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 3

Ngoại lệ

- Ngoại lệ là những lỗi xảy ra trong quá trình mã hóa của chúng ta.
- Trong trình soạn thảo văn bản của chúng tôi, gõ `code hello.py` để tạo một tệp mới. Nhập như sau (bao gồm các lỗi cố ý):

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

  Lưu ý rằng chúng tôi cố tình bỏ qua dấu ngoặc kép.
- Đang chạy `python hello.py` trong cửa sổ terminal của chúng tôi, một lỗi xuất hiện. Trình biên dịch cho biết đó là "lỗi cú pháp". Lỗi cú pháp là những lỗi yêu cầu bạn phải kiểm tra kỹ xem bạn đã gõ đúng mã chưa.

- Bạn có thể tìm hiểu thêm trong tài liệu về Lỗi và Ngoại lệ (https://docs.python.org/3/tutorial/errors.html) của Python .

Lỗi thời gian chạy

- Lỗi thời gian chạy đề cập đến những lỗi được tạo ra bởi hành vi không mong muốn trong mã của bạn. Ví dụ: có thể bạn định cho người dùng nhập một số nhưng thay vào đó họ lại nhập một ký tự. Chương trình của bạn có thể gặp lỗi do đầu vào không mong muốn này của người dùng.
- Trong cửa sổ terminal của bạn, hãy chạy `code number.py`. Mã như sau trong trình soạn thảo văn bản của bạn:

```
x = int(input("What's x? "))
print(f"x is {x}")
```

  Lưu ý rằng bằng cách bao gồm `f`, chúng tôi yêu cầu Python nội suy những gì có trong dấu ngoặc nhọn làm giá trị của `x`. Hơn nữa, khi kiểm tra mã của mình, bạn có thể tưởng tượng cách người ta có thể dễ dàng nhập một chuỗi hoặc một ký tự thay vì một số. Thậm chí, người dùng vẫn không thể gõ gì cả - chỉ cần nhấn phím enter.
- Với tư cách là lập trình viên, chúng ta nên phòng thủ để đảm bảo rằng người dùng của chúng ta đang nhập những gì chúng ta mong đợi. Chúng ta có thể xem xét các "trường hợp góc" như `-1`, `0`, hoặc `cat`.
- Nếu chúng ta chạy chương trình này và gõ "cat", chúng ta sẽ đột nhiên thấy `ValueError: invalid literal for int() with base 10: 'cat'` Về cơ bản, trình thông dịch Python không thích việc chúng ta chuyển "cat" cho hàm `print`.
- Một chiến lược hiệu quả để khắc phục lỗi tiềm ẩn này là tạo ra "xử lý lỗi" để đảm bảo người dùng hành xử như chúng tôi dự định.
- Bạn có thể tìm hiểu thêm trong tài liệu về Lỗi và Ngoại lệ (https://docs.python.org/3/tutorial/errors.html) của Python .

`try`

- Trong Python `try` và `except` là những cách kiểm tra đầu vào của người dùng trước khi xảy ra sự cố. Sửa đổi mã của bạn như sau:

```
try:
    x = int(input("What's x?"))
    print(f"x is {x}")
except ValueError:
    print("x is not an integer")
```

Lưu ý cách chạy mã này, việc nhập `50` sẽ được chấp nhận. Tuy nhiên, việc nhập `cat` sẽ tạo ra lỗi hiển thị cho người dùng, hướng dẫn họ lý do tại sao dữ liệu nhập của họ không được chấp nhận.

- Đây vẫn không phải là cách tốt nhất để triển khai mã này. Lưu ý rằng chúng tôi đang cố gắng thực hiện hai dòng mã. Để có cách thực hành tốt nhất, chúng tôi chỉ nên `try` có ít dòng mã nhất có thể mà chúng tôi lo ngại có thể bị lỗi. Điều chỉnh mã của bạn như sau:

```
try:
    x = int(input("What's x?"))
except ValueError:
    print("x is not an integer")

print(f"x is {x}")
```

Lưu ý rằng mặc dù điều này hoàn thành mục tiêu cố gắng càng ít dòng càng tốt, nhưng giờ đây chúng ta lại gặp phải một lỗi mới! Chúng ta phải đối mặt với một `NameError` nơi `x is not defined`. Hãy xem mã này và cân nhắc: Tại sao x không được xác định trong một số trường hợp?

- Thật vậy, nếu bạn kiểm tra thứ tự các thao tác trong `x = int(input("What's x?"))`, tính từ phải sang trái, nó có thể nhận một ký tự được nhập sai và cố gán nó làm số nguyên. Nếu điều này không thành công, việc gán giá trị sẽ `x` không bao giờ xảy ra. Do đó, không có x để in trên dòng mã cuối cùng của chúng ta.

