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 2

Vòng lặp

- Về cơ bản, vòng lặp là một cách để làm đi làm lại một việc gì đó.
- Bắt đầu bằng cách gõ `code cat.py` vào cửa sổ terminal.
- Trong trình soạn thảo văn bản, hãy bắt đầu bằng đoạn mã sau:

```
print("meow")
print("meow")
print("meow")
```

Chạy mã này bằng cách gõ `python cat.py`, bạn sẽ nhận thấy chương trình kêu meo meo ba lần.

- Khi phát triển với tư cách là một lập trình viên, bạn muốn xem xét cách một người có thể cải thiện các vùng mã của mình khi người ta gõ đi gõ lại cùng một thứ. Hãy tưởng tượng nơi mà một người có thể muốn "meo meo" 500 lần. Có hợp lý không nếu gõ đi gõ `print("meow")` lại cùng một biểu thức đó?
- Vòng lặp cho phép bạn tạo một khối mã thực thi lặp đi lặp lại.

Vòng lặp while

- Vòng `while` lặp gần như phổ biến trong tất cả các ngôn ngữ ~~mã hóa~~ lập trình.
- Vòng lặp như vậy sẽ lặp đi lặp lại một khối mã.
- Trong cửa sổ soạn thảo văn bản, hãy chỉnh sửa mã của bạn như sau:

```
i = 3
while i != 0:
    print("meow")
```

Hãy chú ý rằng mặc dù mã này sẽ thực thi `print("meow")` nhiều lần nhưng nó sẽ không bao giờ dừng lại! Nó sẽ lặp lại mãi mãi. `while` vòng lặp hoạt động bằng cách liên tục hỏi xem điều kiện của vòng lặp có được đáp ứng hay không. Trong trường hợp này, trình biên dịch hỏi "không `i` bằng 0?" Khi bạn bị mắc kẹt trong một vòng lặp thực thi vĩnh viễn, bạn có thể nhấn `control-c` trên bàn phím để thoát ra khỏi vòng lặp.

- Để khắc phục vòng lặp kéo dài mãi mãi này, chúng ta có thể chỉnh sửa mã của mình như sau

```
i = 3
while i != 0:
  print("meow")
  i = i - 1
```

Lưu ý rằng bây giờ mã của chúng tôi thực thi đúng cách, giảm `i` đi `1` cho mỗi lần "lặp" qua vòng lặp. Thuật ngữ lặp lại này có ý nghĩa đặc biệt trong ~~mã hóa~~ lập trình. Khi lặp lại, chúng tôi muốn nói đến một chu kỳ xuyên suốt vòng lặp. Lần lặp đầu tiên là lần lặp "0" xuyên suốt vòng lặp. Lần thứ hai là lần lặp "thứ nhất". Trong lập trình, chúng ta đếm bắt đầu bằng 0, rồi 1, rồi 2.

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

```
i = 1
while i <= 3:
    print("meow")
    i = i + 1
```

Lưu ý rằng khi viết mã, `i = i + 1` chúng ta gán giá trị `i` từ phải sang trái. Ở trên, chúng ta bắt đầu `i` từ một, giống như hầu hết con người đếm (1, 2, 3). Nếu bạn thực thi đoạn mã trên,

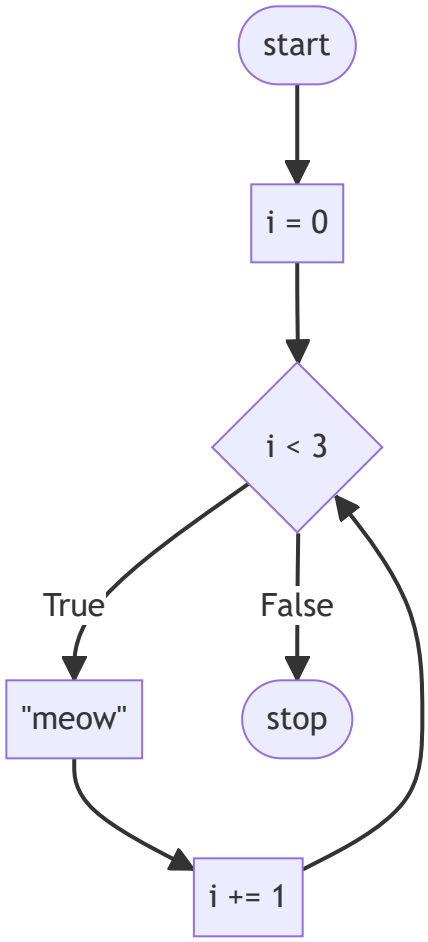
bạn sẽ thấy nó kêu meo meo ba lần. Cách tốt nhất trong lập trình là bắt đầu đếm bằng số 0.

