آموزش رایگان پایتون (python) - آشنایی با مفهوم توابع و تقدم عملگرها در پایتون

در شماره گذشته آموزش پایتون با عملگرها آشنا شدیم. در این مقاله با مبحث تقدم عملگرها و توابع در پایتون آشنا خواهیم شد.

برای مطالعه بخش دهم آموزش رایگان پایتون اینجا کلیک کنید

زمانی که در یک زبان برنامه‌نویسی همچون پایتون دستوری را می‌نویسید که تنها یک عملگر دارد، نحوه اجرای فرمان و عملگر مشخص بوده و خروجی که انتظار دارید را دریافت می‌کنید. اما زمانی که در حال کار با چند عملگر هستید این مسئله فرق کرده و خروجی یک فرمان ممکن است متفاوت از آن چیزی باشد که انتظارش را دارید. این تفاوت به مسئله‌ای باز می‌گردد که دنیای برنامه‌نویسی به آن حق تقدم عملگرها می‌گوید. به‌طور مثال شما روی کاغذ ممکن است نمادهای ضرب، تقسیم، جمع و پرانتزها را نوشته و بدون توجه به آن‌ها محاسبات را انجام دهید، اما کامپیوتر برای هر یک از نمادهای ریاضی و عملگرها ارزش خاصی قائل بوده و ترتیب اجرای محاسبات بر مبنای عملگرهایی که بالاترین اولویت را دارند اجرا می‌شود. به‌طور مثال، مهم است که بدانید خروجی یک عبارت همچون 1+2*3 آیا برابر با 7 یا برابر با 9 خواهد بود. با توجه به تقدم عملگرها خروجی این عبارت برابر با مقدار 7 خواهد بود مگر آن‌که شما با راهکارهایی همچون به‌کارگیری پرانتزها حق تقدم را تغییر دهید. در این حالت اگر عبارت فوق را به صورت (1+2) * 3 بنویسید خروجی 9 را دریافت خواهید کرد، زیرا پرانتزها در **پایتون** بالاترین حق اولویت را دارند. جدول زیر حق تقدم عملگرهای مختلف در **پایتون** را نشان می‌دهد.

حق تقدم عملگرها در **پایتون**	
عملگر	تشریح
()	شما از پرانتزها برای گروه‌بندی دستورات و تغییر حق تقدم پیش‌فرض عملگرها استفاده می‌کنید. در زمان محاسبات ریاضی در بیشتر موارد مجبور هستید برای آن‌که نتیجه یک محاسبه مطابق با آن چیزی شود که شما انتظارش را دارید از پرانتزها استفاده کنید
**	عملگر به توان رساندن یک مقدار
- + ~	عملگرهای یگانه که تنها یک عبارت را متغیر را دریافت می کنند
// % / *	عملگرهای ضرب، تقسیم، باقی‌مانده، تقسیم گرده شده پایین
- +	عملگرهای جمع و تفریق

>> <<	عملگرهای شیفت به راست و چپ
&	عملگر AND
| ^	عملگرهای بیتی
=< < > =>	عملگرهای مقایسه‌ای
=< < > =>	عملگرهای برابری
=* =+ =- =// =/ =% = =**	عمگرهای انتصابی
Is is not	عملگرهای تعیین هویت
In not in	عملگرهای عضویت
not or and	عملگرهای منطقی

ایجاد و به‌کارگیری توابع

برای آن‌که اطلاعات به شکل درستی مدیریت شوند، شما باید از راهکارهایی برای سازمان‌دهی اطلاعات استفاده کنید. در دنیای برنامه‌نویسی هر خط کد کار خاصی انجام داده و این مجموعه کدها هستند که یک کار خاص را انجام می‌دهند. در بیشتر اوقات داده‌های ورودی به سمت برنامه شما فرمت‌های مختلفی دارند که همین مسئله شما را مجبور می‌کند تا کدهایی برای پردازش داده‌های مختلف بنویسید. اگر از راهکار درستی برای پردازش داده‌های مختلف استفاده نکنید، تعداد خطوط برنامه شما یک مرتبه افزایش پیدا کرده، فهم کدها مشکل شده و فرآیند اشکال‌زدایی پیچیده می‌شود. برای حل این مشکل زبان‌های برنامه‌نویسی توابع را پیشنهاد دادند. توابع ابزارهایی هستند که به شما اجازه می‌دهند کدهای خاصی را یک مرتبه نوشته و در بخش‌های مختلف برنامه از آن‌ها استفاده کنید. این راهکار به میزان قابل توجهی حجم کدهای یک برنامه را کاهش داده و در مقابل خوانایی را افزایش می‌دهد.

