Python For Data Science
Thư viện Numpy

Vu Tien Dung - Ngo The Quyen

Nội dung

Giới thiệu về thư viện Numpy

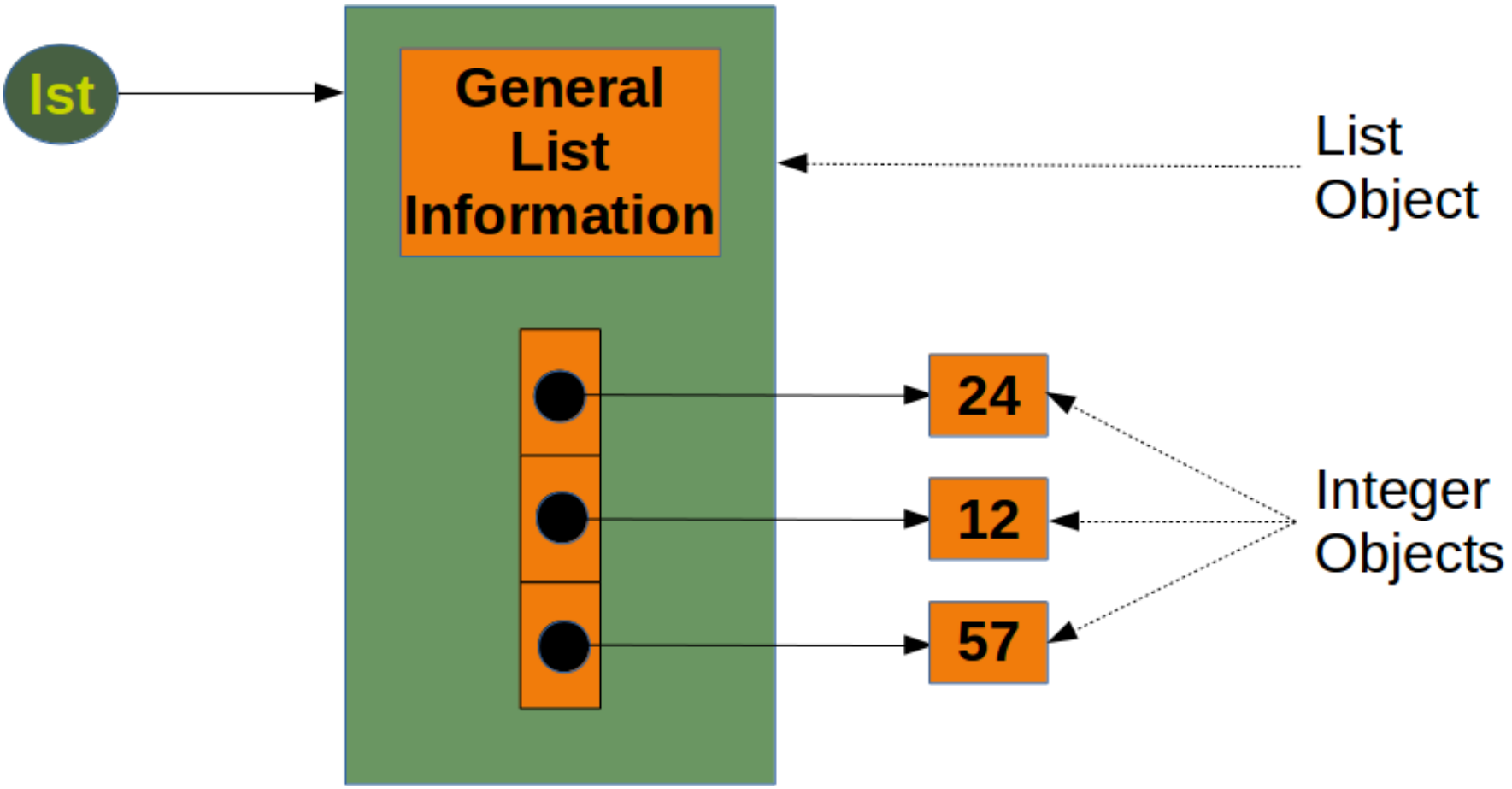
Thư viện Numerical Python (Numpy) là thư viên cơ sở cho tính toán hiệu năng cao và phân tích dữ liệu

