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 4

Thư viện

- Nói chung, thư viện là các đoạn mã do bạn hoặc người khác viết mà bạn có thể sử dụng trong chương trình của mình.
- Python cho phép bạn chia sẻ các chức năng hoặc tính năng với người khác dưới dạng "mô-đun".
- Nếu bạn sao chép và dán mã từ một dự án cũ, rất có thể bạn có thể tạo một mô-đun hoặc thư viện như vậy mà bạn có thể đưa vào dự án mới của mình.

Ngẫu nhiên

- `random` là một thư viện đi kèm với Python mà bạn có thể nhập vào dự án của riêng mình.
- Việc đứng trên vai những lập trình viên tiền nhiệm sẽ dễ dàng hơn khi là một lập trình viên.
- Vì vậy, làm cách nào để tải một mô-đun vào chương trình của riêng bạn? Bạn có thể sử dụng từ này `import` trong chương trình của bạn.
- Bên trong `random` mô-đun có một hàm tích hợp được gọi là `random.choice(seq)`. `random` là mô-đun bạn đang nhập. Bên trong mô-đun đó có chức `choice` năng. Hàm đó lấy một `seq` hoặc một dãy là một danh sách.
- Trong cửa sổ terminal của bạn gõ `code generate.py`. Trong trình soạn thảo văn bản của bạn, mã như sau:

  ```
  import random

  coin = random.choice(["heads", "tails"])
  print(coin)
  ```

  Lưu ý rằng danh sách bên trong `choice` có dấu ngoặc vuông, dấu ngoặc kép và dấu phẩy. Vì bạn đã vượt qua hai mục, Python sẽ thực hiện phép toán và đưa ra 50% cơ hội cho mặt ngửa và mặt sấp. Chạy mã của bạn, bạn sẽ nhận thấy rằng mã này thực sự hoạt động tốt!

- Chúng tôi có thể cải thiện mã của mình. `from` cho phép chúng tôi xác định rất cụ thể về những gì chúng tôi muốn nhập. Trước đó, `import` dòng mã của chúng tôi sẽ đưa toàn bộ nội dung các hàm của `random`. Tuy nhiên, nếu chúng ta chỉ muốn tải một phần nhỏ của mô-đun thì sao? Sửa đổi mã của bạn như sau:

  ```
  from random import choice

  coin = choice(["heads", "tails"])
  print(coin)
  ```

  Lưu ý rằng bây giờ chúng ta chỉ có thể nhập `choice` hàm của `random`. Từ thời điểm đó trở đi, chúng ta không cần phải viết mã nữa `random.choice`. Bây giờ chúng tôi chỉ có thể viết mã `choice` một mình. `choice` được tải rõ ràng vào chương trình của chúng tôi. Điều này giúp tiết kiệm tài nguyên hệ thống và có khả năng làm cho mã của chúng tôi chạy nhanh hơn!

- Tiếp tục, hãy xem xét hàm `random.randint(a, b)`. Hàm này sẽ tạo ra một số ngẫu nhiên giữa `a` và `b`. Sửa đổi mã của bạn như sau:

  ```
  import random

  number = random.randint(1, 10)
  print(number)
  ```

  Lưu ý rằng mã của chúng tôi sẽ tạo ngẫu nhiên một số nằm trong khoảng từ `1` và `10`.

- Chúng ta có thể đưa vào thẻ của mình `random.shuffle(x)` nơi nó sẽ xáo trộn danh sách thành một thứ tự ngẫu nhiên.

```
import random

cards = ["jack", "queen", "king"]
random.shuffle(cards)
for card in cards:
    print(card)
```

  Lưu ý rằng `random.shuffle` sẽ xáo trộn các thẻ vào đúng vị trí. Không giống như các hàm khác, nó sẽ không trả về giá trị. Thay vào đó, nó sẽ lấy `cards` danh sách và xáo trộn chúng trong danh sách đó. Chạy mã của bạn một vài lần để xem mã hoạt động.

