چرا پایتون یاد بگیریم؟

چه تعداد زبان برنامه نویسی وجود دارد و چه فرقی با هم دارند؟ **چرا پایتون** انتخاب مناسبی است؟
زبان‌های برنامه‌نویسی مختلف قابلیت‌ها و کاربردهای مختلفی دارند و هرکدام دارای نقاط قوت و ضعف خاص خود هستند.

طبق آخرین رتبه بندی انجمن برنامه‌نویسیTIOBE ، پایتون یکی از ۱۰ زبان برنامه‌نویسی محبوب جهان است Python .یک زبان برنامه‌نویسی عمومی و سطح بالا است. می توانید از پایتون برای توسعه برنامه‌های رابط کاربری گرافیکی دسکتاپ، وب سایت‌ها و برنامه‌های کاربردی وب استفاده کنید.

در بین این زبان‌ها پایتون یکی از محبوب‌ترین زبان هاست. ولی چرا پایتون؟

۱- یادگیری پایتون بسیار راحت است و غالبا به افرادی که می‌خواهند برنامه‌نویسی را شروع کنند، پایتون معرفی می‌شود.
۲-پایتون زبان پرکاربردی است و در زمینه‌های مختلف می‌توان از آن استفاده کرد. برنامه‌نویسی وب با فریمورک جنگو، طراحی اپلیکیشن موبایل با فریمورکkivy ، برای محاسبات علمی و الگوریتم‌های یادگیری ماشین و کارهای آماری و. . . .
۳- سومین ویژگی مهم پایتون کتابخانه‌های زیاد آن است که برنامه‌نویسی را بسیار راحت کرده و شما می‌توانید از کدهای افرادی دیگر استفاده کنید.

علاوه بر این، پایتون، به عنوان یک زبان برنامه‌نویسی سطح بالا، به شما اجازه می‌دهد تا در کنار توجه به وظایف برنامه‌نویسی رایج، بر روی عملکرد اصلی برنامه تمرکز کنید. قوانین نحوی ساده این زبان برنامه‌نویسی باعث می‌شود که بتوانید پایه و اساس کدها را خوانا بنویسید و برنامه را طوری بنویسید که قابلیت نگه داری و پشتیبانی آن حفظ شود. همچنین دلایل متعددی وجود دارد که چرا باید پایتون را به سایر زبان‌های برنامه‌نویسی ترجیح دهید.

7دلیل برای اینکه باید برنامه‌های نرم افزاریتان را در پایتون بنویسید:

۱- کدهای قابل خواندن و قابل نگهداری

اولین دلیل از دلایل انتخاب پایتون، برای برنامه نویسی، قابلیت نگه داری و خوانایی کدهاست. هنگام نوشتن یک برنامه نرم افزاری، باید روی کیفیت کد منبع آن تمرکز کنید تا نگهداری و به روز رسانی را ساده کنید. قوانین نحوی پایتون به شما امکان می‌دهد مفاهیم را بدون نوشتن کد اضافی بیان کنید.
در عین حال، پایتون برخلاف سایر زبان‌های برنامه‌نویسی، بر خوانایی کد تأکید می‌کند و به شما اجازه می‌دهد از کلمات کلیدی انگلیسی به جای علائم نگارشی استفاده کنید. از این رو، می‌توانید از پایتون برای ساخت برنامه‌های سفارشی بدون نوشتن کد اضافی

استفاده کنید. پایه کد خوانا و تمیز به شما کمک می‌کند تا نرم افزار را بدون صرف زمان و تلاش اضافی، نگهداری و به روز رسانی کنید.

۲-پارادایم‌های برنامه‌نویسی چندگانه

مانند سایر زبان‌های برنامه‌نویسی مدرن، پایتون نیز از چندین پارادایم برنامه‌نویسی پشتیبانی می‌کند. پایتون به طور کامل از برنامه‌نویسی شی گرا و ساخت یافته پشتیبانی می‌کند. علاوه بر این، ویژگی‌های زبان آن از مفاهیم مختلفی در برنامه‌نویسی کاربردی و جنبه گرا پشتیبانی می‌کند. در عین حال، پایتون همچنین دارای یک سیستم نوع پویا و مدیریت خودکار حافظه است. پارادایم‌های برنامه‌نویسی و ویژگی‌های زبان به شما کمک می‌کند تا از پایتون برای توسعه برنامه‌های کاربردی نرم افزاری بزرگ و پیچیده استفاده کنید.

۳- سازگار با پلتفرم‌ها و سیستم‌های اصلی و بزرگ

در حال حاضر پایتون از بسیاری از سیستم عامل‌ها پشتیبانی می‌کند. حتی می‌توانید از مفسرهای پایتون برای اجرای کد روی پلتفرم‌ها و ابزارهای خاص استفاده کنید. همچنین، پایتون یک زبان برنامه‌نویسی مفسری است. این ویژگی، به شما این امکان را می‌دهد که یک کد را بدون کامپایل مجدد بر روی چندین پلتفرم اجرا کنید. بنابراین نیازی نیست بعد از هر تغییر کوچکی، برنامه را مجددا کامپایل کنید. می‌توانید کد برنامه اصلاح شده را بدون کامپایل مجدد اجرا کنید و فوراً تأثیر تغییرات ایجاد شده در کد را بررسی کنید. این ویژگی به شما کمک می‌کند تا بدون اینکه زمان توسعه ی کد بیشتر شود، تغییراتی در کد ایجاد کنید.
یک برنامه نویس ارزش این ویژگی را خیلی خوب متوجه می‌شود.

