self.client.get(self.book.get_absolute_url())
no_response = self.client.get('/books/12345/')
self.assertEqual(response.status_code, 200)
self.assertEqual(no_response.status_code, 404)
self.assertContains(response, 'Harry Potter')
self.assertTemplateUsed(response, 'books/book_detail.html')
```

در ابتدا `TestCase` را در بالا import کردیم. سپس در متدهای `setUp` و `tearDown` نمونه از مدل `Book` ایجاد کردیم که در ادامه تست نویسی ها به آن نیاز خواهیم داشت. بررسی می کند که نحوه نمایش فیلد های نمونه ای که اضافه کردیم با آن چیزی که انتظار می رود برابر باشد. با `reverse` URL و گرفتن `reponse`، بررسی می کند که `HTTP` response body با کمک متدهای `status_code` و `assertEqual` با `200` برابر باشد و یا اینکه در `assertContains` که انتظار می رود موجود باشد و یا اینکه از `Template` صحیح استفاده شده باشد. در آخر `tearDown` می کند تا مطابق تست های قبل همه چیز مطابق انتظار بازگردانده شده باشد. در اینبار برای `page` های اختصاصی کتاب ها پیش برود. همچنین در این تست بررسی شده است که اگر به یک `page` با آدرسی اشتباه مراجعه شود `HTTP status_code` بازگردانده شده برابر با `400` باشد. در تست نویسی مطلوب است که همیشه، هم مواردی که انتظار داریم جواب صحیح بدهند و هم مواردی که انتظار داریم با خطأ مواجه شوند را بررسی کنیم.

بسیار خوب تست ها را اجرا کنید. همگی باید با موفقیت اجرا شوند.

```
$ docker-compose exec web python manage.py test
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
.....
```

---

```
Ran 17 tests in 0.369s
```

OK

Destroying test database for alias 'default'

...

## نتیجه‌گیری

در پایان این بخش طولانی می‌توانیم ببینیم که ساختار پروژه‌ی Bookstore روش‌تر و واضح‌تر از قبل شده است. در این فصل ما یک مدل برای کتاب‌ها ایجاد کردیم، با نحوه‌ی تغییر ساختار URL‌ها آشنا شدیم و یاد گرفتیم که چطور از یک UUID pattern امن استفاده کنیم.

## ۱۱-۸- دسترسی‌ها

در حال حاضر هیچ مجوزی برای پروژه کتابفروشی ما تعیین نشده است. هر کاربری می‌تواند بدون احراز هویت همه‌ی صفحات را ببیند و امکان انجام تمامی کار‌ها را داشته باشد. اگر چه آزاد بودن دسترسی‌ها برای نمونه سازی مناسب است اما باید مجوز‌ها را لازم پیش از تبدیل سایت به محصول قرار داده شوند. جنگو امکانات پیش ساخته‌ای را برای محدودسازی کاربرانی که به سیستم وارد نشده‌اند، قفل کردن صفحات و دسترسی افراد احراز هویت شده، گروه‌های خاص و یا کاربران با دسترسی‌های خاص فراهم می‌کند.

بطور جالبی چندین راه برای اضافه کردن مجوز‌های اساسی وجود دارد که محدود کردن دسترسی به کاربران وارد شده یکی از آن‌ها است. این کار را می‌توان از یک روش ساده و مستقیم با استفاده از تابع دکوریتور login\_required (Decorator) login\_required چون از class استفاده می‌کیم با استفاده از LoginRequired mixin view انجام داد.

ابتدا با محدود کردن دسترسی به صفحات کتاب و ارائه‌ی امکان دسترسی به کاربران وارد شده شروع می‌کنیم، بدین منظور یک لینک در نوار بالایی صفحه وجود دارد. البته این شرایطی نیست که کاربر به طور تصادفی آدرس را پیدا کند (هر چند این امکان وجود دارد) به همین دلیل هم آدرس عمومی می‌باشد.

ابتدا import می‌کنیم و پیش از قرار می‌دهیم. در این حالت ابتدا وارد شدن کاربر به سیستم بررسی می‌شود و اگر کاربر وارد نشده باشد ListView بازگزاری نمی‌شود. بخش دیگر قرار دادن یک Url به منظور هدایت کاربر وارد نشده به آن است تا وارد سیستم شود. این Url به منظور ورود به سیستم است و چون ما از django-

استفاده می کنیم account-login نامیده می شود. اگر از سیستم سنتی احراز هویت جنگو استفاده می کردیم آنگاه این login نامیده میشد.

ساختار کلی برای BookDetailView نیز به صورت مشابه است و باید ساختار و یک آدرس LoginRequiredMixin را اضافه نماییم.

```
# books/views.py
from django.contrib.auth.mixins import LoginRequiredMixin # new
from django.views.generic import ListView, DetailView
from.models import Book
class BookListView(LoginRequiredMixin, ListView): # new
model = Book
context_object_name = 'book_list'
template_name = 'books/book_list.html'
login_url = 'account_login' # new
class BookDetailView(LoginRequiredMixin, DetailView): # new
model = Book
context_object_name = 'book'
template_name = 'books/book_detail.html'
login_url = 'account_login' # new
```

حال اگر از سیستم خارج شده و بروی لینک Books کلیک کنید به صفحه‌ی ورود به سیستم هدایت می شود در حالی که اگروارد سیستم شده باشد با کلیک روی لینک فوق لیست کتاب‌ها نمایش داده می شوند. حتی اگر uuid یک کتاب را نیز بدانید و با وارد کردن آن سعی در دسترسی به اطلاعات یک کتاب بدون ورود به سیستم را داشته باشد مجدداً به صفحه ورود هدایت خواهد شد.

جنگو از یک permission system پایه استفاده می کند که بوسیله‌ی پنل ادمین کنترل می شود. برای نشان دادن آن نیاز به ایجاد یک کاربر داریم. به homepage پنل ادمین رفته و بر روی "Add +" در کنار users کلیک کنید.

صفحه‌ی دوم به ما اجازه می‌دهد تا یک آدرسی ایمیل مناسب را به عنوان قرار دهیم. ما از `dajngo-allauth` استفاده می‌کنیم بنابراین برای ورود، به `email` نیاز داریم اما چون پنل ادمین را شخصی سازی نکرده ایم سیستم برای ایجاد کاربر جدید به نام کاربری نیاز دارد.

اگر نخواهید از سیستم معمول کاربران استفاده کنید یعنی به جای `AbstractUser` از `AbstarctBaseUser` استفاده نمایید به فصل ۳ برگردید، در آن جا یک مدل `User` را به شکل شخصی سازی شده (Customize) ایجاد کرده ایم.

با پایین تر آمدن در صفحه گزینه‌هایی برای تنظیم گروه‌ها به در کنار مجوز‌های کاربران وجود دارد. این لیستی از امکان‌های پیش فرضی می‌باشد که جنگو فراهم نموده است که اکنون از آن استفاده نمیکنیم زیرا در قسمت بعد مجوز‌های دلخواه برای کاربران را قرار خواهیم داد پس بر روی دکمه‌ی `save` در پایین گوشه‌ی سمت راست کلیک کنید.

### مجوز‌های دلخواه (custom permissions)

معمولًا در پروژه‌های جنگو از `custom permissions` استفاده می‌شود که می‌توانیم آن‌ها را با استفاده از `meta class` بر روی مدل‌های دیتابیس تنظیم نماییم.

برای مثال بگذارید شرایط خاصی که در آن نویسده امکان خواندن همه‌ی کتاب‌ها را دارد را اضافه کنیم، در واقع نویسنده‌ها به `DetailView` دسترسی خواهند داشت. می‌توانیم محدودیت‌های خیلی بیشتری را بر اساس کتاب‌ها اعمال کنیم ولی همین برای گام اول مناسب است.

در فایل `books/models.py` یک `Meta class` را اضافه می‌کنیم و `permissions` را به گونه‌ای تنظیم می‌کنیم که در پنل ادمین قابل مشاهده باشد.

خوب هست که هرگاه تغییراتی را در قسمتی از کد ایجاد می‌کنیم تست را اجرا نماییم. در واقع هدف از تست این است که تغییراتی که در کد ایجاد کردیم باعث بروز مشکل در قسمت دیگری از برنامه نشود.

```
$ docker-compose exec web python manage.py test
```

...

Ran 17 tests in 0.519s

FAILED (failures=2)

این نشان می دهد که تعدادی تست رد شده داریم. به طور مشخص test\_book\_list\_view کد موقعیت ۳۰۲ برخورده اند که نشان دهنده ای تغییر مسیر است و در صورت عدم وجود مشکل کد موقعیت آن ها باید برابر ۲۰۰ باشد. علت این مشکل این است که login required به لیست ویو کتاب ها اضافه کرده ایم ولی برای دسترسی به جزئیات صفحه نیز کاربر به مجوز special\_status نیازمند است. گام نخست وارد کردن permission با استفاده از توابع از پیش ساخته شده ای احراز هویت است. سپس داخل BookTests واقع در books/tests.py را به متدها setUp اضافه کنید تا همه ای تست های ما امکان پذیر باشد. test\_book\_list\_view را به قسمت های جداگانه برای کاربران وارد شده و خارج شده تقسیم می کنیم. همچنین deayil veiew set را بروزرسانی می کنیم تا بررسی برای وجود مجوز صحیح برای هر کاربر نیز صورت گیرد.

```
# books/models.py
```

```
from django.contrib.auth import get_user_model
from django.contrib.auth.models import Permission # new
from django.test import Client, TestCase
from django.urls import reverse
from.models import Book, Review
class BookTests(TestCase):
    def setUp(self):
        self.user = get_user_model().objects.create_user(
            username='reviewuser',
            email='reviewuser@email.com',
            password='testpass123'
        )
        self.special_permission = Permission.objects.get(
            codename='special_status' # new
```

```
self.book = Book.objects.create(  
    title='Harry Potter',  
    author='JK Rowling',  
    price='25.00',  
)  
  
self.review = Review.objects.create(  
    book = self.book,  
    author = self.user,  
    review = 'An excellent review',  
)  
  
def test_book_listing(self):  
    ...  
  
    def test_book_list_view_for_logged_in_user(self): # new  
        self.client.login(email='reviewuser@email.com', password='testpass123')  
        response = self.client.get(reverse('book_list'))  
        self.assertEqual(response.status_code, 200)  
        self.assertContains(response, 'Harry Potter')  
        self.assertTemplateUsed(response, 'books/book_list.html')  
  
    def test_book_list_view_for_logged_out_user(self): # new  
        self.client.logout()  
        response = self.client.get(reverse('book_list'))  
        self.assertEqual(response.status_code, 302)  
        self.assertRedirects(  
            response, '%s?next=/books/' % (reverse('account_login')))  
        response = self.client.get(  
            '%s?next=/books/' % (reverse('account_login')))  
        self.assertContains(response, 'Log In')  
  
    def test_book_detail_view_with_permissions(self): # new  
        self.client.login(email='reviewuser@email.com', password='testpass123')
```

```

self.user.user_permissions.add(self.special_permission)
response = self.client.get(self.book.get_absolute_url())
no_response = self.client.get('/books/12345/')
self.assertEqual(response.status_code, 200)
self.assertEqual(no_response.status_code, 404)
self.assertContains(response, 'Harry Potter')
self.assertContains(response, 'An excellent review')
self.assertTemplateUsed(response, 'books/book_detail.html')
...

```

## ۱۲- جستجو

جستجو یکی از ویژگی های اساسی اکثر وبسایت ها و هر چیزی مربوط به تجارت الکترونیک-e-commerce) مانند کتابفروشی ما. در این فصل نحوه پیاده سازی جستجوی ساده را با فرم و فیلتر هارا یاد می گیریم. سپس آن را با بهبود می بخشیم و نگاه عمیق تری به آپشن های سرج در جنگو می اندازیم. در حال حاضر فقط سه کتاب در دیتابیس خود داریم اما این کد به تعداد کتاب های دلخواهمان مقیاس پذیر است.

جستجو شامل دو بخش است:

- یک فرم که کوئری جستجو کاربر را با خود ارسال می کند.

- سپس یک صفحه که نتایج جستجو را بر اساس کوئری ارائه می کند.

تیکین نوع مناسب فیلتر جایی است که جستجو، سخت و جالب می شود. اما ابتدا باید فرم و صفحه نتایج جستجو را ایجاد کنیم.

در این بخش می توان با یکی از دو تا شروع کرد که ما اول بخش فیلتر و سپس فرم را آماده می کنیم.

## صفحه نمایش نتایج

با صفحه نتایج شروع می کنیم. مانند سایر صفحات جنگو باید URL و template view، اختصاصی ایجاد کنیم. ترتیب پیاده سازی اهمیت چندانی ندارد ولی ما به همان ترتیب پیش می رویم.

داخل `books/urls.py` با نام `search_results` اضافه کنید که از `SearchResultsListView` استفاده می کند.

