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 1

Câu điều kiện

- Các điều kiện cho phép bạn, người lập trình, cho phép chương trình của bạn đưa ra quyết định: Như thể chương trình của bạn có quyền lựa chọn giữa việc đi bên trái hoặc đi bên phải dựa trên các điều kiện nhất định.
- Được xây dựng trong Python là một tập hợp các "toán tử" có thể được sử dụng để đặt các câu hỏi toán học.
- `>` và `<` các biểu tượng có lẽ khá quen thuộc với bạn.
- `>=` biểu thị "lớn hơn hoặc bằng".

- `<=` biểu thị "nhỏ hơn hoặc bằng".
- `==` biểu thị "bằng, tuy nhiên hãy chú ý dấu bằng kép! Một dấu bằng duy nhất sẽ gán một giá trị. Dấu bằng kép được sử dụng để so sánh các biến.
- `!=` biểu thị "không bằng.
- Câu lệnh điều kiện so sánh một thuật ngữ bên trái với một thuật ngữ bên phải.

câu lệnh if

- Trong cửa sổ terminal của bạn, gõ `code compare.py`. Thao tác này sẽ tạo một tệp hoàn toàn mới có tên là "so sánh".
- Trong cửa sổ soạn thảo văn bản, hãy bắt đầu bằng lệnh sau:

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

if x < y:
    print("x is less than y")
```

  Lưu ý cách chương trình của bạn lấy dữ liệu đầu vào của người dùng cho cả x và y, chuyển chúng thành số nguyên và lưu chúng vào các biến x và y tương ứng. Sau đó, `if` câu lệnh so sánh x và y. Nếu điều kiện của `x < y` được đáp ứng, `print` câu lệnh sẽ được thực thi.
- Câu lệnh if sử dụng `bool` các giá trị boolean (true hoặc false) để quyết định có thực thi hay không. Nếu câu lệnh `x > y` là đúng, trình biên dịch sẽ đăng ký nó `true` và thực thi mã.

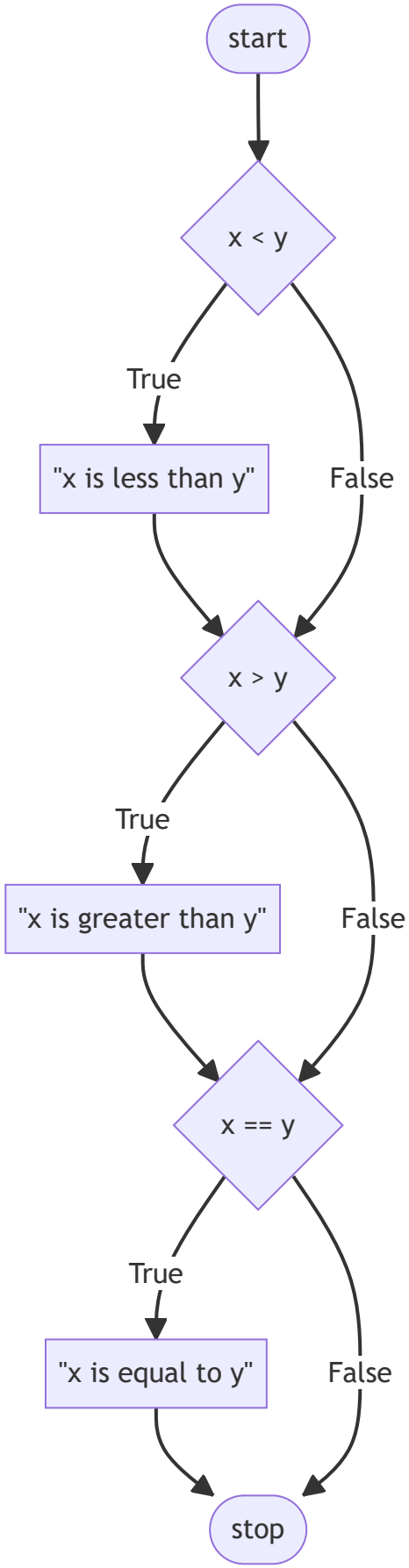
Kiểm soát luồng, Elif và những thứ khác

- Sửa đổi thêm mã của bạn như sau:

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

if x < y:
    print("x is less than y")
if x > y:
    print("x is greater than y")
if x == y:
    print("x is equal to y")
```