بسته‌بندی کدها

توابع روشی برای بسته‌بندی یا به عبارت دقیق‌تر تجمیع کدها در اختیارتان قرار می‌دهند تا به ساده‌ترین شکل به کدها دسترسی داشته و آن‌ها را پیدا کنید. در حقیقت توابع نقش یک سازمان‌دهنده کدها را دارند. تعاریف و برداشت‌های مختلفی از این‌که چرا توابع مهم هستند وجود دارد، اما اگر از همه این تعاریف صرفنظر کنیم به این واقعیت می‌رسیم که توابع یک مکانیسم عالی یکپارچه‌سازی کدها در اختیار ما قرار می‌دهند.

قابلیت استفاده مجدد از کدها

در دنیای واقعی شما به کمد لباس‌ها می‌روید، شلوار و پیراهن را بیرون آورده، آن‌را پوشیده و پس از شست‌وشو در کمد لباس‌ها قرار می‌دهید تا دومرتبه از آن‌ها استفاد کنید. به عبارت دقیق‌تر لباس‌ها را پس از یکبار مصرف در سطل زباله قرار نمی‌دهید. توابع نیز یک چنین قابلیتی در اختیارتان قرار می‌دهند تا از کدهای خود دومرتبه استفاده کنید. هیچ برنامه‌نویسی دوست ندارد کدهای یکنواخت و خسته‌کننده‌ای را بارها و بارها بنویسد. زمانی که تابعی را ایجاد می‌کنید، در حقیقت در حال بسته‌بندی کدهایی هستند که قرار است یک وظیفه یا کار مشخصی را انجام دهند. پس از بسته‌بندی کدها تنها کاری که باید انجام دهید این است که به کامپیوتر بگویید چگونه باید بسته ساخته شده را اجرا کند. کامپیوتر دستوراتی که درون یک تابع قرار داده‌اید را خط به خط اجرا می‌کند. پس از آن‌که کدهای خود را درون یک تابع قرار دادید در مرحله بعد باید به طریقی تابع خود را اجرا کنید. به این فرآیند فراخوانی (calling) گفته

می‌شود. در زمان فراخوانی یک تابع باید اطلاعاتی که تابع بر مبنای آن کار می‌کند را در اختیارش قرار دهید تا بتواند کار مدنظر شما را انجام دهد. در غیر این صورت برنامه با یک پیغام خطا ناتمام مانده یا خروجی اشتباهی ارائه می‌کند. قابلیت استفاده مجدد از کدها و توابع یکی از ارکان اصلی برنامه‌های کاربردی هستند که مزایای زیر را به همراه دارد:

- زمانی توسعه نرم‌افزارها را کوتاه کرده
- خطاهای برنامه‌نویس را کم کرده
- قابلیت اطمینان یک برنامه را افزایش داده
- به سایر اعضا یک تیم اجازه می‌دهند از کدهای نوشته شده استفاده کنند
- فهم کدها ساده‌تر کرده
- عملکرد برنامه را بهبود می‌بخشند.

