CS50's Introduction to Programming with Python

CS50 的 Python 编程简介

OpenCourseWare 开放课件

Donate 捐

David J. Malan 大卫·马兰
malan@harvard.edu

CS50x Puzzle Day 2025...

CS50x 拼图 日 2025 新产品

CS50.ai
Ed Discussion for Q&A
问答的 Ed 讨论
Visual Studio Code Visual Stu...

0. Functions, Variables 函数...

Lecture 0 第 0 讲

Creating Code with Python
使用 Python 创建代码

- VS Code is a special type of text editor that is called a compiler. At the top, you'll notice a text editor. At the bottom, you will see a terminal where you can execute commands.
  VS Code 是一种特殊类型的文本编辑器，称为编译器。在顶部，您会注意到一个文本编辑器。在底部，您将看到一个可以执行命令的终端。
- In the terminal, you can execute `code hello.py` to start coding.
  在终端中，您可以执行 `code hello.py` 开始编码。
- In the text editor above, you can type `print("hello, world")`. This is a famous canonical program that nearly all coders write during their learning process.
  在上面的文本编辑器中，您可以键入 `print（“hello， world”）`。这是一个著名的规范程序，几乎所有编码人员都在学习过程中编写。
- In the terminal window, you can execute commands. To run this program, you are going to need to move your cursor to the bottom of the screen, clicking in the terminal window. You can now type a second command in the terminal window. Next to the dollar sign, type `python hello.py` and press the enter key on your keyboard.
  在终端窗口中，您可以执行命令。要运行此程序，您需要将光标移动到屏幕底部，单击终端窗口。您现在可以在终端窗口中键入第二个命令。在美元符号旁边，键入 `python hello.py` 然后按键盘上的 Enter 键。
- Recall that computers really only understand zeros and ones. Therefore, when you run `python hello.py`, python will interpret the text that you created in `hello.py` and translate it into the zeros and ones that the computer can understand.
  回想一下，计算机实际上只理解 0 和 1。因此，当您运行 `python hello.py` 时，python 会解释您在 `hello.py` 中创建的文本并将其转换为计算机可以理解的 0 和 1。
- The result of running the `python hello.py` program is `hello, world`.
  运行 `python hello.py` 程序的结果是 `hello， world`。
- Congrats! You just created your first program.
  恭喜！您刚刚创建了第一个程序。

Functions 功能

- Functions are verbs or actions that the computer or computer language will already know how to perform.
  函数是计算机或计算机语言已经知道如何执行的动词或动作。
- In your `hello.py` program, the `print` function knows how to print to the terminal window.
  在 `hello.py` 程序中， `print` 函数知道如何打印到终端窗口。
- The `print` function takes arguments. In this case, `"hello, world"` are the arguments that the `print` function takes.
  `print` 函数接受参数。在这种情况下， “hello， world” 是 `print` 函数采用的参数。

Bugs 错误

- Bugs are a natural part of coding. These are mistakes, problems for you to solve! Don't get discouraged! This is part of the process of becoming a great programmer.
  Bug 是编码的自然组成部分。这些都是错误，需要你解决的问题！不要气馁！这是成为一名优秀程序员的过程的一部分。

- Imagine in our `hello.py` program that accidentally typed `print("hello, world"` notice that we missed the final `)` required by the compiler. If I purposefully make this mistake, you'll the compiler will output an error in the terminal window!

  想象一下，在我们的 `hello.py` 程序中，不小心键入了 `print（“hello， world”` notice that we miss the final `）` 编译器所需的）。如果我故意犯这个错误，编译器会在终端窗口中输出一个错误！

- Often, the error messages will inform you of your mistakes and provide you clues on how to fix them. However, there will be many times when the compiler is not this kind.

  通常，错误消息会通知您您的错误并为您提供有关如何修复这些错误的线索。但是，很多时候编译器不是这种。

Improving Your First Python Program
改进您的第一个 Python 程序

- We can personalize your first Python program.

