CS50x Iran

Functions, Variables

Week note

پاییز و زمستان ۱۴۰۴

1 فهرست

VS Code نوع خاصی از ویرایشگر متن است که در قسمت بالا آن، یک ویرایشگر متن و در قسمت پایین، ترمینالی را برای اجرای دستورات مشاهده می‌کنید.

- برای شروع کدنویسی، در ترمینال `code hello.py` را اجرا کنید.
- در ویرایشگر متن بالا، `print("hello, world")` را تایپ کنید که یک کد معروف و متعارف است که تقریباً همه‌ی کدنویس ها در طی فرآیند یادگیری خود می‌نویسند.
- در پنجره‌ی terminal می‌توانید دستورات (commands) را اجرا کنید. برای اجرای این برنامه شما باید با موس خود بر روی پنجره ترمینال پایین صفحه کلیک کنید. اگر پنجره ترمینال برای شما نمایش داده نمی‌شد، می‌توانید با کلیک کردن بر روی گزینه Terminal در منو View بالای صفحه آن را باز کنید. سپس می‌توانید دستور دوم را در پنجره terminal تایپ کنید. در کنار علامت دلار $، `python hello.py` را تایپ کنید و کلید Enter را فشار دهید.
- به خاطر داشته باشید که کامپیوترها تنها صفر و یک را می‌فهمند. بنابراین، وقتی `python hello.py` را اجرا می‌کنید، پایتون متنی را که در hello.py نوشته اید تفسیر می‌کند و آن را به صفر و یک‌های قابل فهم برای کامپیوتر ترجمه می‌کند.
- نتیجه اجرای `python hello.py` نمایش hello, world در ترمینال است.

تبریک میگم! شما اولین برنامه‌ی خود را نوشتید.

- به عملیاتی که کامپیوتر یا زبان کامپیوتر از قبل می‌داند چگونه انجام دهد، تابع می‌گویند.
- در برنامه hello.py، تابع `print` می‌داند که چگونه در پنجره ترمینال متنی چاپ کند.
- توابع می‌توانند ورودی هایی دریافت کنند که به آن آرگومان های ورودی تابع (arguments) می‌گویند. در این مثال، تابع `print`آرگومان “hello, world” را می‌گیرد.

- اشتباه و خطا داشتن بخشی عادی از کدنویسی هستند. اینها اشتباهات و مشکلاتی هستند که باید آنها را حل کنید پس ناامید نشوید. این بخشی از فرآیند تبدیل شدن به یک برنامه نویس عالی است.
- تصور کنید در برنامه hello.py، به طور تصادفی `print("hello, world"` تایپ شده و آخرین `)` که مورد نیاز مفسر پایتون برای اجرا صحیح برنامه هست را از دست دادیم. اگر ما عمداً این اشتباه را انجام دهیم مفسر پایتون در پنجره ترمینال یک خطا را به عنوان خروجی چاپ می‌دهد.
- معمولا پیام های خطا، شما را از اشتباهتان آگاه می‌کند و سرنخ هایی را در مورد نحوه رفع آنها در اختیار شما قرار می‌دهد. با این حال، گاهی پیش می‌آید که مفسر یا مجری برنامه شما این امکان را برای شما فراهم نکند.

- ما می‌توانیم اولین برنامه‌ی پایتون شما را شخصی سازی کنیم.
- ما می‌توانیم در ویرایشگر متن خود در برنامه‌ی hello.py، از تابع دیگری هم استفاده کنیم. `input` یک تابع است که یک متن را به عنوان آرگومان ورودی دریافت می‌کند و بعد از چاپ کردن آن متن منتظر می‌ماند تا کاربر پاسخی را تایپ کند. می‌توانیم کد خود را به این صورت ویرایش کنیم:

hello.py

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

اما این ویرایش به برنامه‌ی شما اجازه نمی‌دهد که کاربر هر آنچه را که به عنوان ورودی وارد می‌کند، در خروجی برنامه ببیند و در هر صورت فقط یک خروجی ثابت بر می‌گرداند. به همین علت ما باید با متغیرها یا Variables آشنا شویم.