`else`

- Hóa ra là có một cách triển khai khác `try` có thể phát hiện ra lỗi kiểu này.
- Điều chỉnh mã của bạn như sau:

```
try:
    x = int(input("What's x?"))
except ValueError:
    print("x is not an integer")
else:
    print(f"x is {x}")
```

Lưu ý rằng nếu không có ngoại lệ nào xảy ra thì nó sẽ chạy khối mã trong `else`. Chạy `python number.py` và cung cấp `50`, bạn sẽ nhận thấy kết quả sẽ được in. Thử lại, lần này cung cấp `cat`, bạn sẽ nhận thấy rằng chương trình hiện đã gặp lỗi.

- Khi xem xét việc cải thiện mã của chúng tôi, hãy lưu ý rằng chúng tôi đang hơi thô lỗ với người dùng của mình. Nếu người dùng của chúng tôi không hợp tác, chúng tôi sẽ kết thúc chương trình của mình. Hãy xem xét cách chúng ta có thể sử dụng vòng lặp để nhắc người dùng `x` và liệu họ có nhắc lại hay không! Cải thiện mã của bạn như sau:

```
while True:
    try:
        x = int(input("What's x?"))
    except ValueError:
        print("x is not an integer")
    else:
        break

print(f"x is {x}")
```

Lưu ý rằng `while True` sẽ lặp lại mãi mãi. Nếu người dùng thành công trong việc cung cấp đầu vào chính xác, chúng ta có thể thoát khỏi vòng lặp và sau đó in đầu ra. Bây giờ, người dùng nhập nội dung nào đó không chính xác sẽ được yêu cầu nhập lại.

Tạo hàm lấy số nguyên

- Chắc chắn có nhiều lúc chúng ta muốn lấy một số nguyên từ người dùng của mình. Sửa đổi mã của bạn như sau:

```
def main():
    x = get_int()
    print(f"x is {x}")


def get_int():
    while True:
        try:
            x = int(input("What's x?"))
        except ValueError:
            print("x is not an integer")
        else:
            break
    return x


main()
```

Lưu ý rằng chúng ta đang thể hiện nhiều đặc tính tuyệt vời. Đầu tiên, chúng ta đã loại bỏ khả năng lấy số nguyên. Bây giờ, toàn bộ chương trình này tóm gọn lại ba dòng đầu tiên của chương trình.

- Thậm chí, chúng tôi vẫn có thể cải thiện chương trình này. Hãy cân nhắc xem bạn có thể làm gì khác để cải thiện chương trình này. Sửa đổi mã của bạn như sau:

```
def main():
    x = get_int()
    print(f"x is {x}")
```

```
def get_int():
    while True:
        try:
            x = int(input("What's x?"))
        except ValueError:
            print("x is not an integer")
        else:
            return x


main()
```

Lưu ý rằng điều đó `return` không chỉ giúp bạn thoát khỏi vòng lặp mà còn trả về một giá trị.

- Một số người có thể cho rằng bạn có thể làm như sau:

```
def main():
    x = get_int()
    print(f"x is {x}")


def get_int():
    while True:
        try:
            return int(input("What's x?"))
        except ValueError:
            print("x is not an integer")


main()
```

Lưu ý rằng điều này thực hiện tương tự như lần lặp lại mã trước đó của chúng tôi, chỉ đơn giản là với ít dòng hơn.

pass

- Chúng tôi có thể làm cho mã của chúng tôi không cảnh báo người dùng mà chỉ cần hỏi lại họ câu hỏi nhắc nhở bằng cách sửa đổi mã của chúng tôi như sau:

```
def main():
    x = get_int()
    print(f"x is {x}")


def get_int():
    while True:
        try:
            return int(input("What's x?"))
        except ValueError:
            pass
```

```
main()
```

Lưu ý rằng mã của chúng tôi sẽ vẫn hoạt động nhưng sẽ không liên tục thông báo cho người dùng về lỗi của họ. Trong một số trường hợp, bạn sẽ muốn nói rõ cho người dùng biết lỗi nào đang được tạo ra. Những lúc khác, bạn có thể quyết định rằng bạn chỉ muốn yêu cầu họ cung cấp lại thông tin.

- Một sàng lọc cuối cùng có thể cải thiện việc thực hiện `get_int` chức năng này. Ngay bây giờ, hãy lưu ý rằng chúng ta hiện đang dựa vào hệ thống danh dự có `x` cả chức năng `main` và `get_int`. Chúng tôi có thể muốn chuyển lời nhắc mà người dùng nhìn thấy khi được yêu cầu nhập liệu. Sửa đổi mã của bạn như sau.

```
def main():
    x = get_int("What's x? ")
    print(f"x is {x}")


def get_int(prompt):
    while True:
        try:
            return int(input(prompt))
        except ValueError:
            pass


main()
```

- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `pass` (https://docs.python.org/3/tutorial/controlflow.html#pass-statements).

Tổng hợp

Lỗi là không thể tránh khỏi trong mã của bạn. Tuy nhiên, bạn có cơ hội sử dụng những gì đã học được ngày hôm nay để giúp ngăn ngừa những lỗi này. Trong bài giảng này, bạn đã tìm hiểu về…

- Ngoại lệ
- Lỗi giá trị
- Lỗi thời gian chạy
- `try`
- `else`
- `pass`