Giới thiệu về lập trình với Python của CS50

OpenCourseWare

Quyên tặng (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan)
(https://twitter.com/davidjmalan)

Bài giảng 0

- Tạo mã bằng Python
- Chức năng
- Lỗi
- Cải thiện chương trình Python đầu tiên của bạn
  - Biến
  - Bình luận
  - Mã giả
- Cải thiện hơn nữa chương trình Python đầu tiên của bạn
- Chuỗi và tham số
  - Một vấn đề nhỏ với dấu ngoặc kép
- Chuỗi định dạng
- Thông tin thêm về dây
- Số nguyên hoặc int
- Khả năng đọc thắng
- Khái niệm cơ bản về phao
- Thông tin thêm về Phao
- chắc chắn
- Giá trị trả về
- Tổng hợp

Tạo mã bằng Python

- VS Code là một loại trình soạn thảo văn bản đặc biệt được gọi là trình biên dịch. Ở phía trên, bạn sẽ thấy một trình soạn thảo văn bản và ở phía dưới, bạn sẽ thấy một thiết bị đầu cuối nơi bạn có thể thực thi các lệnh.
- Trong thiết bị đầu cuối, bạn có thể thực thi `code hello.py` để bắt đầu mã hóa.
- Trong trình soạn thảo văn bản ở trên, bạn có thể nhập `print("hello, world")`. Đây là một chương trình kinh điển nổi tiếng mà gần như tất cả các lập trình viên đều viết trong quá trình học tập.
- Trong cửa sổ terminal, bạn có thể thực thi các lệnh. Để chạy chương trình này, bạn cần di chuyển con trỏ xuống cuối màn hình, nhấp vào cửa sổ terminal. Bây giờ bạn có thể gõ lệnh thứ hai trong cửa sổ terminal. Bên cạnh ký hiệu đô la, hãy nhập `python hello.py` và nhấn phím enter trên bàn phím của bạn.
- Hãy nhớ lại, máy tính thực sự chỉ hiểu số không và số một. Do đó, khi bạn chạy `python hello.py`, python sẽ diễn giải văn bản bạn đã tạo `hello.py` và dịch nó thành số 0 và số 1 mà máy tính có thể hiểu được.
- Kết quả chạy `python hello.py` chương trình là `hello, world`.
- Chúc mừng! Bạn vừa tạo chương trình đầu tiên của mình.

Chức năng /Functions

- Hàm là những động từ hoặc hành động mà máy tính hoặc ngôn ngữ máy tính đã biết cách thực hiện.
- Trong `hello.py` chương trình của bạn, `print` hàm biết cách in ra cửa sổ terminal.
- Hàm `print` lấy đối số. Trong trường hợp này, `"hello, world"` là các đối số mà `print` hàm lấy.

Lỗi /Bugs

- Lỗi là một phần tự nhiên của mã hóa. Đây là những sai lầm, vấn đề cần bạn giải quyết! Đừng nản lòng! Đây là một phần của quá trình trở thành một lập trình viên giỏi.
- Hãy tưởng tượng trong `hello.py` chương trình của chúng ta vô tình gõ phải `print("hello, world"` thông báo rằng chúng ta đã bỏ lỡ phần cuối cùng `)` mà trình biên dịch yêu cầu. Nếu tôi cố tình mắc lỗi này, trình biên dịch sẽ xuất ra lỗi trong cửa sổ terminal!
- Thông thường, các thông báo lỗi sẽ thông báo cho bạn về lỗi của bạn và cung cấp cho bạn manh mối về cách khắc phục chúng. Tuy nhiên, sẽ có nhiều lúc trình biên dịch không thuộc loại này.

Cải thiện chương trình Python đầu tiên của bạn

- Chúng tôi có thể cá nhân hóa chương trình Python đầu tiên của bạn.
- Trong trình soạn thảo văn bản của chúng tôi, `hello.py` chúng tôi có thể thêm một chức năng khác. `input` là một hàm lấy dấu nhắc làm đối số. Chúng tôi có thể chỉnh sửa mã của mình để nói

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

- Tuy nhiên, chỉ chỉnh sửa này sẽ không cho phép chương trình của bạn xuất ra những gì người dùng nhập vào. Để làm được điều đó, chúng tôi sẽ cần giới thiệu cho bạn các biến

Biến /Variables

- Một biến chỉ là nơi chứa một giá trị trong chương trình của riêng bạn.
- Trong chương trình của bạn, bạn có thể giới thiệu biến của riêng mình trong chương trình bằng cách chỉnh sửa nó để đọc

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

Lưu ý rằng `=` dấu bằng ở giữa này `name = input("What's your name? ")` có vai trò đặc biệt trong lập trình. Dấu bằng này thực sự gán những gì ở bên phải cho những gì ở bên trái. Do đó, giá trị trả về `input("What's your name? ")` được gán cho `name`.