```
# books/urls.py
from django.urls import path
from .views import BookListView, BookDetailView, SearchResultsListView # new
```

```
urlpatterns = [
    path('', BookListView.as_view(), name='book_list'),
    path('<uuid:pk>', BookDetailView.as_view(), name='book_detail'),
    path('search/', SearchResultsListView.as_view(),
         name='search_results'), # new
]
```

و بو (view) بعدی `SearchResultsListView` است که در حال حاضر لیستی از تمام کتاب های موجود است. این گزینه اصلی برای استفاده از `ListView` است. آن `template` `search_results.html` نامیده می شود و در پوشه `templates/books/` قرار دارد. تنها کد جدید، مربوط به `SearchResultsListView` است همانطور که قبل از `ListView` و مدل `Book` را در بالای فایل ایمپورت کردیم.

```
# books/views.py
class SearchResultsListView(ListView): # new
    model = Book
    context_object_name = 'book_list'
    template_name = 'books/search_results.html'
```

حال تمام کتاب های موجود، بر اساس عنوان، نویسنده و قیمت لیست می شوند

```
<!-- templates/books/search_results.html -->  
{% extends '_base.html' %}  
  
{% block title %}Search{% endblock title %}  
  
{% block content %}  
<h1>Search Results</h1>  
{% for book in book_list %}  
<div>  
    <h3><a href="{{ book.get_absolute_url }}">{{ book.title }}</a></h3>  
    <p>Author: {{ book.author }}</p>  
    <p>Price: $ {{ book.price }}</p>  
</div>  
{% endfor %}  
/.}endblock content{/.
```

در جنگو از کوئری ست برای فیلتر کردن نتایج مدل در دیتابیس (پایگاه داده) استفاده می شود. در حال حاضر صفحه نتایج جستجوی ما بنظر نمی رسد که یکی باشند زیرا همه نتایج را از مدل Book نشان می دهد. در نهایت ما می خواهیم فیلتر را بر اساس کوئری جستجوی کاربر اجرا کنیم، اما ابتدا چندین روش فیلتر کردن را کار میکنیم.

به نظر می رسد راه های متعددی برای سفارشی کردن کوئری ست وجود دارد، از جمله از طریق یک Manager در خود مدل، اما برای ساده نگه داشتن کارها، میتوانیم فقط با یک خط یک فیلتر اضافه بکنیم. بریم که انجامش بدیم!

ما میتوانیم ویژگی (attribute) کوئری ست را در ListView که به طور پیش فرض همه نتایج را نشان می دهد، لغو کنیم. مستندات کوئری ست کاملاً قوی و جزئیات است، اما اغلب که به حروف کوچک و بزرگ حساس است) یا (contains که به حروف بزرگ و

کوچک حساس نیست) نقطه شروع خوبی هستند. ما فیلتر را بر اساس عنوانی که "شامل" نام "مبتدیان" است اجرا خواهیم کرد.

```
# books/views.py
class SearchResultsListView(ListView):
    model = Book
    context_object_name = 'book_list'
    template_name = 'books/search_results.html'
    queryset = Book.objects.filter(title__icontains='beginners') # new
```

استفاده از `filter()` قدرتمند است و حتی می توان فیلتر هارا با هم زنجیره کرد مانند جستجوی تمام عنوانی که حاوی "django" و "beginners" هستند. با این حال اغلب شما جستجوهای پیچیده تری می خواهید که فقط نیازی نیست از AND استفاده کنید بلکه از OR هم میتوانید استفاده کنید، آن وقت زمان استفاده از `Q objects` فرا میرسد.

در اینجا مثالی وجود دارد که در آن فیلتر را برای جستجوی نتیجه ای که با عنوان "beginners" یا "api" مطابقت دارد، تنظیم می کنیم. این کار به سادگی با import `Q` کردن در بالای فایل انجام میشود و سپس به طور نامحسوس `query` موجود خود را تغییر می دهیم. نماد عملگرد `~` به معنی یا است. ما میتوانیم در هر فیلد در دسترس فیلتر انجام دهیم: نه فقط عنوان، بلکه نویسنده یا قیمت به دلخواه.

با افزایش تعداد فیلتر ها، جدا کردن `get_queryset()` از `queryset` طریق اور را مفید باشد. این کاری است که ما در اینجا انجام خواهیم داد، اما توجه داشته باشید که این کار اختیاری است.

```
# books/views.py
from django.db.models import Q # new
...
class SearchResultsListView(ListView):
    model = Book
    context_object_name = 'book_list'
```

```
template_name = 'books/book_list.html'
```

```
def get_queryset(self): # new  
    return Book.objects.filter(  
        Q(title__icontains='beginners') | Q(title__icontains='api')  
)
```

اساسا فرم وب ساده هستند: ورودی کاربر را می گیرد و آن را از طریق متدهای POST یا URL ارسال می کند. با این حال، در عمل، این رفتار اساسی وب میتواند بسیار پیچیده باشد.

اولین مسئله، ارسال داده های فرم است: داده ها واقعا کجا می روند و ما آن را در آنجا چگونه هندل می کنیم؟ ناگفته نماند که هر زمان که به کاربران اجازه می دهیم داده ها را به یک سایت ارسال کنند، نگرانی های امنیتی متعددی به وجود می آید.

تنها دو گزینه برای نحوه ارسال فرم وجود دارد: از طریق متدهای GET (HTTP) یا POST. داده های POST یک فرم را بسته بندی می کند، آن را برای انتقال رمزگذاری می کند، آن را به سرور ارسال می کند، و سپس یک پاسخ دریافت می کند. هر درخواستی که وضعیت دیتابیس را تغییر دهد (create, edits and deletes) (ایجاد، ویرایش، حذف) باید از متدهای POST استفاده شود.

بسته بندی می کند GET داده های فرم را به صورت string که آن به URL مقصود اضافه می شود. متدهای GET فقط باید برای درخواست هایی استفاده شود که بر وضعیت برنامه تأثیر نمی گذارد، مانند جستجویی که در آن هیچ چیز در دیتابیس تغییر نمی کند اساساً ما فقط یک ویو (view) لیست فیلتر شده را انجام می دهیم.

اگر بعد از بازدید Google.com به URL نگاه کنید عبارت جستجوی خود را در خود صفحه نتایج جستجوی واقعی مشاهده خواهید کرد.

برای اطلاعات بیشتر، موزیلا راهنمای دقیقی در مورد ارسال داده های فرم و اعتبار سنجی داده های فرم دارد که اگر قبلاً با اصول اولیه فرم آشنا نیستید، ارزش بررسی دارد.

بیایید اکنون یک فرم جستجوی اولیه را به صفحه اصلی فعلی اضافه کنیم. به راحتی میتوان آن را در یک navbar یا یک صفحه جستجوی اختصاصی در آینده قرار داد.

ما با `<form>` تگ های شروع می کنیم و از استایل bootstrap استفاده می کنیم تا زیبا به نظر برسد. اکشن (action) مشخص می کند که کاربر پس از ارسال فرم به کجا هدایت شود، که صفحه نتایج جستجو خواهد بود. مثل همه ی لینک URL ها اسم URL این صفحه هم این است. سپس متدهای مورد نظر را مشخص می کنیم که get به جای post باشد.

قسمت دوم فرم ورودی است که شامل عبارت جستجوی کاربر است. ما آن را با متغیر `name` می کنیم، `q`، که بعدا در url قابل مشاهده و در فایل views در دسترس خواهد بود. استایل Placeholder یک `text` را مشخص کنید، یک `label` استرپ را با کلاس اضافه می کنیم، نوع ورودی `text` را مشخص کنید، یک `aria-label` اضافه کنید که متنی پیش فرض است که از کاربر درخواست می کند. آخرین قسمت `aria-label` نامی است که برای کاربران خواننده نمایش داده می شود. دسترسی، بخش بزرگی از توسعه وب است و همیشه باید از ابتدا مورد توجه قرار گیرد: از جمله وجود `aria-labels` در تمامی فرم های شما!

```
<!-- templates/home.html -->
{ % extends '_base.html' %}
{ % load static % }

{ % block title % }Home{ % endblock title % }

{ % block content % }
<h1>Homepage</h1>
<form class="form-inline mt-2 mt-md-0" action="{ % url 'search_results' % }"
method="get">
<input name="q" class="form-control mr-sm-2" type="text"
placeholder="Search"
aria-label="Search">
</form>
```

٪} endblock content {٪.

سعی کنید یک جستجو انجام دهید مثل . "hello" : با زدن Return به صفحه نتایج جستجو هدایت می شوید. توجه داشته باشید URL , به طور خاص حاوی کوئری (عبارت) جستجو است. با این حال نتایج تغییر نکرده است! و این به این دلیل است که ListView نتایج جستجوی شما همچنان مقادیر کدگذاری شده قبلی را دارد. آخرین مرحله این است که درخواست جستجوی کاربر را که در URL با q نشان داده شده است، و آن را به فیلتر های جستجوی واقعی ارسال کنید.

```
# books/views.py
class SearchResultsListView(ListView):
    model = Book
    context_object_name = 'book_list'
    template_name = 'books/search_results.html'
    def get_queryset(self): # new
        query = self.request.GET.get('q')
        return Book.objects.filter(
            Q(title__icontains=query) | Q(author__icontains=query)
        )
```

چه چیزی تغییر کرد ؟ ما یک متغیر query اضافه کردیم که مقدار q را از ارسال فرم می گیرد. سپس فیلترمان را برای استفاده از query در فیلد title و author آپدیت میکنیم. خودشه! صفحه نتایج جستجو را رفرش کنید URL همچنان همان query را دارد و انتظار می رود: نتیجه ای برای "hello" در title یا author وجود نداشته باشد.

به صفحه اصلی برگردید و جستجوی جدیدی مانند "api" یا "beginners" را امتحان کنید تا عملکرد جستجوی کامل را در عمل ببینید.

جستجوی اولیه ما اکنون کامل شده است ، اما فقط یه بخش سطحی از حد مطلوب بالقوه ی جستجو را انجام داده ایم . به عنوان مثال، شاید بخواهیم دکمه ای به فرم جستجو اضافه شود که علاوه بر زدن کلید Return بتوان روی آن کلیک کرد؟ یا بهتر است اعتبار فرم را در نظر

بگیرید. اگر بخواهیم جستجویی با کیفیت گوگل را داشته باشیم، به غیر از فیلتر کردن با AND و OR، فاکتورهای دیگری نیز وجود دارد. چیزهایی مانند ارتباط و موارد دیگر.

چندین پکیج جانبی (*third-party packages*) با ویژگی های پیشرفته مانند django-watson یا django-haystack وجود دارد، با این حال با توجه به این که ما از PostgreSQL به عنوان دیتابیس استفاده می کنیم میتوانیم از full text search و سایر ویژگی هایی که در خود جنگو تعییه شده است استفاده کنیم.

گزینهنه نهایی یا استفاده از یک راه حل در سطح سازمانی مانند ElasticSearch است که باید روی یک سرور جداگانه اجرا شود (که با داکر سخت نیست)، یا به راه حل هاست مانند Algolia یا Swiftype تکیه کنید.

## ۸-۱۳- کارایی

اولویت اول برای هر وبسایتی به درستی اجرا شدن و همچنین دربرداشتن تست های درست و مناسب هست. اما اگر پروژه شما آنقدر خوشانس باشد که حجم زیادی از ترافیک دریافت کند، به سرعت تمرکز به سرعت به عملکرد و کارایی تغییر پیدا میکند و تا حدامکان کارآمدتر می کند. این یک سرگرمی و تمرین چالش برانگیر برای بسیاری از مهندسین است، اما میتواند یک تله نیز باشد.

در عین حال که توجه و نظارت برای بهینه کردن پروژه (کدهایتان) در آینده بسیار مهم است. از همان اول تمرکز زیادی روی آن نگذارید. هیچ راه دقیقی برای شبیه سازی محیط واقعی (پروداکشن) در محیط محلی (*environments locally*) وجود ندارد حتی اگر بخوایم تشخیص دهیم ترافیک یک سایت چگونه خواهد بود. اما ممکن است در مراحل اولیه زمان زیادی را صرف پیدا کردن راهی برای بهینه کردن عملکرد سیستم بجای صحبت با کاربران و بهبود کد های مهم تر کنید.

قبل از اینکه بتوانیم کوئری های دیتابیس خودمان را بهینه کنیم باید آنها را مشاهده کنیم. برای این منظور از پکیج پیشفرض *Third-party django-debug-toolbar* انجمن جنگو request/response میباشد. این ابزار دارای مجموعه ای از پنل ها برای بررسی چرخه کامل

در هر صفحه‌ای که طراحی شده می‌باشد. طبق معمول میتوانیم آن را در Docker نصب کنیم و container‌های در حال اجرا یمان را متوقف کنیم.