  我们可以个性化您的第一个 Python 程序。

- In our text editor in `hello.py` we can add another function. `input` is a function that takes a prompt as an argument. We can edit our code to say

  在 `hello.py` 的文本编辑器中，我们可以添加另一个函数。 `input` 是一个将 prompt 作为参数的函数。我们可以编辑我们的代码来表示

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

- This edit alone, however, will not allow your program to output what your user inputs. For that, we will need to introduce you to variables

  但是，仅此编辑将不允许程序输出用户输入的内容。为此，我们需要向您介绍变量

Variables 变量

- A variable is just a container for a value within your own program.

  变量只是你自己的程序中值的容器。

- In your program, you can introduce your own variable in your program by editing it to read

  在程序中，您可以通过编辑变量来在程序中引入自己的变量，以读取

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

  Notice that this equal `=` sign in the middle of `name = input("What's your name? ")` has a special role in programming. This equal sign literally assigns what is on the right to what is on the left. Therefore, the value returned by `input("What's your name? ")` is assigned to `name`.

  请注意，中间的这个等号 `=` `name = input("What's your name? ")` 在编程中具有特殊作用。这个等号从字面上将右侧的内容分配给左侧的内容。因此，`input（“What's your name? ”）` 返回的值将分配给 `name`。

- If you edit your code as follows, you will notice an error

  如果按如下方式编辑代码，则会注意到一个错误

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

- The program will return `hello, name` in the terminal window regardless of what the user types.

  无论用户键入什么，程序都会在终端窗口中返回 `hello， name`。

- Further editing our code, you could type

  进一步编辑我们的代码，您可以键入

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

- The result in the terminal window would be

  终端窗口中的结果将是

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

- We are getting closer to the result we might intend!

  我们越来越接近我们可能想要的结果！

- You can learn more in Python's documentation on data types.

  您可以在 Python 的数据类型文档中了解更多信息。

Comments 评论

- Comments are a way for programmers to track what they are doing in their programs and even inform others about their intentions for a block of code. In short, they are notes for yourself and others who will see your code!

  注释是程序员跟踪他们在程序中执行的操作的一种方式，甚至可以告知其他人他们对代码块的意图。简而言之，它们是您自己和其他人将看到您的代码的注释！

- You can add comments to your program to be able to see what it is that your program is doing. You might edit your code as follows:

  您可以向程序添加注释，以便能够查看程序正在执行的操作。您可以按如下方式编辑代码：

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

- Comments can also serve as a to-do list for you.

  评论也可以用作您的待办事项列表。

Pseudocode 伪代码

- Pseudocode is an important type of comment that becomes a special type of to-do list, especially when you don't understand how to accomplish a coding task. For example, in your code, you might edit your code to say:

  伪代码是一种重要的注释类型，它成为一种特殊类型的待办事项列表，尤其是当您不了解如何完成编码任务时。例如，在代码中，您可以编辑代码以表示：

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

# Print hello
print("hello,")

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

Further Improving Your First Python Program
进一步改进您的第一个 Python 程序

- We can further edit our code as follows:

  我们可以进一步编辑我们的代码，如下所示：

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

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

- It turns out that some functions take many arguments.

  事实证明，有些函数需要很多参数。

- We can use a comma `,` to pass in multiple arguments by editing our code as follows:

  我们可以使用逗号 `,` 通过编辑我们的代码来传入多个参数，如下所示：

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

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

  The output in the terminal, if we typed "David" we would be `hello, David`. Success.

  终端中的输出，如果我们键入 "David"，我们将是 `hello, David`。成功。

Strings and Paremeters 字符串和参数

- A string, known as a `str` in Python, is a sequence of text.

  字符串在 Python 中称为 `str`，是文本序列。

- Rewinding a bit in our code back to the following, there was a visual side effect of having the result appear on multiple lines:

  将代码中的代码稍微倒回到以下内容，结果显示在多行上有一个视觉副作用：

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

- Functions take arguments that influence their behavior. If we look at the documentation for `print` you'll notice we can learn a lot about the arguments that the print function takes.

  函数采用影响其行为的参数。如果我们查看 `print` 的文档，你会注意到我们可以学到很多关于 print 函数所采用的参数的信息。

- Looking at this documentation, you'll learn that the print function automatically includes a piece of code `end='\n'. This \n indicates that the print function will automatically create a line break when run. The print function takes an argument called end` and the default is to create a new line.