۴- کتابخانه استاندارد قوی

کتابخانه استاندارد بزرگ و قوی آن باعث می‌شود پایتون نسبت به سایر زبان‌های برنامه‌نویسی برتری ویژه ای داشته باشد. کتابخانه استاندارد به شما این امکان را می‌دهد که امکانات و خواسته‌های مورد نظرتان را از بین طیف گسترده ای از ماژول‌ها مطابق با نیازهای دقیق خود انتخاب کنید. هر ماژول به شما امکان می‌دهد تا بدون نوشتن کد اضافی، عملکردی را به برنامه پایتون اضافه کنید. به عنوان مثال، هنگام نوشتن یک برنامه وب در پایتون، می‌توانید از ماژول‌های خاصی برای پیاده سازی خدمات وب، انجام عملیات رشته، مدیریت رابط سیستم عامل یا کار با پروتکل‌های اینترنتی استفاده کنید. شما حتی می‌توانید اطلاعات مربوط به ماژول‌های مختلف را با مرور اسناد کتابخانه استاندارد پایتون جمع آوری کنید.

۵- بسیاری از چارچوب‌ها و ابزارهای منبع باز

متن باز بودن بسیاری از منابع، از دیگر دلایل انتخاب پایتون است. به عنوان یک زبان برنامه‌نویسی متن باز، پایتون به شما کمک می‌کند تا هزینه توسعه نرم افزار را به میزان قابل توجهی کاهش دهید. حتی می‌توانید از چندین چارچوب، کتابخانه و ابزارهای توسعه منبع باز پایتون برای کاهش زمان توسعه بدون افزایش هزینه آن استفاده کنید. شما حتی می‌توانید پایتون و ابزارهای توسعه ی مورد نظرتان را با توجه به نیازهای دقیق خود، از میان طیف گسترده ای از چارچوب‌های منبع باز انتخاب کنید. به عنوان مثال، می‌توانید با استفاده از چارچوب‌های وب قوی پایتون مانند جنگو، فلاسک، پیرامید، بطری وCherrypy ، توسعه برنامه‌های تحت وب را ساده و سرعت ببخشید. به همین ترتیب، می‌توانید توسعه برنامه رابط کاربری گرافیکی دسکتاپ را با استفاده از چارچوب‌ها و جعبه‌ابزارهای Python GUI مانندPyQT ، PyJs، PyGUI، Kivy، PyGTKو WxPython تسریع کنید.

۷- توسعه آزمایش محور را بپذیرید

می‌توانید از پایتون برای ایجاد سریع پروتوتایپ‌ها، استفاده کنید. همچنین، می‌توانید برنامه مورد نظرتان را مستقیماً از نمونه اولیه و به سادگی با بازآفرینی کد، بسازید. پایتون حتی با اتخاذ رویکرد توسعه مبتنی بر آزمایش، اجرای همزمان کدنویسی و آزمایش را برای شما آسان‌تر می‌کند. به راحتی می‌توانید قبل از نوشتن کد تست‌های مورد نیاز را بنویسید و از تست‌ها برای ارزیابی مداوم کد برنامه استفاده کنید. از این آزمایش‌ها می‌توان برای بررسی اینکه آیا برنامه بر اساس کد منبع آن نیازهای از پیش تعریف شده را برآورده می‌کند، استفاده کرد.

۶-با پایتون، توسعه نرم افزارهای پیچیده را ساده کنید

پایتون یک زبان برنامه‌نویسی عمومی است. از این رو، می‌توانید از آن برای توسعه برنامه‌های دسکتاپ و وب استفاده کنید. همچنین می‌توانید از پایتون برای توسعه برنامه‌های پیچیده علمی و عددی استفاده کنید. پایتون با ویژگی‌هایی طراحی شده که تجزیه و تحلیل و تجسم داده‌ها را تسهیل میکنند. شما می‌توانید از ویژگی‌های تجزیه و تحلیل داده‌های پایتون برای ایجاد راه حل‌های کلان داده سفارشی بدون صرف زمان و تلاش اضافی استفاده کنید. در عین حال، کتابخانه‌های تجسم داده‌ها وAPI های ارائه شده توسط پایتون به شما کمک می‌کنند تا داده‌ها را به شیوه‌ای جذاب‌تر و موثرتر تجسم و ارائه کنید. بسیاری از توسعه دهندگان پایتون حتی از پایتون برای انجام وظایف پردازش هوش مصنوعی و زبان طبیعی استفاده می‌کنند.

نتیجه‌گیری:

با وجود تمام دلایل منطقی کار با پایتون، پایتون مانند سایر زبان‌های برنامه‌نویسی دارای کاستی‌های خاص خودش هم هست. برخی از ویژگی‌های داخلی ارائه شده توسط سایر زبان‌های برنامه‌نویسی مدرن را ندارد. از این رو، اگر بخواهید یک نرم افزار سفارشی بسازید ممکن است لازم باشد برای تسریع توسعه ی آن، باید از کتابخانه‌ها، ماژول‌ها و چارچوب‌های پایتون استفاده کنید.
همچنین، چندین مطالعه نشان داده است که پایتون نسبت به چندین زبان برنامه‌نویسی پرکاربرد از جمله جاوا و C++ کندتر است. شما باید با ایجاد تغییراتی در کد برنامه یا استفاده از زمان اجرا سفارشی، سرعت برنامه را افزایش دهید. اما این نکته را همیشه در ذهن داشته باشید که می‌توانید از پایتون برای سرعت بخشیدن به توسعه نرم افزار و ساده سازی نگهداری نرم افزار استفاده کنید.

