💡 Zen of Python فلسفه‌های کدنویسی پایتون (PEP 20)

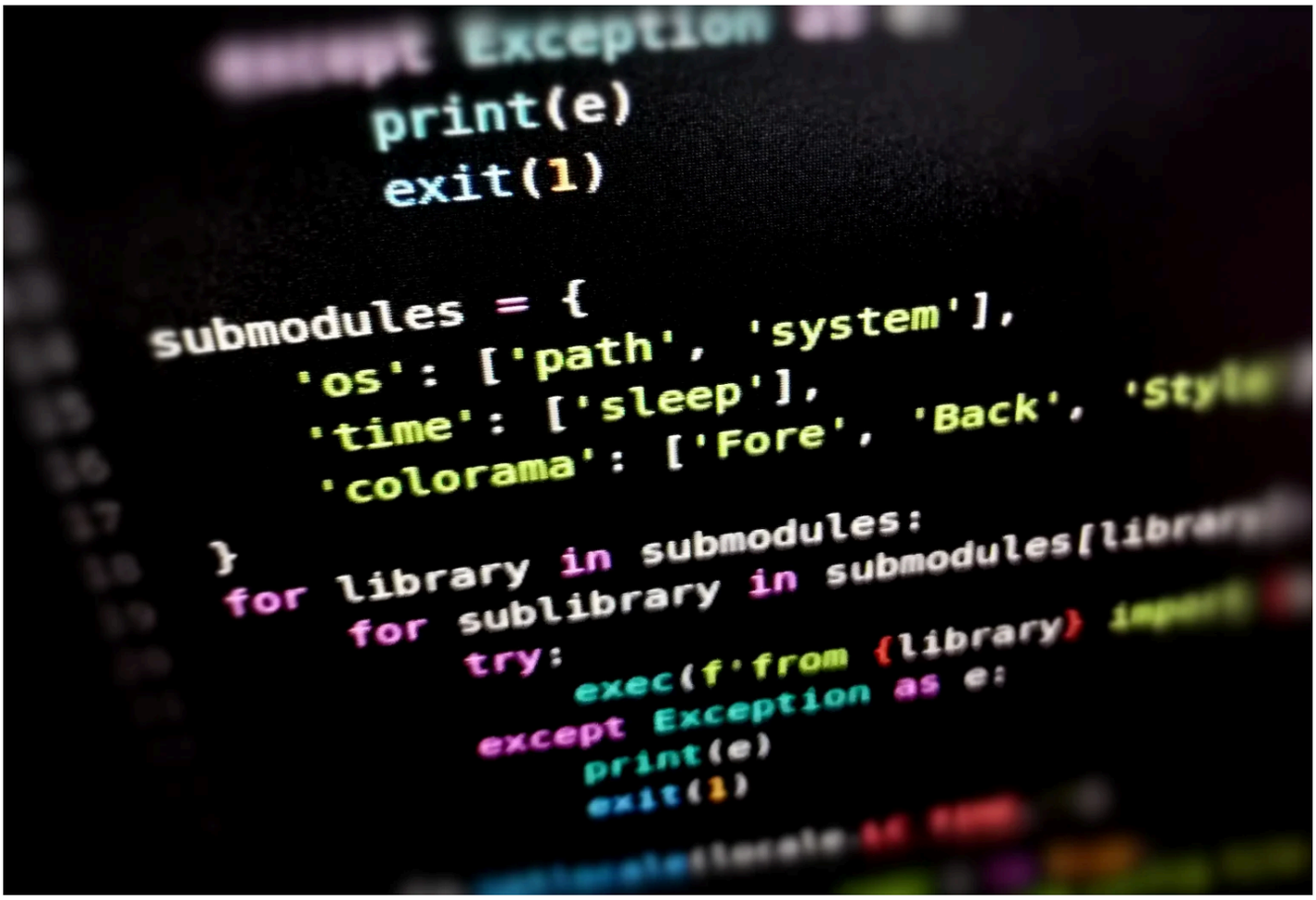
آموزش مقالات هاست پایتون

در زمینه برنامه نویسی، یکی از مهم‌ترین اصولی که به برنامه نویسان کمک می‌کند، این است که زبان‌هایی را انتخاب کنند که از نظر مفهومی، زیبا و قابل فهم باشد. یکی از زبان‌هایی که این اصول را به خوبی رعایت کرده و در خود دارد، پایتون است.

PEP 20 که به **Zen of Python** معروف است، مجموعه‌ای از 19 اصل فلسفی است که برای کمک به برنامه نویسان پایتون در نوشتن کدهای تمیز و خوانا تهیه شده است.