- Nếu bạn chỉnh sửa mã của mình như sau, bạn sẽ thấy lỗi

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

- Chương trình sẽ quay lại `hello, name` trong cửa sổ terminal bất kể người dùng gõ gì.
- Chỉnh sửa thêm mã của chúng tôi, bạn có thể gõ

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

- Kết quả trong cửa sổ terminal sẽ là

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

- Chúng ta đang tiến gần hơn đến kết quả mà chúng ta mong đợi!

- Bạn có thể tìm hiểu thêm trong tài liệu của Python về các kiểu dữ liệu (https://docs.python.org/3/library/datatypes.html) .

Bình luận

- Nhận xét là một cách để các lập trình viên theo dõi những gì họ đang làm trong chương trình của mình và thậm chí thông báo cho người khác về ý định của họ đối với một khối mã. Nói tóm lại, chúng là những ghi chú cho chính bạn và những người khác sẽ thấy mã của bạn!
- Bạn có thể thêm nhận xét vào chương trình của mình để có thể xem chương trình của bạn đang làm gì. Bạn có thể chỉnh sửa mã của mình như sau:

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

- Nhận xét cũng có thể đóng vai trò là danh sách việc cần làm cho bạn.

Mã giả

- Mã giả là một loại nhận xét quan trọng và trở thành một loại danh sách việc cần làm đặc biệt, đặc biệt khi bạn không hiểu cách hoàn thành một nhiệm vụ viết mã. Ví dụ: trong mã của bạn, bạn có thể chỉnh sửa mã của mình để nói:

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

# Print hello
print("hello,")

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

Cải thiện hơn nữa chương trình Python đầu tiên của bạn

- Chúng ta có thể chỉnh sửa thêm mã của mình như sau:

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

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

- Hóa ra một số hàm có nhiều đối số.

- Chúng ta có thể sử dụng dấu phẩy `,` để chuyển nhiều đối số bằng cách chỉnh sửa mã của mình như sau:

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

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

Đầu ra trong thiết bị đầu cuối, nếu chúng ta gõ "David", chúng ta sẽ có `hello, David`. Thành công.

Chuỗi và tham số

- Một chuỗi, được gọi là a `str` trong Python, là một chuỗi văn bản.
- Tua lại một chút đoạn mã của chúng ta về đoạn sau, có một tác dụng phụ trực quan là kết quả xuất hiện trên nhiều dòng:

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

- Các hàm lấy các đối số ảnh hưởng đến hành vi của chúng. Nếu chúng ta xem tài liệu, `print` (https://docs.python.org/3/library/functions.html#print)bạn sẽ nhận thấy rằng chúng ta có thể tìm hiểu rất nhiều về các đối số mà hàm print sử dụng.
- Xem tài liệu này, bạn sẽ biết rằng hàm in tự động bao gồm một đoạn mã `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` và mặc định là tạo một dòng mới.
- Tuy nhiên, về mặt kỹ thuật, chúng ta có thể đưa ra một lập luận cho `end` chính mình để một dòng mới không được tạo ra!
- Chúng ta có thể sửa đổi mã của mình như sau:

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

Bằng cách cung cấp `end=""`, chúng tôi sẽ ghi đè giá trị mặc định `end` sao cho nó không bao giờ tạo dòng mới sau câu lệnh in đầu tiên này. Cung cấp tên là "David", đầu ra trong cửa sổ terminal sẽ là `hello, David`.

- Do đó, các tham số là các đối số có thể được lấy bởi một hàm.

- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `print` (https://docs.python.org/3/library/functions.html#print).

Một vấn đề nhỏ với dấu ngoặc kép

- Hãy lưu ý rằng việc thêm dấu ngoặc kép vào một phần chuỗi của bạn là một thử thách.
- `print("hello,"friend"")` sẽ không hoạt động và trình biên dịch sẽ báo lỗi.
- Nói chung, có hai cách để khắc phục điều này. Đầu tiên, bạn có thể chỉ cần thay đổi dấu ngoặc kép thành dấu ngoặc đơn.
- Một cách tiếp cận khác, được sử dụng phổ biến hơn sẽ là mã dưới dạng `print("hello, \"friend\"")`. Dấu gạch chéo ngược cho trình biên dịch biết rằng ký tự sau phải được coi là dấu ngoặc kép trong chuỗi và tránh lỗi trình biên dịch.