- Chúng tôi có thể cải thiện mã của mình để bắt đầu đếm bằng 0:

```
i = 0
while i < 3:
    print("meow")
    i += 1
```

Lưu ý cách thay đổi toán tử để `i < 3` cho phép mã của chúng ta hoạt động như dự kiến. Chúng ta bắt đầu đếm bằng 0 và nó lặp lại vòng lặp của chúng ta ba lần, tạo ra ba tiếng meo meo. Ngoài ra, hãy chú ý cách `i += 1` nói cũng giống như nói `i = i + 1`.

- Mã của chúng tôi tại thời điểm này được minh họa như sau:



Lưu ý cách vòng lặp của chúng ta đếm i lên tới, nhưng không qua `3`.

Đối với vòng lặp

- Vòng `for` lặp là một loại vòng lặp khác.
- Để hiểu rõ nhất về `for` vòng lặp, tốt nhất bạn nên bắt đầu bằng cách nói về một loại biến mới gọi là a `list` trong Python. Giống như những lĩnh vực khác trong cuộc sống, chúng ta có thể có danh sách thực phẩm, danh sách việc cần làm, v.v.

- Một `for` vòng lặp lặp qua một `list` trong các mục. Ví dụ: trong cửa sổ soạn thảo văn bản, hãy sửa đổi `cat.py` mã của bạn như sau:

```
for i in [0, 1, 2]:
    print("meow")
```

  Hãy chú ý xem mã này sạch như thế nào so với `while` mã vòng lặp trước đó của bạn. Trong mã này, `i` bắt đầu bằng `0`, meo meo, `i` được chỉ định `1`, meo meo và cuối cùng, `i` được chỉ định `2`, meo meo và sau đó kết thúc.

- Mặc dù mã này thực hiện được những gì chúng ta muốn nhưng vẫn có một số khả năng để cải thiện mã của chúng ta cho những trường hợp cực đoan. Thoạt nhìn, mã của chúng tôi trông rất tuyệt. Tuy nhiên, nếu bạn muốn lặp lại tới một triệu thì sao? Tốt nhất là tạo mã có thể hoạt động với những trường hợp cực đoan như vậy. Theo đó, chúng tôi có thể cải thiện mã của mình như sau:

```
for i in range(3):
    print("meow")
```

  Lưu ý cách tự động `range(3)` cung cấp lại ba giá trị (`0`, `1`, và `2`). Mã này sẽ thực thi và tạo ra hiệu ứng như mong muốn, kêu meo meo ba lần.

- Mã của chúng tôi có thể được cải thiện hơn nữa. Lưu ý cách chúng tôi không bao giờ sử dụng `i` rõ ràng trong mã của mình. Nghĩa là, mặc dù Python cần `i` nơi này để lưu trữ số lần lặp của vòng lặp nhưng chúng tôi không bao giờ sử dụng nó cho bất kỳ mục đích nào khác. Trong Python, nếu một biến như vậy không có bất kỳ ý nghĩa nào khác trong mã của chúng ta, chúng ta có thể biểu thị biến này dưới dạng một dấu gạch dưới duy nhất `_`. Do đó, bạn có thể sửa đổi mã của mình như sau:

```
for _ in range(3):
    print("meow")
```

  Lưu ý việc thay đổi `i` thành `_` không có tác động gì đến hoạt động của chương trình của chúng tôi.