زیبا بهتر از زشت است

Beautiful is better than ugly



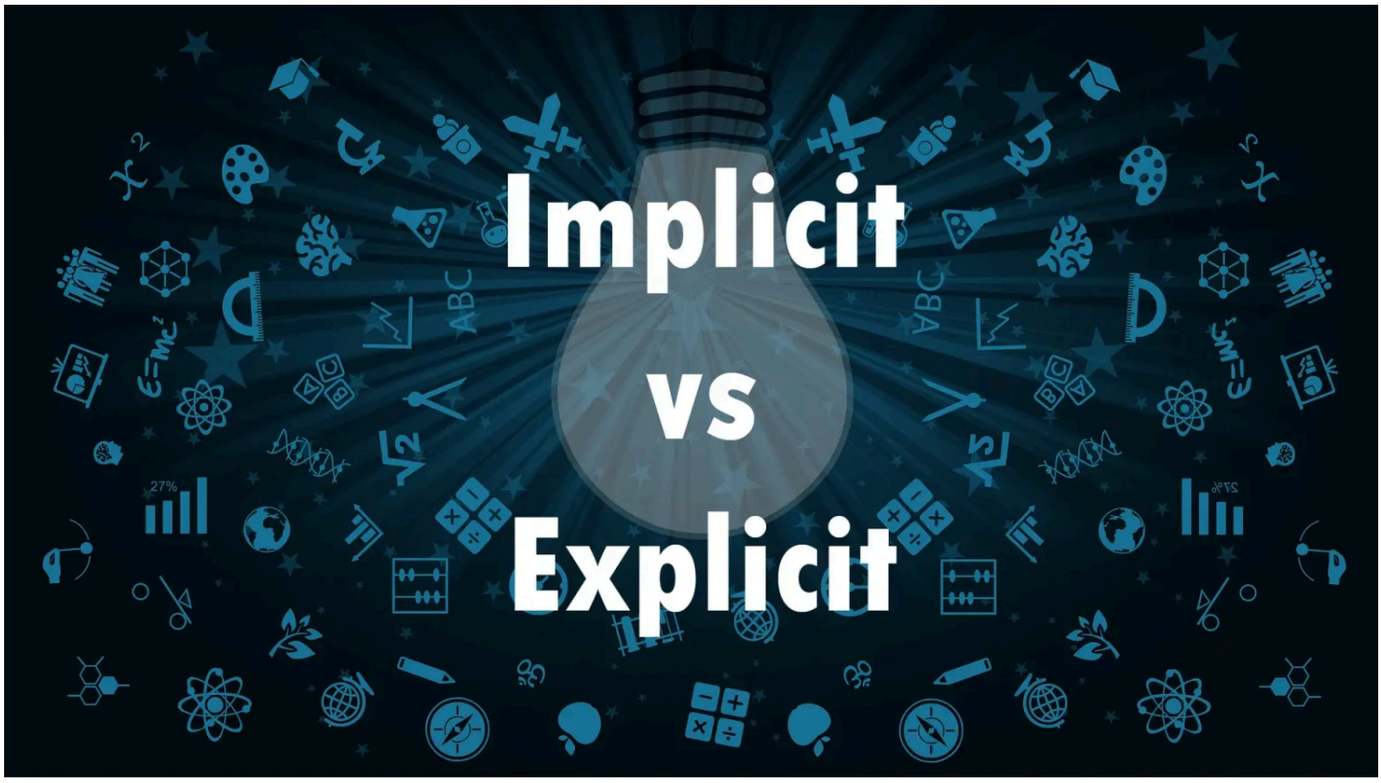
در زمینه برنامه نویسی، زیبایی کد اهمیت زیادی دارد. این زیبایی به معنای ساده و شفاف بودن کد است که باعث می‌شود کد قابل فهم باشد. کد زیبا به راحتی توسط دیگران قابل خواندن است و به برنامه نویسان این امکان را می‌دهد که به سرعت عیب های پروژه را شناسایی کرده و به راحتی تغییرات را اعمال کنند.

در مقابل، کد زشت معمولا به معنای پیچیدگی و عدم یکپارچگی کد است. چنین کدی ممکن است برای برنامه نویسان دیگر یا حتی خود نویسنده در آینده، گیج‌کننده باشد.

زیبایی در کد تنها به ظاهری زیبا مربوط نمی‌شود بلکه به ساختار صحیح و اصولی آن نیز برمی‌گردد که باعث می‌شود برنامه نویسان قادر باشند به راحتی به مفهوم آن کد را درک کنند و به بهبود و ویرایش آن بپردازند.