```
$ docker-compose exec web pipenv install django-debug-toolbar==2.2
```

```
$ docker-compose down
```

سه پیکربندی (configurations) جدا برای تنظیم شدن در فایل تنظیمات وجود دارد config/settings.py :

INSTALLED\_APPS .۱

Middleware .۲

INTERNAL\_IPS .۳

ابتدا INSTALLED\_APPS را به پیکربندی Debug Toolbar اضافه کنید. توجه داشته باشید که نام مناسب django\_debug\_toolbar است نه debug\_toolbar که انتظار می‌رود.

```
# config/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    # Third-party
    'crispy_forms',
    'allauth',
    'allauth.account',
    'debug_toolbar', # new
    # Local
```

```
'accounts',
'pages',
'books',
```

]

دم دوم، Debug Toolbar را به Middleware اضافه کنید که در درجه اول اجرا میشود.

```
# config/settings.py
```

```
MIDDLEWARE = [
```

```
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
'debug_toolbar.middleware.DebugToolbarMiddleware', # new
```

]

و قدم سوم، INTERNAL\_IPS را نیز تنظیم کنید. اگر ما در Docker نبودیم، می توانستیم آن را روی ۱۲۷.۰.۰.۱ تنظیم کنیم، اما از آنجا که ما سرور وب خود را به روی بستر Docker اجرا می کنیم، یک مرحله اضافی لازم است تا با آدرس دستگاه Docker مطابقت داشته باشد. خطوط زیر را در پایین config/settings.py اضافه کنید.

```
# config/settings.py
```

```
...
```

```
# django-debug-toolbar
```

```
import socket
```

```
hostname, _, ips = socket.gethostbyname_ex(socket.gethostname())
```

```
INTERNAL_IPS = [ip[:-1] + "1" for ip in ips[
```

آخرین مرحله به روز رسانی URLconf میباشد . ما فقط می خواهیم Debug Toolbar در صورت DEBUG=True درست ظاهر شود، بنابراین منطق را در این مورد در پایین فایل config(urls.py) اضافه می کنیم تا نمایش داده شود.

```
# config/urls.py
from django.conf import settings
from django.conf.urls.static import static from django.contrib import admin
from django.urls import path, include

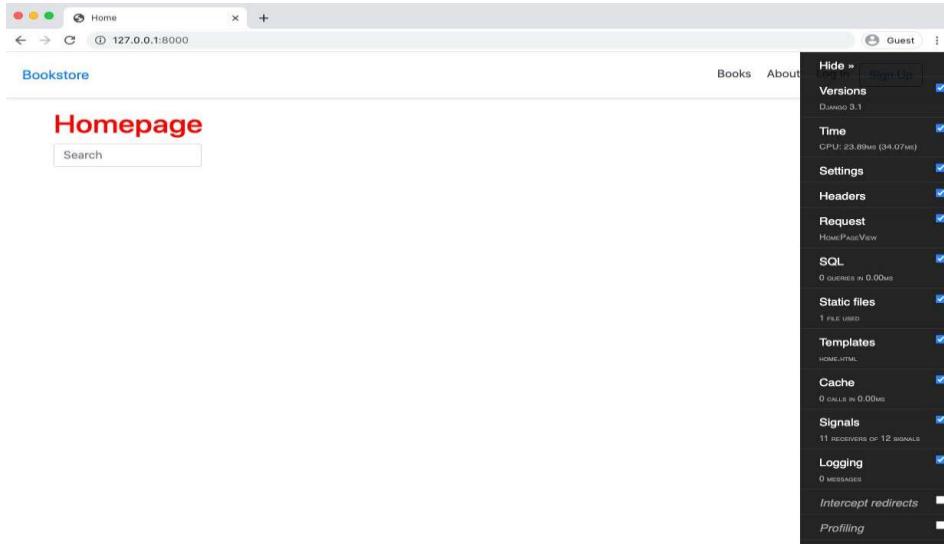
urlpatterns = [
    # Django admin
    path('admin/', admin.site.urls),
    # User management
    path('accounts/', include('allauth.urls')),
    # Local apps
    path("", include('pages.urls')),
    path('books/', include('books.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

if settings.DEBUG: # new
    import debug_toolbar
    urlpatterns = [
        path('__debug__/', include(debug_toolbar.urls)),
    ] + urlpatterns
```

صفحه را رفرش می کنیم

موارد زیادی برای سفارشی سازی دارد اما تنظیمات پیشفرض نیز موارد زیادی debug toolbar را درباره صفحه homepage به ما نشان میدهد . به عنوان مثال، می توانیم نسخه جنگو استفاده شده و همچنین زمان بارگذاری صفحه را مشاهده کنیم . همچنین درخواست مشخصی رو که

به نام HomePageView. ممکن است بدیهی باشد، اما در سیستم هایی با کدبیس بزرگ، بهویژه اگر به عنوان یک توسعه‌دهنده جدید وارد آن می‌شود، ممکن است مشخص نباشد که کدام view کدام صفحه را فراخوانی می‌کند Debug Toolbar. یک شروع سریع و مفید برای درک سایت های (مسیر ها و درخواست ها) موجود است.



اگر متوجه شدید که در یک سایت جنگو با تعداد کوئری SQL بسیار زیادی در هر صفحه کار می‌کنید، چه گزینه هایی وجود دارد؟ با این حال، به طور کلی، تعداد کوئری های سنگین (داده های زیادی را حمل می‌کنند - join های زیاد) سرعت کمتری نسبت به کوئری های سبکتر دارد، اگرچه آزمایش این مورد در عمل ممکن و ضروری است. دو تکنیک رایج برای انجام این کار استفاده از select\_related و prefetch\_related می‌باشد.

برای روابط تک مقداری از طریق یک رابطه رو به جلو یک به چند یا یک به یک استفاده می‌شود. این یک اتصال SQL ایجاد می‌کند و فیلد های شی مرتبط را در عبارت SELECT شامل می‌شود، که در نتیجه همه اشیاء مرتبط در یک کوئری پایگاه داده پیچیده تر گنجانده می‌شوند. این کوئری واحد معمولاً عملکرد بهتری نسبت به کوئری های متعدد و کوچکتر دارد.

برای مجموعه یا لیستی از اشیاء مانند رابطه چند یا چند به یک استفاده می‌شود. در بطن جستجو برای هر رابطه انجام می‌شود و "join" در پایتون اتفاق می

افتد، نه SQL. این به آن اجازه می دهد تا اشیاء چند به چند و چند به یک را از قبل واکشی کند، که با استفاده از `select_related` نمی توان انجام داد، علاوه بر کلید خارجی و روابط یک به یک که توسط `select_related` پشتیبانی می شوند. پیاده سازی یک یا هر دو در یک وب سایت اولین راه رایج برای کاهش درخواست ها و زمان بارگذاری برای یک صفحه مشخص است.

در نظر بگیرید که پروژه کتابفروشی ما یک وب سایت پویا است. هر بار که کاربر صفحه ای را درخواست می کند، سرور ما باید محاسبات مختلفی از جمله کوئری های پایگاه داده، رندر قالب و غیره را قبل از سرویس آن انجام دهد. این زمانبر بوده و بسیار کنتر از خواندن یک فایل از یک سایت ثابت است که در آن محتوا تغییر نمی کند.

با این حال، در سایتها بزرگ، این نوع سربار می تواند بسیار کند باشد و کش کردن یکی از اولین راه حل ها در کیف ابزار توسعه دهندهان وب است. اجرای حافظه پنهان (کش کردن) در پروژه فعلی ما قطعاً بیش از حد است، اما با این وجود گزینه ها را بررسی می کنیم و یک نسخه اولیه را پیاده سازی می کنیم.

کش یک حافظه ذخیره سازی `in-memory` جهت محاسبه قیمت هست. پس از اجرا، نیازی به اجرا مجدد نیست! دو گزینه محبوب Memcached که دارای پشتیبانی بومی جنگو است و Redis که معمولاً با بسته شخص ثالث django-redis پیاده سازی می شود.

جنگو چارچوب کش مخصوص به خود را دارد که شامل چهار گزینه مختلف ذخیره سازی به ترتیب نزولی از جزئیات است:

۱. کش per-site ساده ترین راه اندازی است و کل سایت شما را ذخیره می کند.
۲. حافظه پنهان برای (per-view) به شما امکان می دهد تا نماهای جداگانه را ذخیره کنید.
۳. ذخیره سازی قطعه قالب به شما امکان می دهد بخش خاصی از یک قالب را برای کش تعیین کنید.
۴. API کش سطح پایین به شما امکان می دهد اشیاء خاصی را در حافظه پنهان به صورت دستی تنظیم، بازیابی و نگهداری کنید.

چرا همه چیز را همیشه در حافظه پنهان نگه نمی‌داریم؟ یکی از دلایل این است که حافظه کش گران است، زیرا به عنوان RAM ذخیره می‌شود: به هزینه افزایش رم از ۸ گیگابایت به ۱۶ گیگابایت در لپ تاپ خود در مقابل فضای هارد ۲۵۶ گیگابایت به ۵۱۲ گیگابایت فکر کنید. یکی دیگر از موارد این است که حافظه پنهان باید "گرم" باشد، که با محتوای به روز شده پر شده است، بنابراین بسته به نیاز یک سایت، بهینه سازی حافظه پنهان به گونه‌ای که دقیق باشد، اما بیهوده نباشد، نیاز به تنظیم کمی دارد.

اگر می‌خواهید per-site کش را پیاده‌سازی کنید، که ساده‌ترین روش است، در MIDDLEWARE را در بالای پیکربندی UpdateCacheMiddleware را در پایین اضافه کنید. همچنین config/settings.py سه فیلد اضافی CACHE\_MIDDLEWARE\_ALIAS و CACHE\_MIDDLEWARE\_KEY\_- و CACHE\_MIDDLEWARE\_SECONDS را تنظیم کنید.