محیط یکپارچه توسعه یا IDE:

ابزارهای محیط توسعه یکپارچه یا همانIDE ها نرم افزارهایی هستند که امکاناتی را برای برنامه نویسان جهت کدنویسی، ساخت و توسعه برنامه‌ها و اپلیکیشن‌های دیگر فراهم می‌کنند. IDE مخفف (Integrated Development Environment) است.
IDEبرای دربرگرفتن تمام امکانات لازم برای پیاده‌سازی همه وظایف برنامه نویسی در قالب یک نرم افزار کاربردی طراحی شده است. یکی از اصلی‌ترین مزایایIDE ها این است که این برنامه کاربردی یک واسط (Interface) مرکزی و هسته‌ای برای همه ابزارهای مورد نیاز توسعه نرم افزارهای خاص به شمار می‌رود. در ساخت یک برنامه، نیاز است بخش‌های بسیاری ایجاد شوند که همراه با یکدیگر کار می‌کنند، از جمله آن‌ها می‌توان به کدها، رابط کاربری ، ساختمان پروژه، محیط پیکربندی پروژه و سایر موارد اشاره کرد. با استفاده از IDE همه این بخش‌ها می‌توانند از طریق یک نرم افزار واحد ایجاد و توسعه داده شوند.

کد نویسی در پایتون:

برای نوشتن کد های پایتون ویرایشگرهای کد مختلفی وجود دارند. اما ما در این دوره آموزشی از Visual Studio Code که یک ویرایشگر سبک و قوی از شرکت ماکروسافت است استفاده می کنیم که به اختصار vs code نامیده می شود..

معروف ترین و اولین برنامه ای که هر برنامه نویس آن را می نویسد (“hello, world”) است. برای نوشتن این کد در پایتون تنها کافی است تا این خط کد را در vs code تایپ کنید:

```
print("hello, world")
```

برای اجرا کردن این کد ما می توانیم از دستور python hello.py در ترمینال استفاده کنیم. در پنجره ترمینال می توانید دستورات را اجرا کنید. برای اجرای این برنامه و کلیک در پنجره ترمینال، باید مکان نما خود را به پایین صفحه حرکت دهید. اکنون می توانید دستور دوم را در پنجره ترمینال تایپ کنید. در کنار علامت دالر،python hello.py را تایپ کنید و کلید enter را روی صفحه کلید خود فشار دهید.

به یاد بیاورید، کامپیوترها واقعاً فقط صفر و یک را می فهمند. بنابراین، وقتی python hello.py را اجرا میکنید، پایتون متنی را که در hello.py ایجاد کرده اید تفسیر می کند و آن را به صفرها و یکهایی که رایانه می تواند بفهمد ترجمه می کند .نتیجه اجرای این برنامه، hello, world است .تبریک میگم! حالا شما اولین برنامه خود را نوشتید.

توابع در پایتون:

تابع در پایتون گروهی از عبارت‌های مرتبط است که یک کار مشخص را انجام می‌دهند. توابع کمک می‌کنند تا برنامه به بخش‌های کوچک‌تر و دانه‌بندی شده‌ای ماژولار Modular شکسته شود. در برنامه نویسی ما با دو اصطلاح expression و statement مواجه هستیم. expression کوچکترین واحد کد است که معمولا نتیجه ای در خروجی دارد. اما statement مجموعه ای از دستورالعمل ها است که به کامپیوتر می گوید چگونه رفتار کند. توابع مجموعه ای از این expression ها و statement ها هستند. هرچه برنامه بزرگ و بزرگ‌تر شود، تابع‌ها به سازمان‌یافته‌تر و قابل مدیریت شدن آن کمک می‌کنند. علاوه بر این، توابع مانع از تکرار برنامه‌نویسی برای یک کار واحد می‌شوند و کد را قابل استفاده مجدد می‌کنند. در واقع توابع، افعال یا اعمالی هستند که کامپیوتر یا زبان کامپیوتر از قبل میداند که چگونه باید آن ها را انجام دهد .در برنامهhello.py شما، تابع print می داند که چگونه در پنجره ترمینال چیزی را چاپ کند .تابع print آرگومان ها را می گیرد. در این مورد، " hello, world"آرگومان هایی هستند که تابع print می گیرد.

انواع توابع:

توابع در پایتون به دو دسته کلی تقسیم می‌شوند:

توابع داخلی یا کتابخانه‌ای:(Built-in Functions)

توابعی که به صورت یک مجموعه بخشی از کتابخانه‌ی استاندارد در پایتون هستند توابع کتابخانه استاندارد پایتون (Python Standard Library)، مجموعه ای از توابع از پیش نوشته شده می باشد. با استفاده از این کتابخانه کار برنامه‌نویسی و ارتقای نرم‌افزارها سریع‌تر و ساده‌تر می‌شود .
کتابخانه استاندارد پایتون بسیار گسترده می باشد یکی از مزیت های مهم دیگر آن حرفه‌ای بودن و سهولت در استفاده از آن است. توابع کتابخانه‌ای (library) را سایر برنامه‌نویسان این توابع را توسعه‌داده و در داخل پکیج‌ها و کتابخانه‌ها منتشر کرده‌اند. برای استفاده از این کتابخانه‌ها برنامه‌نویس ابتدا باید پکیج یا کتابخانه مورد نظر را در اینترنت جستجو کرده و پس از نصب آن به کمک دستور import کتابخانه مورد نظر را در برنامه اضافه کرده و از توابع کتابخانه استفاده می‌کنیم.