واضح بهتر از مبهم بودن است

Explicit is better than implicit



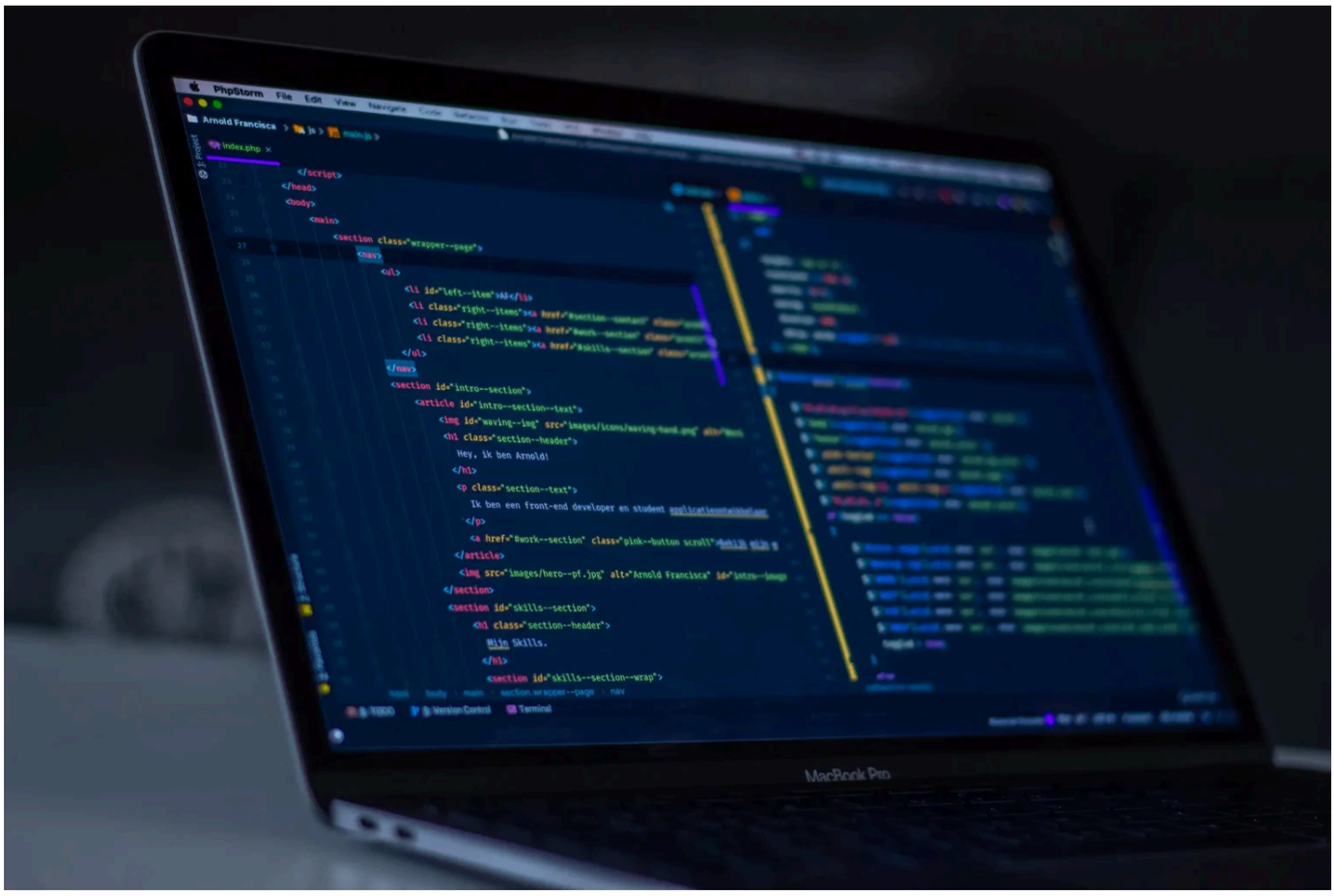
یکی از اصول مهم در نوشتن کد این است که کد باید به‌طور واضح نوشته شود. این بدین معناست که هر اتفاقی که در کد می‌افتد، باید برای دیگران به‌راحتی قابل درک باشد.

استفاده از کدهای واضح به این معنی است که هیچ چیزی پنهان نمی‌ماند و تمامی رفتارهای کدها **بدون ابهام** بیان می‌شوند. این نوع کدها باعث می‌شوند که برنامه نویس ها دیگر به راحتی متوجه شوند که برنامه دقیقا چه عملی انجام می‌دهد.

در مقابل، کدهای مبهم ممکن است نیاز به تفسیر داشته باشد و ممکن است برای کسانی که آن را نوشته‌اند یا برنامه نویسان دیگری که با آن روبرو می‌شوند، گیج‌کننده باشد. به همین دلیل، استفاده از کدهای واضح در برنامه‌نویسی نه تنها باعث افزایش خوانایی می‌شود، بلکه به جلوگیری از خطاها و عیب یابی کدها نیز کمک می‌کند.

ساده بهتر از پیچیده است

Simple is better than complex



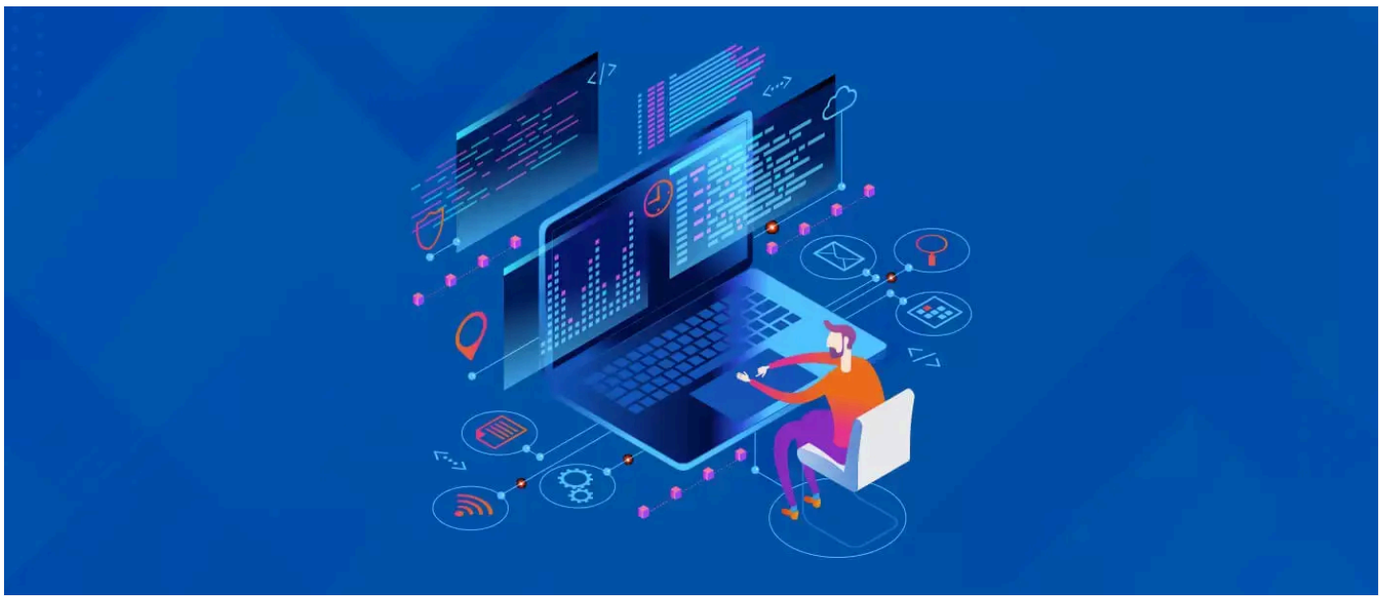
در کدنویسی، ساده بودن کد یکی از مهم‌ترین ویژگی‌ها می‌باشد. کدی که به صورت ساده نوشته شده باشد، معمولا فهم آن برای نویسنده و برنامه نویسان دیگر راحت تر می‌شود. وقتی یک راه‌حل ساده برای یک مساله وجود دارد، رفتن به سراغ گزینه‌های پیچیده، تنها باعث سختی بیشتر در ویرایش و توسعه پروژه خواهد شد. سادگی نه‌تنها در زمان نویسنده صرفه‌جویی می‌کند،

