# 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 6



## Tệp vào/ra

- Cho đến nay, mọi thứ chúng ta lập trình đều lưu trữ thông tin trong bộ nhớ. Nghĩa là, khi chương trình kết thúc, tất cả thông tin được thu thập từ người dùng hoặc do chương trình tạo ra sẽ bị mất.
- I/O tệp là khả năng chương trình lấy tệp làm đầu vào hoặc tạo tệp làm đầu ra.
- Để bắt đầu, trong cửa sổ terminal gõ `code names.py` và mã như sau:

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

  Lưu ý rằng việc chạy mã này có kết quả mong muốn. Người dùng có thể nhập tên. Đầu ra đúng như mong đợi.

- Tuy nhiên, nếu chúng ta muốn cho phép nhập nhiều tên thì sao? Làm thế nào chúng ta có thể đạt được điều này? Hãy nhớ lại rằng a `list` là cấu trúc dữ liệu cho phép chúng ta lưu trữ nhiều giá trị vào một biến duy nhất. Mã như sau:

```
names = []

for _ in range(3):
    name = input("What's your name?" )
    names.append(name)
```

Lưu ý rằng người dùng sẽ được nhắc nhập dữ liệu ba lần. Phương pháp này `append` được sử dụng để thêm `name` vào `names` danh sách của chúng tôi.

- Mã này có thể được đơn giản hóa như sau:

```
names = []

for _ in range(3):
    names.append(input("What's your name?" ))
```

Lưu ý rằng điều này có kết quả tương tự như khối mã trước đó.

  - Bây giờ, hãy kích hoạt khả năng in danh sách tên dưới dạng danh sách được sắp xếp. Mã như sau:

```
names = []

for _ in range(3):
    names.append(input("What's your name?" ))

for name in sorted(names):
    print(f"hello, {name}")
```

Lưu ý rằng khi chương trình này được thực thi, tất cả thông tin sẽ bị mất. Tệp I/O cho phép chương trình của bạn lưu trữ thông tin này để có thể sử dụng sau này.

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

## open

- `open` là một chức năng được tích hợp trong Python cho phép bạn mở tệp và sử dụng nó trong chương trình của mình. Chức năng này `open` cho phép bạn mở một tập tin để bạn có thể đọc hoặc ghi vào nó.
- Để chỉ cho bạn cách bật tệp I/O trong chương trình của bạn, hãy tua lại một chút và viết mã như sau:

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

file = open("names.txt", "w")
file.write(name)
file.close()
```

Lưu ý rằng `open` hàm này sẽ mở một tệp có tên được `names.txt` kích hoạt ghi, như được biểu thị bằng phần mở rộng `w`. Đoạn mã trên gán tệp đã mở đó cho một biến có tên `file`. Dòng `file.write(name)` ghi tên vào tập tin văn bản. Dòng sau đó sẽ đóng tập tin.

- Kiểm tra mã của bạn bằng cách gõ `python names.py`, bạn có thể nhập tên và nó sẽ lưu vào tệp văn bản. Tuy nhiên, nếu bạn chạy chương trình của mình nhiều lần bằng các tên khác nhau, bạn sẽ nhận thấy rằng chương trình này sẽ ghi lại toàn bộ tệp `names.txt` mỗi lần.
- Lý tưởng nhất là chúng tôi muốn có thể thêm từng tên của mình vào tệp. Xóa tệp văn bản hiện có bằng cách nhập `rm names.txt` vào cửa sổ terminal. Sau đó, sửa đổi mã của bạn như sau:

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

file = open("names.txt", "a")
file.write(name)
file.close()
```

Lưu ý rằng thay đổi duy nhất đối với mã của chúng tôi là mã `w` đã được đổi thành `a` "chắp thêm". Chạy lại chương trình này nhiều lần, bạn sẽ nhận thấy rằng các tên sẽ được thêm vào tệp. Tuy nhiên, bạn sẽ nhận thấy một vấn đề mới!