چگونه یک تابع را تعریف کنیم؟

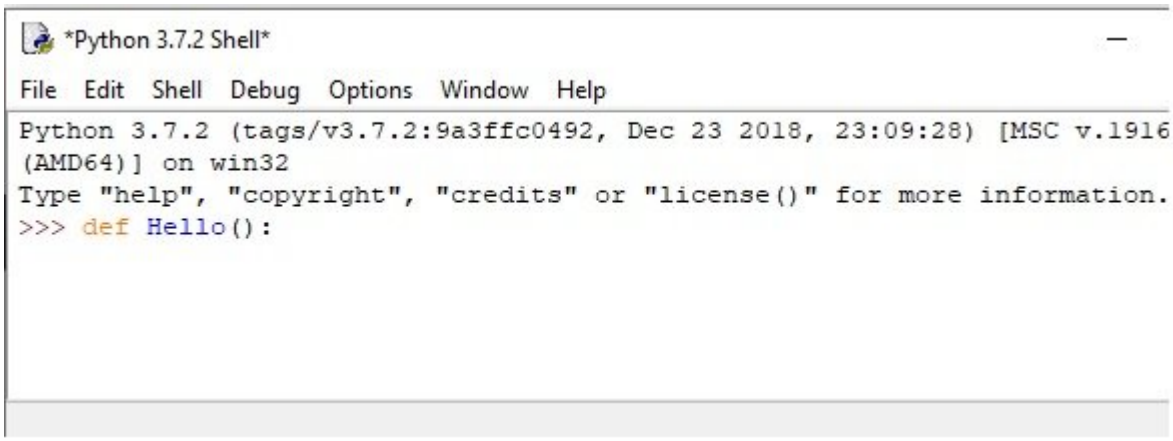
ساخت یک تابع در پایتون به سادگی انجام می‌شود. برای ساخت یک تابع در پایتون باید مراحل زیر را انجام دهید:

1. IDLE را در کادر جست‌وجوی ویندوز 10 وارد کرده و کلید اینتر را فشار دهید.

2. عبارت زیر را تایپ کرده و کلید اینتر را فشار دهید.

```
Def Hello():
```

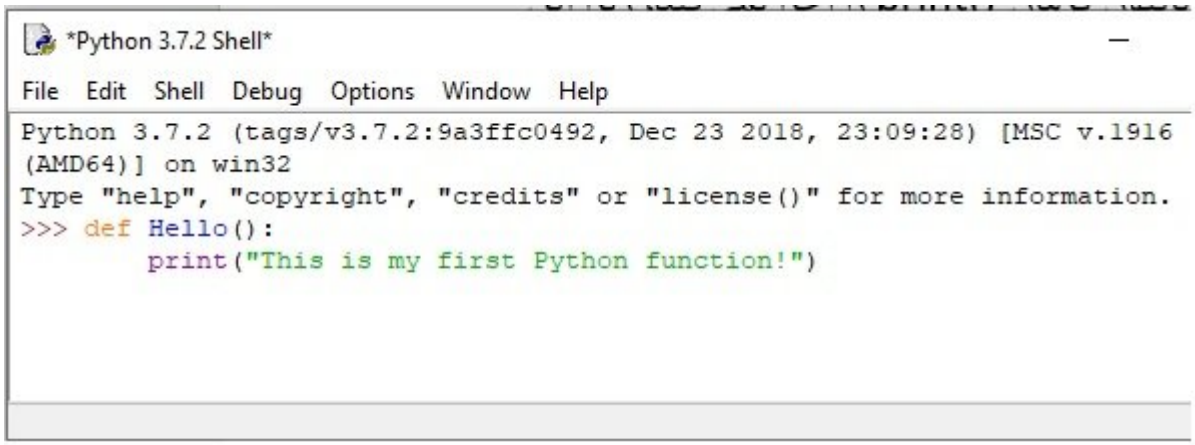
این فرمان به پایتون می‌گوید که تابعی به نام Hello تعریف شده است. پرانتزهایی که مقابل واژه Hello قرار گرفته‌اند از آن جهت حائز اهمیت هستند که برای ارائه ملزومات موردنیاز پایتون استفاده می‌شوند. به‌طور مثال همان‌گونه که در آینده مشاهده خواهید کرد شما در برخی موارد مجبور هستید مقادیری را به عنوان ورودی در اختیار یک تابع قرار دهید تا بتواند کارهای از پیش تعیین شده را انجام دهد. علامت دو نقطه : به پایتون می‌گوید که شما تعریف تابع را کامل کرده‌اید. دقت کنید که با فشار کلید اینتر اشاره‌گر ماوس در ابتدای نام تابع قرار می‌گیرد. این تورفتگی به شما اعلام می‌دارد که کدهای متعلق به تابع را باید در این مکان وارد کنید.