Khai báo thư viện Numpy

```
import numpy as np
```

Giới thiệu về thư viện Numpy

Hình 1: Cơ chế cấp phát bộ nhớ của danh sách



Giới thiệu về thư viện Numpy

Kích thước bộ nhớ của một danh sách

- Kích thước lưu trữ thông tin danh sách chung
- Kích thước cần thiết cho các tham chiếu đến các phần tử
- Kích thước của tất cả các phần tử của danh sách

Thông tin về dung lượng bộ nhớ không kể các phần tử

Sử dụng hàm sys.getsizeof để truy cập thông tin về dung lượng bộ nhớ của danh sách không tính dung lượng các phần tử

Giới thiệu về thư viện Numpy

Kích thước bộ nhớ của một danh sách

```
1 from sys import getsizeof as size
2 lst=[12,24,36]
3 size_of_list_object=size(lst)
4 size_of_elements=len(lst)*size(lst[0])
5 total_list_size=size_of_list_object+size_of_elements
6 print('Size without the size of the elements',size_of_list_object)
7 print('Size of the elements',size_of_elements)
8 print('Total size of list, including elements',total_list_size)
```

Giới thiệu về thư viện Numpy

Kích thước bộ nhớ của một danh sách

```
1  from sys import getsizeof as size
2  lst1=[12,24,36,42]
3  size_of_list_object=size(lst1)
4  size_of_elements=len(lst1)*size(lst1[0])
5  total_list_size=size_of_list_object+size_of_elements
6
7  print('Size without the size of the elements',size_of_list_object)
8  print('Size of the elements',size_of_elements)
9  print('Total size of list, including elements',total_list_size)
10
11 lst1=[]
12 print("Empty list size",size(lst1))
```