برخی از **توابع داخلی پایتون** به شرح زیر هستند:

تابع :()print این تابع مقدار ورودی را چاپ می‌کند.
تابع :()abs این تابع عددی مقدار قدرمطلق ورودی را نمایش می‌دهد.
تابع :()type نوع داده‌ی مشخص شده را به عنوان خروجی نمایش می‌دهد.
تابع :()sum مجموع داده‌های ورودی را محاسبه و نشان می‌دهد.
تابع :()range دنباله ای از اعداد را ایجاد می کند که به صورت پیشفرض از عدد صفر شروع می شود و به ترتیب تا یک عدد قبل از عدد پایانی ادامه می یابد.
تابع :()pow این تابع دو مقدار به عنوان ورودی می‌گیرد و نتیجه را به عنوان یک مقدار توانی نمایش می‌دهد (مقدار اول پایه و مقدار دوم توان است).
تابع :()min در بین اعداد ورودی کوچکترین مقدار را نشان می‌دهد.
تابع :()max در بین اعداد ورودی بزرگترین مقدار را نمایش می‌دهد.
تابع :()complex در ورودی یک عدد حقیقی و یک عدد موهومی می‌گیرد و به عنوان یک عدد مختل نمایش می‌دهد.
تابع :()len طول متن یا همان تعداد کاراکتر آن را نمایش می‌دهد.
تابع :()upper همه‌ی کاراکترهای ورودی را با حروف بزرگ نمایش می‌دهد.
تابع :()lower همه‌ی کاراکترهای ورودی را با حروف کوچک نمایش می‌دهد.
تابع :()bin عدد صحیح را به یک رشته‌ی باینری تبدیل می‌کند.
تابع :()reversed این تابع معکوس یک دنباله را نمایش می‌دهد.
تابع :()zip این تابع، دو دنباله را به یک دیگر پیوند می‌زند. توجه کنید در صورت مساوی نبودن تعداد کاراکترهای دو دنباله قسمتی از دنباله‌ی بزرگتر نادیده گرفته می‌شود.

توابع تعریف شده توسط کاربر:(User-Defined Functions)

توابعی هستند که توسط برنامه‌نویس داخل خود برنامه تعریف می‌شوند و برنامه‌نویس می‌تواند پس از تعریف در بخش‌های مختلف برنامه از این توابع استفاده کند و از نوشتن کد تکراری پرهیز کنند که هم باعث صرفه جویی در زمان شده و هم خوانایی کدها را افزایش می‌دهند.
برای تعریف **تابع در پایتون** از کلیدواژه def استفاده می‌شود که از ابتدای کلمه define انگلیسی به معنای تعریف مشتق شده است. بعد از این کلیدواژه نام تابع مورد نظر خود را انتخاب کرده و داخل پرانتز ورودی هایی که تابع دریافت می‌کند را معرفی می‌کنیم. پس از نام گذاری داخل یک بلاک دستوراتی را که قصد داریم هنگام فراخوانی تابع اتجام شود را می‌نویسیم.
به عنوان مثال برای ایجاد یک تابع برای محاسبه مقدار میانگین ورودی‌ها می‌توان از تابع sum() استفاده کرده و تابع جدید را به صورت تقسیم مقدار تابع sum() به تعداد ورودی‌ها تعریف نمود.

باگ ها:

باگ ها یا اشکالات بخشی طبیعی از کدنویسی هستند. این ها اشتباهاتی هستند که باید آنها را حل کنید! ناامید نشوید! این بخشی از فرآیند تبدیل شدن به یک برنامه نویس عالی است.

تصور کنید در برنامهhello.py ما به طور تصادفی print(“hello, world”) را تایپ کردیم. توجه کنید که ما آخرین (مورد نیاز کامپایلر را ننوشتیم. اگر ما عمداً این اشتباه را انجام دهیم، کامپایلر یک ارور را در پنجره ترمینال خروجی می دهد !اغلب، پیام های ارور شما را از اشتباه تان آگاه می کند و سرنخ هایی را در مورد نحوه رفع آنها در اختیار شما قرار می دهد. با این حال، همیشه قرار نیست که کامپایلر همین اندازه مهربان باشد.

نوشتن اولین برنامه پایتون شما:

ما می توانیم اولین برنامه پایتون شما را شخص ی سازی کنیم. در ویرایشگر متن خود درhello.py می توانیم یک تابع دیگر اضافه کنیم input .تابعی است که یک دستور را به عنوان آرگومان می گیرد. ما می توانیم کد خود را به صورت زیر ویرایش کنیم:

```
input("What's your name? ")
print("hello, world")
```

با این حال، این کار به تنهایی به برنامه شما اجازه نمی دهد تا آنچه را که کاربر شما وارد می کند، خروجی دهد. برای ا ین کار باید متغیرها را به شما معرفی کنیم.

متغیرها:

یک متغیر فقط یک مخزن برای نگهداری یک مقدار در برنامه تان است. متغیرها ساختارهایی هستند که برای ذخیره مقادیر در برنامه‌نویسی مورد استفاده قرار می‌گیرند .**متغیر در پایتون** برای اشاره به مکان حافظه استفاده می‌شود. متغیر پایتون همچنین به عنوان شناسه شناخته می‌شود.
در این بخش علاوه بر یادگیری مفهوم متغیر انواعData type ها نیز معرفی می‌شوند که متغیرها می‌توانند انواع این داده‌ها رو بپذیرند. این Data type ها شامل:
رشته ها یاstring ها به دیتاهایی گفته میشود که به صورت دنباله ای از حروف استفاده میشود.مثل نام ها و عبارت ها و...
اعداد صحیح یاinteger ها که ساده‌ترین مقادیر عددی مورد استفاده در محاسبات هستند.
اعداد اعشاری یا float که برای محاسبات با دقت بالا در برنامه‌نویسی مورد استفاده قرار می‌گیرند.
متغیرهایی که مقادیر آنها به صورت درست یا نادرست (True /False) هستند که به آن Boolean گفته می‌شود.
در تعریف متغیرها دو مورد باید رعایت شود. یکی متغیرها باید با حروف آغاز شوند. همچنین در تعریف متغیرها نباید از کلمات رزرو شده **پایتون** استفاده کرد. مثلا عبارت print در پایتون برای تعریف یک تابع استفاده شده و نباید متغیری به این نام تعریف کرد.