- Mã của chúng tôi có thể được cải thiện hơn nữa. Để mở rộng tâm trí của bạn đến các khả năng trong Python, hãy xem xét đoạn mã sau:

```
print("meow" * 3)
```

  Hãy chú ý nó sẽ kêu meo meo ba lần nhưng chương trình sẽ tạo ra `meowmeowmeow` kết quả. Hãy cân nhắc: Làm cách nào bạn có thể tạo ngắt dòng ở cuối mỗi tiếng meo meo?

- Thật vậy, bạn có thể chỉnh sửa mã của mình như sau:

```
print("meow\n" * 3, end="")
```

Hãy chú ý cách mã này tạo ra ba tiếng meo meo, mỗi tiếng trên một dòng riêng biệt. Bằng cách thêm `end=""` và, `\n` chúng tôi yêu cầu trình biên dịch thêm dấu ngắt dòng vào cuối mỗi tiếng meo meo.

Cải thiện bằng đầu vào của người dùng

- Có lẽ chúng tôi muốn nhận được thông tin đầu vào từ người dùng của mình. Chúng ta có thể sử dụng vòng lặp như một cách xác thực thông tin đầu vào của người dùng.
- Một mô hình phổ biến trong Python là sử dụng `while` vòng lặp để xác thực dữ liệu đầu vào của người dùng.
- Ví dụ: hãy thử nhắc người dùng nhập một số lớn hơn hoặc bằng 0:

```
while True:
    n = int(input("What's n? "))
    if n < 0:
        continue
    else:
        break
```

- Lưu ý rằng chúng tôi đã giới thiệu hai từ khóa mới trong Python `continue` và `break`. `continue` yêu cầu Python chuyển sang lần lặp tiếp theo của vòng lặp. `break` mặt khác, yêu cầu Python "thoát ra" vòng lặp sớm trước khi nó hoàn thành tất cả các lần lặp của nó. Trong trường hợp này, chúng ta sẽ `continue` chuyển sang lần lặp tiếp theo của vòng lặp khi `n` nhỏ hơn 0—cuối cùng sẽ nhắc lại người dùng bằng "What's n?". Tuy nhiên, nếu `n` lớn hơn hoặc bằng 0, chúng tôi sẽ `break` thoát khỏi vòng lặp và cho phép phần còn lại của chương trình chạy.
- Hóa ra `continue` từ khóa là dư thừa trong trường hợp này. Chúng tôi có thể cải thiện mã của mình như sau:

```
while True:
    n = int(input("What's n? "))
    if n > 0:
        break

for _ in range(n):
    print("meow")
```

Lưu ý rằng vòng lặp while này sẽ luôn chạy (mãi mãi) cho đến khi `n` lớn hơn `0`. Khi `n` lớn hơn `0`, vòng lặp sẽ bị ngắt.

- Dựa vào kiến thức đã học trước đây, chúng ta có thể sử dụng các hàm để cải thiện mã của mình hơn nữa:

```
def main():
    number = get_number()
```

```
    meow(number)


def get_number():
    while True:
        n = int(input("What's n? "))
        if n > 0:
            break
    return n


def meow(n):
    for _ in range(n):
        print("meow")
```

Hãy lưu ý rằng chúng tôi không chỉ thay đổi mã của bạn để hoạt động trong nhiều hàm mà còn sử dụng một `return` câu lệnh cho `return` giá trị `n` quay lại hàm `main`.

Thông tin thêm về danh sách

- Hãy xem xét thế giới Hogwarts từ vũ trụ Harry Potter nổi tiếng.
- Trong thiết bị đầu cuối, gõ `code hogwarts.py`.
- Trong trình soạn thảo văn bản, mã như sau:

```
students = ["Hermoine", "Harry", "Ron"]

print(students[0])
print(students[1])
print(students[2])
```

Chú ý cách chúng ta có một `list` học sinh có tên như trên. Sau đó, chúng tôi in ra học sinh ở vị trí thứ 0, "Hermoine". Mỗi học sinh khác cũng được in.