  查看本文档，您将了解 print 函数自动包含一段代码 `end='\n'。this \n indicates that the print function will automatically create a line break when run. The print function takes an argument called end'` 和默认值是创建新行。

- However, we can technically provide an argument for `end` ourselves such that a new line is not created!

  但是，从技术上讲，我们可以为 `end` self 提供一个参数，这样就不会创建新行！

- We can modify our code as follows:

  我们可以按如下方式修改我们的代码：

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

  By providing `end=""` we are overwriting the default value of `end` such that it never creates a new line after this first print statement. Providing the name as "David", the output in the terminal window will be `hello, David`.

  通过提供 `end=""` ，我们将覆盖 `end` 的默认值，这样它就不会在第一个 print 语句之后创建新行。提供名称 "David"，终端窗口中的输出将为 `hello, David`。

- Parameters, therefore, are arguments that can be taken by a function.

  因此，参数是函数可以采用的参数。

因此，参数是函数可以采用的参数。

- You can learn more in Python's documentation on `print`.
  您可以在 Python 的 印刷 文档中了解更多信息。

A small problem with quotation marks
引号的小问题

- Notice how adding quotation marks as part of your string is challenging.
  请注意，将引号添加为字符串的一部分是多么具有挑战性。
- `print("hello,"friend"")` will not work, and the compiler will throw an error.
  `print（“hello，”friend“”）` 将不起作用，编译器将抛出错误。
- Generally, there are two approaches to fixing this. First, you could simply change the quotes to single quotation marks.
  通常，有两种方法可以解决此问题。首先，您可以简单地将引号更改为单引号。
- Another, more commonly used approach would be code as `print("hello, \"friend\"")`. The backslashes tell the compiler that the following character should be considered a quotation mark in the string and avoid a compiler error.
  另一种更常用的方法是 `code as print（“hello， \”friend\“”）` 。反斜杠告诉编译器，应将以下字符视为字符串中的引号，以避免编译器错误。

Formatting Strings 设置字符串格式

- Probably the most elegant way to use strings would be as follows:
  使用字符串的最优雅方式可能如下：

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

Notice the `f` in `print(f"hello, {name}")`. This `f` is a special indicator for Python to treat this string a special way, different than previous approaches we have illustrated in this lecture. Expect that you will be using this style of strings quite frequently in this course.

请注意 `print（f“hello， {name}”）` 中的 `f`。这个 `f` 是 Python 的一个特殊指标，它以特殊的方式处理这个字符串，这与我们在本次讲座中演示的先前方法不同。预计您将在本课程中非常频繁地使用这种样式的琴弦。

More on Strings 更多关于 Strings 的信息

- You should never expect your user to cooperate as intended. Therefore, you will need to ensure that the input of your user is corrected or checked.
  您永远不应该期望您的用户按预期合作。因此，您需要确保正确或检查用户的输入。
- It turns out that built into strings is the ability to remove whitespace from a string.
  事实证明，字符串中内置的功能是从字符串中删除空格。
- By utilizing the method `strip` on `name` as `name = name.strip()`, will strip all the whitespaces on the left and right of the users input. You can modify your code to be:
  通过使用 `name 上 name = name.strip（）` 的方法 `strip`，将去除用户输入左侧和右侧的所有空格。您可以将代码修改为：

```
# 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}")
```

Rerunning this program, regardless of how many spaces you type before or after the name, it will strip off all the whitespace.

重新运行此程序，无论您在名称之前或之后键入多少个空格，它都会去除所有空格。

- Using the `title` method, it would title case the user's name:
  使用 `title` 方法，它将对用户名进行 title 大小写：

```
# 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}")
```

- By this point, you might be very tired of typing `python` repeatedly in the terminal window. You cause use the up arrow of your keyboard to recall the most recent terminal commands you have made.
  此时，您可能已经厌倦了在终端窗口中重复键入 `python`。您需要使用键盘的向上箭头来调用您最近执行的终端命令。
- Notice that you can modify your code to be more efficient:
  请注意，您可以修改代码以提高效率：

```
# 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}")
```

This creates the same result as your previous code.

这将创建与上一个代码相同的结果。

- We could even go further!