- Bây giờ chúng ta có ba cách trên để tạo thông tin ngẫu nhiên.
- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `random` (https://docs.python.org/3/library/random.html).

Số liệu thống kê

- Python đi kèm với một thư viện tích hợp `statistics`. Chúng ta có thể sử dụng mô-đun này như thế nào?
- `average` là một chức năng của thư viện này khá hữu ích. Trong cửa sổ terminal của bạn, gõ `code average.py`. Trong cửa sổ soạn thảo văn bản, sửa đổi mã của bạn như sau:

```
import statistics

print(statistics.mean([100, 90]))
```

  Lưu ý rằng chúng tôi đã nhập một thư viện khác có tên `statistics`. Hàm `mean` lấy một danh sách các giá trị. Điều này sẽ in trung bình của các giá trị này. Trong cửa sổ terminal của bạn, gõ `python average.py`.
- Xem xét khả năng sử dụng `statistics` mô-đun trong các chương trình của riêng bạn.
- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `statistics` (https://docs.python.org/3/library/statistics.html).

Đối số dòng lệnh

- Cho đến nay, chúng tôi đã cung cấp tất cả các giá trị trong chương trình mà chúng tôi đã tạo. Điều gì sẽ xảy ra nếu chúng ta muốn có thể nhận dữ liệu đầu vào từ dòng lệnh? Ví dụ: thay vì gõ `python average.py` vào terminal, điều gì sẽ xảy ra nếu chúng ta muốn có thể gõ `python average.py 100 90` và có thể lấy giá trị trung bình giữa `100` và `90`?

- `sys` là một mô-đun cho phép chúng ta lấy đối số tại dòng lệnh.
- `argv` là một chức năng trong `sys` mô-đun cho phép chúng ta tìm hiểu về những gì người dùng đã nhập vào dòng lệnh. Lưu ý cách bạn sẽ thấy `sys.argv` cách sử dụng trong mã bên dưới. Trong cửa sổ terminal, gõ `code name.py`. Trong trình soạn thảo văn bản, mã như sau:

  ```
  import sys

  print("hello, my name is", sys.argv[1])
  ```

  Lưu ý rằng chương trình sẽ xem những gì người dùng gõ vào dòng lệnh. Hiện tại, nếu gõ `python name.py David` vào cửa sổ terminal, bạn sẽ thấy `hello, my name is David`. Lưu ý rằng đó `sys.argv[1]` là nơi `David` đang được lưu trữ. Tại sao vậy? Chà, trong các bài học trước, bạn có thể nhớ rằng danh sách bắt đầu từ `0` phần tử thứ. Bạn nghĩ hiện tại đang được tổ chức ở đâu `sys.argv[0]`? Nếu bạn đoán `name.py` thì bạn đã đúng!
- Có một vấn đề nhỏ với chương trình của chúng tôi. Nếu người dùng không gõ tên vào dòng lệnh thì sao? Hãy tự mình thử nó. Nhập `python name.py` vào cửa sổ terminal. Một lỗi `list index out of range` sẽ được trình bày bởi trình biên dịch. Lý do cho điều này là không có gì ở đó `sys.argv[1]` vì không có gì được gõ! Đây là cách chúng tôi có thể bảo vệ chương trình của mình khỏi loại lỗi này:

  ```
  import sys

  try:
      print("hello, my name is", sys.argv[1])
  except IndexError:
      print("Too few arguments")
  ```

  Lưu ý rằng bây giờ người dùng sẽ được nhắc nhở bằng một gợi ý hữu ích về cách làm cho chương trình hoạt động nếu họ quên nhập tên. Tuy nhiên, liệu chúng ta có thể phòng thủ hơn để đảm bảo người dùng nhập đúng giá trị không?
- Chương trình của chúng tôi có thể được cải thiện như sau:

  ```
  import sys

  if len(sys.argv) < 2:
      print("Too few arguments")
  elif len(sys.argv) > 2:
      print("Too many arguments")
  else:
      print("hello, my name is", sys.argv[1])
  ```

  Lưu ý rằng nếu bạn kiểm tra mã của mình, bạn sẽ thấy cách xử lý các ngoại lệ này, cung cấp cho người dùng lời khuyên tinh tế hơn. Ngay cả khi người dùng nhập quá nhiều hoặc quá ít đối số, người dùng vẫn được cung cấp hướng dẫn rõ ràng về cách khắc phục sự cố.