بلکه عملکرد پروژه و کدها را بهبود می‌دهد.

از طرف دیگر، پیچیدگی در کد باعث سردرگمی و افزایش احتمال خطا می‌شود. پایتون در فلسفه‌اش توصیه می‌کنند که اگر بتوان مساله‌ای را به شکل ساده حل کرد، نیازی به راه‌حل‌های پیچیده نیست. سادگی در کدنویسی به معنای احترام به زمان خود و دیگران است.

پیچیده بهتر از به هم ریختگی است

Complex is better than complicated



بعضی اوقات در برنامه نویسی، نمی‌توان از پیچیدگی جلوگیری کرد. برخی مساله‌ها نیازمند راه‌حل‌های پیچیده هستند. اما این **پیچیدگی باید ساختاریافته باشد**.

وقتی چیزی پیچیده است اما به درستی طراحی شده است، می‌توان مرحله به مرحله آن را درک کرد و توضیح داد. چنین پیچیدگی‌هایی قابل مدیریت هستند و برنامه نویسان می‌توانند با صرف زمان، عملکرد آن را درک کنند و آن را بهبود دهند.

در مقابل، به هم ریختگی کدها نشانه خوبی نیست و نبود نظم در کدها را نشان می‌دهد.

چنین کدی معمولا بدون فکر و ساختار نوشته شده و حتی نویسنده‌اش هم ممکن است نتواند توضیح دهد که آن کد چگونه کار می‌کند. به هم ریختگی کدها نه تنها ویرایش آن را سخت می‌کند، بلکه احتمال بروز خطا را نیز افزایش میدهد.

فلت بهتر از تو در تو است

Flat is better than nested

در برنامه نویسی، ساختارهای ساده و فلت به خوانایی بهتر کد کمک می‌کنند. زمانی که کد به صورت فلت نوشته شود یعنی بلوک‌های کد در یک سطح قابل پیگیری قرار دارند، نحوه اجرای برنامه واضح‌تر می‌شود.

این نوع ساختار باعث می‌شود که تغییرات با دردسری کمتری اعمال شوند.

از سوی دیگر، کدهای تو در تو، به ویژه زمانیکه بیش از اندازه عمق پیدا کنند، خوانایی کدها را کاهش می‌دهند.

پراکنده بودن بهتر از فشرده بودن است

Sparse is better than dense

در کدنویسی، پراکندگی منظم به خوانایی و درک بهتر کد کمک می‌کند. وقتی کدها بیش از حد فشرده شوند، برنامه نویس باید برای درک هر خط نیاز به صرف زمان بیشتری دارد و احتمال خطلا را افزایش می‌دهد.می‌توان با ایجاد فاصله مناسب بین خطوط، بخش‌بندی مناسب، استفاده از فضاهای خالی می‌توان کدی منظم و پراکنده نوشت. فشرده‌سازی افراطی شاید در نگاه اول موجب کوتاه شدن کد شود، اما در عمل خواندن و درک کد را سخت می‌کند.