```
# config/settings.py

MIDDLEWARE = [
    'django.middleware.cache.UpdateCacheMiddleware', # new
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    'django.middleware.cache.FetchFromCacheMiddleware', # new
]

CACHE_MIDDLEWARE_ALIAS = 'default'
CACHE_MIDDLEWARE_SECONDS = 604800
```

CACHE\_MIDDLEWARE\_KEY\_PREFIX" =

تنها پیش‌فرضی که ممکن است بخواهد تنظیم کنید CACHE\_MIDDLEWARE\_SECONDS است که تعداد پیش‌فرض ثانیه‌ها (۶۰۰) برای کش کردن یک صفحه است. پس از اتمام دوره، حافظه پنهان منقضی می‌شود و خالی می‌شود.

یک پیش‌فرض خوب هنگام شروع، ۶۰ ثانیه یا ۱ هفته (۶۰۴۸۰۰ ثانیه در ۶۰ دقیقه در ۱۶۸ ساعت) برای سایتی با محتوایی است که اغلب تغییر نمی‌کند. اما اگر متوجه شدید که حافظه پنهان شما به سرعت پر می‌شود یا سایتی را اجرا می‌کنید که محتوای آن به طور مکرر تغییر می‌کند، اولین قدم خوب کوتاه کردن این تنظیمات است.

با این حال، پیاده‌سازی حافظه پنهان در این مرحله کاملاً اختیاری است. هنگامی که یک وب سایت راه اندازی و اجرا می‌شود، نیاز به caching\_per\_site، در هر صفحه به سرعت آشکار می‌شود. همچنین پیچیدگی بیشتری وجود دارد زیرا Memcache باید به عنوان یک نمونه جداگانه اجرا شود. در سرویس میزبانی Heroku، که در فصل ۱۸ برای استقرار از آن استفاده خواهیم کرد، یک ردیف رایگان از طریق Memcachier در دسترس است.

ایندکس کردن یک تکنیک رایج برای افزایش سرعت عملکرد پایگاه داده است. این یک ساختار داده جداگانه است که امکان جستجوی سریع‌تر را فراهم می‌کند و معمولاً فقط برای کلید اصلی در یک مدل اعمال می‌شود. نکته منفی این است که ایندکس‌ها به فضای بیشتری روی دیسک نیاز دارند، بنابراین باید با احتیاط از آنها استفاده کرد.

همانطور که وسوسه انگیز است که به سادگی فهرست‌ها را از ابتدا به کلیدهای اصلی اضافه کنید، بهتر است بدون آنها شروع کنید و فقط بعداً بر اساس نیازهای تولید آنها را اضافه کنید. یک قانون کلی این است که اگر یک فیلد معین به طور مکرر مورد استفاده قرار می‌گیرد، مانند ۱۰-۲۵٪ از تمام پرس و جوهای کاندیدای اصلی برای ایندکس شدن است.

از لحاظ تجربی می‌توان با افزودن db\_index=True به هر فیلد مدل، یک فیلد ایندکس ایجاد کرد. به عنوان مثال، اگر بخواهیم یکی را به فیلد idر مدل Book خود اضافه کنیم، به شکل زیر خواهد بود (هر چند در واقع این را اجرا نکنید)

```
# books/models.py
```

```
...
```

```
class Book(models.Model):
    id = models.UUIDField(
        primary_key=True,
        db_index=True, # new
        default=uuid.uuid4,
        editable=False)
```

...

این تغییر باید از طریق یک فایل migration به پایگاه داده اضافه و منتقل شود . با شروع جنگو ۱.۱۱، ایندکس های مدل مبتنی بر کلاس اضافه شدند، بنابراین می توان آن را در بخش متاکلاس قرار داد . همچنین می توانید به جای آن شاخص قبلی را به صورت زیر بنویسید:

```
# books/models.py
...
class Book(models.Model):
    id = models.UUIDField(
        primary_key=True,
        default=uuid.uuid4,
        editable=False)
...
class Meta:
    indexes = [ # new
        models.Index(fields=['id'], name='id_index'),
    ]
    permissions = [
        ("special_status", "Can read all books"),
    ]
```

یکی دیگر از بسته های شخص ثالث بسیار محبوب برای بازرگانی پروژه جنگو، django-extensions است که تعدادی سفارشی مفید را اضافه می کند .

یکی از مواردی که بسیار مفید است shell plus است که همه مدل‌ها را به طور خودکار در پوسته بارگذاری می‌کند که کار با ORM جنگو را بسیار آسان تر می‌کند.

آخرین منبع اصلی تنگناها در یک وب سایت بارگذاری front-end assets است. CSS و Java اسکریپت می‌توانند حجم بسیار زیادی را اشغال کنند و بنابراین ابزارهایی مانند django-compressor می‌توانند به کاهش اندازه آنها کمک کنند.

تصاویر اغلب اولین جایی هستند که از نظر اندازه asset باید بهشون نگاه کرد. فایل استاتیک/رسانه‌ای که ما در اختیار داریم به در مقیاس بزرگ اندازه گیری می‌شود، اما برای سایت‌های واقعاً بزرگ، ارزش بررسی استفاده از یک شبکه تحویل محتوا (CDN) برای تصاویر به جای ذخیره آن‌ها در سیستم فایل سرور را دارد.

همچنین می‌توانید تصاویر با اندازه‌های مختلف را به کاربران ارائه دهید. برای مثال، به جای کوچک کردن یک جلد کتاب بزرگ برای یک فهرست یا صفحه جستجو، می‌توانید به جای آن یک نسخه کوچک‌تر را ذخیره کنید و در صورت نیاز آن را ارائه کنید. بسته به جای ذخیره آن‌ها در سیستم third-party thumbnails محل خوبی برای شروع این کار است.

یک کتاب الکترونیکی فوق العاده رایگان در این زمینه، Essential Image Optimization اثر Addy Osmani است که به عمق بهینه‌سازی تصویر و اتوماسیون می‌پردازد.

به عنوان آخرین بررسی، تست‌های خودکاری برای سرعت front\_end مانند Google's PageSpeed Insights وجود دارد که بر اساس سرعت بارگیری یک صفحه امتیازی را تعیین می‌کند.

لیست تقریباً بی پایانی از بهینه‌سازی عملکرد وجود دارد که می‌تواند در یک پروژه اعمال شود. اما مراقب باشید و توصیه حکیم دونالد کنوت را به خاطر بسپارید که زیاد دیوانه نشوید. تنگناها خود را در تولید نشان خواهند داد و باید تا حد زیادی در آن زمان برطرف شوند. نه از قبل باید به یاد داشته باشید که مشکلات عملکرد مشکل خوبی است! آنها قابل تعمیر هستند و به این معنی است که پروژه شما به شدت مورد استفاده قرار می‌گیرد.

## ۱۴-۸- امنیت

دنیای اینترنت، دنیای خطرناکی هست. در آن تعداد زیادی افراد بد حتی بیشتر از آنها ربات های هستند که بصورت خودکار تلاش میکنند و بسایت شما را هک و ایجاد مشکل کنند. از این رو فهمیدن و پیاده سازی فیچر های امنیتی در هر وبسایتی الزامی است.

خوبشخانه، جنگو به دلیل سال ها تجربه در رسیدگی به مسایل مربوط به امنیت وب و همچنین یک چرخه بروز رسانی امنیتی قوی از سابقه بسیار خوبی بر خوردار می باشد.

فیچر های جدید تقریبا هر ۹ ماه یک بار مثل ورژن ۲,۰ تا ۳,۰ منتشر می شوند اما وصله های امنیتی تقریبا هر ماه بصورت ۲,۰,۲ تا ۲,۰,۳ منتشر می شوند.

به هر حال، مانند هر ابزاری، این مهم است که فیچر های امنیتی بطور صحیح پیاده سازی شوند و ما در این قسمت آموزش می دهیم که چطور این موارد امنیتی را در پروژه کتاب فروشی (Book Store) پیاده سازی کنید.

### مهندسی اجتماعی (Social Engineering)

در نهایت بزرگترین ریسک برای هر وبسایتی مشکلات فنی نیست، مشکلات انسانی است! اصطلاح مهندسی اجتماعی به مجموعه روش های پیدا کردن افراد با دسترسی به یک سیستم اشاره می کند که خواسته یا ناخواسته اطلاعات اهراز هویت خود را در اختیار فرد حمله کنند قرار میدهد (مترجم: فرد حمله کننده با اطلاعات فرد مورد اطمینان وارد سیستم می شود)

این روز ها بزرگترین مقصو، فیشنگ می باشد. همه این ها از یک کلیک بر روی لینک مخرب که در یک ایمیل قرار دارد شروع می شود که باعث می شود که دسترسی کارمند را به فرد هکر بدهد یا لااقل دسترسی اش را به خطر بیندازد. برای جلوگیری از این اتفاق (فیشنگ)، باید دسترسی های مختلف ایجاد کنید و به افراد فقط دسترسی هایی که به آن لازم دارن داده شود و نه بیشتر! آیا همه مهندسین احتیاج دسترسی بع دیتابیس محصول را دارند؟ احتمالا نه. آیا سایر کارمندان احتیاج به دسترسی نوشتن(در دیتابیس) را نیاز دارند؟ دوباره، احتمالا نه. بهترین کار این است که بطور پیشفرض فقط دسترسی های مورد نیاز داده شود، نه اینکه پیشفرض همه را superuser درنظر بگیریم!

## بروزرسانی های جنگو (Django updates)

یکی از راه های مهم امن نگهداشتن پروژه شما، بروز نگهداشتن ورژن به آخرین ورژن جنگو می باشد. این به این معنی نیست که فقط به ورژن های آخرین بروزرسانی (latest feature release) مانند ورژن های ۲,۲,۰، ۳,۰، ۳,۱... یا... بروزرسانی کنید که حدودا هر ۹ ماه منتشر می شود. همچنانی بروزرسانی های امنیتی هر ماه منتشر می شود که به صورت ۱,۲,۲,۲، ۲,۲,۳... یا... منتشر می شوند. (متوجه به این معنا می باشد که هر ماه باید بروز رسانی های امنیتی را دریافت کنید)

نسخه های feature (releases) یا به اختصار LTS چه هستند؟ بعضی از releases نامیده می شوند که برای مدت مشخصی (حدودا ۳ سال) اطمینان داده می شود که بروزرسانی های امنیتی و data fixes را دریافت می کند. برای مثال ورژن ۲,۲ LTS هست که تا سال ۲۰۲۲ بروزرسانی می شوند، یعنی تا زمانی که نسخه ۴,۰ جنگو بعنوان ورژن بعد LTS انتخاب می شود. آیا می شود روی نسخه LTS ماند؟ بله. آیا مجبورید؟ نه. برای امن بودن بهتر هست همیشه بروز باشیم.

مقاومت در مقابل وسوسه (بروزرسانی) و واقعیت تعداد زیادی از پروژها این نیست که وقت توسعه را به رفع اشکالات بعد از بروزرسانی جنگو بگذارند. یک وبسایت همانند یک خودرو: نیاز به نگهداری منظم برای کار کردن در بهترین حالت خود را دارد. اگر از خیر بروز رسانی ها بگردد، تنها مشکلات را باهم ترکیب می کنید.

چگونه بروزرسانی کنیم؟ امکان depreciation warning جنگو، باید قبل از هر ریلیز با تایپ دستور python -Wa manage.py test اجرا شود. بهتر است برای بروزرسانی نسخه به نسخه اقدام شود (مثلا از ۲,۰ به ۲,۱ و الی آخر) و هر دفعه depreciation warning را اجرا کرد تا اینکه یک دفعه به نسخه آخر رفت. (چندین نسخه بین را بدون اجرا دستور گفته شده رد کردند).

```
$ docker-compose exec web python manage.py check --deploy
System check identified some issues:
```

**WARNINGS:**

...

.(System check identified 5 issues (0 silenced

درنهایت، تنظیمات ما برای محیط توسعه (development) و محصول (production) متفاوت هست. ما در حال حاضر، تنظیمات را در فصل ۸ انجام دادیم. اگر یادتون باشه متغیرهای محلی (environment variables) را برای DEBUG و پایگاه داده اضافه کردیم. اما مقادیری برای production تنظیم نکردیم یا راهی موثر برای تغییر بین محیط توسعه و محصول ارائه نکردیم.

راه های زیادی برای برخورد با این چالش وجود دارد. با توجه به اینکه قرار است در Heroku استقرار دهیم، رویکرد ما ایجاد فایل docker-compose-prod.yml است که به واسطه آن میتوانیم محیط محصول را تست کنیم و همچنین میتوانیم بصورت دستی متغیرهای محلی به محیط محصول اضافه کنیم.

بطور پیشفرض، گیت تمامی فایل ها و پوشه ها را در پروژه ما دنبال میکند. ما نمیخواهیم این فایل جدید را دنبال کند، چون حاوی اطلاعات مهم و حساسی هست. راه حل آن ایجاد فایل ignore گذشت که درون آن باید نام فایل ها و پوشه هایی که مایلیم توسط گیت دنبال نشود قرار دهیم.

یک فایل جدید ایجاد میکنیم.

```
$touch.gitignore
docker-compose-prod.yml
__pycache__/
db.sqlite3
.DS_Store # Mac only
```

حالا، محتویات فایل docker-compose-prod.yml را در docker-compose.yml کپی میکنیم.

```
version: '3.8'
```

```
services:
```

```
web:
```

```

build:
  command: python /code/manage.py runserver 0.0.0.0:8000
  volumes:
    - .:/code
  ports:
    - 8000:8000
  depends_on:
    - db
  environment:
    - "DJANGO_SECRET_KEY=)*_s#exg*#w+#+
      xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8*n"
    - "DJANGO_DEBUG=True"
db:
  image: postgres:11
  volumes:
    - postgres_data:/var/lib/postgresql/data/
  environment:
    - "POSTGRES_HOST_AUTH_METHOD=trust"
volumes:
  postgres_data:

```

برای اجرای فایل جدید، داکر را با فلگ `-f` ریستارت کنید تا فایل کامپوز جایگزین انتخاب شود.