  我们甚至可以走得更远！

```
# 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}")
```

- You can learn more about strings in Python's documentation on `str`

  您可以在 `Python 的 str` 文档中了解有关字符串的更多信息

Integers or int 整数或 int

- In Python, an integer is referred to as an `int`.

  在 Python 中，整数称为 `int`。

- In the world of mathematics, we are familiar with +, -, *, /, and % operators. That last operator `%` or modulo operator may not be very familiar to you.

  在数学领域，我们熟悉 +、-、*、/ 和 % 运算符。最后一个运算符 `%` 或模运算符您可能不是很熟悉。

- You don't have to use the text editor window in your compiler to run Python code. Down in your terminal, you can run `python` alone. You will be presented with `>>>` in the terminal window. You can then run live, interactive code. You could type `1+1`, and it will run that calculation. This mode will not commonly be used during this course.

  您不必使用编译器中的文本编辑器窗口来运行 Python 代码。在终端中，您可以单独运行 `python`。您将在终端窗口中显示 `>>>`。然后，您可以运行实时的交互式代码。您可以键入 `1+1`，它将运行该计算。本课程中不常使用此模式。

- Opening up VS Code again, we can type `code calculator.py` in the terminal. This will create a new file in which we will create our own calculator.

  再次打开 VS Code，我们可以在终端中键入 `code calculator.py`。这将创建一个新文件，我们将在其中创建自己的计算器。

- First, we can declare a few variables.

  首先，我们可以声明一些变量。

```
x = 1
y = 2

z = x + y

print(z)
```

Naturally, when we run `python calculator.py` we get the result in the terminal window of `3`. We can make this more interactive using the `input` function.

自然地，当我们运行 `python calculator.py` 时，我们会在终端窗口 `3` 中得到结果。我们可以使用 `input` 函数使其更具交互性。

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

z = x + y

print(z)
```

- Running this program, we discover that the output is incorrect as `12`. Why might this be?

  运行此程序时，我们发现输出不正确，为 `12`。为什么会这样呢？

- Prior, we have seen how the `+` sign concatenates two strings. Because your input from your keyboard on your computer comes into the compiler as text, it is treated as a string. We, therefore, need to convert this input from a string to an integer. We can do so as follows:

  之前，我们已经看到了 `+` 号是如何连接两个字符串的。由于您从计算机上键盘输入的输入作为文本进入编译器，因此它被视为字符串。因此，我们需要将此输入从字符串转换为整数。我们可以这样做：

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

z = int(x) + int(y)

print(z)
```

The result is now correct. The use of `int(x)` is called "casting," where a value is temporarily changed from one type of variable (in this case, a string) to another (here, an integer).

现在结果正确。`int（x）` 的使用称为 "casting"，其中值从一种类型的变量（在本例中为字符串）临时更改为另一种类型的变量（此处为整数）。

- We can further improve our program as follows:

  我们可以进一步改进我们的计划，如下所示：

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

print(x + y)
```

This illustrates that you can run functions on functions. The inner function is run first, and then the outer one is run. First, the `input` function is run. Then, the `int` function.

这说明了您可以在函数上运行函数。首先运行内部函数，然后运行外部函数。首先，运行 `输入` 函数。然后，`使用 int` 函数。

- You can learn more in Python's documtenation of `int`.

  您可以在 Python 的 `int` 文档中了解更多信息。

Readability Wins 可读性胜出

Readability Wins 可读性取胜

- When deciding on your approach to a coding task, remember that one could make a reasonable argument for many approaches to the same problem.

  在决定编码任务的方法时，请记住，对于同一问题，可以提出多种方法的合理论点。

- Regardless of what approach you take to a programming task, remember that your code must be readable. You should use comments to give yourself and others clues about what your code is doing. Further, you should create code in a way that is readable.

  无论您采用何种方法执行编程任务，请记住，您的代码必须是可读的。您应该使用注释为自己和他人提供有关代码正在做什么的线索。此外，您应该以可读的方式创建代码。

Float Basics 浮点数基础知识

- A floating point value is a real number that has a decimal point in it, such as `0.52`.

  浮点值是包含小数点的实数，例如 `0.52`。

- You can change your code to support floats as follows:

  您可以更改代码以支持浮点数，如下所示：

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

print(x + y)
```

  This change allows your user to enter `1.2` and `3.4` to present a total of `4.6`.