- یک متغیر، یک ظرف یا مکان برای نگهداری مقادیر و داده ها در برنامه‌ی شماست.
- در برنامه ای که نوشته بودیم، اکنون می‌توانیم با معرفی متغیرهای خود، آن را ویرایش کنیم تا بصورت زیر شود:

hello.py

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

- توجه کنید که علامت مساوی `=` که در وسط `name = input("What's your name? ")` وجود دارد، نقش ویژه ای در برنامه نویسی دارد. این علامت مساوی به این معناست که آنچه در سمت راست آمده است به آنچه که در سمت چپ است، اختصاص داده می‌شود. بنابراین مقدار دریافتی و بازگردانده شده توسط `input("What's your name? ")` به `name` اختصاص داده می‌شود.

- اگر برنامه‌ی خود را بصورت زیر ویرایش کنید با یک مشکل مواجه می‌شوید:

```
hello.py

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

- برنامه در جواب، بدون این که در نظر بگیرد کاربر چه چیزی تایپ کرده، hello, name را به عنوان خروجی در پنجره‌ی ترمینال بر می‌گرداند.
- در ادامه‌ی ویرایش کد، می‌توانید به این صورت تایپ کنید:

```
hello.py

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

- نتیجه ای که در پنجره‌ی ترمینال مشاهده خواهید کرد:

```
What's your name? David
hello
David
```

- اکنون ما به نتیجه ای که در نظر داریم نزدیک تر شدیم.
- در اینجا می‌توانید با انواع داده ها در پایتون بیشتر آشنا شوید: data types

- کامنت در واقع راهی است که برنامه‌نویس می‌تواند از طریق آن، آنچه را که در برنامه‌اش پیاده سازی کرده را پیگیری کند و حتی دیگران را از قصد و منظور خود در کدی که نوشته است آگاه کند. به طور خلاصه، کامنت‌ها، یادداشت‌هایی برای خودتان و کسانی که کد شما را مشاهده می‌کنند هستند.
- شما می‌توانید کامنت‌هایی را به برنامه‌ی خود اضافه کنید تا ببینید برنامه‌ی شما چه کاری انجام می‌دهد. می‌توانید کد خود را بصورت زیر ویرایش کنید:

hello.py

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

- همچنین کامنت ها می‌توانند برای شما بعنوان لیست کارهایی که می‌خواهید انجام دهید باشند.
- کامنت‌ها هیچ تاثیری در اجرای برنامه شما ندارند و صرفا همانند یادداشتی در کد های شما هستند.
- کامنت ها در پایتون همیشه با # شروع می‌شوند.

- Pseudocode یا شبه کد، نوع خاصی از کامنت است و معمولا برای مواقعی به کار می‌رود که نحوه‌ی انجام یک کار کدنویسی را می‌خواهید توضیح دهید. برای مثال به کد زیر و کامنت ها توجه کنید:

hello.py

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

# Print hello
print("hello,")

# Print the name inputted
print(name)
```

- ما می‌توانیم برای بهبود بهتر، کد را بصورت زیر ویرایش کنیم:

hello.py

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

# Print hello and the inputted name
print("hello, " + name)
```

- به نظر می‌رسد که برخی از توابع بیشتر از یک آرگومان ورودی دریافت می‌کنند.
- ما می‌توانیم برای پاس دادن چندین آرگومان به یک تابع از کاما , استفاده کنیم:

hello.py

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

# Print hello and the inputted name
print("hello ", name)
```

در آخر، اگر David را تایپ کنیم، خروجی ما در ترمینال hello David خواهد بود.

- یک رشته یا String که بصورت `str` در پایتون شناخته می‌شود، توالی و دنباله ای هستند از حروف و کارکتر.
- در کدهایمان کمی به عقب بر می‌گردیم تا به کد زیر برسیم. عوارض جانبی ظاهری نوشتن این کد در چند خط را مشاهده می‌کنید:

hello.py

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

- توابع آرگومان هایی را دریافت می‌کنند که این قضیه بر طرز کار آنها اثر می‌گذارند. اگر به مستنداتی که برای `print` وجود دارند نگاهی بیاندازید، متوجه خواهید شد که می‌توانیم چیزهای زیادی درباره‌ی آرگومان‌هایی که تابع `print` می‌گیرد، یاد بگیریم.

- با مشاهده این مستندات، خواهید آموخت که تابع `print` به طور خودکار شامل یک قطعه کد `end="\n"` است. مقدار `\n` نشان می‌دهد که تابع `print` به طور خودکار بعد از چاپ کردن مقداری که آن ورودی داده بودید یک خط شکسته ایجاد می‌کند. تابع printآرگومانی به نام `end` دریافت می‌کند که تعیین می‌کند، چه مقداری در انتها متن چاپ شده قرار بگیرد.
- ما می‌توانیم آرگومان `end` را طوری تغییر بدهیم که خط جدیدی ایجاد نشود:

hello.py

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

- با درنظر گرفتن `end=""` ما در واقع مقدار پیش فرض `end` را بازنویسی می‌کنیم، به گونه ای که بعد از چاپ کردن مقدار مورد نظر هیچوقت خط جدید ایجاد نمی‌کند. اگر به `name` مقدار David را ورودی دهیم، خروجی که در ترمینال چاپ می‌شود به این صورت است: hello, David
- بنابراین پارامترها، آرگومان هایی هستند که می‌توانند توسط یک تابع گرفته شوند.

- توجه داشته باشید که چگونه افزودن علامت نقل‌قول به عنوان بخشی از رشته شما، چالش برانگیز است.
- کد `print("hello,"friend"")` کار نمی‌کند و با خطا مواجه می‌شود.
- به طور کلی، دو راه برای رفع این مشکل وجود دارد. ابتدا می‌توانیم به سادگی نقل قول ها را به علامت نقل قول تکی ' تغییر دهیم.
- روش دیگری که بیشتر مورد استفاده قرار می‌گیرد به‌صورت `print("hello, \"friend\"")` می‌باشد. در اینجا بک اسلش ها به مفسر پایتون می‌گوید که کاراکتر زیر باید به عنوان علامت نقل‌قول در رشته در نظر گرفته شود و از خطا جلوگیری می‌شود.

- می‌توان گفت زیباترین راه برای استفاده از رشته ها به شرح زیر است:

hello.py

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

- به `f` در `print(f"hello, {name}")` توجه کنید. این `f` یک نشانگر ویژه در پایتون برای برخورد متفاوت با رشته است و باعث می‌شود شما قادر باشید به صورت مستقیم از متغیر مد نظر درون یک رشته استفاده کنید. توجه کنید که اسم متغیر را حتما باید درون `{}` قرار دهید. انتظار می‌رود که در این دوره به طور مکرر از این سبک رشته ها استفاده کنید.

- شما هرگز نباید انتظار داشته باشید که کاربر شما به همان شیوه‌ای که شما انتظار دارید با برنامه شما کار کند. بنابراین، شما باید اطمینان حاصل کنید که ورودی کاربر اصلاح یا بررسی شده است.
- در رشته‌ها، قابلیت حذف فضای خالی از یک رشته وجود دارد.
- با استفاده از متد `strip` در `name` بصورت `name = name.strip()`، باعث می‌شود تمام فضاهای خالی سمت چپ و راست ورودی کاربر پاک شود. می‌توانید کد خود را بصورت زیر تغییر دهید:

hello.py

```
# 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 برنامه، صرف نظر از اینکه چند فاصله قبل یا بعد از `name` تایپ می‌کنید، تمام فضای خالی حذف می‌شود.

- با استفاده از تابع `title`، نام کاربر بصورت عنوان تغییر داده می‌شود:

hello.py

```
# 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}")
```

- تا این لحظه ممکن است از تکراری تایپ کردن کلمه python در ترمینال خسته شده باشید.
- شما می‌توانید از کلید فلش رو به بالا ↑ در صفحه کلید خود، استفاده کنید تا دستورات ترمینالی را که اخیراً وارد کرده‌اید را، دوباره فراخوانی کنید.