Chuỗi định dạng /Định dạng các Chuỗi

- Có lẽ cách thanh lịch nhất để sử dụng chuỗi sẽ như sau:

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

Lưu ý `f` trong `print(f"hello, {name}")`. Đây `f` là một chỉ báo đặc biệt để Python xử lý chuỗi này theo một cách đặc biệt, khác với các phương pháp trước đây mà chúng tôi đã minh họa trong bài giảng này. Hy vọng rằng bạn sẽ sử dụng kiểu dây này khá thường xuyên trong khóa học này.

Thông tin thêm về ~~dây~~ Strings/Chuỗi/Xâu

- Bạn không bao giờ nên mong đợi người dùng của mình sẽ hợp tác như dự định. Do đó, bạn sẽ cần đảm bảo rằng thông tin đầu vào của người dùng đã được sửa hoặc kiểm tra.
- Hóa ra, chuỗi được tích hợp sẵn là khả năng loại bỏ khoảng trắng khỏi chuỗi.
- Bằng cách sử dụng phương thức `strip` trên `name` as `name = name.strip()`, nó sẽ loại bỏ tất cả các khoảng trắng ở bên trái và bên phải đầu vào của người dùng. Bạn có thể sửa đổi mã của mình thành:

```
# 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}")
```

Chạy lại chương trình này, bất kể bạn nhập bao nhiêu khoảng trắng trước hay sau tên, nó sẽ loại bỏ tất cả khoảng trắng.

- Sử dụng `title` phương thức này, nó sẽ đặt tiêu đề cho tên người dùng:

```
# 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}")
```

- Đến thời điểm này, bạn có thể đã cảm thấy mệt mỏi khi phải gõ đi gõ `python` lại nhiều lần trong cửa sổ terminal. Bạn khiến chúng tôi sử dụng mũi tên lên trên bàn phím để gọi lại các lệnh đầu cuối gần đây nhất mà bạn đã thực hiện.
- Lưu ý rằng bạn có thể sửa đổi mã của mình để hiệu quả hơn:

```
# 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}")
```

Điều này tạo ra kết quả tương tự như mã trước đó của bạn.

- Chúng tôi thậm chí có thể đi xa hơn!

```
# Ask the user for their name, remove whitespace from the str and capitalize the
name = input("What's your name? ").strip().title()

# Print the output
print(f"hello, {name}")
```

- Bạn có thể tìm hiểu thêm về chuỗi trong tài liệu của Python trên `str` (https://docs.python.org/3/library/stdtypes.html#str)

Số nguyên hoặc int

- Trong Python, một số nguyên được gọi là `int`.
- Trong thế giới toán học, chúng ta đã quen thuộc với các toán tử +, -, *, / và %. Toán tử cuối cùng hoặc toán tử modulo đó `%` có thể không quen thuộc lắm với bạn.
- Bạn không cần phải sử dụng cửa sổ soạn thảo văn bản trong trình biên dịch để chạy mã Python. Xuống trong thiết bị đầu cuối của bạn, bạn có thể chạy `python` một mình. Bạn sẽ được trình bày `>>>` trong cửa sổ terminal. Sau đó bạn có thể chạy mã tương tác trực tiếp. Bạn có thể gõ `1+1` và nó sẽ chạy phép tính đó. Chế độ này sẽ không được sử dụng phổ biến trong khóa học này.
- Mở lại VS Code, chúng ta có thể nhập `code calculator.py` vào terminal. Điều này sẽ tạo một tệp mới trong đó chúng tôi sẽ tạo máy tính của riêng mình.
- Đầu tiên, chúng ta có thể khai báo một vài biến.

```
x = 1
y = 2

z = x + y

print(z)
```

Đương nhiên, khi chạy `python calculator.py` chúng ta nhận được kết quả trong cửa sổ terminal của `3`. Chúng ta có thể làm cho điều này trở nên tương tác hơn bằng cách sử dụng `input` hàm.

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

z = x + y

print(z)
```

- Chạy chương trình này, chúng tôi phát hiện ra rằng đầu ra không chính xác vì `12`. Tại sao điều này có thể xảy ra?
- Trước đây, chúng ta đã thấy `+` dấu nối hai chuỗi như thế nào. Vì dữ liệu đầu vào từ bàn phím trên máy tính của bạn được đưa vào trình biên dịch dưới dạng văn bản nên nó được xử lý dưới dạng chuỗi. Do đó, chúng ta cần chuyển đổi đầu vào này từ một chuỗi thành một số nguyên. Chúng ta có thể làm như vậy như sau:

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

z = int(x) + int(y)

print(z)
```

Kết quả bây giờ là chính xác. Việc sử dụng `int(x)`, được gọi là "truyền" trong đó một giá trị được thay đổi tạm thời từ một loại biến (trong trường hợp này là một chuỗi) sang một loại

biến khác (ở đây là số nguyên).