  此更改允许用户输入 `1.2` 和 `3.4` 来显示总计 `4.6`。

- Let's imagine, however, that you want to round the total to the nearest integer. Looking at the Python documentation for `round`, you'll see that the available arguments are `round(number[n, ndigits])`. Those square brackets indicate that something optional can be specified by the programmer. Therefore, you could do `round(n)` to round a digit to its nearest integer. Alternatively, you could code as follows:

  但是，假设您要将总数四舍五入到最接近的整数。查看 `round` 的 Python 文档，您将看到可用的参数是 `round（number[n， ndigits]）`。这些方括号表示程序员可以指定一些可选内容。因此，您可以执行 `round（n）` 将数字舍入到最接近的整数。或者，您可以按如下方式编写代码：

```
# 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)
```

  The output will be rounded to the nearest integer.

  输出将四舍五入到最接近的整数。

- What if we wanted to format the output of long numbers? For example, rather than seeing `1000`, you may wish to see `1,000`. You could modify your code as follows:

  如果我们想要格式化长数字的输出怎么办？例如，您可能希望看到 `1,000`，而不是 `1000`。您可以按如下方式修改代码：

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

  Though quite cryptic, that `print(f"{z:,}")` creates a scenario where the outputted `z` will include commas where the result could look like `1,000` or `2,500`.

  虽然非常神秘，但 `print（f“{z:，}”）` 会创建一个场景，其中输出的 `z` 将包含逗号，结果可能看起来像 `1,000` 或 `2,500`。

More on Floats 有关 Float 的更多信息

- How can we round floating point values? First, modify your code as follows:

  我们如何对浮点值进行四舍五入？首先，按如下方式修改您的代码：

```
# 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)
```

  When inputting `2` as x and `3` as y, the result z is `0.6666666666`, seemingly going on to infinite as we might expect.

  当输入 `2` 作为 `x，3` 作为 y 时，结果 z 是 `0.6666666666`，似乎正如我们预期的那样走向无限。

- Let's imagine that we want to round this down. We could modify our code as follows:

  假设我们想要将其四舍五入。我们可以按如下方式修改我们的代码：

```
# 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)
```

As we might expect, this will round the result to the nearest two decimal points.

正如我们所料，这会将结果四舍五入到最接近的小数点后两位。

- We could also use `fstring` to format the output as follows:

  我们还可以使用 `fstring` 来格式化输出，如下所示：

```
# 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}")
```

This cryptic `fstring` code displays the same as our prior rounding strategy.

这个神秘的 `fstring` 代码与我们之前的舍入策略显示相同。

- You can learn more in Python's documentation of `float`.

  您可以在 Python 的 `float` 文档中了解更多信息。

Def 防守

- Wouldn't it be nice to create our own functions?

  创建我们自己的函数不是很好吗？

- Let's bring back our final code of `hello.py` by typing `code hello.py` into the terminal window. Your starting code should look as follows:

  让我们通过在终端窗口中键入 `code hello.py` 来恢复我们的最终 `hello.py` 代码。您的起始代码应如下所示：

```
# 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}")
```

We can better our code to create our own special function that says "hello" for us!

我们可以改进我们的代码来创建自己的特殊函数，为我们说 "hello"！

- Erasing all our code in our text editor, let's start from scratch:

  擦除文本编辑器中的所有代码，让我们从头开始：

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

Attempting to run this code, your compiler will throw an error. After all, there is no defined function for `hello`.

尝试运行此代码时，编译器将引发错误。毕竟，`hello` 没有定义的函数。

- We can create our own function called `hello` as follows:

  我们可以创建自己的名为 `hello` 的函数，如下所示：

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


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

Notice that everything under `def hello()` is indented. Python is an indented language. It uses indentation to understand what is part of the above function. Therefore, everything in the `hello` function must be indented. When something is not indented, it treats it as if it is not inside the `hello` function. Running `python hello.py` in the terminal window, you'll see that your output is not exactly as you may want.