  Hãy chú ý cách bạn đưa ra một loạt `if` các câu phát biểu. Đầu tiên, `if` tuyên bố đầu tiên được đánh giá. Sau đó, `if` câu lệnh thứ hai chạy đánh giá của nó. Cuối cùng, `if` câu lệnh cuối cùng chạy đánh giá của nó. Luồng quyết định này được gọi là "luồng điều khiển".
- Mã của chúng tôi có thể được biểu diễn như sau:



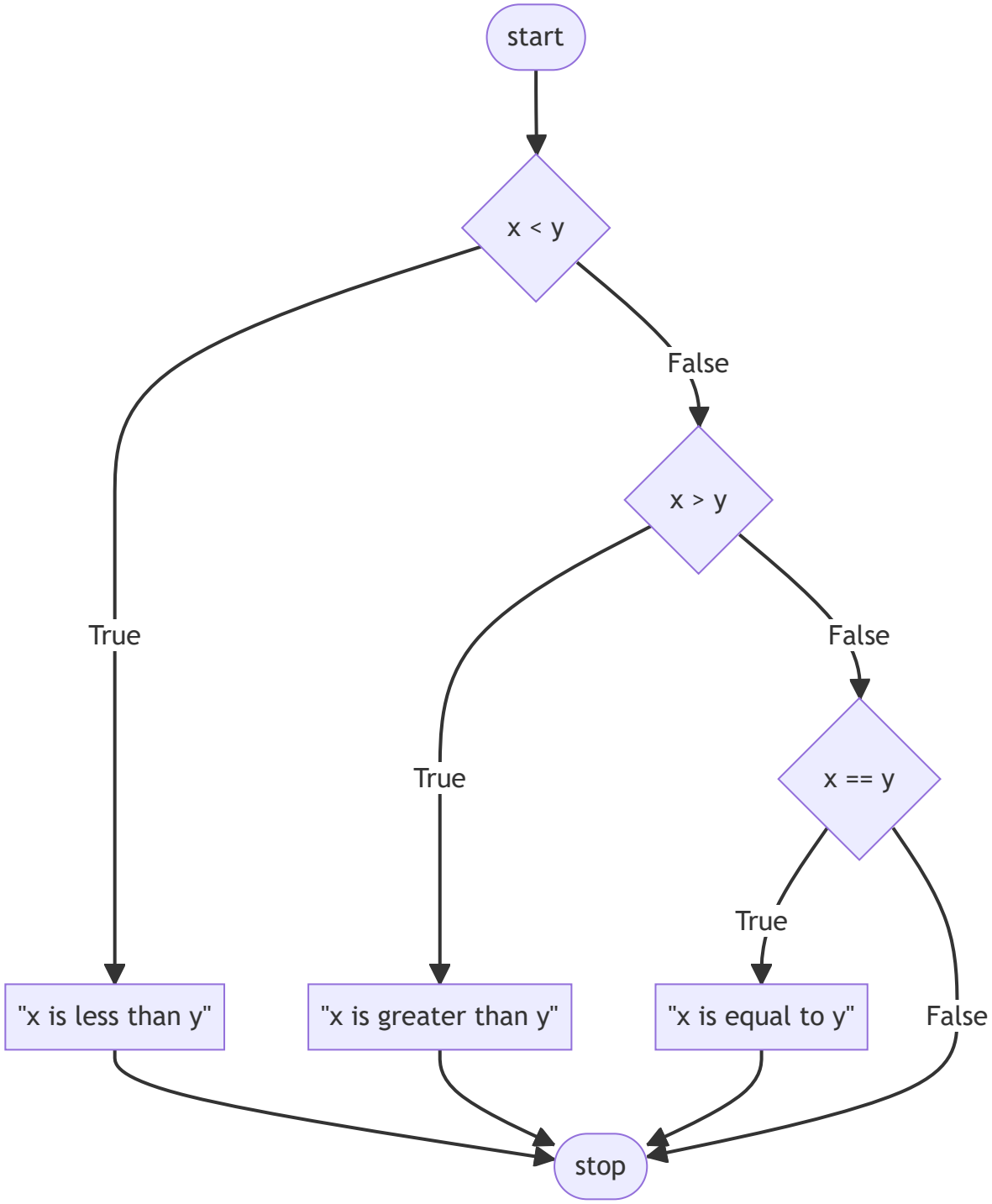
- Chương trình này có thể được cải thiện bằng cách không hỏi ba câu hỏi liên tiếp. Rốt cuộc, không phải cả ba câu hỏi đều có thể có kết quả đúng! Sửa lại chương trình của bạn như sau:

