



tinkling_ @Minh.Khai

Theo dõi

★ 38 👤 0 ✎ 4

Đã đăng vào thg 12 17, 2020 10:01 PM - 12 phút đọc

👁 2.7K 💬 2 📌 0

Tìm hiểu về thư viện Numpy trong Python(Phần 1)

...

Trong khoảng thời gian vừa rồi tôi có cơ hội được làm việc bằng ngôn ngữ Python thông qua một dự án về sức khỏe của công ty. Sau một khoảng thời gian tìm hiểu, làm việc tôi nhận thấy đây là một ngôn ngữ quá mạnh mẽ, nó mạnh mẽ vì sự đa dụng của nó, nó có thể làm được mọi việc liên quan đến lập trình từ làm web, app hay game, nhưng điều khiến python nổi tiếng có lẽ đến từ việc nó có thể code được cả trong lĩnh vực phân tích khoa học dữ liệu, xây dựng trí tuệ nhân tạo. Một điều nữa cũng khiến Python được yêu thích đó là nó có một hệ sinh thái các thư viện hỗ trợ cực kì lớn. Nó lớn đến mức đôi khi chúng ta phải phân vân nên chọn sử dụng thư viện nào cho hợp lí.

Trong số hàng triệu thư viện hỗ trợ Python thì có một thư viện lại gần như được đóng đinh mà bắt buộc lập trình viên nào khi làm việc với Python cũng cần phải tìm hiểu đó chính là thư viện NumPy. Trong bài ngày hôm nay tôi xin giới thiệu với các bạn về thư viện NumPy trong Python và cách sử dụng nó trong python.

1. Giới thiệu về thư viện NumPy

Numpy (Numeric Python): là một thư viện toán học rất phổ biến và mạnh mẽ của Python. NumPy được trang bị các hàm số đã được tối ưu, cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng Python đơn thuần.

Nếu bạn muốn trở thành một lập trình viên khoa học dữ liệu chuyên sâu, bạn cần phải nắm rõ numpy. Đây là một trong những thư viện hữu ích nhất của python, đặc biệt là nếu bạn đang tìm hiểu về các con số. Vì phần lớn Khoa học Dữ liệu và Máy học xoay quanh Thống kê, nên việc thực hành trở nên quan trọng hơn nhiều.

NumPy được phát triển bởi Jim Hugunin. Phiên bản ban đầu là Numarray được phát triển, có một số chức năng bổ sung. Năm 2005, Travis Oliphant đã tạo ra gói NumPy bằng cách kết hợp các tính năng của Numarray và gói Numeric.

Sử dụng NumPy, lập trình viên có thể thực hiện các thao tác sau:

- Các phép toán toán học và logic trên mảng.
- Các biến đổi Fourier và các quy trình để thao tác shape.
- Các phép toán liên quan đến đại số tuyến tính. NumPy tích hợp sẵn các hàm cho đại số tuyến tính và tạo số ngẫu nhiên.



↑ +1 ↓



NumPy thường được sử dụng cùng với các gói như SciPy (Python Scientific) và Matplotlib (thư viện vẽ đồ thị). Sự kết hợp này được sử dụng rộng rãi để thay thế cho MatLab, một nền tảng phổ biến cho tính toán kỹ thuật. Tuy nhiên, Python thay thế cho MatLab hiện được xem như một ngôn ngữ lập trình hoàn thiện và hiện đại hơn. Điều quan trọng hơn cả là Numpy là một thư viện mã nguồn mở, miễn phí so với MatLab là một thư viện mã nguồn đóng và phải trả phí.