请注意，`def hello()` 下的所有内容都是缩进的。Python 是一种缩进语言。它使用缩进来理解上述函数的一部分。因此，`hello` 函数中的所有内容都必须缩进。当某些内容没有缩进时，它会将其视为不在 `hello` 函数中。在终端窗口中运行 `python hello.py`，您会发现您的输出并不完全符合您的预期。

- We can further improve our code:

  我们可以进一步改进我们的代码：

```
# Create our own function
def hello(to):
    print("hello,", to)


# Output using our own function
name = input("What's your name? ")
hello(name)
```

Here, in the first lines, you are creating your `hello` function. This time, however, you are telling the compiler that this function takes a single parameter: a variable called `to`. Therefore, when you call `hello(name)` the computer passes `name` into the `hello` function as `to`. This is how we pass values into functions. Very useful! Running `python hello.py` in the terminal window, you'll see that the output is much closer to our ideal presented earlier in this lecture.

在这里，在第一行中，您将创建 `hello` 函数。但是，这一次，您告诉编译器此函数采用单个参数：调用 `to` 的变量。因此，当您调用 `hello(name)` 时，计算机会将 `name` 传递到 `hello` 函数中。这就是我们将值传递给函数的方式。非常有用！在终端窗口中运行 `python hello.py`，您将看到输出更接近于本讲座前面介绍的理想情况。

- We can change our code to add a default value to `hello`:

  我们可以更改代码以向 `hello` 添加默认值：

```
# 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()
```

Test out your code yourself. Notice how the first `hello` will behave as you might expect, and the second hello, which is not passed a value, will, by default, output `hello, world`.

自己测试您的代码。请注意第一个 `hello` 的行为方式，默认情况下，第二个 hello （未传递值）将输出 `hello, world`。

- We don't have to have our function at the start of our program. We can move it down, but we need to tell the compiler that we have a `main` function and a separate `hello` function.

  我们不必在程序开始时就有我们的函数。我们可以将其向下移动，但需要告诉编译器我们有一个 `main` 函数和一个单独的 `hello` 函数。

```
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)
```

This alone, however, will create an error of sorts. If we run `python hello.py`, nothing happens! The reason for this is that nothing in this code is actually calling the `main` function and bringing our program to life.

然而，仅此一项就会产生各种错误。如果我们 hello.py 运行 `python`，什么都不会发生！这样做的原因是，此代码中的任何内容实际上都没有调用 `main` 函数并使我们的程序栩栩如生。

- The following very small modification will call the `main` function and restore our program to working order:

  以下非常小的修改将调用 `main` 函数并将我们的程序恢复到工作状态：

```
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()
```

Returning Values 返回值

- You can imagine many scenarios where you don't just want a function to perform an action but also to return a value back to the main function. For example, rather than simply printing the calculation of `x + y`, you may want a function to return the value of this calculation back to another part of your program. This "passing back" of a value we call a `return` value.

  您可以想象许多场景，您不仅希望函数执行操作，还希望将值返回给 main 函数。例如，您可能希望函数将此计算的值返回给程序的另一部分，而不是简单地打印 `x + y` 的计算。这种“传回”的值称为 `返回` 值。

- Returning to our `calculator.py` code by typing `code calculator.py`. Erase all code there. Rework the code as follows:

  通过键入 `code calculator.py` 返回到我们的 `calculator.py` 代码。擦除那里的所有代码。按如下方式重做代码：

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


def square(n):
    return n * n


main()
```

Effectively, `x` is passed to `square`. Then, the calculation of `x * x` is returned back to the main function.

实际上，`x` 被传递给 `square`。然后，`x * x` 的计算返回给 main 函数。

Summing Up 总结

Through the work of this single lecture, you have learned abilities that you will use countless times in your own programs. You have learned about…

通过这一次讲座的工作，您学到了将在自己的程序中无数次使用的能力。您已经了解了…

- Creating your first programs in Python;

  在 Python 中创建您的第一个程序;

- Functions; 功能;
- Bugs; 错误

- Bugs; 错误;
- Variables; 变量;
- Comments; 评论;
- Pseudocode; 伪代码;
- Strings; 字符串;
- Parameters; 参数;
- Formatted Strings; 格式化字符串;
- Integers; 整数;
- Principles of readability; 可读性原则;
- Floats; 浮;
- Creating your own functions; and 创建您自己的函数;和
- Return values. 返回值。