- توجه داشته باشید که می‌توانید کد خود را بصورت کارآمد تر هم ویرایش کنید تا نتیجه کد قبلی شما را نشان دهد:

hello.py

```
# 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}")
```

- حتی می‌توانستیم به این صورت هم پیش برویم:

hello.py

```
# 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}")
```

- شما می‌توانید از طریق این لینک اطلاعات بیشتری در مورد رشته ها در پایتون بدست آورید.

- در پایتون، عدد صحیح را بصورت int نشان می‌دهند.
- ما در دنیای ریاضیات، با عملگرهای `+ - * % /` آشنا شدیم اما ممکن است با آخرین عملگر یعنی `%` یا modulo آشنایی زیادی نداشته باشید.
- شما اجباری به استفاده از پنجره ویرایشگر متن برای اجرای کد پایتون ندارید. در ترمینال خود، می‌توانید فقط با اجرای دستور `python`، کد پایتون خود را اجرا کنید. شما در پنجره‌ی ترمینال با علامت `>>>` مواجه خواهید شد. سپس می‌توانید کد خود را به صورت تعاملی و زنده را اجرا کنید. شما می‌توانید `1+1` را بنویسید و با کلید Enter آن را اجرا کنید و آن محاسبه را انجام دهید. این حالت در طول این دوره چندان مورد استفاده قرار نخواهد گرفت.
- ما می‌توانیم `code calculator.py` را در ترمینال تایپ کنیم تا فایل جدیدی ساخته شود که در آن ماشین حساب خودمان را بسازیم.

- ابتدا می‌خواهیم چند متغیر را تعریف کنیم:

calculator.py

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

- وقتی ما `python calculator.py` را اجرا می‌کنیم، نتیجه‌ای که در ترمینال بدست می‌آوریم 3 است. حال ما می‌توانیم با استفاده از تابع `input`، برنامه‌ی کارآمدتری را بنویسیم:

calculator.py

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

با اجرا کردن این برنامه، خروجی ما 12 می‌شود و نتیجه غلط است. چرا این اتفاق افتاد ؟

- قبلاً دیدیم که چگونه علامت `+` دو رشته را به هم متصل می‌کند. از آنجا که ورودی شما از کیبورد کامپیوتر به صورت متن وارد مفسر پایتون می‌شود، به عنوان یک رشته در نظر گرفته می‌شود. بنابراین، باید این ورودی را از یک رشته به یک عدد صحیح تبدیل کنیم. ما می‌توانیم این کار را به صورت زیر انجام دهیم:

calculator.py

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

z = int(x) + int(y)

print(z)
```

اکنون نتیجه‌ی ما درست است. به عملیاتی که `int(x)` انجام می‌دهد، Casting می‌گویند، که در آن یک مقدار به طور موقت از یک نوع داده (در اینجا یک رشته) به نوع دیگری (در اینجا یک عدد صحیح) تغییر می‌کند.

- ما می‌توانیم برنامه‌ی خودت را بهتر از قبل و بصورت زیر ویرایش کنیم:

calculator.py

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

print(x + y)
```

این نشان می‌دهد که می‌توانید از توابع درون توابع استفاده کنید. درونی ترین تابع ابتدا اجرا می‌شود و سپس تابع بیرونی اجرا می‌شود. ابتدا تابع `input`، و سپس تابع `int` اجرا می‌شود.

- شما می‌توانید از طریق این لینک اطلاعات بیشتری در مورد int در پایتون بدست آورید.

- هنگام تصمیم‌گیری در مورد نحوه کدنویسی برنامه خود، به خاطر داشته باشید که می‌توانید برای رفع یک مشکل، رویکرد ها و راه حل های مختلفی ارائه دهید.

- صرف نظر از اینکه چه رویکردی در نظر می‌گیرید، به یاد داشته باشید که کد شما باید خوانا باشد. شما باید از کامنت‌ها استفاده کنید تا به خود و دیگران توضیحاتی در مورد کاری که کدتان انجام می‌دهد بدهید. علاوه بر این، باید کد را به گونه ای بنویسید که قابل خواندن باشد.