Cách cài đặt NumPy

Trong bài này tôi sẽ thực hành trên ubuntu và Framework Django, nếu bạn đang chạy trên HĐH khác thì chỉ cần gg là đều có hướng dẫn chi tiết

Đầu tiên mở Terminal lên và nhập

```
sudo apt install python3-pip
pip install numpy
```

Bạn cần phải cài đặt Numpy thông qua pip

Sau khi đã cài đặt xong NumPy, chúng ta cần import nó để sử dụng như các thư viện khác của Python để sử dụng các hàm số của NumPy:

```
import numpy as np
```

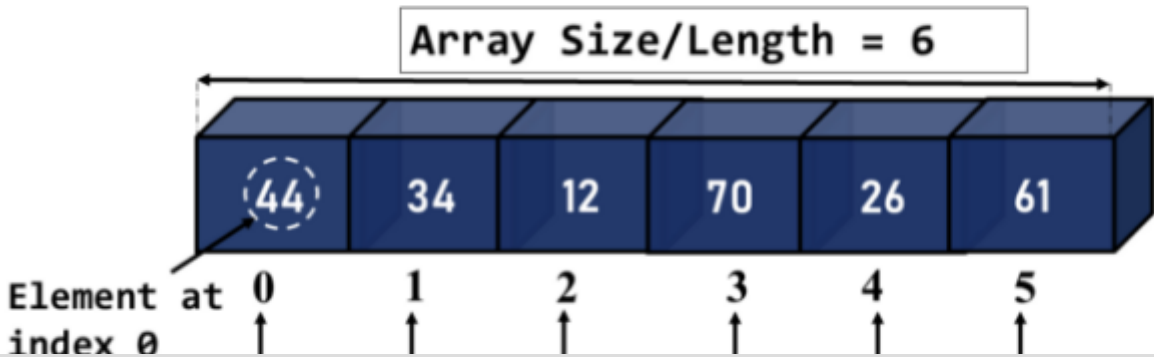
Sau khi cài đặt xong chúng ta sẽ tìm hiểu về các kiểu dữ liệu trong Numpy

1. Mảng

Mảng là một cấu trúc dữ liệu chứa một nhóm các phần tử. Thông thường, tất cả các phần tử này có cùng kiểu dữ liệu, chẳng hạn như số nguyên hoặc chuỗi. Chúng thường được sử dụng trong các chương trình để sắp xếp dữ liệu để một bộ giá trị liên quan có thể dễ dàng được sắp xếp hoặc tìm kiếm.

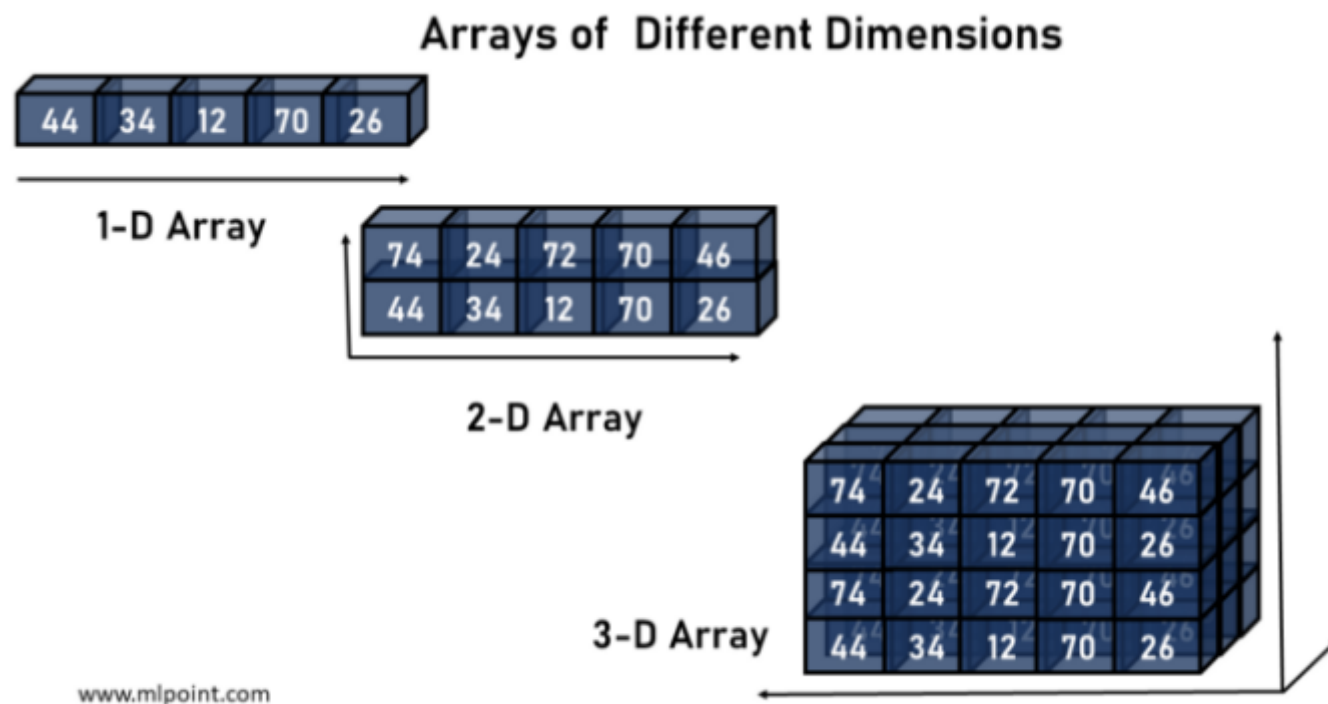
Khi nói đến NumPy, một mảng là một cấu trúc dữ liệu trung tâm của thư viện. Đó là một lưới các giá trị và nó chứa thông tin về dữ liệu thô, cách xác định vị trí của một phần tử và cách diễn giải một phần tử. Nó có một lưới các phần tử có thể được lập chỉ mục theo nhiều cách khác nhau. Tất cả các phần tử có cùng kiểu, được gọi là kiểu mảng (kiểu dữ liệu).