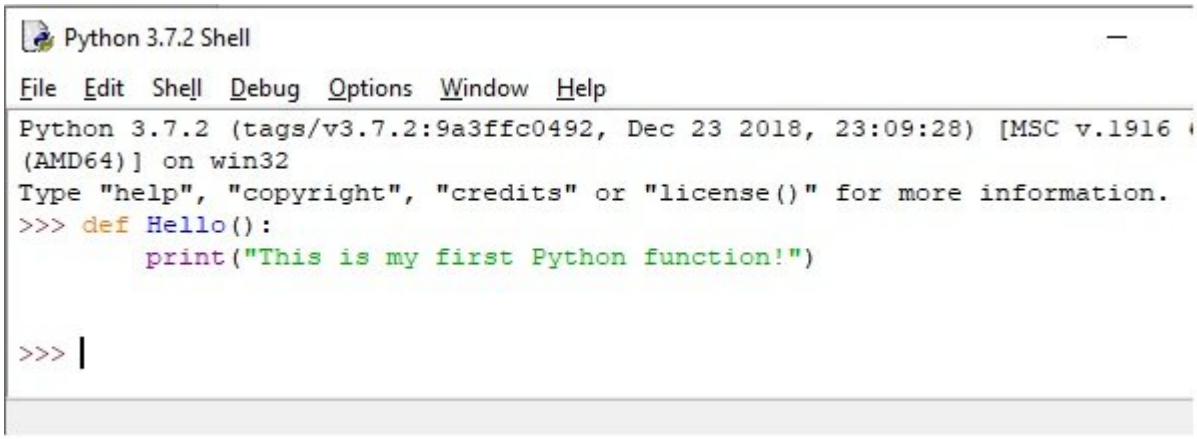
3. فرمان زیر را در مکانی که اشاره‌گر قرار دارد تایپ کرده و کلید اینتر را فشار دهید.

```
print(“This is my first Python function!”)
```

در این قطعه کد، دو نکته مهم حائز اهمیت است. اول آن‌که پس از تایپ فرمان فوق و فشار کلید اینتر، اشاره‌گر هنوز هم در ابتدای مکانی که نام تابع نوشته شده است قرار می‌گیرد، زیرا IDLE منتظر است تا شما دستورات بعدی را درون تابع درج کنید. دوم آن‌که پایتون فرمان print() را اجرا نمی‌کند، زیرا این فرمان جزیی از تابع است و به عنوان یک دستور جداگانه در نظر گرفته نمی‌شود.



4. کلید اینتر را فشار دهید. فشار دومرتبه کلید اینتر اشاره‌گر را به ابتدای خط بعد جایی که کاراکترهای <<< قرار دارند انتقال می‌دهد. این‌کار به معنای آن است که فرآیند تعریف تابع تمام شده است.

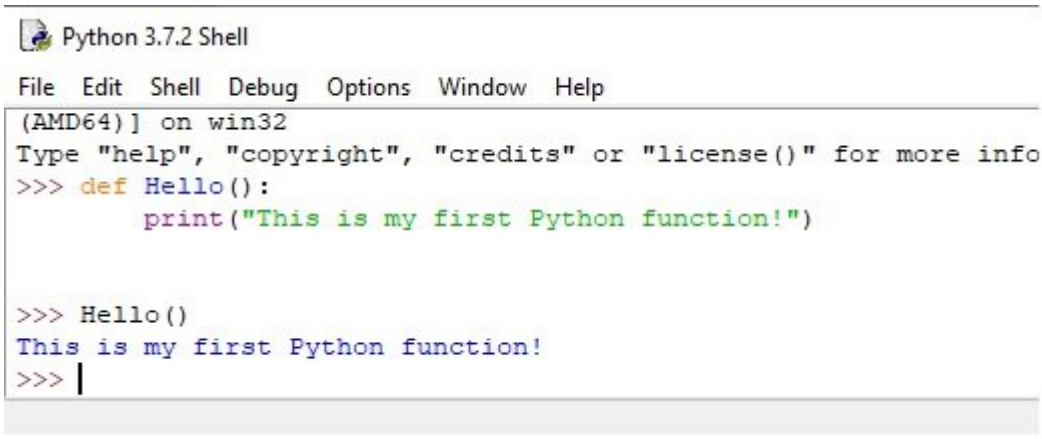


درست است که شما یک تابع ساده را تعریف کردید، اما این ترکیب نحوی در زمان تعریف هر تابعی در پایتون استفاده می‌شود. تعریف یک تابع به شرح زیر انجام می‌شود:

1. درج کلمه کلیدیdef
2. تعریف نام تابع (هر نام دلخواهی)
3. پرانتزها (همراه با پارامترهای اختیاری)
4. دو نقطه
5. مجموعه دستورات که بدنه تابع را شکل می‌دهند.
6. فشار کلید اینتر که به پایتون می‌گوید تعریف دستورات درون یک تابع به اتمام رسیده‌اند.