Giới thiệu về thư viện Numpy

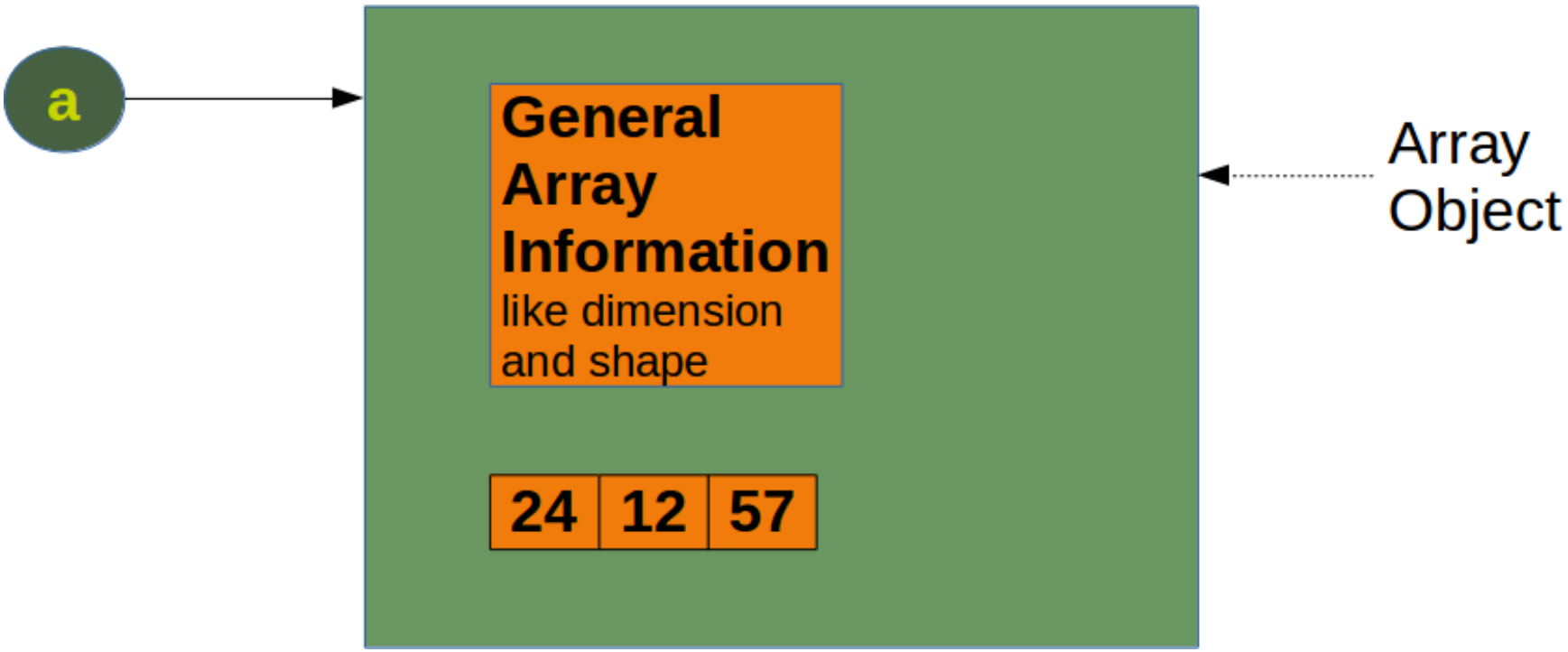
Kích thước bộ nhớ của một danh sách

- Với mỗi phần tử mới, chúng ta cần thêm 8 byte để tham chiếu đến đối tượng mới
- Kích thước của mỗi đối tượng số nguyên là 28 byte
- Kích thước của danh sách "lst" không kể kích thước của các phần tử có thể được tính bằng:

$$64 + 8 * len(lst)$$

Giới thiệu về thư viện Numpy

Hình 2: Cơ chế cấp phát bộ nhớ của mảng



Giới thiệu về thư viện Numpy

Kích thước bộ nhớ của một mảng

```
1  from sys import getsizeof as size
2  import numpy as np
3  a=np.array([12,24,36])
4  size_of_array_object=size(a)
5  print("Total size of array",size_of_array_object)
6  a=np.array([12,24,36,42])
7  size_of_array_object=size(a)
8  print("Total size of array",size_of_array_object)
9  e=np.array([])
10 print("Empty array size",size(e))
```

Giới thiệu về thư viện Numpy

Kích thước bộ nhớ của một mảng

- Kích thước của mảng gồm n phần tử nguyên có thể được tính bằng:

$$96 + 4 * n$$

- Chúng ta có thể xác định kích thước của các số nguyên, khi chúng ta định nghĩa một mảng

Ví dụ

```
1 a=np.array([12,24,36],np.int8)
2 print(size(a)-96)
3 a=np.array([12,24,36],np.int16)
4 print(size(a)-96)
```

Giới thiệu về thư viện Numpy

So sánh về thời gian thực thi

```
1  import time
2  import numpy as np
3  size_of_vec = 1000000
4  def pure_python_version():
5      t1 = time.time()
6      X = range(size_of_vec)
7      Y = range(size_of_vec)
8      Z = [X[i] + Y[i] for i in range(len(X)) ]
9      return time.time() - t1
```

Giới thiệu về thư viện Numpy

So sánh về thời gian thực thi

```
1  def numpy_version():
2      t1 = time.time()
3      X = np.arange(size_of_vec)
4      Y = np.arange(size_of_vec)
5      Z = X + Y
6      return time.time() - t1
7  if __name__=='__main__':
8       t1 = pure_python_version()
9       t2 = numpy_version()
10      print(t1, t2)
11      print("Numpy is in this example " + str(t1/t2) + " faster!")
```

Khởi tạo mảng

Khởi tạo mảng bởi hàm

np.zeros(tupe)	Khởi tạo dãy gồm các phần tử 0
np.ones(tupe)	Khởi tạo dãy gồm các phần tử 1
np.empty(tupe)	Khởi tạo dãy gồm các phần tử ngẫu nhiên
np.eye(tupe)	Khởi tạo ma trận đơn vị
np.range(start,end,step)	Tạo ra một dãy các số
np.linspace(start,end,num)	Tạo dãy gồm num giá trị trong khoảng từ start tới end

Khởi tạo mảng

Ví dụ

```
1  import numpy as np
2  num1=np.zeros(5);
3  print(num1)
4  num2=np.zeros((5,3));
5  print(num2)
6  num3=np.ones((3,4))
7  print(num3)
8  num4=np.empty(4)
9  print(num4)
10 num5=np.eye(4)
11 print(num5)
12 num6=np.arange(2,7,2)
13 print(num6)
14 num7=np.linspace(2,6,4)
15 print(num7)
```

Khởi tạo mảng

Khơi tạo mảng ngẫu nhiên

np.random.rand	Tạo mảng gồm các phần tử là một số ngẫu nhiên có phân bố đều trong [0, 1)
np.random.normal	Tạo mảng gồm các phần tử là một số ngẫu nhiên có phân bố chuẩn chính tắc
np.random.seed	Khởi tạo các bộ sinh số ngẫu nhiên
np.random.randint	Tạo mảng các số tự nhiên ngẫu nhiên
np.random.permutation	Sinh ra một dãy hoán vị