- Hiện tại, mã của chúng tôi đã đúng về mặt logic. Tuy nhiên, có một điều rất hay về việc tách biệt việc kiểm tra lỗi với phần còn lại của mã. Làm cách nào chúng tôi có thể tách biệt việc xử lý lỗi của mình? Sửa đổi mã của bạn như sau:

```
import sys

if len(sys.argv) < 2:
    sys.exit("Too few arguments")
elif len(sys.argv) > 2:
    sys.exit("Too many arguments")

print("hello, my name is", sys.argv[1])
```

  Lưu ý cách chúng tôi đang sử dụng chức năng tích hợp được `sys` gọi `exit` cho phép chúng tôi thoát khỏi chương trình nếu người dùng gặp lỗi. Bây giờ chúng ta có thể yên tâm rằng chương trình sẽ không bao giờ thực thi dòng mã cuối cùng và gây ra lỗi. Do đó, `sys.argv` cung cấp một cách để người dùng có thể giới thiệu thông tin từ dòng lệnh. `sys.exit` cung cấp một phương tiện để chương trình có thể thoát ra nếu có lỗi phát sinh.
- Bạn có thể tìm hiểu thêm trong tài liệu của Python về `sys` (https://docs.python.org/3/library/sys.html).

`slice`

- `slice` là một lệnh cho phép chúng ta lấy a `list` và cho trình biên dịch biết nơi chúng ta muốn trình biên dịch xem xét điểm bắt đầu `list` và kết thúc của `list`. Ví dụ: sửa đổi mã của bạn như sau:

```
import sys

if len(sys.argv) < 2:
    sys.exit("Too few arguments")

for arg in sys.argv:
    print("hello, my name is", arg)
```

  Lưu ý rằng nếu bạn nhập `python name.py David Carter Rongxin` vào cửa sổ terminal, trình biên dịch sẽ không chỉ xuất ra kết quả dự định của các tên mà còn cả các tệp `hello, my name is name.py`. Làm thế nào chúng ta có thể đảm bảo rằng trình biên dịch bỏ qua phần tử đầu tiên của danh sách `name.py` hiện đang được lưu trữ?
- `slice` có thể được sử dụng trong mã của chúng tôi để bắt đầu danh sách ở một nơi khác! Sửa đổi mã của bạn như sau:

```
import sys

if len(sys.argv) < 2:
```

```
    sys.exit("Too few arguments")

for arg in sys.argv[1:]:
    print("hello, my name is", arg)
```

Lưu ý rằng thay vì bắt đầu danh sách tại `0`, chúng tôi sử dụng dấu ngoặc vuông để yêu cầu trình biên dịch bắt đầu tại `1` và đi đến cuối bằng cách sử dụng `1:` đối số. Chạy mã này, bạn sẽ nhận thấy rằng chúng tôi có thể cải thiện mã của mình bằng cú pháp tương đối đơn giản.