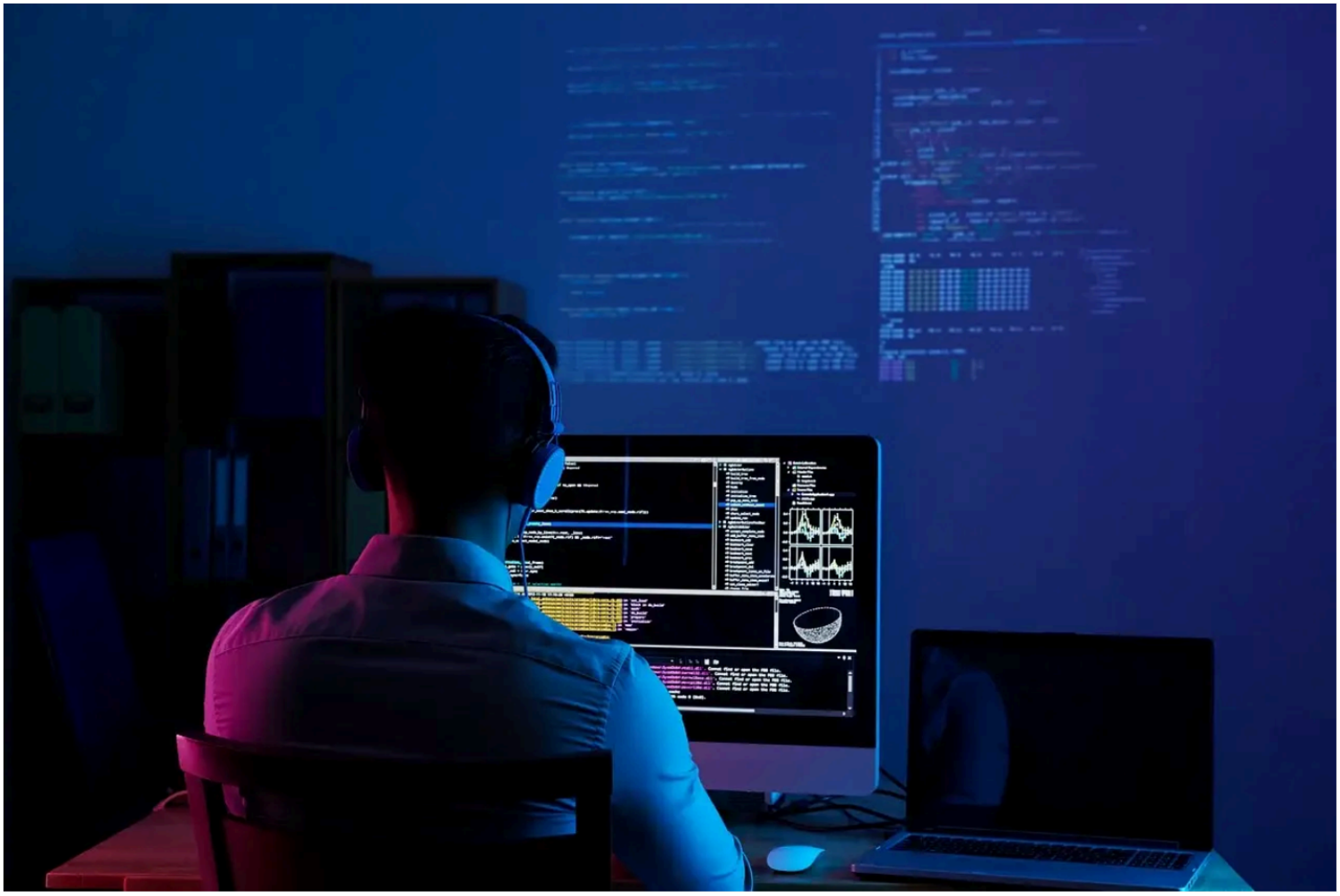
خوانایی مهم است

Readability counts

کدی که نوشته می‌شود نباید فقط برای کامپیوتر قابل فهم باشد بلکه در قدم اول باید برای برای انسان‌ها قابل فهم باشد. زمانیکه کدی خوانا باشد، یعنی ساختار مناسب، نام‌گذاری‌های معنادار و شفافیت داشته باشد، برنامه نویسان می‌توانند بدون سردرگمی عملکرد کد را درک کنند، عیب یابی را سریع‌تر انجام دهند و تغییرات لازم را به راحتی اعمال کنند. نادیده گرفتن خوانایی، حتی اگر کد به درستی کار کند و از نظر فنی درست باشد، در بلند مدت اشتباهات پیش‌بینی نشده‌ای به وجود خواهد آمد.

موارد خاص آن قدر مهم نیستند که قوانین را زیر پا گذاشت

Special cases aren’t special enough to break the rules



در برنامه‌نویسی، ممکن است با شرایطی مواجه شویم که در نگاه اول نیاز به راه‌حلی متفاوت و خارج از قواعد داشته باشد، اما حتی در چنین مواردی باید قواعد کلی حفظ شوند، چون اگر به خاطر هر مورد خاص، ساختار اصلی را زیر پا بگذاریم، به‌تدریج نظم و انسجام کد از بین می‌رود و در نهایت با سیستمی روبه‌رو می‌شویم که پر از استثنا و تصمیم‌های لحظه‌ای است و دیگر به راحتی قابل توسعه یا درک نخواهد بود.