*در پایتون، نیازی به تعیین نوع متغیر نداریم زیرا پایتون یک‌زبان استنباطی است و به‌اندازه کافی باهوش است که نوع متغیر را به دست آورد.

قوانین ایجاد و نام گذاری متغیرها در پایتون به‌صورت زیر است:

- نام متغیر باید با یک حرف یا کاراکتر زیرخط (اندرلاین_) شروع شود.
- نام متغیر نمی‌تواند با عدد شروع شود.
- یک نام متغیر فقط می‌تواند شامل کاراکترهای عددی و آندرلاین(A-z) (_) ، ۰-۹ و _ باشد.
- نام متغیرها به حروف کوچک و کوچک حساس هستند ALI و ali متفاوت هستند.
- از کلمات رزرو شده در پایتون (کلمات کلیدی) نمی‌توان برای نام‌گذاری متغیر استفاده کرد.
- نام شناسه نباید حاوی هیچ فضای سفید (space)یا کاراکتر خاصی مانند!، @، #، ^، &، *باشد.

نمونه نام‌گذاری متغیر در پایتون:

نمونه‌هایی از شناسه‌های معتبرa123 ،. n_، n_9و غیره.

نمونه‌هایی از شناسه‌های نامعتبر: ۱a، n٪ ۴، n 9و غیره.

نام متغیرها می‌تواند گروهی از حروف و ارقام باشد، اما آن‌ها باید با یک حرف یا آندرلاین شروع شوند. توصیه می‌شود برای نام متغیر از حروف کوچک استفاده کنید .

اعلام متغیر و تخصیص ارزش

پایتون ما را ملزم به اعلام متغیر قبل از استفاده از آن در برنامه نمی‌کند. این به ما امکان می‌دهد یک متغیر در زمان موردنیاز ایجاد کنیم. ما نیازی به اعلام صریح **متغیر در پایتون** نداریم. وقتی هر مقداری را به متغیر اختصاص می‌دهیم، آن متغیر به‌طور خودکار اعلام می‌شود.

برای تخصیص مقدار به یک متغیر از عملگر Equal (=) استفاده می‌شود .

وقتی ما یک متغیر را اعلان می‌کنیم، لازم است درک کنیم که مفسر پایتون چگونه کار می‌کند. فرایند ایجاد متغیرها تا حدودی با بسیاری از زبان‌های برنامه‌نویسی متفاوت است. پایتون زبان برنامه‌نویسی بسیار شیء گرا است. به همین دلیل است که هر مورد داده به نوع خاصی از کلاس تعلق دارد. به مثال زیر توجه کنید.

در پایتون، متغیرها یک نام نمادین هستند که مرجع یا اشاره‌گر یک شیء هستند. متغیرها برای نشان دادن اشیاء با آن نام استفاده می‌شوند.

در برنامه تان می توانید متغیر خود را ایجاد کنید تا ورودی خوانده شود و داخل متغیرتان قرار گیرد:

```
name = input(“What’s your name? “)
print(“hello, world”)
```

توجه داشته باشید که علامت برابر = در میان کدname = input(whats your name) نقش ویژه ای در برنامه نویسی دارد. این علامت مساوی آنچه در سمت راست قرار دارد را به آنچه در سمت چپ است اختصاص می دهد. بنابراین، مقدار بازگردانده شده توسط input(whats your name) درون name قرار می گیرد.

اگر کد خود را به صورت زیر ویرایش کنید، متوجه یک خطا می شوید:

```
name = input("What's your name? ")
print("hello, name")
```

برنامه بدون در نظر گرفتن آنچه که کاربر تایپ کرده است، hello, name را در پنجره ترمینال بر می گرداند.

برای ویرایش کدتان، می توانید به صورت زیر تایپ کنید:

```
name = input(“What’s your name? “)
print(“hello,”)
print(name)
```

نتیجه این کد در پنجره ترمینال به صورت زیر خواهد بود:

```
What’s your name? David
hello,
David
```

داریم به نتیجه دلخواهمان نزدیک می شویم!

کامنت ها:

کامنت ها راه کار برنامه نویسان برای پیگیری کارهایی ست که در برنامه های خود انجام می دهند و حتی می توانند دیگران را در مورد اهداف خود برای یک بلوک کد آگاه کنند. به طور خلاصه، کامنت ها یادداشت هایی برای خود و دیگران هستند که کد شما را توضیح می دهند !می توانید کامنت هایی را به برنامه خود اضافه کنید تا بتوانید نشان دهید برنامه شما در حال انجام چه کاری است. می توانید کد خود را به صورت زیر ویرایش کنید:

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,")
print(name)
```

کامنت ها همچنین می توانند به عنوان لیست کارهایی که باید انجام دهید برای شما باشند.

شبه کد:

شبه کد نوع مهم ی از کامنت است که به نوع خاص ی از فهرست کارها تبدیل میشود، بهخصوص زمانی که نم یدانید چگونه یک کار کدنویسی را انجام ده ید. به عنوان مثال، ممکن است کد خود را به صورت زیر ویرایش کنید:

```
# Ask the user for their name
name = input("What's your name? ")
# Print hello
print("hello,")
# Print the name inputted
print(name)
```

بهبود بیشتر اولین برنامه پایتون شما:

ما می توانیم کد خود را به صورت زیر ویرایش کنیم:

```
# Ask the user for their name
name = input("What's your name? ")
# Print hello and the inputted name
print("hello, " + name)
```

به نظر می رسد که برخی از توابع آرگومان ها ی زیادی را می گیرند .ما میتوانیم از یک کاما برای ارسال آرگومانهای متعدد با ویرایش کد خود به صورت زیر استفاده کنیم:

اگر David را تایپ کنیم، خروجی در ترمینال hello ,David خواهد بود. اکنون به هدف خود رسیدیم!

رشته ها و پارامتر ها:

یک string که در پایتون با str شناخته می شود، دنباله ای از متن است .کمی در کد خود به عقب برگردیم، مشکل کد زیر چاپ خروجی در دو خط متفاوت بود:

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,")
print(name)
```