دسترسی به توابع و فراخوانی آن‌ها

پس از آن‌که در یک برنامه کاربردی توابع را تعریف کردید، در مکان‌های مختلفی باید توابع ایجاد شده را فراخوانی کنید. در پاراگرف قبل شما تابعی به نام Hello ایجاد کردید. برای فراخوانی و اجرای این تابع کافی است نام تابع را همراه با پرانتزها تایپ کرده و کلید اینتر را فشار دهید.



کاری که شما انجام دادید، فراخوانی تابع Hello است. این فراخوانی باعث می‌شود دستورات درون تابع اجرا شوند.

ارسال اطلاعات به توابع

تابعی که در پاراگراف قبل ایجاد کردید از آن جهت مطلوب است که شما برای فراخوانی آن نیازی به انجام هیچ کار خاصی ندارید و تنها نام تابع را برای فراخوانی آن می‌نویسید. با این حال، این مدل تعریف توابع محدودیت‌هایی به همراه دارد، زیرا شما نمی‌توانید هیچ‌گونه اطلاعاتی به تابع ارسال کنید تا درون تابع پردازش شده و خروجی در اختیارتان قرار گیرد. در حالت ایده‌آل توابع باید انعطاف‌پذیر باشند و به شما اجازه دهند کارهای مختلفی را انجام دهید. اگر این‌گونه نباشد شما مجبور هستید توابع متعددی را بنویسید که هریک داده‌های مختلفی را به عنوان ورودی دریافت کنند. برای حل این مشکل و منعطف کردن توابع ما از پارامترها یا به عبارت دقیق‌تر آرگومان‌ها (اختیار) استفاده می‌کنیم تا بتوانیم اطلاعاتی را به عنوان ورودی در اختیار توابع قرار دهیم.

آرگومان‌ها چه هستند؟

واژه آرگومان به معنای آن نیست که قرار است کارهای پیچیده و سختی انجام دهید. آرگومان‌ها با هدف ارائه اطلاعاتی که قرار است در یک تابع پردازش شوند تعریف شده و استفاده می‌شوند. تابعی که ما در مرحله قبل ایجاد کردیم انعطاف‌پذیر نبود، زیرا تنها رشته‌ای که درون تابع print قرار داشت را چاپ می‌کرد. زمانی که تابعی یک آرگومان را به عنوان ورودی دریافت کند؛ انعطاف‌پذیری زیادی به دست می‌آورد، زیرا شما می‌توانید رشته‌های مختلفی برای تابع ارسال کنید تا روی صفحه‌نمایش چاپ شوند. برای تعریف تابعی که بتواند آرگومانی به عنوان ورودی دریافت کند، مراحل زیر را انجام دهید.

1. در محیط IDLE دستورات زیر را وارد کنید.

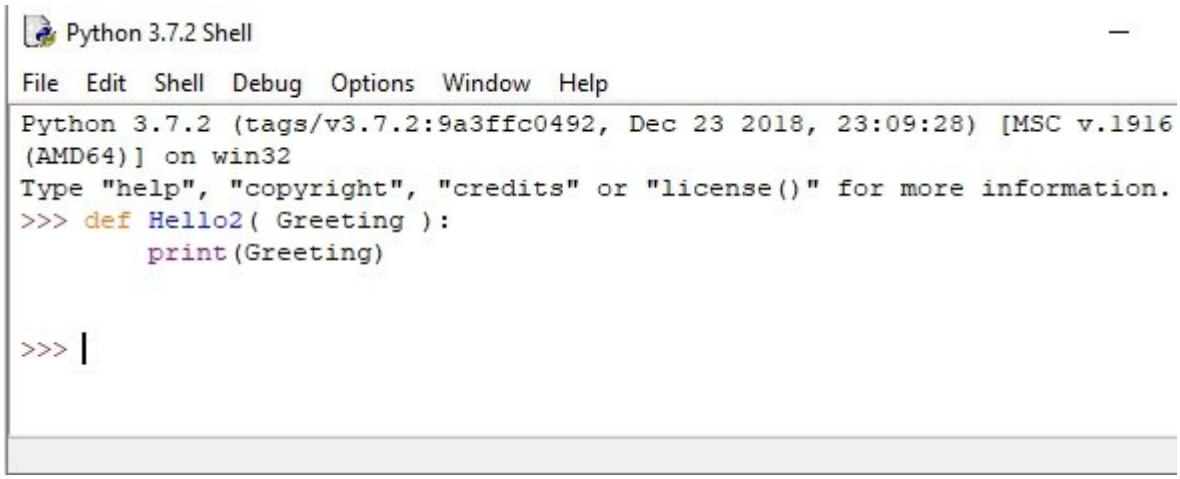
```
def Hello2( Greeting ):

print(Greeting)
```