- Kiểm tra tệp văn bản của bạn sau khi chạy chương trình nhiều lần, bạn sẽ nhận thấy rằng các tên đang chạy cùng nhau. Các tên đang được nối thêm mà không có bất kỳ khoảng trống nào giữa mỗi tên. Bạn có thể khắc phục vấn đề này. Một lần nữa, xóa tệp văn bản hiện có bằng cách nhập `rm names.txt` vào cửa sổ terminal. Sau đó, sửa đổi mã của bạn như sau:

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

file = open("names.txt", "a")
file.write(f"{name}\n")
file.close()
```

Lưu ý rằng dòng with `file.write` đã được sửa đổi để thêm dấu ngắt dòng ở cuối mỗi tên.

- Mã này đang hoạt động khá tốt. Tuy nhiên, có nhiều cách để cải thiện chương trình này. Điều đó xảy ra là bạn rất dễ quên đóng tập tin.
- Bạn có thể tìm hiểu thêm trong tài liệu về open (https://docs.python.org/3/library/functions.html#open) .

## `with`

- Từ khóa `with` cho phép bạn tự động hóa việc đóng tệp.
- Sửa đổi mã của bạn như sau:

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

with open("names.txt", "a") as file:
    file.write(f"{name}\n")
```

  Lưu ý rằng dòng bên dưới `with` được thụt lề.
- Cho đến thời điểm này, chúng tôi chỉ ghi vào một tập tin. Điều gì sẽ xảy ra nếu chúng ta muốn đọc từ một tập tin. Để kích hoạt chức năng này, hãy sửa đổi mã của bạn như sau:

```
with open("names.txt", "r") as file:
    lines = file.readlines()

for line in lines:
    print("hello,", line)
```

  Lưu ý rằng `readlines` có khả năng đặc biệt để đọc tất cả các dòng của tệp và lưu trữ chúng trong một tệp có tên là dòng. Chạy chương trình của bạn, bạn sẽ nhận thấy rằng kết quả đầu ra khá xấu. Dường như có nhiều ngắt dòng trong đó lẽ ra chỉ nên có một ngắt dòng.
- Có nhiều cách tiếp cận để khắc phục vấn đề này. Tuy nhiên, đây là một cách đơn giản để sửa lỗi này trong mã của chúng tôi:

```
with open("names.txt", "r") as file:
    lines = file.readlines()

for line in lines:
    print("hello,", line.rstrip())
```

  Lưu ý rằng nó `rstrip` có tác dụng loại bỏ ngắt dòng không liên quan ở cuối mỗi dòng.
- Tuy nhiên, mã này có thể được đơn giản hóa hơn nữa:

```
with open("names.txt", "r") as file:
    for line in file:
        print("hello,", line.rstrip())
```

  Lưu ý rằng việc chạy mã này là chính xác. Tuy nhiên, hãy lưu ý rằng chúng tôi không sắp xếp tên.
- Mã này có thể được cải thiện hơn nữa để cho phép sắp xếp tên:

```
names = []

with open("names.txt") as file:
    for line in file:
        names.append(line.rstrip())
```

```
for name in sorted(names):
    print(f"hello, {name}")
```

Lưu ý rằng đó `names` là một danh sách trống nơi chúng tôi có thể thu thập tên. Mỗi tên được thêm vào `names` danh sách trong bộ nhớ. Sau đó, mỗi tên trong danh sách đã sắp xếp trong bộ nhớ sẽ được in. Chạy mã của bạn, bạn sẽ thấy rằng các tên hiện đã được sắp xếp chính xác.

- Điều gì sẽ xảy ra nếu chúng ta muốn có khả năng lưu trữ nhiều thứ hơn là chỉ tên của học sinh? Điều gì sẽ xảy ra nếu chúng ta muốn lưu trữ cả tên của học sinh và nhà của họ?

## CSV

- CSV là viết tắt của "các giá trị được phân tách bằng dấu phẩy".
- Trong cửa sổ terminal của bạn, gõ `code students.csv`. Đảm bảo tệp CSV mới của bạn trông giống như sau:

```
Hermoine,Gryffindor
Harry,Gryffindor
Ron,Gryffindor
Draco,Slytherin
```

- Hãy tạo một chương trình mới bằng cách gõ `code students.py` và mã như sau:

```
with open("students.csv") as file:
    for line in file:
        row = line.rstrip().split(",")
        print(f"{row[0]} is in {row[1]}")
```

Lưu ý rằng `rstrip` sẽ xóa phần cuối của mỗi dòng trong tệp CSV của chúng tôi. `split` cho trình biên dịch biết nơi tìm phần cuối của mỗi giá trị trong tệp CSV của chúng tôi. `row[0]` là phần tử đầu tiên trong mỗi dòng của tệp CSV của chúng tôi. `row[1]` là phần tử thứ hai trong mỗi dòng trong tệp CSV của chúng tôi.