توابع آرگومان هایی را می گیرند که بر رفتار آنها تأثیر می گذارند. اگر به فایل راهنمای print نگاه کنید، متوجه خواهید شد که می توانیم چیزهای زیادی در مورد آرگومان هایی که تابع print می گیرد بیاموزیم .با نگاهی به فایل راهنما، خواهید دید که تابع print به طور خودکار شامل یک قطعه کد end = ‘/n’می شود .این/n نشان میدهد که تابع print به طور خودکار هنگام اجرا یک enter ایجاد میکند. تابع print آرگومانی به نام end می گیرد و به طور پیش فرض به خط بعدی می رود .با این حال، از نظر فنی می توانیم کاری کنیم که خط جدیدی ایجاد نشود!

ما می توانیم کد خود را به صورت زیر تغییر دهیم:

```
# Ask the user for their name
name = input("What's your name? ")
print("hello,", end="")
print(name)
```

با نوشتن end = “” مقدار پیشفرض end را روی آن م ینویسیم به طور ی که هرگز بعد از اولین دستور چاپ خط جدیدی ایجاد نمی کند. با وارد کردن نام“David” ، خروجی در پنجره ترمینال hello, David خواهد بود.

بنابراین، پارامترها آرگومان هایی هستند که می توانند توسط یک تابع گرفته شوند .شما می توانید در فایل راهنمای پایتون اطالعات بیشتری درباره تابع print کسب کنید.

یک مشکل کوچک با علامتquotation :

توجه داشته باشید که در string ها افزودن عالمت quotation (“) چالش برانگیز است .مثال print(“hello,”frinds””) کار نمی کند و کامپایلر ارور می دهد .به طور کلی، رویکردی برای رفع این مشکل وجود دارد ، می توانید به سادگی quotation ها را به علامت (‘) quotation single تغییر دهید.

قالب بندی رشته ها:

احتمالا زیباترین راه برای استفاده از رشته ها به شرح زیر است:

```
# Ask the user for their name
name = input("What's your name? ")
print(f"hello, {name}")
```

به f در print(f"hello,{name}) دقت کنید. این f یک نشانگر ویژه پایتون برای برخورد با این رشته به روشی خاص است، متفاوت از رویکردهای قبلی که در این جلسه توضیح داده ایم. انتظار می رود که در این دوره به طور مکرر از این سبک رشته ها استفاده کنید.

F-strings، که به آنها Formatted String Literals نیز گفته می‌شود، یک روش نوین برای فرمت کردن رشته‌ها در پایتون است. این ویژگی برای اولین بار در نسخه ۳.۶ پایتون ارائه شد. با استفاده از f-strings ، می‌توانید به راحتی مقادیر متغیرها، اعداد و عبارت‌های دیگر را درون یک رشته قرار دهید. برای استفاده از f-strings ، شما باید قبل از رشته خود حرف `f` قرار دهید. سپس، می‌توانید مقادیر متغیرها یا عبارت‌ها را درون `{ }` قرار دهید. این مقادیر در زمان اجرا با مقدار واقعی آنها جایگزین می‌شوند.

مطالب بیشتر در مورد رشته ها:

هرگز نباید انتظار داشته باشید که کاربر شما همانطور که دلخواهتان است همکاری کند. بنابراین، باید اطمینان حاصل کنید که ورودی کاربر تصحیح یا بررسی شده است .در رشته ها قابلیت حذف فضا ی خالی از یک رشته وجود دارد .با استفاده از متد strip بر روی name به شکل name = name.strip() ، تمام فضاهای خالی سمت چپ و راست ورودی کاربر حذف می شود. می توانید کد خود را به صورت زیر تغییر دهید:

```
# Ask the user for their name
name = input("What's your name? ")
# Remove whitespace from the str
name = name.strip()
# Print the output
print(f"hello, {name}")
```

با اجرای مجدد این برنامه، صرف نظر از اینکه چند فاصله قبل یا بعد از نام تایپ می کنید، تمام فضای خالی حذف می شود. همچنین با استفاده از این متد می توانید به برنامه بگویید که چه کارکتری از برنامه حذف شود.

متد title() یکی دیگر از متدهای کلاس String در پایتون است. با استفاده از این متد حروف اول کلیه کلمات موجود در متن بزرگ شده و در حالت title قرار می گیرند.

```
# Ask the user for their name
name = input("What's your name? ")
# Remove whitespace from the str
name = name.strip()
# Capitalize the first letter of each word
name = name.title()
# Print the output
print(f"hello, {name}")
```

در این مرحله، ممکن است از تایپ مکرر python در پنجره ترمینال بسیار خسته شده باشید .اگر فلش رو به بالا صفحه کلیدتان را فشار دهید، دستورات اخیری که در ترمینال نوشته بودید نمایش داده می شود .توجه داشته باشید که می توانید کد خود را تغییر ده ید تا کارآمدتر شود:

```
# Ask the user for their name
name = input("What's your name? ")
# Remove whitespace from the str and capitalize the first letter of
each word
name = name.strip().title()
# Print the output
print(f"hello, {name}")
```

این همان نتیجه کد قبلی شما را ایجاد می کند .

حتی می توانیم بیشتر پیش برویم!

Int یا integers :

در پایتون به یک عدد صحیح int گفته می شود.