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

if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
elif x == y:
    print("x is equal to y")
```

Lưu ý cách sử dụng `elif` cho phép chương trình đưa ra ít quyết định hơn. Đầu tiên, `if` tuyên bố được đánh giá. Nếu tuyên bố này được cho là đúng, tất cả các `elif` câu lệnh sẽ không được chạy. Tuy nhiên, nếu `if` câu lệnh được đánh giá và phát hiện là sai thì câu lệnh đầu tiên `elif` sẽ được đánh giá. Nếu điều này đúng, nó sẽ không chạy đánh giá cuối cùng.

- Mã của chúng tôi có thể được biểu diễn như sau:



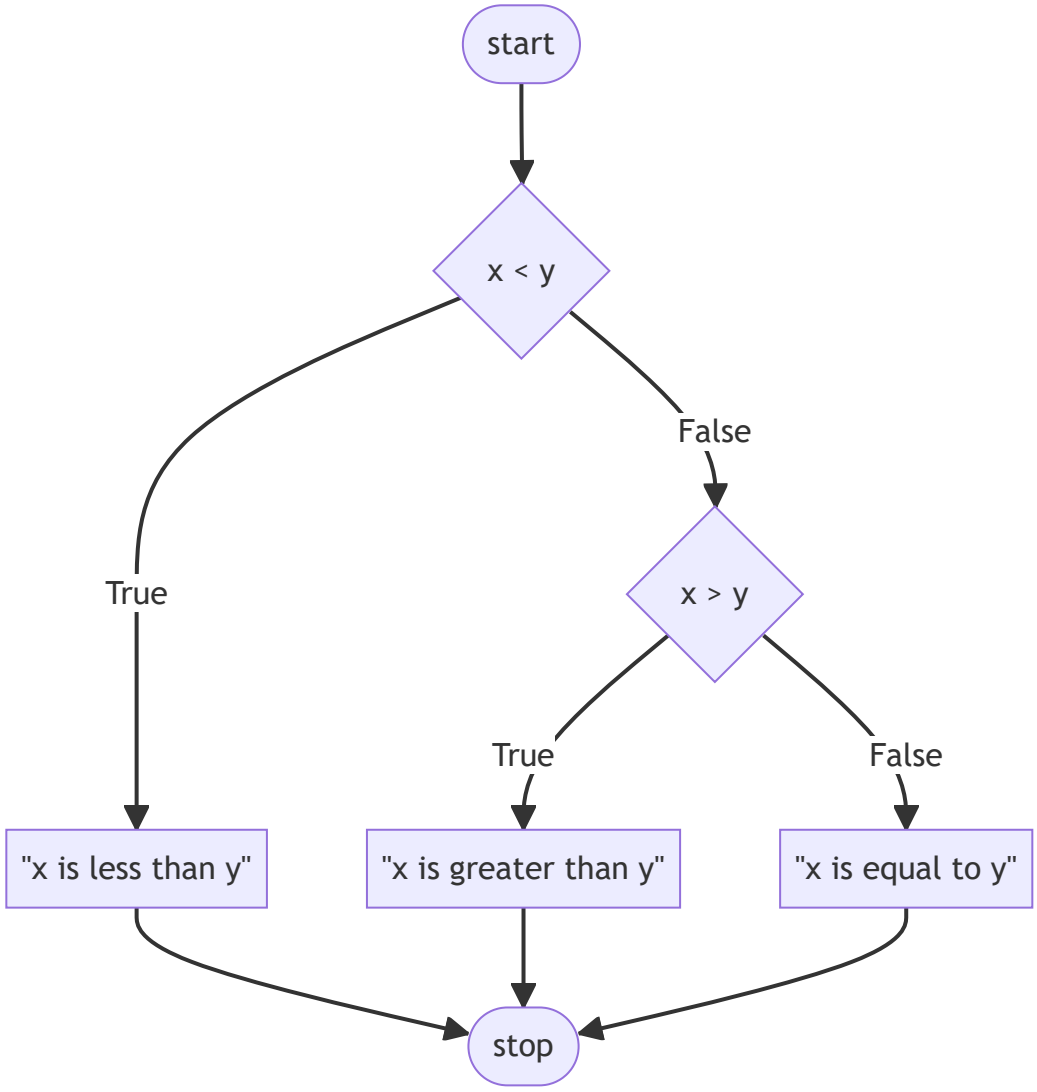
- Mặc dù máy tính của bạn có thể không nhận thấy sự khác biệt về tốc độ giữa chương trình đầu tiên của chúng tôi và chương trình sửa đổi này, hãy xem xét cách một máy chủ trực tuyến chạy hàng tỷ hoặc hàng nghìn tỷ loại phép tính này mỗi ngày chắc chắn có thể bị ảnh hưởng bởi một quyết định mã hóa nhỏ như vậy.
- Có một cải tiến cuối cùng mà chúng tôi có thể thực hiện cho chương trình của mình. `elif x == y` Lưu ý rằng việc đánh giá không cần thiết phải thực hiện một cách hợp lý như thế nào . Xét cho cùng, nếu về mặt logic thì x không nhỏ hơn y VÀ x không lớn hơn y, thì x PHẢI bằng y. Vì vậy, chúng ta không cần phải chạy `elif x == y` . Chúng ta có thể tạo một kết quả mặc định "tất cả" bằng cách sử dụng một `else` câu lệnh. Chúng ta có thể sửa lại như sau:

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

if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x is equal to y")
```