Khởi tạo mảng

Ví dụ

```
1  import numpy as np
2  np.random.seed(0);
3  x=np.random.randint(5,10,size=(3,4))
4  print(x)
5  y=np.random.rand(3,4)
6  print(y)
7  z=np.random.normal(0,1,size=(3,2))
8  print(z)
9  t=np.random.permutation(9)
10 print(t)
11
```

Thuộc tính của đối tượng mảng

Các thuộc tính cơ bản

ndim	Số chiều của mảng
shape	Kích thước của mỗi chiều của mảng
size	Tổng số phần tử của mảng
dtype	Kiểu dữ liệu của phần tử mảng
itemsize	Kích thước của mỗi phần tử mảng
nbytes	Tổng số byte dùng lưu trữ

Thuộc tính của đối tượng mảng

Ví dụ

```
1
2
3
4
5
6
```

```
import numpy as np
np.random.seed(0)
x=np.random.randint(5,10,size=(3,4,5))
print(x.ndim,x.shape,x.size)
print(x.dtype,x.itemsize,x.nbytes)
```

Truy cập phần tử của mảng

Chỉ số

- Mỗi thành phần trong mảng 1 chiều tương ứng với một chỉ số. Chỉ số trong numpy bắt đầu bằng 0. Nếu mảng 1 chiều có d phần tử thì các chỉ số chạy từ 0 đến d - 1
- Các phần tử của mảng có thể được truy cập và gán giá trị theo cách tượng tự như kiểu dữ liệu danh sách
  - Đối với mảng một chiều, giá trị thứ i có thể được truy cập bằng chỉ số trong ngoặc vuông
  - Đối với mảng nhiều chiều, phần tử có thể truy cập bằng bộ chỉ số đặt cách nhau bởi dấu phẩy
  - Để truy cập vào phần tử cuối cùng của mảng một chiều gồm d phần tử ta có thể dùng chỉ số -1.

Truy cập phần tử của mảng

Ví dụ

```
1
2
3
4
5
6
7
8
9
```

```
import numpy as np
a=np.arange(10)
print(a)
print(a[1],a[2],a[-1])
x=np.random.randint(3,5,size=(3,4))
print(x)
print(x[1,3])
print(x[0])
```

Truy cập các phần tử của mảng một chiều

Truy cập tới nhiều phần tử của mảng x

Cú pháp truy cập nhiều phần tử của mảng một chiều

```
x[start:stop:step]
```

Truy cập các phần tử của mảng một chiều

Ví dụ

```
1  import numpy as np
2  x=np.arange(10)
3  print(x)
4  print(x[:5])
5  print(x[5:])
6  print(x[4:7])
7  print(x[::2])
8  print(x[1::2])
9  print(x[::-1])
10
```

Truy cập đối tượng trên mảng nhiều chiều

Ví dụ

```
1  import numpy as np
2  x=np.random.randint(1,12,size=(3,4))
3  print(x)
4  print(x[:2,:3])
5  print(x[:2,::2])
6  print(x[::-1,::-1])
7
```

Khung nhìn và mảng con

Khung nhìn

Mỗi thao tác cắt lát tạo ra một khung nhìn trên mảng ban đầu và đây là một cách truy cập dữ liệu mảng. Điều đó có nghĩa là khi thực hiện thao tác cắt lát không có vùng nhớ mới nào được tạo ra.

- Chúng ta có thể sử dụng hàm np.may_share_memory() để kiểm tra hai mảng khác nhau có cùng một vùng nhớ chia sẻ.
- Khi thay đổi dữ liệu trên khung nhìn thì dữ liệu trên mảng ban đầu cũng thay đổi theo
- Khi chúng ta thực hiện với dữ liệu lớn, chúng ta có thể truy cập và xử lý những khối dữ liệu mà không cần sao chép dữ liệu vào vùng nhớ

Khung nhìn và mảng con

Ví dụ

```
1  import numpy as np
2  a=np.arange(10)
3  print(a)
4  b=a[::2]
5  print(b)
6  print(np.may_share_memory(a,b))
7  b[0]=12;
8  print(b,a)
9
```

Khung nhìn và mảng con

Mảng con

Để sao chép dữ liệu trong một mảng hoặc một mảng con ta dùng phương thức copy().