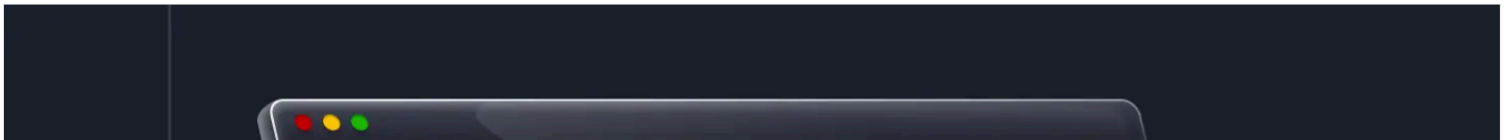
کارآمد بودن بر خلوص برتری دارد

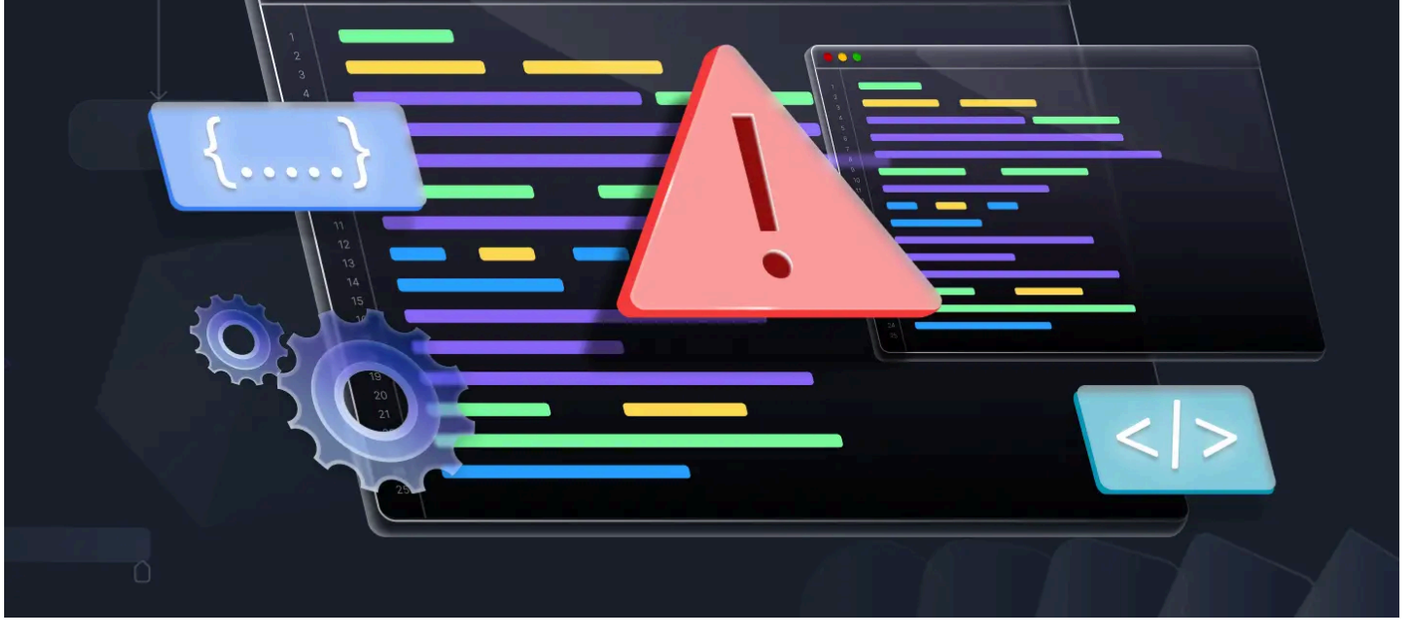
Although practicality beats purity

در دنیای واقعی برنامه‌نویسی، آنچه در عمل جواب می‌دهد معمولا ارزشمندتر از راه‌حل‌های کاملا ایده‌آل و بی‌نقص است؛ گاهی لازم است از برخی اصول صرف‌نظر کرد تا بتوان مساله‌ای را به‌صورت عملی حل کرد. خلوص در طراحی و پیروی کامل نظریه‌ها ممکن است زیبا باشد، اما اگر مانع انجام کار شود یا باعث پیچیدگی گردد، دیگر ارزشش را از دست می‌دهد. این اصل یادآوری می‌کند که هدف اصلی برنامه‌نویسی، ساختن ابزارهایی کارآمد و قابل استفاده است، نه رسیدن به کدی غیر عملی که فقط روی کاغذ بی‌نقص به نظر می‌رسد.

خطاها باید همیشه دیده و مدیریت شوند

Errors should never pass silently





وقتی در کد خطایی رخ می‌دهد، باید آن را به‌وضوح دید و بررسی کرد، نه اینکه بی‌صدا از کنارش گذشت، چون پنهان ماندن خطاها می‌تواند در آینده باعث بروز مشکلاتی بزرگ‌تر و غیرقابل پیش‌بینی شود.

کدی که خطاهایش را گزارش نمی‌دهد، مانند دستگاهی است که خراب شده اما هیچ علامتی نمی‌دهد؛ ظاهرش ممکن است سالم باشد، اما درونش دیگر قابل اعتماد نیست.

اگرچه ممکن است در برخی شرایط خاص نیاز به نادیده گرفتن کنترل‌شده خطاها وجود داشته باشد، اما در حالت کلی، شفاف بودن در مواجهه با خطاها نشانه طراحی درست در پروژه است.

بعضی وقت ها عمدا خطایی نادیده گرفته می شود

Unless explicitly silenced

در برخی موارد خاص، ممکن است نیاز باشد که خطایی عمدا نادیده گرفته شود، اما این کار باید آگاهانه، نه اینکه به‌صورت اتفاقی انجام شود. وقتی خطایی با قصد قبلی نادیده گرفته می‌شود، مسولیت آن بر عهده‌ی برنامه نویس است و باید دلیلش را در کد یا مستندات آن بنویسد.

در برابر ابهام، از حدس زدن پرهیز کنید

In the face of ambiguity, refuse the temptation to guess

زمانی که با موقعیتی مبهم روبه‌رو می‌شویم، نباید با حدس زدن ابهام را برطرف کنیم. در برنامه‌نویسی، ابهام می‌تواند به خطاهای بزرگی منجر شود، چرا که حدس زدن اغلب نتیجه‌های نادرست به‌دنبال دارد.

این اصل به ما یادآوری می‌کند که در صورت عدم وضوح، بهتر است تا رسیدن به درک کامل از مساله صبر کنیم و از انجام تغییرات یا نوشتن کد بر اساس فرضیات خودداری کنیم. این فرآیند باعث می‌شود که کد دقیق‌تر باشد و باعث بروز مشکلات احتمالی نشود.

باید یک و ترجیحا فقط یک راه واضح برای انجام آن وجود داشته باشد

There should be one– and preferably only one — obvious way to do it

در برنامه‌نویسی، داشتن **یک راه‌حل واضح و استاندارد** برای انجام یک کار خاص، موجب می‌شود که کد خواناتر و با نظم بیشتری نوشته شود.

وقتی یک مشکل را می‌توان به روش‌های مختلف حل کرد، احتمال سردرگمی و اشتباهات بیشتر می‌شود. برای همین، در زبان‌های برنامه‌نویسی مانند پایتون تاکید بر این است که برای هر کار خاص، یک روش استاندارد و روشن وجود داشته باشد که همه برنامه نویسان به آن پایبند باشند.