```

$ docker-compose down
$ docker-compose -f docker-compose-prod.yml up -d --build
$ docker-compose exec web python manage.py migrate

```

فلگ `--build` برای بیلد اولیه ایمیج به همراه تمامی پکیج های نرم افزاری برای فایل کامپوز جدید اضافه شده است. همچنین دستور `migrate` دیتابیس جدید را اجرا میکند. این یک نمونه کاملاً جدید از پروژه ما است! به این ترتیب، نه اکانت ابر کاربر (super user) وجود دارد.

نه اطلاعات کتاب های ما. اما برای الان اشکالی ندارد، این اطلاعات بعدا می توانیم روی محیط production اضافه کنیم و تمام تمکزمان را روی ایجاد محیط production محلی برای تست محصول میگذاریم.

در این بخش هدف ما این هست که با استفاده از docker-compose-prod.yml تمام چک لیست های استقرار (deployment) جنگو را پاس کنیم. باید با تغییر DEBUG شروع کنیم که تنظیم شده است به True در صورتی که در production باید False باشد.

environment:

```
- "DJANGO_SECRET_KEY=)*_s#exg*#w+#+xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8*n"
```

```
" -DJANGO_DEBUG=False" # new
```

ما از متغیرهای محلی (Enviroment Variables) به دو دلیل استفاده میکنیم: ذخیره مقادیری مانند SECRET\_KEY درواقع امن هستند و یک راه برای سویچ کردن تنظیمات از production به local هستند. مشکلی پیش نمیاد اگر دو تا متغیرهای محلی برای تنظیمات مانند DEBUG داشته باشیم، شاید بهتر باشه مقدار پیشفرض برای آن قرار بدیم زمانی که لازم نیست چیزی را مخفی (secret) نگهداریم.

```
# config/settings.py
(DEBUG = env.bool("DJANGO_DEBUG", default=False)
```

به این معنا می باشد که در محیط پروداکشن اگر متغیر محلی نداشته باشیم، مقدار آن بطور پیشفرض برابر False می باشد. ما باید متغیر DJANGO\_DEBUG را در فایل محلی docker-compose.yml حذف کنیم. نتایج docker-compose-prod.yml این رویکرد باعث می شود که فایل docker-compose-prod.yml کوچکتر شود و همچنین شاید باعث امنیت بیشتر بشود اگر به دلایلی متغیرهای محلی به درستی لود نشوند، ما اشتباه تنظیمات محلی development را فعال نمی کنیم.

بریم با حذف کردن DJANGO\_DEBUG از فایل docker-compose.prod.yml را بروزرسانی کنیم.

environment:

```
" DJANGO_SECRET_KEY=)*_s#exg*#w+#
xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8*n
"
```

در حال حاضر SECRET\_KEY ما در فایل docker-compose.yml قابل رویت می باشد. برای امنیت بیشتر، ما باید کلید جدید ایجاد کنیم و توسط docker-compose-prod.yml تست کنیم SECRET\_KEY. یک رشته تصادفی ۵۰ کارکتری است که هر دفعه دستور startproject را اجرا می کنید، ساخته می شود. برای ساخت کلید جدید میتوانید از کتابخانه داخلی پایتون یعنی مazzo secret استفاده کنید.

```
$ docker-compose exec web python -c 'import secrets;
print(secrets.token_urlsafe(38))'
```

```
ldBHq0YGYxBzaMJnLVOiNG7hruE8WKzGG2zGpYxoTNmphB0mdBo
```

پارامتر token\_urlsafe تعداد بایت های در یک رشته a URL-safe را بازمیگرداند. با انکدین Base64 بطور میانگین هر بایت برابر ۱,۳ کارکتر می باشد. در این مثال با استفاده از ۳۸ ما ۵۱ کارکتر داریم. مهمترین نکته ای که باید به آن توجه کنید، مقدار SECRET\_KEY حداقل باید ۵۰ کارکتر باشد. هر دفعه که این دستور را اجرا کنید، مقدار جدید دریافت می کنید.

به روش زیر می توانید SECRET\_KEY جدید را به docker-compose-prod.yml اضافه کنید:

```
# docker-compose-prod.yml
```

```
environment:
```

```
-
```

```
"DJANGO_SECRET_KEY=ldBHq0YGYxBzaMJnLVOiNG7hruE8WKzGG2z
GpYxoTNmphB0mdB"
```

حال کانتینر داکری خود را ریستارت کنید، به درستی SECRET\_KEY جدید جایگزین شد.

```
$ docker-compose down
```

```
$ docker-compose -f docker-compose-prod.yml up -d --build
```

حتی اگر جنگو به طور پیش فرض با رایج ترین مسائل امنیتی برخورد کند، هنوز درک آن حیاتی است که روش ها و حمله های مکرر وجود دارد که جنگو تلاش می کند تا آنها را

کاهش دهد. شما می توانید یک نمای کلی در صفحه امنیتی جنگو بیابید، اما در اینجا به عمق بیشتری خواهیم پرداخت.

جنگو به طور پیش فرض با تعدادی میان افزار امنیتی اضافی همراه است که نگهبان در برابر سایر حملات چرخه درخواست/پاسخ (request/response) است.

بیا با حمله تزریق SQL شروع کنیم که زمانی اتفاق می افتد که یک کاربر مخرب بتواند به صورت دلخواه در دیتابیس ما کد SQL اجرا کند. یک فرم ورود به سایت را در نظر بگیرید. چه اتفاقی می افتد اگر یک کاربر مخرب بتواند DELETE from users WHERE user\_id=user\_id را تایپ کند؟

اگر این کد در مقابل پایگاه داده بدون حفاظت های مناسب اجرا شود، می تواند منجر به حذف تمام رکوردهای کاربر شود! خوب نیست. این XKCD comic یک تفکر طنز آمیز به طور بالقوه مثال دقیقی از چگونگی وقوع این امر ارائه می دهد.

خوبی بختانه Django ORM به طور پیش فرض ورودی های کاربر را هنگام ساخت querysets برای جلوگیری از این نوع حملات پاکسازی می کند. زمانی که باید مراقب باشید این است که جنگو گزینه ای برای اجرای sql سفارشی یا پرس و جوهای خام ارائه می دهد. هر دو باید با احتیاط مورد استفاده قرار گیرند زیرا میتوانند آسیب پذیری را در برابر تزریق SQL باز کنند.

تزریق اسکریپت از طریق سایت (XSS) این یک حمله کلاسیک دیگر است که زمانی اتفاق می افتد که مهاجم (attacker) قادر است تکه های کوچکی از کد را به صفحات مشاهده شده توسط افراد دیگر تزریق کند. این کد، معمولاً به زبان جاوا اسکریپت است و در صورت ذخیره در پایگاه داده، بازیابی شده و برای سایر کاربران نمایش داده می شود.

برای مثال: فرمی که برای نوشتن بررسی (نقد) کتاب استفاده می شود را در نظر بگیرید. چه می شود اگر به جای تایپ کردن ، "این کتاب عالی بود "کاربر یک چیزی را با جاوا اسکریپت تایپ کند ؟ برای مثال <script>alert('hello');</script> . اگر این اسکریپت در دیتابیس ذخیره میشد صفحه هر کاربری یک alert با عنوان hello به وجود می آمد. در حالی که این مثال خواص بیشتر آزار دهنده است تا خطرناک ، سایتی که در مقابل حمله XSS آسیب پذیر است بسیار خطرناک است زیرا هر کاربر مخرب میتواند هر جاوا اسکریپتی را در صفحه قرار دهد ، از جمله کد جاوا اسکریپتی که میتواند اطلاعات کاربران ناشناس را هم بدزد.

برای جلوگیری از حمله XSS قالب های جنگو به صورت خودکار automatically escape کاراکتر های خاصی که به طور بالقوه خطرناک هستند از جمله : براکت ها < > ، ( <سینگل کوت ، ' ) دابل کوت " (double quotes) و علامت &دوری می کند. مواردی وجود دارد که ممکن است بخواهد به صورت خودکار autoescape off کنید اما باید با احتیاط زیاد انجام شود.

این Cross-Site Request Forgery (CSRF) سومین نوع عمده حمله است اما عموما نسبت به تزریق (SQL Injection) XSS کمتر شناخته شده است. این حمله اساسا از اعتماد سایت به مرورگر کاربر استفاده می کند.

زمانی که کاربر وارد یک سایت می شود ، اجازه دهید آن را یک سایت بانکی برای اهداف تصاویر سازی بنامیم ، سرور session token را برای آن کاربر ارسال می کند. هدر های HTTP در آینده شامل همه request ها و احراز هویت های کاربر می شود. اما اگر یک بازیگر مخرب (هکر) (به نحوی به session token پیدا کند چه اتفاقی می افتد؟

برای مثال ، کاربری را در نظر بگیرید که در یک تب مرورگر به بانک خود وارد می شود. سپس آن در یک تب دیگر ایمیلش را باز می کند و روی یک link email که از طرف یک بازیگر مخرب (هکر) است کلیک می کند. این لینک قانونی به نظر می رسد ، اما در واقع به بانک کاربر اشاره می کند که هنوز در آن login است ! بنابراین به جای ترک کردن کامنت و بلاگ در این سایت جعلی ، در پشت صحنه ، از اطلاعات کاربری برای انتقال پول از حساب آن به حساب هکر ها استفاده می شود.

در عمل روش های زیادی برای به دست آوردن اعتبار کاربر از طریق حمله CSRF وجود دارد ، نه فقط لینک ها ، حتی form های پنهان ، برچسب های مخصوص عکس ، و حتی AJAX های .

CSRF protection جنگو با قرار دادن secret key رندوم به عنوان کوکی از طریق Middleware و در فرم ها از طریق تمپلیت تگ csrf token (template tag) محافظت می کند. سایت تیم سوم به کوکی های کاربر دسترسی نخواهد داشت بنابراین هرگونه مغایرت بین دو کلید باعث ارور می شود.

جنگو امکان سفارشی سازی را مثل همیشه می دهد : شما میتوانید CSRF middleware را غیر فعال کنید و از تمپلیت تگ `(template tag) csrf_protect` در view های خاص استفاده بکنید. با این حال این مرحله را با نهایت احتیاط انجام دهید.

این OWASP [CSRF Cheat Sheet](#) نگاه جامعی به موضوع ارائه می دهد. تغريبا همه می سایت های اصلی در مقاطعی از زمان قربانی حملات CSRF شده اند.

کلیک دزدی حمله دیگری است که در آن یک سایت مخرب کاربر را فریب می دهد تا روی یک frame مخفی شده کلیک کند. یک frame داخلی، که به عنوان iframe شناخته می شود ، معمولا برای جاسازی یک سایت در داخل یکی دیگر استفاده می شود. برای مثال ، اگر میخواهید Google Map یا ویدیوی YouTube را در سایت خود قرار دهید `iframe` tag شامل است که آن سایت را در سایت شما قرار می دهد. این بسیار راحت است.

اما این یک خطر امنیتی دارد که می تواند یک frame را از دید کاربر مخفی کند. در نظر بگيريد که یک کاربر قبلا به حساب کاربری اش در آمازون login شده است و سپس از یک سایت مخرب بازدید می کند که به نظر می رسید عکس بچه گربه باشد. کاربر برای مشاهده بچه گربه های بیشتر بر روی سایت مخرب گفته شده کلیک می کند ، اما در واقع آنها روی یک `iframe` از محصول آمازون که ناخودآگاه خریداری شده کلیک می کنند. این تنها یک نمونه از clickjacking است.

برای جلوگیری از این (حمله) جنگو با یک پیش فرض همراه است clickjacking که HTTP header X-FrameOptions middleware که load یک frame یا iframe را تنظیم می کند ، که نشان می دهد که یک منبع مجاز به load یک frame یا iframe یا load است یا خیر. در صورت تمایل می توانید این محافظ را خاموش کنید ، یا آن را در سطح نمایش تنظیم کنید. با این حال این کار را با احتیاط و تحقيق بالا (انجام دهید).

تمام سایت های مدرن باید از HTTPS استفاده کنند ، که ارتباط رمز گذاری را بین مشتری و سرور فراهم میکند HTTP (Hypertext Transfer Protocol). ستون فقرات وب مدرن است ، اما به طور پیش فرض رمز گذاری ندارد.

حرف "s" در HTTPS به ماهیت رمز گذاری شده آن ابتدا به دلیل SSL (Secure Sockets Layer) اشاره می کند و این روزها آن جانشین TLS (Transport Layer Security) هستند.

با فعال بودن , HTTPS که ما خود در فصل دیپلولیمنت (deployment) انجام خواهیم داد ، بازیگران مخرب نمی توانند ترافیک ورودی و خروجی را برای داده هایی مانند اعتبار احراز هویت (authentication credentials) یا کلید API تشخیص دهند (بو بکشند.).

یکی از ۴ مسائل باقی مانده در چک لیست دیپلولیمنت جنگو ما این است که SECURE\_SSL\_REDIRECT در حال حاضر روی False تنظیم شده است. به دلایل امنیتی ، خیلی بهتر است که این را مجبور کنیم در حالت پروداکشن True (production) باشد . بباید آن را با پیش فرض پیکربندی به True و افزودن مقدار توسعه محلی به فایل docker-compose.yml تغییر دهیم.

Clickjacking حمله دیگری است که در آن یک سایت مخرب کاربر را فریب می دهد تا روی یک frame مخفی شده کلیک کند. یک frame داخلی، که به عنوان iframe شناخته می شود ، معمولا برای جاسازی یک سایت در داخل یکی دیگر استفاده می شود. برای مثال ، اگر میخواهید آن را با پیش فرض پیکربندی به True و افزودن مقدار قرار دهید iframe شامل tag YouTube یا ویدیوی Google Map را در سایت خود قرار دهید است که آن سایت را در سایت شما قرار می دهد. این بسیار راحت است.

اما این یک خطر امنیتی دارد که می تواند یک frame را از دید کاربر مخفی کند. در نظر بگیرید که یک کاربر قبلا به حساب کاربری اش در آمازون login شده است و سپس از یک سایت مخرب بازدید می کند که به نظر می رسید عکس بچه گربه باشد. کاربر برای مشاهده بچه گربه های بیشتر بر روی سایت مخرب گفته شده کلیک می کند ، اما در واقع آنها روی یک iframe از محصول آمازون که ناخودآگاه خریداری شده کلیک می کنند. این تنها یک نمونه از clickjacking است.

برای جلوگیری از این (حمله) جنگو با یک پیش فرض همراه است clickjacking که X-FrameOptions HTTP header middleware را تنظیم می کند ، که نشان می دهد که یک منبع مجاز به load یک frame یا iframe است یا خیر. در صورت تمایل می توانید این محافظت را خاموش کنید ، یا آن را در سطح نمایش تنظیم کنید. با این حال این کار را با احتیاط و research تحقیق بالا (انجام دهید).

مام سایت های مدرن باید از HTTPS استفاده کنند ، که ارتباط رمز گذاری را بین مشتری و سرور فراهم میکند HTTP (Hypertext Transfer Protocol) ستون فقرات وب مدرن است ، اما به طور پیش فرض رمزگذاری ندارد

"s" در HTTPS به ماهیت رمزگذاری شده آن ابتدا به دلیل (SSL (Secure Sockets Layer) اشاره می کند و این روزها آن جانشین (TLS (Transport Layer Security) هستند. با فعال بودن ، HTTPS که ما خود در فصل دیپلویمنت (deployment) انجام خواهیم داد ، بازیگران مخرب نمی توانند ترافیک ورودی و خروجی را برای داده هایی مانند اعتبار احراز هویت (authentication credentials) یا کلید API تشخیص دهند (بو بکشنده).

یکی از ۴ مسائل باقی مانده در چک لیست دیپلویمنت جنگو ما این است که در حال حاضر روی SECURE\_SSL\_REDIRECT تنظیم شده است. به دلیل امنیتی خیلی بهتر است که این را مجبور کنیم در حالت پروداکشن True (production) باشد. باید آن را با پیش فرض پیکربندی به True و افزودن مقدار توسعه محلی به فایل docker-compose.yml تغییر دهیم.