1-D Array with 6 Elements



Một mảng có thể được lập chỉ mục bởi một bộ số nguyên không âm, bởi boolean, bởi một mảng khác hoặc bởi số nguyên. Thứ hạng của mảng là số thứ nguyên. Hình dạng của mảng là một loạt các số nguyên cho biết kích thước của mảng dọc theo mỗi chiều. Một cách chúng ta có thể khởi tạo mảng NumPy là từ danh sách Python lồng nhau.

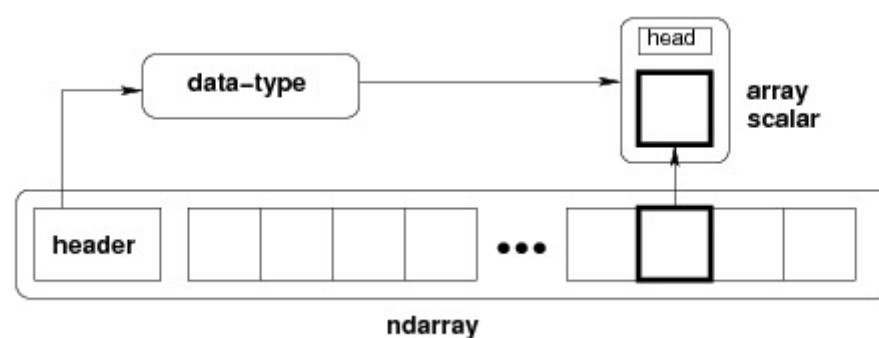
Một mảng được gọi là cấu trúc dữ liệu trung tâm của thư viện NumPy. Mảng trong NumPy được gọi là NumPy Array.



Đối tượng quan trọng nhất được định nghĩa trong NumPy là một kiểu mảng N chiều được gọi là ndarray. Nó mô tả bộ sưu tập các mặt hàng cùng loại. Các mục trong bộ sưu tập có thể được truy cập bằng chỉ mục dựa trên số không.

Mọi mục trong một ndarray có cùng kích thước khối trong bộ nhớ. Mỗi phần tử trong ndarray là một đối tượng của đối tượng kiểu dữ liệu (được gọi là dtype).