این اصل به راحت‌تر کردن همکاری اعضای یک تیم کمک می‌کند و فرآیند توسعه پروژه را ساده‌تر می‌کند.

اگرچه این راه ممکن است در ابتدا واضح نباشد مگر اینکه هلندی باشید

Although that way may not be obvious at first unless you're Dutch

این جمله به نوعی شوخی در فرهنگ پایتون اشاره دارد، که به دلیل تاثیرگذار بودن خالق زبان پایتون، گویدو ون روسوم، که اهل هلند است، به‌وجود آمده است. در ابتدای کار با پایتون، ممکن است برخی از اصول و راه‌حل‌های آن برای تازه‌کارها کمی گیج‌کننده به نظر برسند، اما به‌مرور زمان و با تجربه بیشتر، این روش‌ها روشن و کارآمد خواهند شد. این **اشاره به فرهنگ هلندی** درواقع نوعی شوخی است که از تمایز خاص پایتون نسبت به زبان‌های دیگر و سادگی آن می‌باشد.

الان بهتر از هیچ وقت است

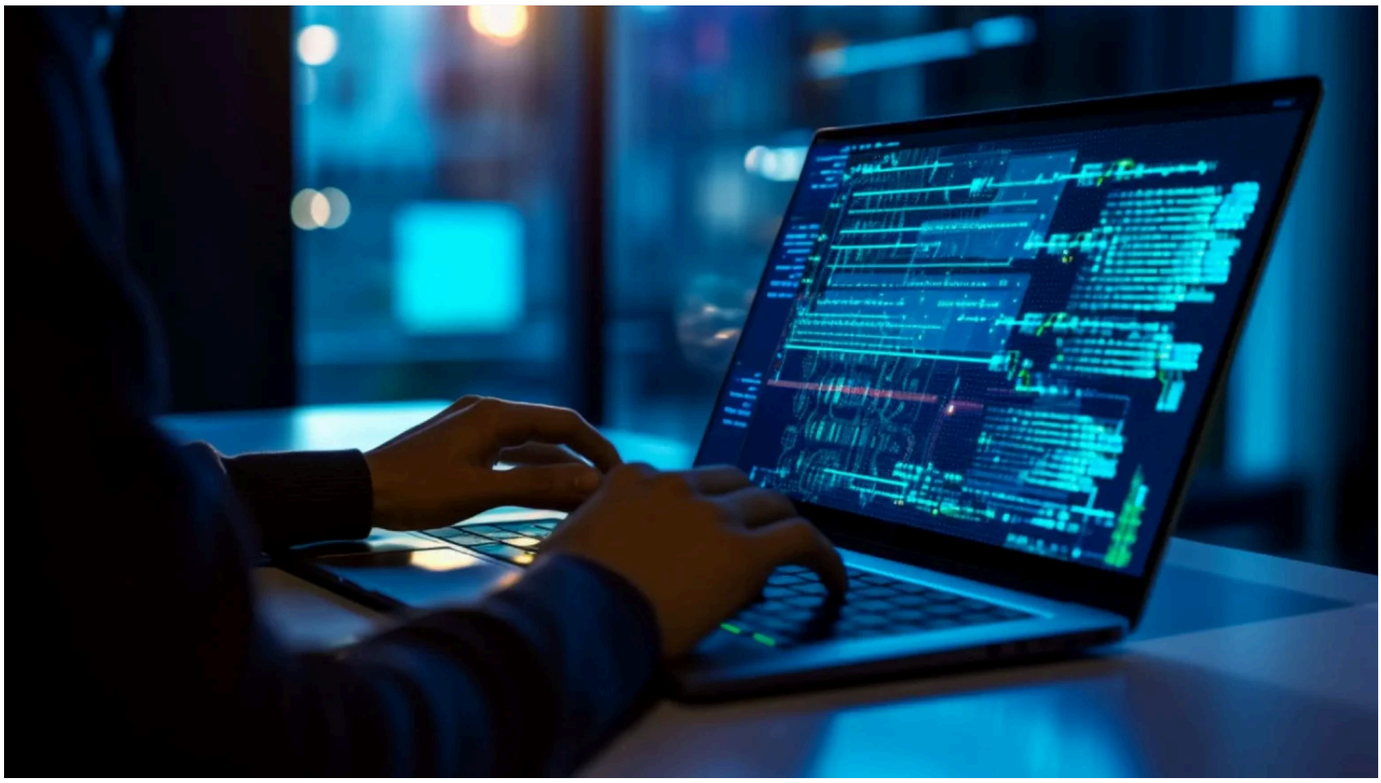
Now is better than never

در برنامه‌نویسی، گاهی ممکن است به‌خاطر ترس از اشتباه یا عدم آمادگی کامل، انجام کاری را به تعویق بیندازیم. اما این اصل به ما یادآوری می‌کند که **انجام دادن کارها، حتی اگر در ابتدا کامل نباشد، بهتر از اقدام نکردن است.** با شروع کردن، می‌توانیم تجربه کسب کنیم، اشتباهات را اصلاح کنیم و به تدریج به بهبود پروژه بپردازیم. به‌جای منتظر ماندن برای شرایط ایده‌آل، بهتر است که هم‌اکنون اقدام کنیم تا به تدریج پیشرفت کنیم و به هدف برسیم.

شاید اگر قصد خرید هاست پایتون دارید، الان وقتش باشد.