```
# config/settings.py
SECURE_SSL_REDIRECT =
env.bool("DJANGO_SECURE_SSL_REDIRECT", default=True)
```

```
$ docker-compose down
$ docker-compose -f docker-compose-prod.yml up -d --build
$ docker-compose exec web python manage.py check --deploy
```

انتقال اکیدا ایمن (HSTS) یک سیاست امنیتی است که به سرور ما اجازه می دهد که بر اینکه مرورگر ها باید از طریق HTTPS ارتباط داشته باشند به اضافه کردن Strict-Transport-Security Security header ،

سه پیکربندی ضمنی HSTS در فایل settings.py ما وجود دارد که باید برای حالت production به روز شود:

- SECURE\_HSTS\_SECONDS = 0

- SECURE\_HSTS\_INCLUDE\_SUBDOMAINS = False
- SECURE\_HSTS\_PRELOAD = False

متغير SECURE\_HSTS\_SECONDS در setting به طور پيش فرض روی ۰ تنظيم شده است اما برای اهداف امنیتی هرچه بیشتر بهتر. ما آن را برای یک ماه ، ۲،۵۹۲،۰۰۰ ثانیه ، در پروژه مان تنظیم می کنیم.

متغير SECURE\_HSTS\_INCLUDE\_SUBDOMAINS زیر دامنه هارا مجبور می کند تا به طور انحصاری از SSL استفاده کنند ، بنابراین ما آن را در production روی True تنظیم می کنیم.

متغير SECURE\_HSTS\_PRELOAD فقط زمانی تاثیر می گذارد که یک مقدار غیر ۰ برای وجود داشته باشد ، اما از آنجا که ما فقط یکی را تنظیم کرده ایم، باید این را روی True تنظیم کنیم.

</div>

ما به ۳ موضوع می پردازیم.

<div dir="ltr">

### ### HTTP Strict Transport Security (HSTS)

</div>

یک سیاست امنیتی است که به سرور ما اجازه HTTP Strict Transport Security (HSTS) می دهد که بر اینکه مرورگر ها باید از طریق HTTPS ارتباط داشته باشند با اضافه کردن Strict-Transport-Security header ، تأکید می کند.

سه پیکربندی ضمنی HSTS در فایل settings.py ما وجود دارد که باید برای حالت production به روز شود:

<div dir="ltr">

- `SECURE\_HSTS\_SECONDS = 0`
- `SECURE\_HSTS\_INCLUDE\_SUBDOMAINS = False`
- `SECURE\_HSTS\_PRELOAD = False`

</div>

در setting به طور پیش فرض روی . تنظیم شده است اما هرچه بیشتر بهتر برای اهداف امنیتی. ما آن را برای یک ماه ، ۲،۵۹۲،۰۰۰ ثانیه ، در پروژه مان تنظیم می کنیم.

SECURE\_HSTS\_INCLUDE\_SUBDOMAINS زیر دامنه هارا مجبور می کند تا به طور انحصاری از SSL استفاده کند ، بنابراین ما آن را در production روی True تنظیم می کنیم. SECURE\_HSTS\_PRELOAD فقط زمانی تاثیر می گذارد که یک مقدار غیر . برای وجود داشته باشد ، اما از آنجا که ما فقط یکی را تنظیم کرده ایم، باید این را روی True تنظیم کنیم.

در اینجا باید ببینید که تنظیمات به روز شده چگونه باید باشد.

<div dir="ltr">

```
```python
# config/settings.py

SECURE_HSTS_SECONDS=
env.int("DJANGO_SECURE_HSTS_SECONDS", default=2592000)

SECURE_HSTS_INCLUDE_SUBDOMAINS=
env.bool("DJANGO_SECURE_HSTS_INCLUDE_SUBDOMAINS",
default=True)

SECURE_HSTS_PRELOAD=
env.bool("DJANGO_SECURE_HSTS_PRELOAD", default=True)
```

سپس فایل docker-compose.yml با مقادیر توسعه محلی به روز کنید.

# docker-compose.yml

environment:

```
- "DJANGO_SECRET_KEY=)*_s#exg*#w+#
xt=vu8b010% %a&p@4edwyj0=(nqq90b9a8*n"
- "DJANGO_DEBUG=True"
- "DJANGO_SECURE_SSL_REDIRECT=False"
- "DJANGO_SECURE_HSTS_SECONDS=0" # new
- "DJANGO_SECURE_HSTS_INCLUDE_SUBDOMAINS=False" # new
" - "DJANGO_SECURE_HSTS_PRELOAD=False" # new
```

دامر را ری استارت کرده و دوباره چک لیست دیپلومینت را اجرا کنید.

```
$ docker-compose down
```

```
$ docker-compose -f docker-compose-prod.yml up -d --build
```

```
$ docker-compose exec web python manage.py check --deploy
```

کوکی HTTP برای ذخیره اطلاعات در کامپیوتر مشتری استفاده می شود مانند احراز هویت اعتبارنامه (authentication credentials). این امر ضروری است زیرا پروتکل HTTP از نظر طراحی بدون حالت است : راهی وجود ندارد برای اینکه آیا کاربر به غیر از شناسه در HTTP احراز هویت شده است یا نه !Header

جنگو نیز مانند اکثر وب سایت ها از cookies و sessions برای این کار استفاده می کند. اما (cookies کوکی ها) می توانند و باید باشند از طریق کانفیگ کردن SESSION\_COOKIE\_SECURE به HTTPS وادر می شوند. تنظیمات جنگو به طور پیش فرض False است بنابراین باید آن را در production به True تغییر دهیم.

دومین مسئله CSRF\_COOKIE\_SECURE است ، که به طور پیش فرض روی False است اما در حالت production باید True باشد تا فقط کوکی هایی که با عنوان("secure" امن) مشخص شده اند با اتصال HTTPS ارسال شوند.

HTTP Cookie برای ذخیره اطلاعات در کامپیوتر مشتری استفاده می شود مانند احراز هویت اعتبارنامه (authentication credentials). این امر ضروری است زیرا پروتکل HTTP از نظر طراحی بدون حالت است : راهی وجود ندارد برای اینکه آیا کاربر به غیر از شناسه در HTTP احراز هویت شده است یا نه !Header

جنگو نیز مانند اکثر وب سایت ها از sessions برای این کار استفاده می کند. اما cookies (کوکی ها) می توانند و باید باشند از طریق کانفیگ کردن SESSION\_COOKIE\_SECURE به HTTPS وادر می شوند. تنظیمات جنگو به طور پیش فرض False است بنابراین باید آن را در production به True تغییر دهیم.

دومین مسئله CSRF\_COOKIE\_SECURE است، که به طور پیش فرض روی False است. اما در حالت production باید True باشد تا فقط کوکی هایی که با عنوان("secure" امن) مشخص شده اند با اتصال HTTPS ارسال شوند.

```
# config/settings.py
```

```
SESSION_COOKIE_SECURE=
env.bool("DJANGO_SESSION_COOKIE_SECURE", default=True)

CSRF_COOKIE_SECURE = env.bool("DJANGO_CSRF_COOKIE_SECURE",
default=True)
```

سپس فایل docker-compose.yml را آپدیت کنید.

```
# docker-compose.yml
```

environment:

```
- "DJANGO_SECRET_KEY=)*_s#exg*#w+-#
xt=vu8b010%a&p@4edwyj0=(nqq90b9a8*n"
- "DJANGO_DEBUG=True"
- "DJANGO_SECURE_SSL_REDIRECT=False"
- "DJANGO_SECURE_HSTS_SECONDS=0"
- "DJANGO_SECURE_HSTS_INCLUDE_SUBDOMAINS=False"
- "DJANGO_SECURE_HSTS_PRELOAD=False"
- "DJANGO_SESSION_COOKIE_SECURE=False" # new
" - DJANGO_CSRF_COOKIE_SECURE=False" # new
```

دامری استارت کرده و دوباره چک لیست دیپلوبیمنت را اجرا کنید.

```
$ docker-compose down
```

```
$ docker-compose -f docker-compose-prod.yml up -d --build
```

```
$ docker-compose exec web python manage.py check --deploy
```

System check identified no issues (0 silenced).

تا اینجا ممکن است به نظر برسد که توصیه عمومی امنیت این است که به پیش فرض های جنگو تکیه کنید ، از HTTPS استفاده کنید ، به فرم ها تک csrf\_token را اضافه کنید ، و یک ساختار permissions تنظیم کنید. همگی صحیح است. یک قدم دیگر که جنگو از طرف ما قبول نمی کند ارتقا امنیت ادمین جنگو است.