Bất kỳ mục nào được trích xuất từ đối tượng ndarray (bằng cách cắt) được đại diện bởi một đối tượng Python thuộc một trong các kiểu vô hướng mảng. Biểu đồ sau đây cho thấy mối quan hệ giữa ndarray, đối tượng kiểu dữ liệu (dtype) và kiểu vô hướng mảng:



Một instance của ndarray có thể được xây dựng bằng các quy trình tạo mảng khác nhau được mô tả ở phần sau của hướng dẫn. Ndarray cơ bản được tạo bằng một array function trong NumPy như sau:

```
numpy.array
```

Nó tạo ra một ndarray từ bất kỳ đối tượng nào hiển thị giao diện mảng hoặc từ bất kỳ phương thức nào trả về một mảng.

```
numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)
```

STT	Mô tả thông số
1	object - Bất kỳ đối tượng nào hiển thị phương thức giao diện mảng sẽ trả về một mảng hoặc bất kỳ chuỗi (lồng nhau) nào..
2	dtype - Kiểu dữ liệu mong muốn của mảng, optional
3	copy - Không bắt buộc. Theo mặc định (true), đối tượng được sao chép
4	order - C (hàng chính) hoặc F (cột chính) hoặc A (bất kỳ) (mặc định)
5	subok - Theo mặc định, mảng trả về buộc phải là mảng lớp cơ sở. Nếu đúng, các lớp con được chuyển qua
6	ndmin - Chỉ định kích thước tối thiểu của mảng kết quả

Ví dụ để hiểu thêm:

```
import numpy as np
a = np.array([1,2,3])
print a
```

Kết quả : [1, 2, 3]

```
# more than one dimensions
import numpy as np
a = np.array([[1, 2], [3, 4]])
print a
```

Kết quả : [[1, 2][3, 4]]

```
# minimum dimensions
import numpy as np
a = np.array([1, 2, 3,4,5], ndmin = 2)
print a
```

Kết quả : [[1, 2, 3, 4, 5]]

```
Bản thử trực tiếp
# dtype parameter
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print a
```

Kết quả : [1.+0.j, 2.+0.j, 3.+0.j]

Sự khác biệt giữa Python List và Numpy Array

- Python List có thể chứa các phần tử với các kiểu dữ liệu khác nhau trong khi các phần tử của Numpy Array luôn đồng nhất (cùng một kiểu dữ liệu)

- NumPy Array sử dụng bộ nhớ cố định để lưu trữ dữ liệu và ít bộ nhớ hơn Python List.
- Cấp phát bộ nhớ liền kề trong NumPy Array

Loại dữ liệu trong NumPy

NumPy hỗ trợ nhiều kiểu số hơn nhiều so với Python. Bảng sau đây cho thấy các kiểu dữ liệu vô hướng khác nhau được xác định trong NumPy.

STT	Loại dữ liệu & mô tả
1	bool_ - Boolean (Đúng hoặc Sai) được lưu trữ dưới dạng byte
2	int_ - Kiểu số nguyên mặc định (giống C long; thường là int64 hoặc int32)
3	intc - Giống hệt với int C (thường là int32 hoặc int64)
4	intp - Số nguyên được sử dụng để lập chỉ mục (giống như C ssize_t; thông thường là int32 hoặc int64)
5	int8 - Byte (-128 đến 127)
6	int16 - Số nguyên (-32768 đến 32767)







[↗ Đăng nhập/Đăng ký](#)

8	int64 - Số nguyên (-9223372036854775808 đến 9223372036854775807)
9	uint8 - Số nguyên không dấu (0 đến 255)
10	uint16 - Số nguyên không dấu (0 đến 65535)
11	uint32 - Số nguyên không dấu (0 đến 4294967295)
12	uint64 - Số nguyên không dấu (0 đến 18446744073709551615)
13	float_ - Viết tắt cho float64
14	float16 - float: bit dấu, số mũ 5 bit, phần định trị 10 bit
15	float32 - float: bit dấu, số mũ 8 bit, phần định trị 23 bit
16	float64 - float: bit dấu, số mũ 11 bit, phần định trị 52 bit
17	complex_ - Viết tắt cho complex128
18	complex64 - Số phức, được biểu diễn bằng hai số thực 32 bit (thành phần thực và ảo)
19	complex128 - Số phức, được biểu thị bằng hai số thực 64 bit (thành phần thực và ảo)