Hãy lưu ý rằng độ phức tạp tương đối của chương trình này đã giảm đi như thế nào qua bản sửa đổi của chúng tôi.

- Mã của chúng tôi có thể được biểu diễn như sau:



hoặc

- `or` cho phép chương trình của bạn quyết định giữa một hoặc nhiều lựa chọn thay thế. Ví dụ: chúng tôi có thể chỉnh sửa thêm chương trình của mình như sau:

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

if x < y or x > y:
    print("x is not equal to y")
else:
    print("x is equal to y")
```

Lưu ý rằng kết quả của chương trình của chúng tôi là như nhau, nhưng độ phức tạp giảm đi và hiệu quả mã của chúng tôi tăng lên.

- Tại thời điểm này, mã của chúng tôi khá tuyệt vời. Tuy nhiên, thiết kế có thể được cải thiện hơn nữa? Chúng tôi có thể chỉnh sửa thêm mã của mình như sau:

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

if x != y:
    print("x is not equal to y")
else:
    print("x is equal to y")
```

Hãy chú ý cách chúng tôi loại bỏ `or` hoàn toàn và chỉ hỏi "x có bằng y không?" Chúng tôi hỏi một và chỉ một câu hỏi. Rất hiệu quả!

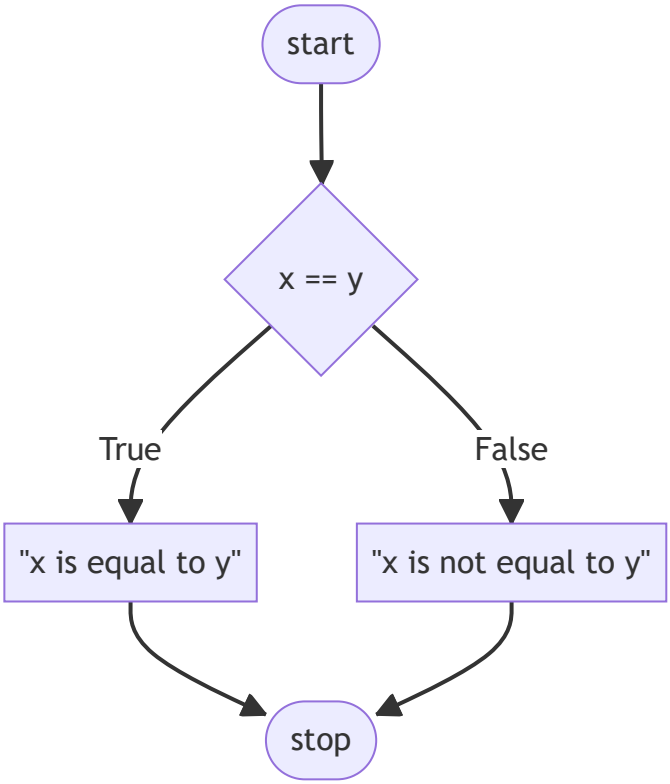
- Với mục đích minh họa, chúng ta cũng có thể thay đổi mã của mình như sau:

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

if x == y:
    print("x is equal to y")
else:
    print("x is not equal to y")
```

Lưu ý rằng `==` toán tử đánh giá xem phần bên trái và bên phải có bằng nhau hay không. Việc sử dụng dấu bằng kép là rất quan trọng. Nếu bạn chỉ sử dụng một dấu bằng, trình biên dịch có thể sẽ đưa ra lỗi.

- Mã của chúng tôi có thể được minh họa như sau:



Và

- Tương tự như `or`, `and` có thể được sử dụng trong các câu lệnh có điều kiện.
- Thực hiện trong cửa sổ terminal `code grade.py`. Bắt đầu chương trình mới của bạn như sau:

```
score = int(input("Score: "))

if score >= 90 and score <= 100:
    print("Grade: A")
elif score >=80 and score < 90:
    print("Grade: B")
elif score >=70 and score < 80:
    print("Grade: C")
elif score >=60 and score < 70:
    print("Grade: D")
else:
    print("Grade: F")
```

  Lưu ý rằng khi thực hiện, `python grade.py` bạn sẽ có thể nhập điểm và nhận điểm. Tuy nhiên, hãy chú ý xem có khả năng xảy ra lỗi như thế nào.

- Thông thường, chúng tôi không bao giờ muốn tin tưởng người dùng của mình sẽ nhập thông tin chính xác. Chúng tôi có thể cải thiện mã của mình như sau:

```
score = int(input("Score: "))

if 90 <= score <= 100:
```

```
        print("Grade: A")
    elif 80 <= score < 90:
        print("Grade: B")
    elif 70 <= score < 80:
        print("Grade: C")
    elif 60 <= score < 70:
        print("Grade: D")
    else:
        print("Grade: F")
```

Lưu ý cách Python cho phép bạn xâu chuỗi các toán tử và điều kiện lại với nhau theo cách khá hiếm gặp đối với các ngôn ngữ lập trình khác.

- Tuy nhiên, chúng tôi có thể cải thiện hơn nữa chương trình của mình:

```
score = int(input("Score: "))

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
elif score >= 60:
    print("Grade: D")
else:
    print("Grade: F")
```

Lưu ý cách chương trình được cải thiện bằng cách đặt ít câu hỏi hơn. Điều này làm cho chương trình của chúng tôi dễ đọc hơn và dễ bảo trì hơn trong tương lai.

- Bạn có thể tìm hiểu thêm trong tài liệu của Python về luồng điều khiển (https://docs.python.org/3/tutorial/controlflow.html) .

Modulo

- Trong toán học, tính chẵn lẻ đề cập đến việc một số là chẵn hay lẻ.
- Toán tử modulo `%` trong lập trình cho phép người ta xem hai số chia đều hay chia và có phần dư.
- Ví dụ: 4 % 2 sẽ cho kết quả bằng 0, vì nó chia đều. Tuy nhiên, 3 % 2 không chia đều và sẽ dẫn đến một số khác 0!
- Trong cửa sổ terminal, tạo một chương trình mới bằng cách gõ `code parity.py`. Trong cửa sổ soạn thảo văn bản, nhập mã của bạn như sau:

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

if x % 2 == 0:
    print("Even")
```

```
else:
    print("Odd")
```

Lưu ý cách người dùng của chúng tôi có thể nhập bất kỳ số 1 hoặc lớn hơn nào để xem số đó là số chẵn hay số lẻ.