1. دقت کنید در تعریف تابع Hello2 پرانتزها خالی نیستند. آن‌ها شامل یک کلمه کلیدی به نام Greeting هستند که به عنوان آرگومانی برای تابع Hello2 در نظر گرفته می‌شود. آرگومان Greeting یک نام دلخواه بوده و در اصل متغیری است که مقداری را نگه داشته و برای تابع print ارسال می‌کند. (دقت کنید که نام آرگومان‌ها می‌تواند هر کلمه‌ای به جزء کلمات کلیدی پایتون باشد.)

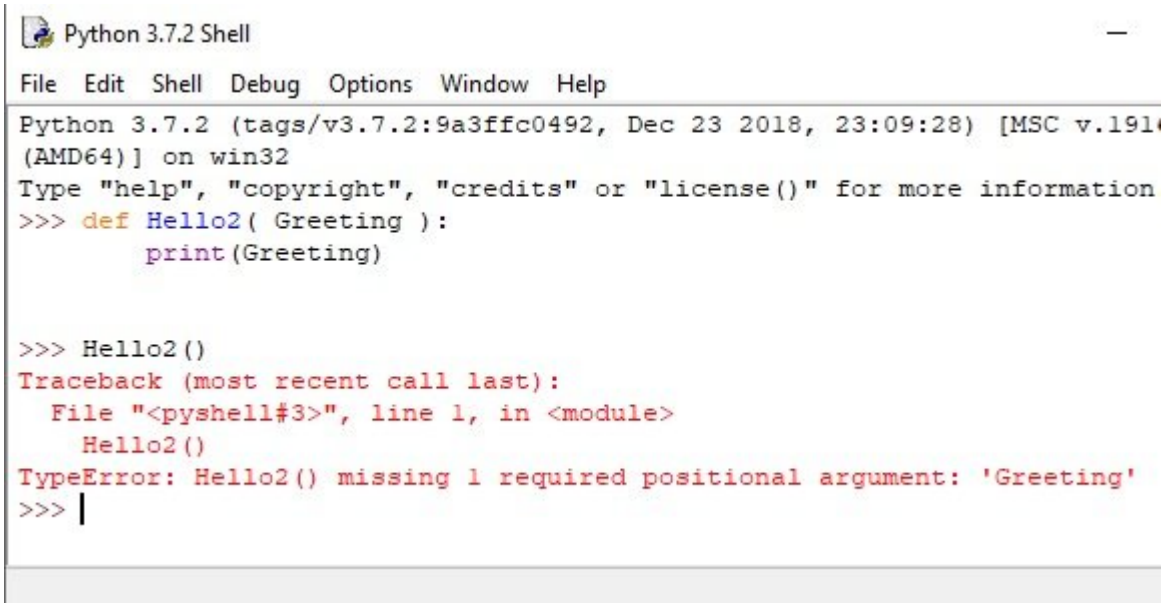
ارسال آرگومان‌ها به توابع

اکنون تابعی در اختیار دارید که متغیری را به عنوان ورودی دریافت می‌کند. دقت کنید زمانی که توابع دارای آرگومان‌های اجباری هستند، نحوه فراخوانی آن‌ها متفاوت از قبل می‌شود، زیرا شما باید آرگومان موردنیاز تابع را در اختیارش قرار دهید. اگر تابع Hello2 را همانند قبل و بدون وارد کردن آرگومانی برای آن فراخوانی کنید، مفسر پایتون پیغام خطایی نشان می‌دهد.