Đối tượng kiểu dữ liệu (dtype)

Một đối tượng kiểu dữ liệu mà ta diện giải nội bộ nno có định tương ứng với một mảng, tùy thuộc vào các kiến trúc sau:

- Loại dữ liệu (số nguyên, đối tượng float hoặc Python)
- Kích thước của dữ liệu
- Thứ tự Byte (little-endian hoặc big-endian)
- Trong trường hợp kiểu có cấu trúc, tên của các trường, kiểu dữ liệu của từng trường và một phần của khối bộ nhớ được lấy bởi từng trường.
- Nếu kiểu dữ liệu là một mảng con, hình dạng và kiểu dữ liệu của nó

Thứ tự byte được quyết định bằng cách thêm tiền tố '<' hoặc '>' vào kiểu dữ liệu. '<' có nghĩa là mã hóa có giá trị nhỏ (ít quan trọng nhất được lưu trữ ở địa chỉ nhỏ nhất). '>' có nghĩa là mã hóa là big-endian (byte quan trọng nhất được lưu trữ ở địa chỉ nhỏ nhất).

Một đối tượng dtype được xây dựng bằng cú pháp sau:

```
numpy.dtype(object, align, copy)
```

Các thông số là :

- Đối tượng - Được chuyển đổi thành đối tượng kiểu dữ liệu
- Căn chỉnh - Nếu đúng, hãy thêm phần đệm vào trường để làm cho trường tương tự như C-struct
- Sao chép - Tạo một bản sao mới của đối tượng dtype. Nếu sai, kết quả là tham chiếu đến đối tượng kiểu dữ liệu nội trang

Array Indexing

NumPy cung cấp một số cách để truy xuất phần tử trong mảng

Indexing và slicing: Mỗi thành phần trong mảng 1 chiều tương ứng với một chỉ số. Chỉ số trong NumPy, cũng giống như chỉ số trong python, bắt đầu bằng 0. Nếu mảng 1 chiều có n phần tử thì các chỉ số chạy từ 0 đến n - 1. Và tương tự như list trong python, NumPy arrays cũng có thể được cắt (slicing).

```
# Khởi tạo numpy array có shape = (3, 4) như sau:
a = np.array([[1,2,3,4],
              [5,6,7,8],
              [9,10,11,12]])
# Dùng chỉ số để lấy phần tử hàng 1, cột 2
print(a[1][2]) # 7
print(a[1, 2]) # 7
# Dùng slicing để lấy 2 hàng đầu tiên của 2 cột đầu tiên
print(a[:2][:2])
# [[1 2]
#  [5 6]]
# Kết hợp dùng slicing và indexing
# Chú ý: sẽ tạo ra mảng có rank thấp hơn mảng cũ
r1 = a[1, :] # Rank 1, hàng 1 của a
print(r1, r1.shape) # [[5 6 7 8]] (4,)
```

Cho phép bạn chọn ra các phần tử tùy ý của một mảng, thường được sử dụng để chọn ra các phần tử thỏa mãn điều kiện nào đó

```
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2) # Tìm các phần tử lớn hơn 2;
# Trả về 1 numpy array of Booleans có shape như mảng a
# và giá trị tại mỗi phần tử là
# True nếu phần tử của a tại đó > 2,
# False cho trường hợp ngược lại.
print(bool_idx)
# [[False False]
#  [ True  True]
#  [ True  True]]”
```

Kết luận

NumPy là một thư viện toán học phổ biến và mạnh mẽ của Python. Nó cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng Python thuần.