Gói

- Một trong những lý do khiến Python trở nên phổ biến là vì có rất nhiều thư viện mạnh mẽ của bên thứ ba bổ sung chức năng. Chúng tôi gọi những thư viện của bên thứ ba này, được triển khai dưới dạng thư mục, là "gói".
- PyPI là kho lưu trữ hoặc thư mục của tất cả các gói của bên thứ ba hiện có.
- `cowsay` là một gói nổi tiếng cho phép một con bò nói chuyện với người dùng.
- Python có một trình quản lý gói được gọi `pip` là cho phép bạn cài đặt các gói nhanh chóng vào hệ thống của mình.
- Trong cửa sổ terminal, bạn có thể cài đặt `cowsay` gói bằng cách gõ `pip install cowsay`. Sau một chút đầu ra, bây giờ bạn có thể bắt đầu sử dụng gói này trong mã của mình.
- Trong cửa sổ terminal của bạn gõ `code say.py`. Trong trình soạn thảo văn bản, mã như sau:

```
import cowsay
import sys

if len(sys.argv) == 2:
    cowsay.cow("hello, " + sys.argv[1])
```

Lưu ý rằng trước tiên chương trình sẽ kiểm tra xem người dùng đã nhập ít nhất hai đối số vào dòng lệnh hay chưa. Sau đó, con bò sẽ nói chuyện với người dùng. Gõ `python say.py David` và bạn sẽ thấy một con bò đang nói "xin chào" với David.

- Sửa đổi thêm mã của bạn:

```
import cowsay
import sys

if len(sys.argv) == 2:
    cowsay.trex("hello, " + sys.argv[1])
```

Lưu ý rằng một con T-rex hiện đang nói "xin chào".

- Bây giờ bạn có thể thấy cách cài đặt các gói của bên thứ ba.
- Bạn có thể tìm hiểu thêm về mục nhập của PyPI cho `cowsay` (https://pypi.org/project/cowsay/)
- Bạn có thể tìm thấy các gói của bên thứ ba khác tại PyPI (https://pypi.org/)

API

- API hoặc "giao diện chương trình ứng dụng" cho phép bạn kết nối với mã của người khác.
- `requests` là một gói cho phép chương trình của bạn hoạt động như một trình duyệt web.
- Trong thiết bị đầu cuối của bạn, nhập `pip install requests`. Sau đó, gõ `code itunes.py`.
- Hóa ra Apple iTunes có API riêng mà bạn có thể truy cập trong các chương trình của mình. Trong trình duyệt internet, bạn có thể truy cập https://itunes.apple.com/search?entity=song&limit=1&term=weezer (https://itunes.apple.com/search?entity=song&limit=1&term=weezer) và một tệp văn bản sẽ được tải xuống. David đã xây dựng URL này bằng cách đọc tài liệu API của Apple. Lưu ý cách truy vấn này đang tìm kiếm một `song`, với `limit` một kết quả, liên quan `term` đến `weezer`. Nhìn vào tệp văn bản được tải xuống này, bạn có thể thấy định dạng tương tự như định dạng mà chúng tôi đã lập trình trước đây bằng Python.
- Định dạng trong tệp văn bản đã tải xuống được gọi là JSON, một định dạng dựa trên văn bản được sử dụng để trao đổi dữ liệu dựa trên văn bản giữa các ứng dụng. Theo nghĩa đen, Apple đang cung cấp một tệp JSON mà chúng tôi có thể diễn giải trong chương trình Python của riêng mình.
- Trong cửa sổ terminal, gõ `code itunes.py`. Mã như sau:

```
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()

response = requests.get("https://itunes.apple.com/search?entity=song&limit=1&term
print(response.json())
```

  Lưu ý cách giá trị trả về của `requests.get` sẽ được lưu trữ trong `response`. David, sau khi đọc tài liệu của Apple về API này, biết rằng thứ được trả về là một tệp JSON. Chạy `python itunes.py weezer`, bạn sẽ thấy file JSON được Apple trả về. Tuy nhiên, phản hồi JSON được Python chuyển đổi thành từ điển. Nhìn vào đầu ra, nó có thể khá chóng mặt!