- Giống như chúng ta đã minh họa trước đây, chúng ta có thể sử dụng vòng lặp để lặp lại danh sách. Bạn có thể cải thiện mã của mình như sau:

```
students = ["Hermoine", "Harry", "Ron"]

for student in students:
    print(student)
```

Lưu ý rằng đối với mỗi `student` danh `students` sách, nó sẽ in ra học sinh như dự định. Bạn có thể thắc mắc tại sao chúng tôi không sử dụng `_` tên gọi như đã thảo luận trước đó. Chúng tôi chọn không làm điều này vì `student` nó được sử dụng rõ ràng trong mã của chúng tôi.

- Bạn có thể tìm hiểu thêm trong tài liệu về danh sách (https://docs.python.org/3/tutorial/datastructures.html#more-on-lists) của Python .

Chiều dài

- Chúng ta có thể sử dụng `len` như một cách để kiểm tra độ dài `list` của `students`.
- Hãy tưởng tượng rằng bạn không chỉ muốn in tên của học sinh mà còn cả vị trí của họ trong danh sách. Để thực hiện điều này, bạn có thể chỉnh sửa mã của mình như sau:

```
students = ["Hermoine", "Harry", "Ron"]

for i in range(len(students)):
    print(i + 1, students[i])
```

  Lưu ý cách thực thi mã này không chỉ mang lại kết quả là vị trí của mỗi học sinh cộng với một bằng cách sử dụng `i + 1`, mà còn in ra tên của từng học sinh. `len` cho phép bạn tự động xem danh sách học sinh dài bao nhiêu bất kể nó tăng lên bao nhiêu.
- Bạn có thể tìm hiểu thêm trong tài liệu len (https://docs.python.org/3/library/functions.html?highlight=len#len) của Python .

Từ điển

- `dict`s hoặc từ điển là cấu trúc dữ liệu cho phép bạn liên kết các khóa với các giá trị.
- Trong đó a `list` là danh sách nhiều giá trị, a `dict` liên kết một khóa với một giá trị.
- Xem xét các ngôi nhà ở Hogwarts, chúng ta có thể chỉ định những học sinh cụ thể vào những ngôi nhà cụ thể.

Hermione	Harry	Ron	Draco
Gryffindor	Gryffindor	Gryffindor	Slytherin

- Chúng ta có thể sử dụng `list`s một mình để thực hiện điều này:

```
students = ["Hermoine", "Harry", "Ron", "Draco"]
houses = ["Gryffindor", "Gryffindor", "Griffindor", "Slytherin"]
```

  Lưu ý rằng chúng tôi có thể hứa rằng chúng tôi sẽ luôn giữ các danh sách này theo thứ tự. Cá nhân ở vị trí đầu tiên `students` được liên kết với ngôi nhà ở vị trí đầu tiên của danh `houses` sách, v.v. Tuy nhiên, điều này có thể trở nên khá cồng kềnh khi danh sách của chúng tôi tăng lên!
- Chúng ta có thể cải thiện mã của mình bằng cách sử dụng a `dict` như sau:

```
students = {
    "Hermoine": "Gryffindor",
    "Harry": "Gryffindor",
    "Ron": "Gryffindor",
    "Draco": "Slytherin",
}
print(students["Hermoine"])
print(students["Harry"])
print(students["Ron"])
print(students["Draco"])
```

Hãy chú ý cách chúng ta sử dụng `{}` dấu ngoặc nhọn để tạo từ điển. Trong đó `list` sử dụng số để lặp qua danh sách, `dict` thì cho phép chúng ta sử dụng từ.

- Chạy mã của bạn và đảm bảo đầu ra của bạn như sau:

```
$ python hogwarts.py
Gryffindor
Gryffindor
Gryffindor
Slytherin
```

- Chúng tôi có thể cải thiện mã của mình như sau:

```
students = {
    "Hermoine": "Gryffindor",
    "Harry": "Gryffindor",
    "Ron": "Gryffindor",
    "Draco": "Slytherin",
}
for student in students:
    print(student)
```

Lưu ý cách thực thi mã này, vòng lặp for sẽ chỉ lặp qua tất cả các khóa, dẫn đến danh sách tên của các học sinh. Làm cách nào chúng tôi có thể in ra cả giá trị và khóa?