Trong bài viết tôi đã giới thiệu cho bạn về NumPy, lợi ích của nó, cách cài đặt nó để sử dụng, tìm hiểu về Mảng trong NumPy, kiểu dữ liệu trong NumPy. Trong bài viết tiếp theo chúng ta sẽ tiếp tục tìm hiểu về các kiểu dữ liệu khác trong NumPy

Tham khảo:

[Numpy Tutorial](#) [Numpy.org](#) [Numpy Medium](#)

Basic Python Numpy

All rights reserved



Bài viết liên quan

[Tổng hợp các cú pháp lệnh for thường gặp trong swift...](#)

[Lê Văn Tuấn](#)
1 phút đọc
👁 829 📌 4 💬 0 ⬆ 4

[Học Python từ con số 0 \(Phần 2\) - Function và...](#)

[Điệp Trần](#)
4 phút đọc
👁 2750 📌 3 💬 0 ⬆ 3

[Toán tử trong Python](#)

[Nguyễn Hữu Kim](#)
4 phút đọc
👁 3921 📌 0 💬 0 ⬆ 1

[Let's Learn N](#)

[Dat Nguyen](#)
1 phút đọc
👁 3845 📌 0

[Ứng dụng thuật toán Naive Bayes trong.giải quyết bài toán chuẩn đoán bệnh tiểu đường](#)

[Phạm Văn Toàn](#)

[So sánh các mô hình dự đoán trong bài toán nhận dạng khuôn mặt và ví dụ thực tế](#)

[Phạm Văn Toàn](#)

Big-O giải thích bởi 1 lập trình viên tự học

Nguyen Thanh Hai

8 phút đọc

6289 9 0 8

Một số hàm thông dụng trong matlab để vẽ đồ thị

Nguyễn Thùy Dương

13 phút đọc

👁 108789 📖 8 💬 6 ⬆️ 17

Bài viết khác từ tinkling

Tìm hiểu về I3 Window manager và cách dùng trên Ubuntu

tinkling

11 phút đọc

👁 484 📖 1 💬 2 ⬆ 3

Tìm hiểu về thư viện Numpy trong Python(Phần 3).

tinkling

3 phút đọc

👁 270 📖 0 💬 0 ⬆️ 2

Tìm hiểu về thư viện Numpy trong Python(Phần 2).

tinkling

11 phút đọc

712 1 0 2

Tìm hiểu về I3 Window manager và cách dùng trên Ubuntu

tinkling

11 phút đọc

👁 484 📖 1 💬 2 ⬆ 3

Tìm hiểu về thư viện Numpy trong Python(Phần 3).

tinkling

3 phút đọc

👁 270 📖 0 💬 0 ⬆ 2

Tìm hiểu về thư viện Numpy trong Python(Phần 2).

tinkling

11 phút đọc

712 1 0 2

Bình luận

🗨 Đăng nhập để bình luận

 [Huy_Nguyen](#) @kev26

thg 3 27, 9:58 PM

Sao trong bài hướng dẫn của bạn dùng [] thay cho () khi khai báo phần tử, mình có đọc bài chỗ khác thì họ dùng tuple (), vậy dùng cái nào sẽ chuẩn hơn ?, vì theo mình biết thì tuple không cho sửa xóa !

^ 0 v | [Trả lời](#) [Chia sẻ](#) ...

 [tinkling](#) @Minh.Khai

thg 3 30, 1:31 PM

[@kev26](#) Bài này là mình viết Mảng mà bạn ơi, mình xem lại thấy mình không có đề cập đến tuple đâu nhỉ. còn tuple thì phải là () còn mảng thì là [] bạn nhé. Thanks

0 | [Trả lời](#) [Chia sẻ](#) [...](#)

[Danh sách](#)

[Câu hỏi](#)

[Videos](#)

[Thảo luận](#)

[Công cụ](#)

[Trạng thái hệ thống](#)

[Tài liệu](#)

[Tags](#)

[Tác giả](#)

[Đề xuất hệ thống](#)

[Machine Learning](#)

 [Viblo Code](#)

 [Viblo CV](#)


 [Viblo CTF](#)

 [Viblo Learning](#)



LIÊN KẾT



 Tiếng Việt 

[Về chúng tôi](#)

[Phản hồi](#)

[Giúp đỡ](#)

[FAQs](#)

[RSS](#)

[Điều khoản](#)



© Viblo 2021



↑ +1 ↓