- اعداد گویا (float) اعدادی هستند که دارای یک نقطه اعشار مانند 0.52 می‌باشند.
- می‌توانید کد خود را برای دریافت از اعداد گویا به صورت زیر تغییر دهید:

calculator.py

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

print(x + y)
```

این تغییر به کاربر شما اجازه می‌دهد تا اعداد 1.2 و 3.4 را وارد کند و در جواب 4.6 را دریافت کند.

- با این حال، بیایید تصور کنیم که می‌خواهید حاصل یک عملیات جمع را به نزدیکترین عدد صحیح گرد کنید. با نگاهی به اسناد پایتون برای round، خواهید دید که آرگومان های موجود برای این تابع round(number [n , ndigits]) هستند. براکت ها نشان می‌دهند که برنامه‌نویس می‌تواند آرگومانی را بصورت اختیاری وارد کند. بنابراین، می‌توانید برای گرد کردن یک رقم به نزدیکترین عدد صحیح آن تابع را به این صورت round(n) بنویسید:

calculator.py

```
# 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 به نزدیکترین عدد صحیح گرد می‌شود.

- چه اتفاقی می‌افتد اگر می‌خواستیم خروجی اعداد طولانی را قالب‌بندی کنیم؟ برای مثال، به جای نمایش 1000، ممکن است بخواهید 1,000 را نمایش دهید. می‌توانید کد خود را به صورت زیر تغییر دهید:

calculator.py

```
# 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:,}")
```

اگرچه کاملاً ناواضح است، اما `print(f"{z:,}")` سناریویی را ایجاد می‌کند که در آن خروجی z شامل کاما می‌شود.

- چگونه می‌توانیم مقادیر اعشاری را گرد کنیم؟ ابتدا کد خود را به صورت زیر تغییر دهید:

calculator.py

```
# 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 2 به عنوان مقدار x و 3 به عنوان y، در نتیجه مقدار z برابر 0.6666666666 است که ظاهراً همانطور که انتظار داریم تا بی‌نهایت ادامه می‌یابد.

- می‌خواهیم آن عدد را گرد کنیم، می‌توانیم کد خود را به صورت زیر تغییر دهیم:

calculator.py

```
# 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، نتیجه را به نزدیکترین دو رقم اعشار گرد می‌کند.

- همچنین می‌توانیم از fstring برای فرمت خروجی به صورت زیر استفاده کنیم:

calculator.py

```
# 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}")
```

این کد fstring، همان استراتژی گرد کردن قبلی ما را انجام می‌دهد.

- شما می‌توانید از طریق این لینک اطلاعات بیشتری در مورد float در پایتون بدست آورید.

- جالب نیست اگر بتوانیم توابع مخصوص به خودمان را بنویسیم؟
- بیایید کد نهایی hello.py را با تایپ `code hello.py` در پنجره ترمینال برگردانیم. کد شما باید به صورت زیر باشد:

hello.py

```
# 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 می‌توانیم کد خود را بهتر کنیم و تابع خاص خود را ایجاد کنیم که برای ما "hello" برمی‌گرداند!

- با پاک کردن تمام کدهای خود در ویرایشگر متن، بیایید از ابتدا شروع کنیم:

hello.py

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

- با تلاش برای اجرای این کد، با پیام خطا مواجه می‌شوید. بالاخره هیچ تابع تعریف شده ای برای `hello` وجود ندارد.

- ما می‌توانیم تابع خود به نام `hello` را به صورت زیر ایجاد کنیم:

hello.py

```
def hello():
    print("hello")


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

توجه کنید که هر چیزی که زیر `def hello()` قرار دارد، باید به اندازه یک Tab یا (4 فاصله) از ابتدا خط فاصله بگیرد. پایتون یک زبان حساس به فاصله است. این زبان از فاصله‌گذاری برای تشخیص محدوده کد های تابع، استفاده می‌کند. بنابراین، هر چیزی که در تابع `hello` قرار دارد، باید فاصله گذاری شود. وقتی یک چیزی فاصله ندارد، به عنوان اینکه داخل تابع `hello` نیست، در نظر گرفته می‌شود. اگر دستور `python hello.py` را در ترمینال اجرا کنید، خواهید دید که خروجی شما دقیقاً همانطور که شما می‌خواهید نیست.