- Sửa đổi mã của bạn như sau:

```
students = {
    "Hermoine": "Gryffindor",
    "Harry": "Gryffindor",
    "Ron": "Gryffindor",
    "Draco": "Slytherin",
}
for student in students:
    print(student, students[student])
```

Chú ý cách `students[student]` đi tới chìa khóa của từng học sinh và tìm giá trị căn nhà của họ. Thực thi mã của bạn và bạn sẽ nhận thấy đầu ra hơi lộn xộn.

- Chúng ta có thể cải thiện chức năng in bằng cách cải thiện mã của mình như sau:

```
students = {
    "Hermoine": "Gryffindor",
    "Harry": "Gryffindor",
    "Ron": "Gryffindor",
    "Draco": "Slytherin",
}
for student in students:
    print(student, students[student], sep=", ")
```

Lưu ý cách điều này tạo ra sự tách biệt rõ ràng `, ` giữa mỗi mục được in.

- Nếu bạn thực thi `python hogwarts.py`, bạn sẽ thấy như sau:

```
$ python hogwarts.py
Hermoine, Gryffindor
Harry, Gryffindor
Ron, Gryffindor
Draco, Slytherin
```

- Điều gì sẽ xảy ra nếu chúng tôi có thêm thông tin về học sinh của mình? Làm thế nào chúng ta có thể liên kết nhiều dữ liệu hơn với mỗi học sinh?

	name	house	patronus
0	Hermione	Gryffindor	Otter
1	Harry	Gryffindor	Stag
2	Ron	Gryffindor	Jack Russell terrier
3	Draco	Slytherin	

- Bạn có thể tưởng tượng muốn có nhiều dữ liệu liên kết nhiều thứ bằng một khóa. Cải thiện mã của bạn như sau:

```
students = [
    {"name": "Hermoine", "house": "Gryffindor", "patronus": "Otter"},
    {"name": "Harry", "house": "Gryffindor", "patronus": "Stag"},
    {"name": "Ron", "house": "Gryffindor", "patronus": "Jack Russell terrier"},
    {"name": "Draco", "house": "Slytherin", "patronus": None},
]
```

Lưu ý cách mã này tạo ra `list`s `dict`. Cái `list` gọi `students` có bốn `dicts` bên trong nó: Một cho mỗi học sinh. Ngoài ra, hãy lưu ý rằng Python có một `None` ký hiệu đặc biệt trong đó không có giá trị nào liên quan đến khóa.

- Giờ đây, bạn có quyền truy cập vào một loạt dữ liệu thú vị về những sinh viên này. Bây giờ, sửa đổi thêm mã của bạn như sau:

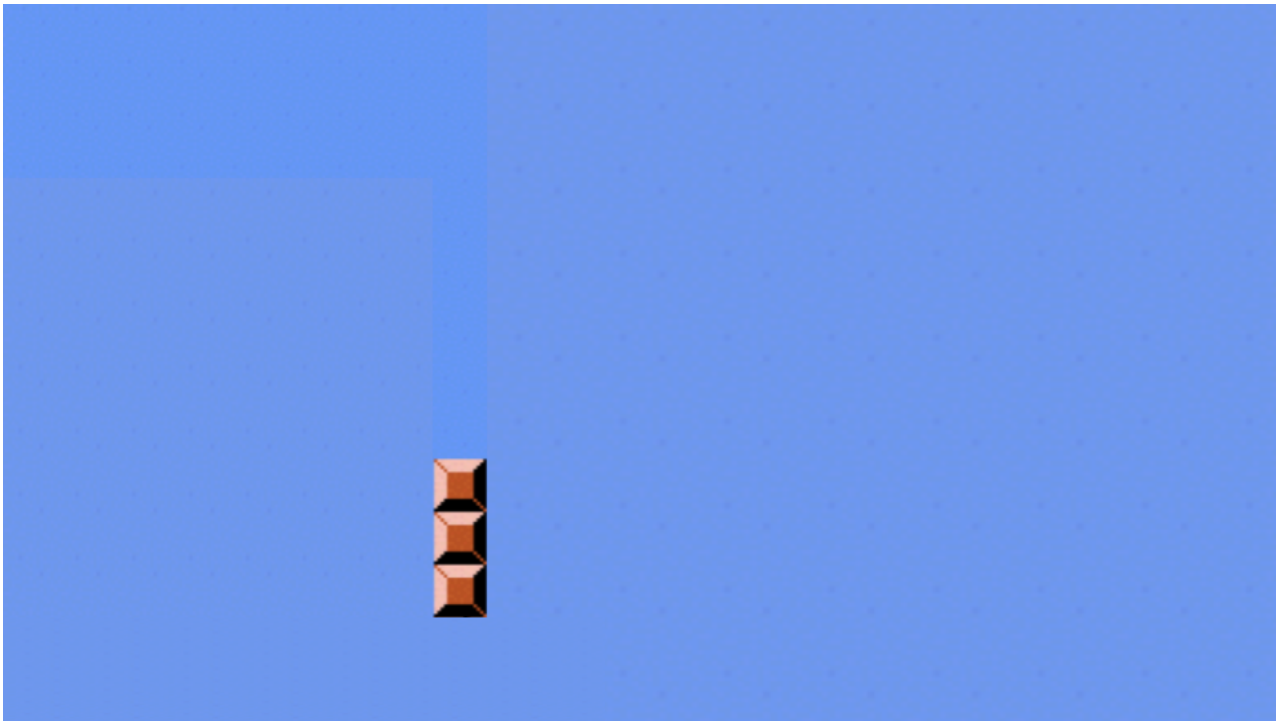
```
students = [
    {"name": "Hermoine", "house": "Gryffindor", "patronus": "Otter"},
    {"name": "Harry", "house": "Gryffindor", "patronus": "Stag"},
    {"name": "Ron", "house": "Gryffindor", "patronus": "Jack Russell terrier"},
    {"name": "Draco", "house": "Slytherin", "patronus": None},
]

for student in students:
    print(student["name"], student["house"], student["patronus"], sep=", ")
```

  Lưu ý cách `for` vòng lặp sẽ lặp qua từng `dict` s bên trong được `list` gọi `students` .
- Bạn có thể tìm hiểu thêm trong Tài liệu của Python về `dict` s (https://docs.python.org/3/tutorial/datastructures.html#dictionaries) .

Mario

- Hãy nhớ rằng trò chơi cổ điển Mario có một anh hùng nhảy qua những viên gạch. Hãy tạo một bản trình bày bằng văn bản của trò chơi này.



- Bắt đầu mã hóa như sau:

```
print("#")
print("#")
print("#")
```

  Hãy chú ý cách chúng ta sao chép và dán đi dán lại cùng một đoạn mã.

- Hãy xem xét cách chúng ta có thể viết mã tốt hơn như sau:

```
for _ in range(3):
    print("#")
```

  Lưu ý cách điều này thực hiện cơ bản những gì chúng ta muốn tạo.

- Hãy cân nhắc: Liệu sau này chúng ta có thể tóm tắt thêm để giải quyết các vấn đề phức tạp hơn bằng mã này không? Sửa đổi mã của bạn như sau:

```
def main():
    print_column(3)


def print_column(height):
    for _ in range(height):
        print("#")


main()
```

  Lưu ý cách cột của chúng tôi có thể phát triển nhiều như chúng tôi muốn mà không cần bất kỳ mã hóa cứng nào.