در نظر بگیرید که هر سایت جنگو admin را به طور پیش فرض به آدرس /admin تنظیم می کند. هر هکری که تلاش می کند به پنل ادمین جنگو دسترسی یابد ابتدا به آن (/admin) شک می کند. بنابراین ، یک قدم آسان این است که به سادگی URL آدرس admin (را به معنای واقعی کلمه تغییر دهید ! ادیتور را باز کنید و مسیر URL را تغییر دهید. در این مثال چنین است/anything-but-admin .

```
# config/urls.py

from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    # Django admin
    path('anything-but-admin/', admin.site.urls), # new

    # User management
    path('accounts/', include('allauth.urls')),

    # Local apps
    path("", include('pages.urls')),
    path('books/', include('books.urls')),
```

]

```
if settings.DEBUG:
    import debug_toolbar
    urlpatterns = [
        path('__debug__/', include(debug_toolbar.urls)),
    ] + urlpatterns
```

یک پکیج ثالث سرگرم کننده django-admin-honeypot است که یک صفحه ورود admin جعلی در صفحه ایجاد می کند و IP هر کسی را که تلاش می کند به سایت شما در این آدرس (/admin) حمله کند به ادمین سایت ایمیل می زند. این آدرس های IP سپس می تواند به لیست آدرس مسدود شده سایت اضافه شود.

همچنین از طریق django-two-factor-auth توانید احراز هویت دوگانه را برای admin خود برای یک لایه محافظتی بیشتر اضافه کنید.

نگرانی اصلی هر سایت امنیت است. با استفاده از یک docker-compose-prod.yml میتوانیم قبل از دیپلوی مستقیم سایت ، در داکر تنظیمات production ما را به طور دقیق آزمایش کنید. و با استفاده از مقادیر پیش فرض هم می توانیم متغیرهای محیط را در فایل ساده کرده و هم اطمینان حاصل کنیم که اگر در مورد متغیرهای محیط چیزی خراب شود، مقادیر production را پیش فرض می کنیم، نه مقادیر محلی نامن! جنگو دارای بسیاری از ویژگی های امنیتی داخلی است و با افزودن چک لیست دیپلویمنت ، اکنون ما می توانیم سایت خود را که اطمینان داریم دارای امنیت بالایی است دیپلوی کنیم.

در نهایت، امنیت یک نبرد دائمی است و در حالی که مراحل این فصل بیشتر مناطق را پوشش می دهد نگرانی، به روز نگه داشتن وب سایت خود با آخرین نسخه جنگو برای ادامه ایمنی بسیار مهم است.

## ۱۵-۸ - توسعه پروژه

تا به این جای کار روی سیستم شخصی به عنوان محیط توسعه کار می‌کردیم. حال زمان آن فرا رسیده که پروژه را برای دسترسی عمومی (دسترسی از طریق اینترنت) نیز دیپللوی کنیم. عنوان دیپللوی موضوع گسترده‌ای می‌باشد، به گونه‌ای که می‌توان در کتاب جداگانه‌ای مفصل به توضیح آن پرداخت. در مقایسه با دیگر فریم ورک‌ها، جنگو در این موضوع بسیار کاربردی می‌باشد. امکان دیپللوی اپلیکیشن جنگو تنها با یک کلیک رو اکثر پلتفرم‌های هاستینگ وجود نداشته و همین دلیل نیازمند کار بیشتر سمت توسعه دهنده‌گان می‌باشد، با این حال امکان سفارشی سازی بالایی را در مقایسه با روش معمول در جنگو برای ما فراهم می‌آورد. در فصل قبل با پیکره بندی کامل فایل docker-compose-prod.yml به صورت جداگانه، همچنین به روزرسانی فایل config/setting.py، اپلیکیشن جنگو را آماده قرار گیری در محیط پروداکشن کردیم. در این فصل به ترتیب اقدام به بررسی نحوه انتخاب هاستینگ، اضافه کردن وب سرور برای محیط پروداکشن و همچنین اطمینان از پیکره‌بندی فایل‌های static/media، قبل از استقرار فروشگاه کتاب آنلاین، خواهیم نمود. لین سوال این است که از پلتفرم ابری استفاده کنیم یا رایانش ابری. پلتفرم ابری یکی از گزینه‌های هاستینگ معتبر می‌باشد، که اکثر پیکره‌بندی‌های اولیه آن انجام شده و امکان مقیاس پذیری را نیز برای سایت شما فراهم می‌کند. از سرویس دهنده‌های مشهور پلتفرم ابری، می‌توان به PythonAnyWhere و Dokku در بین دیگر سرویس دهنده‌ها اشاره کرد. اگرچه پلتفرم ابری در مقایسه با رایانش ابری دارای هزینه بیشتری می‌باشد ولی این سرویس زمان زیادی از توسعه دهنده‌گان را صرفه‌جویی می‌کند. در مقابل پلتفرم ابری، رایانش ابری وجود دارد که اگر چه انعطاف پذیری بیشتری نسبت به پلتفرم ابری داشته و همچنین هزینه کمتری دارد، اما نیازمند دانش فنی بالا و همچنین اطمینان کافی از نصب و راه اندازی را دارد. از سرویس دهنده‌گان رایانش ابری هم می‌توان به Google Compute Engine و Amazon EC2، DigitalOcean و Linode اشاره کرد.

حال باید از کدام استفاده کرد؟ توسعه دهنده‌گان جنگو تمایل دارند از یکی از این دو سرویس را استفاده کنند: اگر آن‌ها اقدام به دیپللوی پایپ لاین روی سرویس رایانش ابری کرده باشند، انتخاب آن‌ها رایانش ابری خواهد بود، در غیر این صورت سراغ پلتفرم ابری خواهند رفت.

از آنجایی که استفاده از سرویس رایانش ابری پیچیده بوده و نیازمند پیکره‌بندی‌های زیادی می‌باشد، ما در اینجا از پلتفرم ابری Heroku استفاده خواهیم کرد. اگر چه پلتفرم ابری یک تکنولوژی بالغ بوده و همچنین این سرویس دهنده دارای پلن رایگان مناسب برای فروشگاه کتاب آنلاین ما می‌باشد، اما انتخاب Heroku یک موضوع اختیاری می‌باشد.

در توسعه لوکال، جنگو بر اساس staticfiles app اقدام به جمع آوری و ارائه فایل‌های استاتیک در سراسر پروژه می‌کرد، اگر چه راه حل آسانی می‌باشد ولی کاربردی و امن نیست. برای محیط پروداکشن باید collectstatic اجرا شود، تا تمامی فایل‌های استاتیکی که توسط STATIC\_ROOT مشخص شده اند را در یک دایرکتوری مشخص کامپایل کند. با به روزرسانی STATICFILES\_STORAGE می‌توان آن‌ها را در یک سرور واحد در کنار اپلیکیشن جنگو یا یک سرور جداگانه و یا همچنین سرویس‌های ابری مانند CDN، ارائه داد. در این پروژه ما فایل‌های سرور خود را به کمک WhiteNoise ارائه خواهیم داد، که این پکیج کاملاً با سرویس Heroku سازگار بوده و سرعت بسیار بیشتری نسبت به جنگو در حالت عادی به ما ارائه می‌دهد. در محله اول اقدام به نصب پکیج whitenoise در داکر و همچنین اقدام به توقف کانتینر خواهیم کرد:

```
$ docker-compose exec web pipenv install whitenoise==5.1.0
```

```
$docker-compose down
```

تا انجام تغییرات در تنظیمات اپلیکیشن جنگو، ما اقدام به ساخت دوباره کانتینر از ایمیج اپلیکیشن نخواهیم کرد. تا زمانی که که از داکر استفاده می‌کنیم، امکان تغییر به whitenoise به صورت لوکالی همانند محیط پروداکشن را داریم. در حالی که می‌توان باست کردن آرگومان nostatic این کار را انجام داد، ولی در عمل این کار خسته کننده خواهد بود. رویکرد بهتر این می‌باشد که در پیکره بندی INSTALLED\_APPS دستور whitenoise را قبل از django.contrib.staticfiles اضافه کنیم. ما همچنین برای استفاده از whitenoise پیکره بندی‌های آن را در بخش MIDDLEWARE دقیقاً پایین خط SecurityMiddleware و بخش STATICFILES\_STORAGE اضافه می‌کنیم.

```
# config/settings.py
INSTALLED_APPS = [
```

```

'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'whitenoise.runserver_nostatic', # new
'django.contrib.staticfiles',
'django.contrib.sites',
...
]
MIDDLEWARE = [
    'django.middleware.cache.UpdateCacheMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware', # new
...
]
STATICFILES_STORAGE=
'whitenoise.storage.CompressedManifestStaticFilesStorage' # new

```

با انجام تغییرات، ما می‌توانیم پروژه را در حالت توسعه به صورت لوکال اجرا کنیم.

```
$docker-compose exec web python manage.py collectstatic
```

متاسفانه white-noise عملکرد خوبی در مواجه با فایل‌های چندسانه‌ای آپلود شده توسط کاربران ندارد. کاورهای کتاب ما توسط جنگو ادمین اضافه شده است ولی عملکرد آن شبیه به فایل‌های آپلود شده توسط کاربران می‌باشد. در نتیجه، در حالی که در توسعه لوکال به صورت دلخواه ظاهر می‌شوند، ولی در تنظیمات محیط پروداکشن ظاهر نمی‌شوند. رویکرد پیشنهادی استفاده از پکیج محبوب django-storage کنار یک CDN اختصاصی مانند S3 می‌باشد. این رویکرد نیازمند پیکره‌بندی‌های اضافی که توضیحات مربوط به آن، خارج از محدوده کتاب می‌باشد.

فایل wsgi.py با پیکره بندی های پیش فرض (Web Server Gateway Interface) ساخته شد. این مشخصات برای ارتباط بین وب اپ ساخته شده (مانند پروژه فروشگاه کتاب آنلاین) با وب سرور استفاده می‌گردد. برای محیط پروداکشن رایج است که از Unicorn یا آنلاین) به جای WSGI پیش فرض استفاده کنیم، هر دو این‌ها باعث بهبود عملکرد می‌گردند ولی به دلیل سادگی در پیاده سازی انتخاب ما Unicorn می‌باشد. مرحله اول نصب پکیج Gunircorn در پروژه و سپس متوقف کردن کانتینرها می‌باشد:

```
$ docker-compose exec web pipenv install gunicorn==20.0.4
```

```
docker-compose down $
```

اکنون وارد وبسایت Heroku شده و به صورت رایگان ثبت نام کنید. بعد تایید ایمیل، CLI (Command Line Interface) مربوط به Heroku را نصب کنید تا بتوانیم از طریق "heroku login" دستور اسکریپتی منتقل خواهد کرد. سپس مطمئن شوید که Heroku از طریق command line با این جایگزین استقرار را انجام دهیم. جزئیات نصب مرحله اخیر، ورود با مشخصات کاربری از طریق command line با تایپ دستور heroku login می‌باشد. از ایمیل و رمزعبوری که برای ساخت اکانت در Heroku استفاده کرده‌اید، برای لاگین استفاده کنید.

```
$heroku login
```

حال با ما دو انتخاب روبه‌رو هستیم: استقرار از طریق راه حل پیشین یا با استفاده از کانتینر داکر. مورد اخر (استفاده از کانتینر داکر) یک رویکرد جدید در Heroku و دیگر ارائه دهنده‌گان خدمات هاستینگ می‌باشد که به تازگی اضافه شده است. همانطور که داکر توسعه لوکال را بر عهده گرفته بود، استقرار را نیز بر عهده خواهد گرفت. اگر شما یک بار پیکره‌بندی‌های لازم برای استقرار کانتینر را انجام دهید، تعویض هاستینگ بسیار ساده تری در مقایسه با روش‌ها معمول هر هاستینگ خواهد داشت. پس ما استقرار را با استفاده از کانتینرهای داکر انجام خواهیم داد. با این حال ما با یک انتخابی دیگر در بین ویژگی‌ها کانتینرها، روبه‌رو هستیم: استفاده از ایمیج‌ها از پیش ساخته شده موجود در داکر رجیستری (مانند داکرhab) و یا افزودن فایل heroku.yml. به دلیل اجازه استفاده از دستورات بیشتر و همچنین شباهت بیشتر به رویکرد پیشین Heroku با استفاده از Procfile در پیکره‌بندی، از رویکرد دوم استفاده خواهیم کرد.