در دنیای ریاضیات با عملگرهای +، -، *، / و ٪ آشنا هستیم. آخرین اپراتور ٪ یا اپراتور modulo ممکن است برای شما چندان آشنا نباشد .برای اجرای کد پایتون لازم نیست از پنجره ویرایشگر متن در کامپایلر خود استفاده کنید. در پایین ترمینال خود، می توانید پایتون را به تنهایی اجرا کنید. در پنجره ترمینال >>>به شما نمایش داده می شود. شما می توانید در لحظه و به شکل تعاملی کدها را اجرا کنید. می توانید 1+1را تایپ کنید و محاسبه انجام می شود. این حالت معمولا در طول این دوره استفاده نمی شود .مجددا Code VS را باز کنید وcode calculator.py را در پنجره ترمینال تایپ کنید. با این کار یک فایل جدید ایجاد می کنیم که در آن ماشین حساب خود را خواهیم ساخت .ابتدا می توانیم چند متغیر را تعریف کنیم.

```
x = 1
y = 2
z = x + y
print(z)
```

طبیعتا اگر python calculator.py را اجرا کنیم، در پنجره ترمینال عدد ۳ را خواهیم دید. با استفاده از تابع input می توانیم کد خود را تعاملی تر کنیم:

```
x = input("What's x? ")
y = input("What's y? ")
z = x + y
print(z)
```

با اجرای این کد می بینیم که جواب به غلط ۱۲ اعلام می شود. چرا این اتفاق می افتد؟

قبلا دیدیم که چگونه علامت + دو رشته را به هم متصل می کند. از آنجا که ورودی شما از صفحه کلید کامپیوترتان به صورت متن وارد کامپایلر می شود، به عنوان یک رشته در نظر گرفته می شود. بنابراین، باید این ورودی را از یک رشته به یک عدد صحیح تبدیل کنیم. ما می توانیم این کار را به صورت زیر انجام دهیم:

```
x = input("What's x? ")
y = input("What's y? ")
z = int(x) + int(y)
print(z)
```

اکنون نتیجه حاصل صحیح است. این کار استفاده از int(x) ,casting نامیده می شود که به طور موقت یک نوع متغیر در اینجا یک رشته(را به نوع دیگری از متغیر)در اینجا عدد صحیح تبدیل می کند .ما می توانیم برنامه خود را به شرح زیر بهتر کنیم:

```
x = int(input("What's x? "))
y = int(input("What's y? "))
print(x + y)
```

این موضوع نشان می دهد که می توانید توابع را روی توابع اجرا کنید. درونی ترین تابع ابتدا اجرا می شود و سپس تابع بیرونی اجرا می شود. یعنی ابتدا تابع input و سپس تابع int اجرا می شود .شما می توانید در فایل راهنمای پایتون در مورد int اطلاعات بیشتری کسب کنید.

هنگام تصمیم گیری در مورد رویکرد خود برای یک کار کدنویسی، به یاد داشته باشید که می توان رویکردهای بسیاری را برای یک مشکل، با استدلال منطقی ارائه داد .صرف نظر از اینکه چه رو یکردی برای یک کار برنامه نویسی دارید، به یاد داشته باشید که کد شما باید خوانا باشد. شما باید از کامنت ها استفاده کنید تا به خود و دیگران سرنخ هایی در مورد کاری که کدتان انجام می دهد بدهید. علاوه بر این، باید کد را به گونه ای ایجاد کنید که قابل خواندن باشد.

:float

مقدار float یک عدد حقیقی است که دارای یک نقطه اعشار مانند ۰.۵۲ است .می توانید کد خود را برای پشتیبانی از floats به صورت زیر تغییر دهید:

```
x = float(input("What's x? "))
y = float(input("What's y? "))
print(x + y)
```

این تغییر به کاربر شما اجازه می دهد تا ۱.۲ و ۳.۴ را وارد کند تا در مجموع ۴.۶ را دریافت کند .با این حال، بیایید تصور کنیم که می خواهید مجموع را به نزدیکترین عدد صحیح گرد کنید.

اگر به فایل راهنمای پایتون در مورد round نگاه کنید، خواهید دید که آرگومان های موجود round(number[n, ndigits]) خواهند بود. براکت ها نشان میدهند که برنامه نویس می تواند مقادیری اختیاری را تعیین کند. بنابراین، می توانید برای گرد کردن یک رقم به نزدیکترین عدد صحیح آن، از round(n) استفاده کنید .می توانید به صورت زیر کد خود را بنویسید:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))
# Create a rounded result
z = round(x + y)
# Print the result
print(z)
```

خروجی ما به نزدیک ترین عدد صحیح گرد شده است .اگر بخواه یم اعداد خروجی را قالب گذاری کنیم چه؟ برای مثال، به جای دیدن ۱۰۰۰، ممکن است بخواه ید ۱,۰۰۰ را ببینید. می توانید کد خود را به صورت زیر تغییر دهید:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))
# Create a rounded result
z = round(x + y)
# Print the formatted result
print(f"{z:,}")
```

با اینکه کمی عجیب است اما print(f"{z:,}") سناریویی را ایجاد می کند که در آن z شامل ویرگول خواهد شد که در آن نتیجه می تواند ۱,۰۰۰ یا ۲,۵۰۰ باشد.