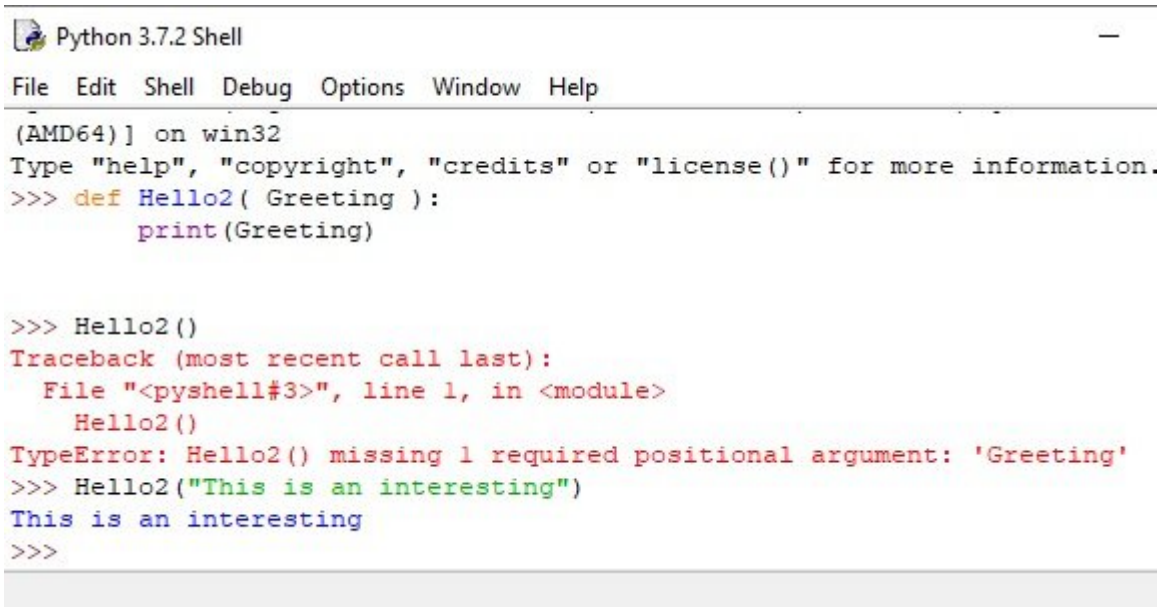


همان‌گونه که در تصویر مشاهده می‌کنید پایتون به شما اعلام می‌دارد که این تابع آرگومانی دارد که مقداردهی نشده است. برای فراخوانی توابعی که دارای آرگومان هستند باید مقدار مناسب را در زمان فراخوانی تابع در اختیارش قرار دهیم. در مثال ما آرگومان فوق یک رشته است، در نتیجه در زمان فراخوانی این تابع باید رشته دلخواه خود را مطابق با ترکیب زیر در اختیار تابع قرار دهیم.

Hello2(“This is an interesting")



اکنون تابع فوق رشته‌ای که به عنوان ورودی در اختیارش قرار داده‌اید را دریافت کرده، در اختیار تابع Print قرار داده و تابع این رشته را روی صفحه‌نمایش چاپ کرده است. این تابع هر رشته‌ای که در اختیارش قرار دهید را می‌تواند روی صفحه چاپ کند، دقت کنید که رشته‌ها باید درون دو علامت کوتیشن قرار گیرند، اما مقادیر را بدون کوتیشن هم می‌توانید برای تابع ارسال کنید.



در شماره آینده آموزش **پایتون** این مبحث را ادامه خواهیم داد.

تاریخ انتشار:
21 بهمن 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/programming/14577/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%B1%D8%A7%DB%8C%DA%AF%D8%A7%D9%86-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86-python-%E2%80%93-%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-%D9%85%D9%81%D9%87%D9%88%D9%85-%D8%AA%D9%88%D8%A7%D8%A8%D8%B9-%D9%88-%D8%AA%D9%82%D8%AF%D9%85-%D8%B9%D9%85%D9%84%DA%AF%D8%B1%D9%87%D8%A7-%D8%AF%D8%B1