- Chúng tôi có thể cải thiện hơn nữa chương trình của mình như sau:

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

print(x + y)
```

Điều này minh họa rằng bạn có thể chạy các hàm trên hàm. Hàm bên trong nhất được chạy trước, sau đó mới chạy hàm bên ngoài. Đầu tiên, `input` chức năng được chạy. Sau đó, `int` chức năng.

- Bạn có thể tìm hiểu thêm trong Tài liệu về `int` (https://docs.python.org/3/library/functions.html?highlight=float#int).

Khả năng đọc thắng

- Khi quyết định cách tiếp cận của bạn đối với một nhiệm vụ viết mã, hãy nhớ rằng người ta có thể đưa ra lập luận hợp lý cho nhiều cách tiếp cận đối với cùng một vấn đề.
- Bất kể bạn thực hiện nhiệm vụ lập trình theo cách nào, hãy nhớ rằng mã của bạn phải dễ đọc. Bạn nên sử dụng nhận xét để cung cấp cho bản thân và những người khác manh mối về chức năng mà mã của bạn đang thực hiện. Hơn nữa, bạn nên tạo mã theo cách dễ đọc.

Khái niệm cơ bản về ~~phao~~ Float/Số thực kiểu Float

- Giá trị dấu phẩy động là số thực có dấu thập phân trong đó, chẳng hạn như `0.52`.
- Bạn có thể thay đổi mã của mình để hỗ trợ float như sau:

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

print(x + y)
```

Thay đổi này cho phép người dùng của bạn nhập `1.2` và `3.4` hiển thị tổng cộng `4.6`.

- Tuy nhiên, hãy tưởng tượng rằng bạn muốn làm tròn tổng số đến số nguyên gần nhất. Nhìn vào tài liệu Python, `round` bạn sẽ thấy các đối số khả dụng là `round(number[n, ndigits])`. Những dấu ngoặc vuông đó cho biết rằng người lập trình có thể chỉ định một số tùy chọn. Vì vậy, bạn có thể làm `round(n)` tròn một chữ số đến số nguyên gần nhất của nó. Ngoài ra, bạn có thể viết mã như sau:

```
# 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)
```

Kết quả đầu ra sẽ được làm tròn đến số nguyên gần nhất.

- Điều gì sẽ xảy ra nếu chúng ta muốn định dạng đầu ra của số dài? Ví dụ, thay vì nhìn thấy `1000`, bạn có thể muốn nhìn thấy `1,000`. Bạn có thể sửa đổi mã của mình như sau:

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

Mặc dù khá khó hiểu nhưng điều đó `print(f"{z:,}")` tạo ra một kịch bản trong đó kết quả xuất ra `z` sẽ bao gồm dấu phẩy trong đó kết quả có thể trông giống như `1,000` hoặc `2,500`.

Thông tin thêm về ~~Phao~~ Float

- Làm cách nào chúng ta có thể làm tròn các giá trị dấu phẩy động? Đầu tiên, sửa đổi mã của bạn như sau:

```
# 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)
```

Khi nhập dưới `2` dạng x và `3` y, kết quả z `0.6666666666` dường như tiến tới vô hạn như chúng ta mong đợi.

- Hãy tưởng tượng rằng chúng ta muốn làm tròn số này xuống, chúng ta có thể sửa đổi mã của mình như sau:

```
# 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)
```

Như chúng ta có thể mong đợi, điều này sẽ làm tròn kết quả đến hai chữ số thập phân gần nhất.

- Chúng tôi cũng có thể sử dụng `fstring` để định dạng đầu ra như sau:

```
# 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}")
```

Mã khó hiểu này `fstring` hiển thị giống như chiến lược làm tròn trước đó của chúng tôi.

- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `float` (https://docs.python.org/3/library/functions.html?highlight=float#float).

~~chắc chắn~~ Định nghĩa Hàm/Function

- Sẽ thật tuyệt nếu chúng ta tạo ra các chức năng của riêng mình phải không?
- Hãy lấy lại mã cuối cùng của chúng ta `hello.py` bằng cách gõ `code hello.py` vào cửa sổ terminal. Mã bắt đầu của bạn sẽ trông như sau:

```
# Ask the user for their name, remove whitespace from the str and capitalize the
name = input("What's your name? ").strip().title()

# Print the output
print(f"hello, {name}")
```

Chúng ta có thể cải thiện mã của mình để tạo ra hàm đặc biệt của riêng mình có chức năng "xin chào" dành cho chúng ta!