اطلاعات بیشتر درباره floats :

چگونه می توانیم مقادیر float را گرد کنیم؟ ابتدا کد خود را به صورت زیر تغییر دهید:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))
# Calculate the result
z = x / y
# Print the result
print(z)
```

هنگام وارد کردن ۲ به عنوان x و ۳ به عنوانy ، نتیجهz ، ۰,٦٦٦٦٦٦٦٦٦٦ است که ظاهراً همانطور که انتظار داریم تا بی نهایت ادامه می یابد .بیایید تصور کنیم که می خواهیم این را گرد کنیم، می توانیم کد خود را به صورت زیر تغییر دهیم:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))
# Calculate the result and round
z = round(x / y, 2)
# Print the result
print(z)
```

همانطور که انتظار داریم، این کد نتیجه را به نزدیکترین عدد با دو رقم اعشار گرد می کند .همچنین می توانیم از fstring برای فرمت خروجی به صورت زیر استفاده کنیم:

```
# Get the user's input
x = float(input("What's x? "))
y = float(input("What's y? "))
# Calculate the result
z = x / y
# Print the result
print(f"{z:.2f}")
```

fstring، استراتژی گرد کردنی مانند کد قبلی ما دارد .شما می توانید در فایل راهنمای پایتون در مورد floats اطلاعات بیشتری کسب کنید.

def:

بسیار جالب می شود اگر بتوانیم توابع خودمان را ایجاد کنیم.

بیایید کد نهایی hello.py را با تایپ code hello.py در پنجره ترمینال برگردانیم .کد شروع شما باید به صورت زیر باشد:

```
# Ask the user for their name, remove whitespace from the str and capitalize the first letter
of each word
name = input("What's your name? ").strip().title()
# Print the output
print(f"hello, {name}")
```

ما می توانیم با ایجاد یک تابع خاص که به ما" hello" می گوید کد خود را بهتر کنیم!

با پاک کردن تمام کدها ی خود در ویرایشگر متن، بیایید از ابتدا شروع کنیم:

```
name = input("What's your name? ")
hello()
print(name)
```

با اجرای این کد، کامپایلر شما با خطا مواجه می شود. زیرا هیچ تابع تعریف شده ای برای helloوجود ندارد.

ما می توانیم تابع خود به نام hello را به صورت زیر ایجاد کنیم:

```
def hello():
 print("hello")
name = input("What's your name? ")
hello()
print(name)
```

توجه داشته باشید که کد زیر print("hello") فاصله ای از ابتدای خط دارد. زبان پایتون بر اساس این قانون عمل می کند. از این فاصله از ابتدای خط برای درک این استفاده می شود که کدام بخش مربوط به تابع فوق است. بنابراین، همه چیز در تابع helloباید از ابتدای خط فاصله داشته باشد. وقتی کدی فاصله نداشته باشد، طوری با آن رفتار می شود که انگار داخل تابع hello نیست. با اجرای python hello.py در پنجره ترمینال، خواهید دید که خروجی شما دقیقاً آنطور که می خواهید نیست.

ما می توانیم کد خود را بیشتر بهبود ببخشیم:

```
# Create our own function
def hello(to):
 print("hello,", to)
# Output using our own function
name = input("What's your name? ")
hello(name)
```

در اینجا، در خطوط اول، شما در حال ساخت تابع hello خود هستید . این بار به کامپایلر می گویید که این تابع یک پارامتر واحد را می گیرد: یک متغیر فراخوانی شده به نام .to بنابراین، هنگامی که شما hello(name) را صدا می زنید، کامپیوتر name را به تابع helloبه عنوان to ارسال می کند. به این ترتیب ما مقادیر را به توابع منتقل می کنیم .بسیار مفید! با اجرای hello.py pythonدر پنجره ترمینال، خواهید دید که خروجی به ایده آل ما که قابل ارائه شد بسیار نزدیکتر است.

ما می توانیم کد خود را تغییر دهیم تا یک مقدار پیش فرض به hello اضافه کنیم:

```
# Create our own function
def hello(to="world"):
 print("hello,", to)
# Output using our own function
name = input("What's your name? ")
hello(name)
# Output without passing the expected arguments
hello()
```

کد خود را خودتان تست کنید. توجه داشته باشید که hello اول همانطور که انتظار دارید رفتار می کند و hello دوم که مقداری به آن وارد نمی شود، به طور پیش فرض خروجی hello, worldرا نشان می دهد .ما مجبور نیستیم در ابتدای برنامه تابع خود را داشته باشیم. میتوانیم آن را به پایین منتقل کنیم، اما باید به کامپایلر بگوییم که یک تابع main و یک تابع hello جداگانه داریم.

```
def main():
 # Output using our own function
 name = input("What's your name? ")
 hello(name)
 # Output without passing the expected arguments
 hello()
# Create our own function
def hello(to="world"):
 print("hello,", to)
```

با این حال، این به تنهایی یک نوع خطا ایجاد می کند. اگر python py.hello را اجرا کنیم هیچ اتفاقی نمی افتد! دلیل این امر این است که هیچ چیز در این کد در واقع تابع main را فراخوانی نمی کند .تغییر بسیار کوچک زیر تابع main را فراخوانی م ی کند و برنامه ما را به کار می اندازد:

```
def main():
 # Output using our own function
 name = input("What's your name? ")
 hello(name)
 # Output without passing the expected arguments
 hello()
# Create our own function
def hello(to="world"):
 print("hello,", to)
main()
```

بازگرداندن مقادیر:

شما می توانید سناریوها ی زیادی را تصور کنید که در آنها نه تنها می خواهید یک تابع یک عمل را انجام دهد، بلکه می خواهید یک مقدار را به تابع اصلی برگرداند. به عنوان مثال، به جای چاپ محاسباتx + y ، ممکن است تابعی بخواهید که این مقدار را محاسبه کرده و به قسمت دیگری از برنامه شما برگرداند. این "بازگشت" یک مقدار را value return می نامیم .با تایپ code py.calculatorبه کد calculator .py برگردید. همه کدها را در آنجا پاک کنید. کد را به صورت زیر تغییر دهید:

```
def main():
 x = int(input("What's x? "))
 print("x squared is", square(x))
def square(n):
 return n * n
main()
```

xبه square منتقل می شود. سپس، حاصل x * x به تابع main باز گردانده می شود.