- ما می‌توانیم کد خود را بیشتر بهبود ببخشیم:

hello.py

```
# 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 خط اول، شما تابع `hello` را تعریف می‌کنید. همچنین، شما به مفسر می‌گویید که این تابع یک آرگومان ورودی دارد. یک متغیر به نام `to`. بنابراین، وقتی شما `hello(name)` فراخوانی می‌کنید، کامپیوتر `name` را بعنوان `to` به تابع `hello` ارسال می‌کند. این روش برای ارسال مقادیر به توابع استفاده می‌شود.
- با اجرای `python hello.py` در پنجره‌ی ترمینال، خواهید دید که خروجی بسیار نزدیکتر به بحثی که در ابتدا داشتیم است.

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

hello.py

```
# 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 تست کنید. توجه داشته باشید که اگر در ورودی چیزی تایپ کنید، تابع همانطور که انتظار داریم کار می‌کند اما اگر چیزی تایپ نکنید و Enter را بزنید، کد `to="world"` باعث می‌شود که مقدار پیش فرض `world` درون تابع `print` قرار بگیرد و خروجی به این صورت است: hello, world

- ما مجبور نیستیم در ابتدای برنامه تابع خود را بنویسیم. ما می‌توانیم آن را پایین بیاوریم اما حتما باید به مفسر پایتون بگوییم که ما یک تابع `main` و یک تابع `hello` جداگانه داریم.

hello.py

```
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 کد یک نوع خطا ایجاد می‌کند. اگر ما `python hello.py` را اجرا کنیم هیچ اتفاقی نمی‌افتد. علت آن این است که هیچ بخشی از این کد در واقع تابع `main` را فراخوانی نمی‌کند تا برنامه ما را زنده کند.

- تغییر بسیار کوچک زیر، تابع `main` را فراخوانی می‌کند و برنامه ما را به کار می‌اندازد:

hello.py

```
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()
```

- شما می‌توانید سناریو های زیادی را تصور کنید که در آنها نه تنها می‌خواهید یک تابع یک عمل را انجام دهد، بلکه می‌خواهید یک مقدار را به تابع اصلی برگردانید. به عنوان مثال، به جای چاپ نتیجه `x + y`، ممکن است بخواهید که مقدار این محاسبه را به قسمت دیگری از برنامه برگردانید. این برگرداندن مقدار از تابع را `return` می‌نامند.
- با تایپ `code calculator.py` به کد calculator.py برگردید. همه کدها را در آنجا پاک کنید. کد را به صورت زیر دوباره بنویسید:

calculator.py

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

def square(n):
    return n * n

main()
```

به طور مؤثر، `x` به تابع توان `square` منتقل می‌شود. سپس، محاسبه `x * x` به تابع اصلی باز می‌گردد.

در این هفته مباحث زیر را مورد بررسی قرار دادیم:

- ✓ **نوشتن کد با پایتون**
- ✓ **توابع**
- ✓ **باگ‌ها**
- ✓ **بهبود اولین برنامه‌ی پایتون**
- ✓ **متغیرها**
- ✓ **کامنت‌ها**
- ✓ **شبه کد**
- ✓ **رشته‌ها و پارامترها**
- ✓ **یک مشکل با نقل‌قول**
- ✓ **قالب بندی رشته‌ها**
- ✓ **اعداد صحیح**
- ✓ **اعداد گویا**
- ✓ **خوانا بودن کد**

حالا شما توانایی این رو دارید که در پایتون از دستورات شرطی استفاده کنید و متناسب با نتیجه دریافتی، فرایند موردنظر خود را اجرا کنید.

CS50x Iran

Harvard's Computer Science 50x Iran