- Hóa ra Python có thư viện JSON tích hợp có thể giúp chúng tôi diễn giải dữ liệu nhận được. Sửa đổi mã của bạn như sau:

```
import json
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()
```

```
response = requests.get("https://itunes.apple.com/search?entity=song&limit=1&term
print(json.dumps(response.json(), indent=2))
```

Lưu ý rằng nó `json.dumps` được triển khai sao cho nó sử dụng `indent` để làm cho đầu ra dễ đọc hơn. Chạy `python itunes.py weezer`, bạn sẽ thấy cùng một tệp JSON. Tuy nhiên, lần này, nó dễ đọc hơn nhiều. Bây giờ hãy chú ý rằng bạn sẽ thấy một từ điển có tên `results` bên trong đầu ra. Bên trong từ điển được gọi `results` đó có rất nhiều khóa. Nhìn vào `trackName` giá trị ở đầu ra. Bạn nhìn thấy tên bản nhạc nào trong kết quả của mình?

- Làm thế nào chúng ta có thể đơn giản xuất ra tên của tên bản nhạc đó? Sửa đổi mã của bạn như sau:

```
import json
import requests
import sys

if len(sys.argv) != 2:
    sys.exit()

response = requests.get("https://itunes.apple.com/search?entity=song&limit=50&ter

o = response.json()
for result in o["results"]:
    print(result["trackName"])
```

Lưu ý cách chúng tôi lấy kết quả `response.json()` và lưu trữ nó `o` (như trong chữ thường). Sau đó, chúng tôi lặp qua phần `results` in `o` và in từng tệp `trackName`. Cũng lưu ý cách chúng tôi đã tăng số lượng kết quả giới hạn lên `50`. Chạy chương trình của bạn. Xem kết quả.

- Bạn có thể tìm hiểu thêm `requests` thông qua tài liệu của thư viện (https://docs.python-requests.org/) .
- Bạn có thể tìm hiểu thêm về JSON trong tài liệu JSON (https://docs.python.org/3/library/json.html) của Python .

Tạo thư viện của riêng bạn

- Là một lập trình viên Python, bạn có khả năng tạo thư viện của riêng mình!
- Hãy tưởng tượng những tình huống mà bạn có thể muốn sử dụng lại nhiều lần các đoạn mã hoặc thậm chí chia sẻ chúng với người khác!
- Cho đến nay, chúng tôi đã viết rất nhiều mã để nói "xin chào" trong khóa học này. Hãy tạo một gói để cho phép chúng ta nói "xin chào" và "tạm biệt". Trong cửa sổ terminal của bạn, gõ `code sayings.py`. Trong trình soạn thảo văn bản, mã như sau:

```
def hello(name):
    print(f"hello, {name}")


def goodbye(name):
    print(f"goodbye, {name}")
```

Lưu ý rằng bản thân mã này không làm được gì cho người dùng. Tuy nhiên, nếu một lập trình viên nhập gói này vào chương trình của riêng họ thì các khả năng do các hàm trên tạo ra có thể được triển khai trong mã của họ.

- Hãy xem cách chúng tôi có thể triển khai mã bằng cách sử dụng gói mà chúng tôi đã tạo này. Trong cửa sổ terminal, gõ `code say.py`. Trong tệp mới này trong trình soạn thảo văn bản của bạn, hãy nhập thông tin sau:

  ```
  import sys

  from saying import goodbye

  if len(sys.argv) == 2:
      goodbye(sys.argv[1])
  ```

  Lưu ý rằng mã này nhập các khả năng của `goodbye` gói `sayings`. Nếu người dùng nhập ít nhất hai đối số vào dòng lệnh, nó sẽ nói "tạm biệt" cùng với chuỗi được nhập ở dòng lệnh.

Tổng hợp

Thư viện mở rộng khả năng của Python. Một số thư viện được bao gồm theo mặc định trong Python và bạn chỉ cần nhập vào. Các gói khác là các gói của bên thứ ba cần được cài đặt bằng `pip`. Bạn có thể tự tạo các gói để sử dụng cho chính mình hoặc người khác! Trong bài giảng này, bạn đã tìm hiểu về...

- Thư viện
- Ngẫu nhiên
- Số liệu thống kê
- Đối số dòng lệnh
- Lát cắt
- Gói
- API
- Tạo thư viện của riêng bạn