ر استقرارهای غیرداکری Heroku، از فایل Procfile برای استقرار یک سایت استفاده می‌کنیم، برای کانتینرهای Heroku نیز از یک رویکر مشابه استفاده می‌کنیم، در این رویکرد یک فایل docker-compose.yml با نام heroku.yml در دایرکتوری root ساخته می‌شود. این فایل شبیه فایل docker-compose.yml است که برای ساخت کانتینر در محیط توسعه لوکال استفاده می‌کردیم، می‌باشد.

با دستور زیر فایل heroku.yml را می‌سازیم.

```
$touch heroku.yml
```

این فایل دارای چهار قسمت نصب (setup)، ساخت (build)، انتشار (release) و اجرا (run) می‌باشد. وظیفه اصلی بخش نصب، مشخص کردن افزونه‌های مورد نیاز می‌باشد. معمولاً این افزونه‌ها توسط Heroku با پرداخت هزینه ارائه می‌شود. بزرگترین آن پایگاه داده ما بوده، که روی افزونه رایگان Heroku-postgresql قرار دارد. با ارائه سرویس و به روزرسانی‌های امنیتی آن باعث می‌شود ما به راحتی سایز و دسترسی پذیری سرویس را بر اساس نیاز ارتقا دهیم. بخش ساخت (build) مشخص می‌کنیم که فایل Dockerfile چگونه باید ساخته شود، این به فایل Dockerfile که در دایرکتوری root قرار دارد، بستگی دارد. مرحله انتشار برای انجام وظایف قبل از انتشار نسخه جدید، کاربرد دارد. برای مثال، ما باید مطمئن شویم که collectstatic در هر دیپلوی به صورت خودکار اجرا می‌گردد. در نهایت در مرحله اجرا، ما مشخص می‌کنیم که کدام پروسس باید اپلیکیشن را اجرا کند، به ویژه استفاده از Gunicorn به عنوان وب سرور می‌باشد.

setup:

addons:

- plan: heroku-postgresql

build:

docker:

web: Dockerfile

release:

image: web

command:

```
- python manage.py collectstatic --noinput
web: gunicorn config.wsgi
```

حال یک اپ جدید برای پروژه فروشگاه کتاب ایجاد می‌کنیم. اگر شما عبارت heroku create را تایپ کنید، Heroku یک نام تصادفی به اپ اختصاص خواهد داد. تا زمانی که اسمی در به صورت سراسری می‌باشند، امکان استفاده از اسمی معمول مانند "blog" یا Heroku وجود ندارد. نام اپ می‌تواند همیشه در داخل Heroku تغییر یابد تا در فضای نام سراسری در دسترس باشد.

```
$ heroku create
Creating app... done, ⏺ fast-ravine-89805
https://fast-ravine-89805.herokuapp.com/
https://git.heroku.com/fast-ravine-89805.git
```

قبل از ارسال کدهای مان به Heroku، پایگاه داده PostgreSQL میزبان را مشخص کنید. در مورد ما، نسخه رایگان hobby-dev به خوبی کار می‌کند. همچنین قابلیت آپدیت در آینده را دارد.

```
$ heroku addons:create heroku-postgresql:hobby-dev -a fast-ravine-89805
Creating heroku-postgresql:hobby-dev on ⏺ fast-ravine-89805... free
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pg:copy
Created postgresql-curved-34718 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
```

ما آماده ایم! با ساخت یک Heroku ریموت، یک نسخه از کدهای ما در سرورهای میزبان وجود خواهد داشت. مطمئن شوید a- و نام اپ وجود داشته باشد، سپس کدها رو به Heroku ارسال کنید که نتیجه آن ساخت ایمیج و کانتینر داکری خواهد بود.

```
$ heroku git:remote -a fast-ravine-89805
set git remote heroku to https://git.heroku.com/fast-ravine-89805.git
$git push heroku master
```

ارسال اولیه ممکن است کمی طول بکشد. شما می‌توانید با کلیک رو تب "Activity" در داشبورد مدیریتی Heroku، میزان پیشرفت هر فعالیت را مشاهده کنید. حال پروژه فروشگاه کتاب ما به صورت آنلاین در دسترس می‌باشد. به خاطر داشته باشید در حالی که کدهای ما دقیقاً در Heroku کپی شده است، ولی پروژه ما در محیط پروداکشن دارای پایگاه داده مخصوص به خود می‌باشد که در حال حاضر هیچ داده‌ای داخل آن قرار ندارد. برای اجرا دستورات در Heroku باید heroku run را اجرا کنید. برای مثال ما باید پایگاه داده اولیه را کرده و سپس یک اکانت superuser بسازیم.

```
$ heroku run python manage.py migrate
$heroku run python manage.py createsuperuser
```

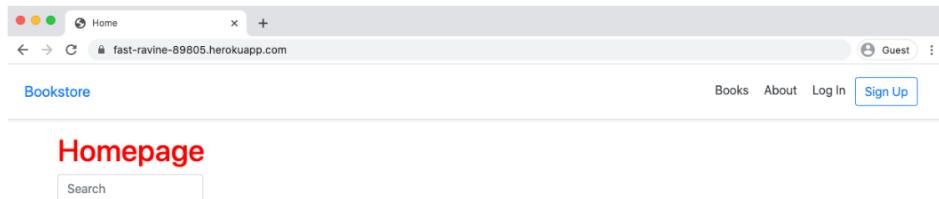
بعد از باز کردن اپ، با خطای ریدایرکت روبرو می‌شویم!

بررسی‌های دقیق نشان می‌دهد که این مشکل مربوط به تنظیمات SECURE\_SSL\_REDIRECT می‌باشد Heroku. از پروکسی‌ها استفاده می‌کند، پس ما باید هدر مربوطه را پیدا کرده و هدر SECURE\_PROXY\_SSL\_HEADER را آپدیت کنیم. به صورت پیش فرض هدر ذکر شده برابر None می‌باشد، به دلیل اعتمادی که به Heroku داریم مقدار آن را برابر (HTTP\_X\_FORWARDED\_PROTO, 'https') این تنظیمات باعث آسیب به توسعه لوكال ما نشده و ما می‌توانیم به صورت مستقیم آن را در مسیر config/settings.py قرار دهیم:

```
# config/settings.py
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https') # new

تغییرات را در گیت commit کرده و سپس کدهای به روز شده را به Heroku انتقال دهید:
$ git status
$ git commit -m 'secure_proxy_ssl_header and allowed_hosts update'
```

\$git push heroku master



بسیار معمول است که گاهی اوقات شما هنگام استقرار اپ با خطاهایی مواجه شوید. کافی است شما دستور `heroku logs --tail` را وارد کنید تا لگ های مربوط به خطاهای، اطلاعات و دیباگ را مشاهده کنید. امیدوارم روند استقرار بدون مشکل انجام شود، ولی در عمل حتی در سرویس‌های پلتفرمی (PaaS) مانند Heroku نیز، ممکن است مشکلاتی پیش آید. اگر شما صفحه خطا را مشاهده کردید، دستور `heroku logs -tail` را تایپ کنید تا خطاهای را مشاهده کرده و مشکل را تشخیص دهید.

Heroku دارای لیست بزرگی سرویس‌های افزونه‌ای می‌باشد که با پرداخت هزینه آن، به سرعت می‌توانید آن را به سایت خود اضافه کنید. برای مثال، برای فعالسازی کشینگ با `Memcache`، افزونه `Memcachier` نیاز است. پشتیبان گیری روزانه اختیاری می‌باشد ولی یک ویژگی ضروری برای پایگاه‌داده‌های محیط پروداکشن می‌باشد. اگر شما از یک دامنه اختصاصی برای سایت خودتان استفاده می‌کنید، اطمینان از گواهینامه SSL برای هر وبسایتی حیاتی می‌باشد، برای فعل کردن این قابلیت، شما باید در یک لایه غیررایگان قرار داشته باشید.

کدهای بسیار زیادی در فصل وجود داشت، اگر با مشکلی روبرو شدید، لطفا `official source code on Github` را چک کنید. حتی با وجود مزایای بسیاری که سرویس‌هایی پلتفرمی (PaaS) مانند Heroku دارند، استقرار یک کار پیچیده و خسته کننده برای توسعه دهنده‌گان می‌باشد. من شخصاً، نیاز دارم که سایت من فقط کار کند، ولی اغلب مهندسان از چالش‌های ایجاد شده برای بهبود در عملکرد، امنیت و مقیاس پذیری لذت می‌برند. اندازه گیری پیشرفت سایت بسیار آسان می‌باشد: آیا سرعت سایت افزایش یافته است؟ آیا سایت همیشه در دسترس است؟ آیا امنیت آن به روزرسانی شده است؟ کار کردن روی این مشکلات اغلب بهتر از اضافه کردن قابلیت جدید به سایت می‌باشد.



كلام آخر

ساختن یک وبسایت «حروفهای» کار کوچکی نیست، حتی با تمام کمک هایی که یک فریمورک وب مانند جنگو به ما می کند. داکر یک ویژگی و مزیت کلی در اختیار ما قرار می دهد مخصوصا در زمینه کار تیمی استانداردسازی پروژه در محیط های لوکال و پروداکشن بدون توجه به سیستم کاربر است. به هر حال داکر برای خودش یک ابزار پیچیده است در حالی که ما در این کتاب از آن به طور عاقلانه ای استفاده کردیم که بسته به نیازهای یک پروژه بسیار بیشتر از آن کار می تواند انجام دهد.

خود جنگو با پروژه های کوچک رابطه خوبی دارد به دلیل اینکه پیش فرض و دیفالت جنگو بر توسعه سریع محیط لوکال تاکید دارد اما این تنظیمات باید به صورت سیستماتیک برای محیط پروداکشن آپدیت شوند از جمله ارتقا دیتابیس به پستگرس ، استفاده از مدل یوزر شخصی سازی شده، متغیرهای محیطی، پیکربندی جریان ثبت نام کاربر، تنظیمات فایل های استاتیک، ایمیل و ...

خبر خوب این است که مراحل مورد نیاز برای رسیدن به مرحله پروداکشن رویکرد کاملا مشابهی دارد. از این رو نیمه اول این کتاب به طور عمده درباره پروژه نهایی ساخته شده است. متوجه خواهید شد که این مراحل تقریباً در هر پروژه جدید جنگو استاندارد می باشد. نیمه دوم کتاب بر ایجاد یک سایت کتابفروشی واقعی با بهترین شیوه های مدرن، افزودن نظرات، آپلود تصاویر، تنظیم دسترسی ها، اضافه کردن سرج، بررسی عملکرد و مسائل امنیتی متمرکز شد و در نهایت در Heroku با کانتینرها دیپلوی شد.

با تمام محتوایی که در این کتاب پوشش داده شده است، ما فقط سطح آنچه جنگو می تواند انجام دهد را بررسی کرده ایم. این ماهیت توسعه وب مدرن است: تکرار مداوم.

جنگو یک یار عالی برای ساختن یک وبسایت حرفه ای می باشد به خاطر اینکه بسیاری از ابزارهای مورد نیاز قبلاً در نظر گرفته شده و در فریمورک گنجانده شده است. اما دانشی لازم است تا بدانیم چگونه این کلید تولید محصول را روشن کنیم تا بتوانیم از مزایای شخصی سازی که جنگو اجازه می دهد استفاده کنیم.

در نهایت هدف این کتاب این است: تا به شما به عنوان یک خواننده طيف کاملی از آنچه جنگو و وب سایت های حرفه ای نیاز دارند نشان دهد.

همانطور که در مورد توسعه وب و جنگو بیشتر یاد می گیرید توصیه می کنم در مورد بهینه سازی وبسایت قبل از موعد احتیاط کنید. مواردی مثل اضافه کردن ویژگی و بهینه سازی پروژه که فکر می کنید بعدا برای پروژه لازم است و سوشه انگیز می باشد. لیست کوتاهی از این موارد می تواند شامل استفاده از cdn ها برای فایل های استاتیک و مدیا فایل ها آنالیز هوشمندانه کوئری های دیتابیس اضافه کردن ایندکس به مدل ها و... می باشد.

حقیقت این است که در هر پروژه وبسایتی، همیشه کارهای بیشتری نسبت به زمانی که در اختیار داریم وجود دارد که می توانیم آن ها را انجام دهیم. این کتاب مبانی را پوشش داده است که برای درست کردن آن ارزش این زمان محدود را دارد. همچنین مراحل اضافی در مورد امنیت، عملکرد و ویژگی ها را به صورت همزمان به شما نشان می دهد. سعی کنید تا زمانی که ضروری نیست در برابر اضافه کردن پیچیدگی جدید به پروژه مقاومت کنید.