Tạo hàm chẵn lẻ của riêng chúng ta

- Như đã thảo luận trong Bài giảng 0, bạn sẽ thấy việc tạo một chức năng của riêng mình là rất hữu ích!
- Chúng ta có thể tạo hàm riêng để kiểm tra xem một số là chẵn hay lẻ. Điều chỉnh mã của bạn như sau:

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
        print("Odd")


def is_even(n):
    if n % 2 == 0:
        return True
    else:
        return False


main()
```

Lưu ý rằng một lý do khiến `if` câu lệnh của chúng tôi `is_even(x)` hoạt động, mặc dù không có toán tử nào ở đó. Điều này là do hàm của chúng ta trả về một `bool` (hoặc boolean), đúng hoặc sai, quay lại hàm chính. Câu `if` lệnh chỉ đơn giản đánh giá xem `is_even` of `x` là đúng hay sai.

Pythonic

- Trong thế giới lập trình, có những loại lập trình được gọi là "Pythonic". Tức là có nhiều cách lập trình đôi khi chỉ thấy trong lập trình Python. Hãy xem xét bản sửa đổi sau đây cho chương trình của chúng tôi:

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
```

```
        print("Odd")


def is_even(n):
    return True if n % 2 == 0 else False


main()
```

Lưu ý rằng câu lệnh return này trong mã của chúng ta gần giống như một câu trong tiếng Anh. Đây là cách mã hóa độc đáo chỉ thấy trong Python.

- Chúng ta có thể sửa đổi thêm mã của mình và làm cho nó ngày càng dễ đọc hơn:

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
        print("Odd")


def is_even(n):
    return n % 2 == 0


main()
```

Lưu ý rằng chương trình sẽ đánh giá những gì đang xảy ra bên trong `n % 2 == 0` là đúng hoặc sai và chỉ cần trả lại điều đó cho hàm chính.

`match`

- Tương tự như `if`, `elif` và `else` các câu lệnh, `match` các câu lệnh có thể được sử dụng để chạy mã có điều kiện khớp với các giá trị nhất định.
- Hãy xem xét chương trình sau:

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

if name == "Harry":
    print("Gryffindor")
elif name == "Hermione":
    print("Gryffindor")
elif name == "Ron":
    print("Gryffindor")
elif name == "Draco":
    print("Slytherin")
else:
    print("Who?")
```

Lưu ý ba câu lệnh điều kiện đầu tiên in ra cùng một phản hồi.

- Chúng tôi có thể cải thiện mã này một chút bằng cách sử dụng từ `or` khóa:

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

if name == "Harry" or name == "Hermione" or name == "Ron":
    print("Gryffindor")
elif name == "Draco":
    print("Slytherin")
else:
    print("Who?")
```

Lưu ý rằng số lượng `elif` câu lệnh đã giảm, cải thiện khả năng đọc mã của chúng tôi.

- Ngoài ra, chúng ta có thể sử dụng `match` các câu lệnh để ánh xạ tên tới các ngôi nhà. Hãy xem xét đoạn mã sau:

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

match name:
    case "Harry":
        print("Gryffindor")
    case "Hermione":
        print("Gryffindor")
    case "Ron":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```

Lưu ý việc sử dụng `_` ký hiệu trong trường hợp cuối cùng. Điều này sẽ khớp với bất kỳ đầu vào nào, dẫn đến hành vi tương tự như một `else` câu lệnh.

- Câu lệnh so khớp so sánh giá trị theo sau `match` từ khóa với từng giá trị theo sau `case` từ khóa. Trong trường hợp tìm thấy kết quả khớp, phần mã thụt lề tương ứng sẽ được thực thi và chương trình sẽ dừng việc so khớp.
- Chúng tôi có thể cải thiện mã:

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

match name:
    case "Harry" | "Hermione" | "Ron":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```

Lưu ý, việc sử dụng thanh dọc đơn `|`. Giống như `or` từ khóa, điều này cho phép chúng ta kiểm tra nhiều giá trị trong cùng một `case` câu lệnh.

Tổng hợp

Giờ đây, trong Python, bạn có khả năng sử dụng các câu lệnh có điều kiện để đặt câu hỏi và yêu cầu chương trình của bạn thực hiện hành động tương ứng. Trong bài giảng này, chúng tôi đã thảo luận...

- Điều kiện;
- `if` Các câu lệnh;
- Kiểm soát luồng, `elif` và `else` ;
- `or` ;
- `and` ;
- Modulo;
- Tạo chức năng của riêng bạn;
- Mã hóa Pythonic;
- và `match` .