- Bây giờ, hãy thử in một hàng theo chiều ngang. Sửa đổi mã của bạn như sau:

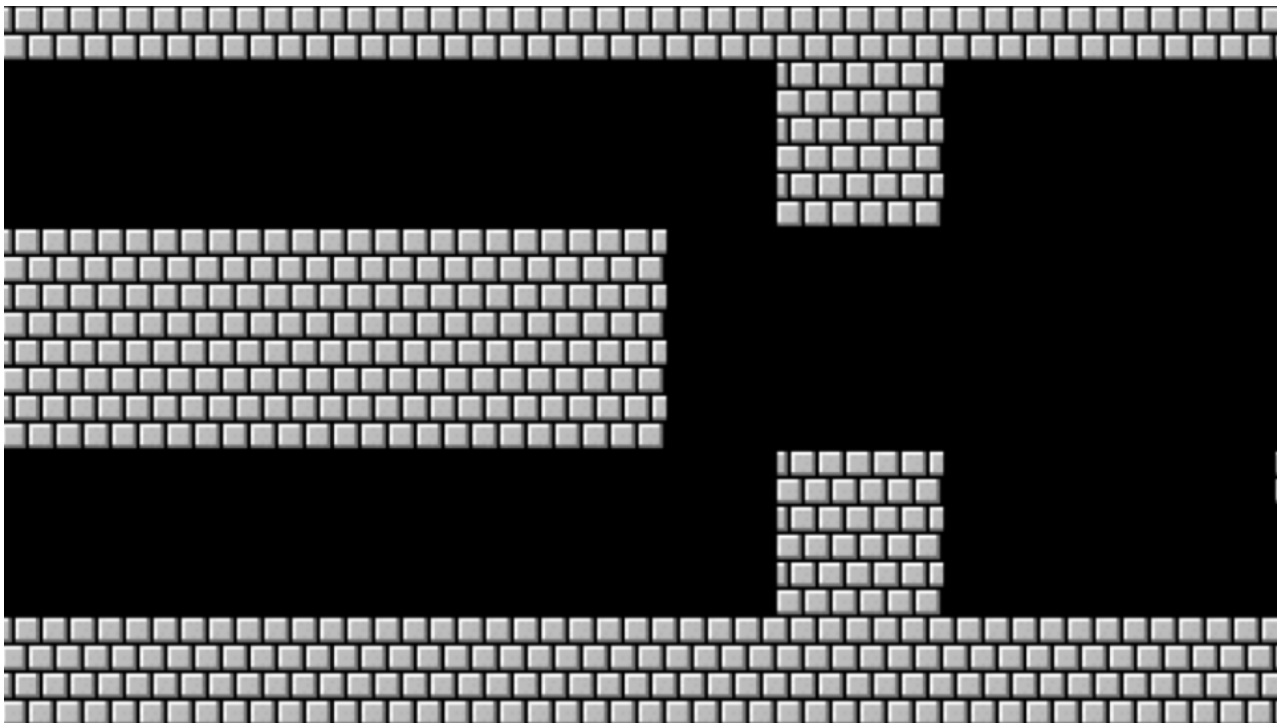
```
def main():
    print_row(4)


def print_row(width):
    print("?" * width)


main()
```

  Lưu ý bây giờ chúng ta có mã có thể tạo các khối từ trái sang phải.

- Xem xét slide bên dưới, hãy chú ý cách Mario có cả hàng và cột khối.



- Hãy xem xét làm cách nào chúng ta có thể triển khai cả hàng và cột trong mã của mình? Sửa đổi mã của bạn như sau:

```
def main():
    print_square(3)


def print_square(size):

    # For each row in square
    for i in range(size):

        # For each brick in row
        for j in range(size):

            #  Print brick
            print("#", end="")

        # Print blank line
        print()


main()
```

Lưu ý rằng chúng ta có một vòng lặp bên ngoài đánh địa chỉ mỗi hàng trong hình vuông. Sau đó, chúng ta có một vòng lặp bên trong in ra một viên gạch ở mỗi hàng. Cuối cùng, chúng ta có một `print` câu lệnh in một dòng trống.

- Chúng ta có thể trừu tượng hóa hơn nữa mã của mình:

```
def main():
    print_square(3)
```

```
def print_square(size):
    for i in range(size):
        print_row(size)


def print_row(width):
    print("#" * width)


main()
```

Tổng hợp

Bây giờ bạn có một sức mạnh khác trong danh sách ngày càng tăng các khả năng Python của mình. Trong bài giảng này chúng tôi đã đề cập...

- Vòng lặp
- `while`
- `for`
- `len`
- `list`
- `dict`