- Đoạn mã trên có hiệu quả trong việc chia từng dòng hoặc "bản ghi" của tệp CSV của chúng tôi. Tuy nhiên, sẽ hơi khó hiểu nếu bạn không quen với loại cú pháp này. Python có khả năng tích hợp sẵn có thể đơn giản hóa mã này hơn nữa. Sửa đổi mã của bạn như sau:

```
with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        print(f"{name} is in {house}")
```

Lưu ý rằng `split` hàm thực sự trả về hai giá trị: Giá trị trước dấu phẩy và giá trị sau dấu phẩy. Theo đó, chúng ta có thể dựa vào chức năng đó để gán hai biến cùng một lúc thay vì một biến!

- Hãy tưởng tượng rằng chúng tôi lại muốn cung cấp danh sách này dưới dạng đầu ra được sắp xếp? Bạn có thể sửa đổi mã của mình như sau:

```
students = []

with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        students.append(f"{name} is in {house}")

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

  Lưu ý rằng chúng tôi tạo một `list` cuộc gọi `students`. Chúng tôi `append` từng chuỗi vào danh sách này. Sau đó, chúng tôi xuất ra một phiên bản đã sắp xếp của danh sách của mình.

- Hãy nhớ lại rằng Python cho phép `dictionaries` một khóa có thể được liên kết với một giá trị. Mã này có thể được cải thiện hơn nữa

```
students = []

with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        student = {}
        student["name"] = name
        student["house"] = house
        students.append(student)

for student in students:
    print(f"{student['name']} is in {student['house']}")
```

  Lưu ý rằng chúng tôi tạo một từ điển trống có tên `student`. Chúng tôi thêm các giá trị cho mỗi học sinh, bao gồm tên và nhà của họ vào từ `student` điển. Sau đó, chúng tôi thêm sinh viên đó vào tên `list` gọi `students`.

- Chúng tôi có thể cải thiện mã của mình để minh họa điều này như sau:

```
students = []

with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        student = {"name": name, "house": house}
        students.append(student)

for student in students:
    print(f"{student['name']} is in {student['house']}")
```

  Lưu ý rằng điều này tạo ra kết quả mong muốn, trừ đi việc sắp xếp học sinh.

- Thật không may, chúng tôi không thể sắp xếp học sinh như trước đây vì mỗi học sinh giờ đây là một từ điển bên trong danh sách. Sẽ rất hữu ích nếu Python có thể sắp xếp `students` danh `student` sách từ điển sắp xếp danh sách từ điển này theo tên của học sinh.
- Để triển khai điều này trong mã của chúng tôi, hãy thực hiện các thay đổi sau:

```
students = []

with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        students.append({"name": name, "house": house})


def get_name(student):
    return student["name"]


for student in sorted(students, key=get_name):
    print(f"{student['name']} is in {student['house']}")
```

  Lưu ý rằng `sorted` cần phải biết cách lấy chìa khóa của từng học sinh. Python cho phép một tham số được gọi là `key` nơi chúng ta có thể xác định "khóa" nào mà danh sách học sinh sẽ được sắp xếp. Do đó, `get_name` hàm chỉ trả về khóa của `student["name"]`. Chạy chương trình này, bây giờ bạn sẽ thấy danh sách hiện đã được sắp xếp theo tên.

- Tuy nhiên, mã của chúng tôi có thể được cải thiện hơn nữa. Điều đó xảy ra là nếu bạn chỉ sử dụng một hàm như `get_name` một lần, bạn có thể đơn giản hóa mã của mình theo cách được trình bày bên dưới. Sửa đổi mã của bạn như sau:

```
students = []

with open("students.csv") as file:
    for line in file:
        name, house = line.rstrip().split(",")
        students.append({"name": name, "house": house})

for student in sorted(students, key=lambda student: student["name"]):
    print(f"{student['name']} is in {student['house']}")
```

  Hãy lưu ý cách chúng ta sử dụng một `lambda` hàm, một hàm ẩn danh, có nội dung "Này Python, đây là một hàm không có tên: Cho một `student`, truy cập vào chúng `name` và trả về hàm đó cho `key`.

- Thật không may, mã của chúng tôi hơi dễ vỡ. Giả sử rằng chúng tôi đã thay đổi tệp CSV sao cho chỉ ra nơi mỗi học sinh lớn lên. Điều này sẽ có tác động gì đến chương trình của chúng ta? Đầu tiên, sửa đổi `students.csv` tập tin của bạn như sau:

```
Harry,"Number Four, Privet Drive"
Ron,The Burrow
```

```
Draco,Malfoy Manor
```

Lưu ý rằng việc chạy chương trình của chúng ta sẽ tạo ra một số lỗi như thế nào.

- Bây giờ chúng ta đang xử lý nhà thay vì nhà, hãy sửa đổi mã của bạn như sau:

  ```
  students = []

  with open("students.csv") as file:
      for line in file:
          name, home = line.rstrip().split(",")
          students.append({"name": name, "home": home})

  for student in sorted(students, key=lambda student: student["name"]):
      print(f"{student['name']} is in {student['home']}")
  ```

  Lưu ý rằng việc chạy chương trình của chúng tôi vẫn không hoạt động bình thường. Bạn có thể đoán tại sao không?

- Lỗi `ValueError: too many values to unpack` do trình biên dịch tạo ra là do trước đây chúng tôi đã tạo chương trình này với mong đợi tệp CSV đang `split` sử dụng `,` (dấu phẩy). Chúng ta có thể dành nhiều thời gian hơn để giải quyết vấn đề này, nhưng thực sự ai đó đã phát triển một cách để "phân tích cú pháp" (nghĩa là đọc) các tệp CSV!
- Thư viện tích hợp của Python `csv` đi kèm với một đối tượng được gọi là `reader`. Như tên cho thấy, chúng tôi có thể sử dụng a `reader` để đọc tệp CSV của mình mặc dù có thêm dấu phẩy trong "Số Bốn, Privet Drive". A `reader` hoạt động theo `for` vòng lặp, trong đó mỗi lần lặp sẽ `reader` cung cấp cho chúng tôi một hàng khác từ tệp CSV của chúng tôi. Bản thân hàng này là một danh sách, trong đó mỗi giá trị trong danh sách tương ứng với một phần tử trong hàng đó. `row[0]`, ví dụ: là phần tử đầu tiên của hàng đã cho, trong khi `row[1]` là phần tử thứ hai.

  ```
  import csv

  students = []

  with open("students.csv") as file:
      reader = csv.reader(file)
      for row in reader:
          students.append({"name": row[0], "home": row[1]})

  for student in sorted(students, key=lambda student: student["name"]):
      print(f"{student['name']} is from {student['home']}")
  ```

  Lưu ý rằng chương trình của chúng tôi hiện hoạt động như mong đợi.

- Cho đến thời điểm này, chúng tôi đã dựa vào chương trình của mình để quyết định cụ thể phần nào trong tệp CSV là tên và phần nào là nhà. Tuy nhiên, thiết kế tốt hơn là đưa tệp này trực tiếp vào tệp CSV của chúng tôi bằng cách chỉnh sửa tệp như sau:

  ```
  name,home
  Harry,"Number Four, Privet Drive"
  ```

```
Ron,The Burrow
Draco,Malfoy Manor
```

Lưu ý cách chúng tôi nói rõ ràng trong tệp CSV của mình rằng bất kỳ thứ gì đọc nó đều phải có giá trị tên và giá trị nhà trong mỗi dòng.

- Chúng tôi có thể sửa đổi mã của mình để sử dụng một phần của `csv` thư viện được gọi là a `DictReader` để xử lý tệp CSV của chúng tôi một cách linh hoạt hơn nữa:

```
import csv

students = []

with open("students.csv") as file:
    reader = csv.DictReader(file)
    for row in reader:
        students.append({"name": row["name"], "home": row["home"]})

for student in sorted(students, key=lambda student: student["name"]):
    print(f"{student['name']} is in {student['home']}")
```

Lưu ý rằng chúng tôi đã thay thế `reader` bằng `DictReader`, trả về từng từ điển một. Ngoài ra, hãy lưu ý rằng trình biên dịch sẽ truy cập trực tiếp vào `row` từ điển, lấy số `name` và `home` của từng học sinh. Đây là một ví dụ về mã hóa phòng thủ. Miễn là người thiết kế tệp CSV đã nhập thông tin tiêu đề chính xác trên dòng đầu tiên, chúng tôi có thể truy cập thông tin đó bằng chương trình của mình.

- Cho đến thời điểm này, chúng tôi đã đọc các tệp CSV. Nếu chúng ta muốn ghi vào tệp CSV thì sao?
- Để bắt đầu, hãy dọn dẹp các tập tin của chúng ta một chút. Đầu tiên, xóa `students.csv` tập tin bằng cách gõ `rm students.csv` vào cửa sổ terminal. Lệnh này sẽ chỉ hoạt động nếu bạn ở trong cùng thư mục với `students.csv` tệp của mình.
- Sau đó, trong `students.py`, sửa đổi mã của bạn như sau:

```
import csv

name = input("What's your name? ")
home = input("Where's your home? ")

with open("students.csv", "a") as file:
    writer = csv.DictWriter(file, fieldnames=["name", "home"])
    writer.writerow({"name": name, "home": home})
```

Lưu ý cách chúng tôi tận dụng chức năng tích hợp của `DictWriter`, có hai tham số: `file` được ghi vào và được `fieldnames` ghi. Hơn nữa, hãy chú ý cách `writerow` hàm lấy từ điển làm tham số. Theo đúng nghĩa đen, chúng ta đang yêu cầu trình biên dịch viết một hàng có hai trường được gọi là `name` và `home`.

- Lưu ý rằng có nhiều loại tệp mà bạn có thể đọc và ghi vào.

- Bạn có thể tìm hiểu thêm trong tài liệu CSV (https://docs.python.org/3/library/csv.html) của Python .

## Tệp nhị phân và `PIL`

- Một loại tệp nữa mà chúng ta sẽ thảo luận hôm nay là tệp nhị phân. Tệp nhị phân chỉ đơn giản là tập hợp các số 1 và số 0. Loại tệp này có thể lưu trữ mọi thứ, bao gồm dữ liệu âm nhạc và hình ảnh.
- Có một thư viện Python phổ biến được gọi `PIL` là hoạt động tốt với các tệp hình ảnh.
- GIF động là một loại tệp hình ảnh phổ biến có nhiều tệp hình ảnh bên trong được phát đi phát lại theo trình tự, tạo ra hiệu ứng hoạt hình hoặc video đơn giản.
- Hãy tưởng tượng rằng chúng ta có một loạt trang phục, như minh họa bên dưới.
- Đây là `costume1.gif`.

- Đây là một cái khác được gọi là `costume2.gif`. Chú ý vị trí của chân hơi khác nhau một chút.

- Trước khi tiếp tục, vui lòng đảm bảo rằng bạn đã tải xuống các tệp mã nguồn từ trang web của khóa học. Bạn sẽ không thể viết mã những nội dung sau nếu không sở hữu và lưu trữ hai hình ảnh trên trong IDE của mình.
- Trong cửa sổ terminal gõ `code costumes.py` và mã như sau:

```
import sys

from PIL import Image

images = []

for arg in sys.argv[1:]:
    image = Image.open(arg)
    images.append(image)

images[0].save(
    "costumes.gif", save_all=True, append_images=[images[1]], duration=200, loop=
)
```

Lưu ý rằng chúng tôi nhập `Image` chức năng từ `PIL`. Lưu ý rằng `for` vòng lặp đầu tiên chỉ lặp qua các hình ảnh được cung cấp dưới dạng đối số dòng lệnh và lưu trữ chủ đề vào `list` tệp `images`. Việc `1:` bắt đầu cắt `argv` ở phần tử thứ hai của nó. Dòng mã cuối cùng lưu hình ảnh đầu tiên và cũng thêm hình ảnh thứ hai vào đó, tạo ra một ảnh gif động. Đang gõ `python costumes.py costume1.gif costume2.gif` vào thiết bị đầu cuối. Bây giờ, hãy nhập `code costumes.gif` vào cửa sổ terminal và bây giờ bạn có thể thấy ảnh GIF động.

- Bạn có thể tìm hiểu thêm trong tài liệu PIL (https://pillow.readthedocs.io/) của Pillow .

## Tổng hợp

Bây giờ, chúng ta không chỉ thấy rằng chúng ta có thể ghi và đọc tệp văn bản—chúng ta còn có thể đọc và ghi tệp bằng cách sử dụng số một và số không. Chúng tôi rất nóng lòng muốn xem bạn đạt được những gì với những khả năng mới này tiếp theo.

- Tệp vào/ra
- `open`
- `with`
- CSV
- `PIL`