- Xóa tất cả mã trong trình soạn thảo văn bản của chúng tôi, hãy bắt đầu lại từ đầu:

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

Cố gắng chạy mã này, trình biên dịch của bạn sẽ báo lỗi. Rốt cuộc, không có hàm nào được xác định cho `hello`.

- Chúng ta có thể tạo hàm riêng của mình được gọi `hello` như sau:

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


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

Lưu ý rằng mọi thứ bên dưới `def hello()` đều được thụt vào. Python là một ngôn ngữ thụt lề. Nó sử dụng thụt lề để hiểu đâu là một phần của hàm trên. Vì vậy, mọi thứ trong `hello` hàm đều phải được thụt lề. Khi một cái gì đó không được thụt lề, nó sẽ xử lý nó như thể nó không nằm trong hàm `hello`. Chạy `python hello.py` trong cửa sổ terminal, bạn sẽ thấy đầu ra của mình không chính xác như bạn mong muốn.

- Chúng tôi có thể cải thiện hơn nữa mã của mình:

```


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


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

Ở đây, trong những dòng đầu tiên, bạn đang tạo `hello` hàm của mình. Tuy nhiên, lần này bạn nói với trình biên dịch rằng hàm này nhận một tham số duy nhất: một biến có tên là `to`. Vì vậy, khi bạn gọi `hello(name)` máy tính sẽ chuyển `name` sang `hello` hàm dưới dạng `to`. Đây là cách chúng ta chuyển giá trị vào hàm. Rất hữu dụng! Chạy `python hello.py` trong cửa sổ terminal, bạn sẽ thấy rằng kết quả đầu ra gần hơn với lý tưởng của chúng tôi đã trình bày trước đó trong bài giảng này.

- Chúng tôi có thể thay đổi mã của mình để thêm giá trị mặc định vào `hello`:

```
# 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()
```

Hãy tự kiểm tra mã của bạn. Lưu ý cách đầu tiên `hello` sẽ hoạt động như bạn mong đợi và lời chào thứ hai, không được truyền giá trị, theo mặc định sẽ xuất ra `hello, world`.

- Chúng tôi không cần phải có chức năng của mình khi bắt đầu chương trình. Chúng ta có thể di chuyển nó xuống, nhưng chúng ta cần nói với trình biên dịch rằng chúng ta có một `main` hàm và chúng ta có một `hello` hàm riêng.

```
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)
```

Tuy nhiên, chỉ điều này thôi cũng sẽ tạo ra một số lỗi. Nếu chúng ta chạy thì `python hello.py` không có gì xảy ra cả! Lý do cho điều này là không có gì trong mã này thực sự gọi hàm `main` và đưa chương trình của chúng ta vào cuộc sống.

- Sửa đổi rất nhỏ sau đây sẽ gọi hàm `main` và khôi phục chương trình của chúng tôi về trạng thái hoạt động:

```
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()
```

Giá trị trả về

- Bạn có thể tưởng tượng nhiều tình huống trong đó bạn không chỉ muốn một hàm thực hiện một hành động mà còn trả về một giá trị cho hàm chính. Ví dụ, thay vì chỉ in phép tính của `x`

`+ y`, bạn có thể muốn một hàm trả về giá trị của phép tính này cho một phần khác trong chương trình của bạn. Việc "trả lại" một giá trị này mà chúng ta gọi là `return` giá trị.

- Quay lại `calculator.py` mã của chúng tôi bằng cách gõ `code calculator.py`. Xóa tất cả mã ở đó. Làm lại mã như sau:

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


def square(n):
    return n * n


main()
```

Thực tế, `x` được chuyển đến `square`. Sau đó, phép tính `x * x` được trả về hàm chính.

Tổng hợp

Thông qua bài giảng này, bạn đã học được những khả năng mà bạn sẽ sử dụng vô số lần trong các chương trình của riêng mình. Bạn đã tìm hiểu về...

- Tạo chương trình đầu tiên của bạn bằng Python;
- Chức năng;
- Lỗi;
- Biến;
- Bình luận;
- Mã giả;
- ~~Dây~~; Strings/Chuỗi/Xâu
- Thông số;
- Chuỗi được định dạng;
- Số nguyên;
- Nguyên tắc dễ đọc;
- ~~Phao~~; Float
- Tạo các chức năng của riêng bạn; Và
- Trả về các giá trị.