برخی اوقات، هیچ وقت بهتر از الان است

Although never is often better than right now



این اصل به **اهمیت تصمیم‌گیری آگاهانه و عدم شتاب‌زدگی** اشاره دارد. گاهی اوقات اتفاق می‌افتد که بلافاصله به یک تصمیم یا اقدام بپردازیم اما در بسیاری از مواقع، می‌توان از اجتناب از انجام فوری کارها نتیجه بهتری داشت.

این نکته به ویژه در مواقعی که ممکن است **تصمیمات عجولانه به مشکلات غیرمنتظره منجر شود**، بسیار اهمیت دارد. در واقع گاهی با تاخیر و بررسی دقیق‌تر پیش از اقدام، می‌توان تصمیمات بهتری گرفت.

اگر روش پیاده‌سازی برای توضیح دادن سخت باشد، ایده‌ی بدی است

If the implementation is hard to explain, it’s a bad idea

یکی از نشانه‌های یک طراحی ضعیف و یا پیچیده این است که وقتی تلاش می‌کنید آن را توضیح دهید نمی‌توانید به‌سادگی منظور خود را منتقل کنید.

اگر پیاده‌سازی یک راه‌حل پیچیده باشد و نتوانید آن را به‌راحتی برای دیگران شرح دهید، احتمالا آن ایده کارآمد و یا موثر نخواهد بود. **سادگی و شفافیت در طراحی** به این معنی است که هر برنامه نویسی بتواند به‌سرعت آن را درک کند و به‌راحتی با آن کار کند.

وقتی توضیح دادن پیاده‌سازی سخت می‌شود، نشانه این است که راه‌حل به‌اندازه کافی بهینه یا مناسب نیست.

اگر روش پیاده‌سازی برای توضیح دادن آسان باشد، ممکن است ایده‌ی خوبی باشد

If the implementation is easy to explain, it may be a good idea

پیاده‌سازی هایی که به راحتی قابل توضیح هستند اغلب ساده، شفاف و قابل فهم هستند و این خود نشان‌دهنده کارآمد بودن کدها است.

زمانی که می‌توانیم یک راه‌حل را به‌سادگی و بدون پیچیدگی برای دیگران توضیح دهیم نشان می‌دهد که طراحی اصولی و مفید است. این نوع پیاده‌سازی‌ها نه تنها به راحتی توسط تیم‌های مختلف قابل درک و استفاده هستند، بلکه گسترش آن‌ها نیز ساده‌تر خواهد بود.

namespace یکی از بهترین ایده‌هاست، بیایید از آن‌ها بیشتر استفاده کنیم!

Namespaces are one honking great idea — let’s do more of those!

namespace، روشی موثر برای **نظم دادن به کد و جلوگیری از تداخل** بین اجزای مختلف پروژه یا کد هستند کمک می‌کند تا متغیرها، توابع و یا کلاس‌هایی که نام‌های مشابهی دارند، در محدوده‌های جداگانه قرار گیرند و بدون اختلال و در کنار هم کار کنند.

namespace باعث می‌شود که در پروژه‌های بزرگ و یا در زمان استفاده از کتابخانه‌های مختلف کدها منظم‌تر، خواناتر و توسعه پذیرتر باشند.

وقتی از namespace به‌درستی استفاده شود، ساختار کلی برنامه شفاف‌تر می‌شود و از بروز اشتباهات ناخواسته در نام‌گذاری جلوگیری خواهد شد.
یک مثال برای درک بهتر موضوع:

```
def greet():
    print("greet!")

class MyGreeter:
    def greet(self):
        print("MyGreeter!")

greet()

greeter = MyGreeter()
greeter.greet()
```

ما دوتا تابع به اسم greet را در کد بالا تعریف کرده‌ایم.
اما چون یکی از آن‌ها خارج از کلاس و یکی درون کلاس MyGreeter تعریف شده است، هرکدام namespace خود را دارند و تداخلی ایجاد نمی‌کنند.

- ()greet به تابعی که خارج از کلاس تعریف شده است اشاره دارد.
- ()greeter.greet به متدی اشاره داره که در داخل کلاس MyGreeter تعریف شده.

فلسفه پایتون، که در قالب Zen of Python بیان شده، بر اصل‌هایی مانند **سادگی**، **وضوح**، **خوانایی** و **ساختار** تاکید دارد. رعایت این اصول نه‌تنها به نوشتن کدی قابل فهم و توسعه‌پذیر کمک می‌کند، بلکه موجب می‌شود برنامه نویسان در کارهای تیمی، دیباگ کردن و اعمال تغییرات در پروژه با چالش‌های کمتری مواجه شوند.

از اینکه با این پست پارس وب سرور با ما همراه بودید سپاسگزاریم.

💡 Zen of Python فلسفه‌های کدنویسی پایتون (PEP 20)

👈 منابع یادگیری برنامه‌نویسی (python-nodejs-php) سال ۲۰۲۵

⭐ خداحافظی با کامنت های اسپم وردپرس، بدون نصب افزونه

💫 آخرین انتقال، انتقال رایگان وب سایت به پارس وب سرور

دسترسی سریع

ورود | عضویت

ارسال تیکت پشتیبانی

قوانین سرویس ها

برخی از خدمات ما

هاست اختصاصی (ویژه)

هاست نود جی اس

هاست پایتون

PARS WEBSERVER

ما را در شبکه های اجتماعی دنبال کنید

© - تمامی حقوق مادی و معنوی این وب سایت متعلق به پارس وب سرور می باشد. 2025