Ví dụ

```
1
2
3
4
5
6
7
8
9
```

```
import numpy as np
a = np.arange(10)
print(a)
c = a[::2].copy()  # force a copy
print(c)
c[0] = 12
print(a)
print(np.may_share_memory(a,c))
```

Các phép toán hiệu năng cao

Ví dụ

```
1  import numpy as np
2  np.random.seed(0)
3
4  def compute_reciprocals(values):
5      output = np.empty(len(values))
6      for i in range(len(values)):
7          output[i] = 1.0 / values[i]
8      return output
9
10 big_array = np.random.randint(1, 100, size=1000000)
11 %timeit compute_reciprocals(big_array)
12 %timeit (1.0 / big_array)
```

Các phép toán vector

Các phép toán hiệu năng cao (vector hóa) trong thư viện Numpy được thực hiện thông qua các hàm ufuncs.

Các phép toán số học

+	np.add	Phép cộng
-	np.subtract	Phép trừ
-	np.negative	Phép phủ định
*	np.multiply	Phép nhân
/	np.divide	Phép chia
//	np.floor_divide	Phép chia nguyên
**	np.power	Phép lũy thừa
%	np.mod	Phép toán lấy dư

Các phép toán vector

Ví dụ

```
1  import numpy as np
2  import numpy as np
3  x=np.arange(10)
4  print("x=    ",x)
5  print("x+5=",x+5)
6  print("x-5=",x-5)
7  print("x*2=",x*2)
8  print("x/2=",x/2)
9  print("x//2=",x//2)
10
```

Các phép toán vector

Các phép toán hiệu năng cao (vector hóa) trong thư viện Numpy được thực hiện thông qua các hàm ufuncs.

Các hàm hiệu năng cao

np.abs	hàm lấy trị tuyệt đối
np.log	Hàm logarit
np.exp	Hàm e mũ
np.exp2	Hàm lũy thừa của 2
np.pow	Hàm lũy thừa

Các phép toán vector

Ví dụ

```
1  import numpy as np
2  x=np.array([-3+4j,3-4j,0-3j])
3  print(np.abs(x))
4  x=np.array([-2,3,-4,5,-6])
5  print(np.abs(x))
6  x=[1,2,3]
7  print("x  =",x)
8  print("e^x=",np.exp(x))
9  print("2^x=",np.exp2(x))
10 print("2^x=",np.power(2,x))
11 print("3^x=",np.power(3,x))
12
```

Các phép toán vector

Ví dụ

```
1
2
3
4
5
6
7
```

```
import numpy as np
x=[1,2,4,10]
print("x  =",x)
print("log(x) =",np.log(x))
print("log2(x)=",np.log2(x))
print("log10(x)=",np.log10(x))
```

Các tính năng nâng cao

Xác định Output

Sử dụng tham biến out

Ví dụ

```
1 import numpy as np
2 x=np.arange(5)
3 y=np.zeros(5)
4 np.multiply(x,10,out=y)
5 print(y)
6 y=np.zeros(10)
7 np.power(2,x,out=y[::2])
8 print(y)
```

Các tính năng nâng cao

Phép toán tổng hợp

Sử dụng phương thức reduce hoặc accumulate với bất kỳ phép toán hiệu năng cao

Ví dụ

```
1 import numpy as np
2 x=np.arange(1,5)
3 print("Sum=",np.add.reduce(x))
4 print("Product=",np.multiply.reduce(x))
5 print("Sums=",np.add.accumulate(x))
6 print("Products=",np.multiply.accumulate(x))
```

Các tính năng nâng cao

Phép toán tổng hợp

Trong các trường hợp cụ thể, ta nên sử dụng các hàm chuyên dụng: np.sum, np.prod, np.cumsum, np.cumprod

Ví dụ

```
1 import numpy as np
2 big_array=np.random.random(1000000)
3 %timeit sum(big_array)
4 %timeit np.sum(big_array)
```

Các tính năng nâng cao

Ví dụ

```
1 import numpy as np
2 M=np.random.randint(1,100,size=(3,4))
3 print(M)
4 print(np.sum(M))
5 print(np.sum(M,axis=0))
6 print(np.sum(M,axis=1))
```

Các tính năng nâng cao

Danh sách phép toán tổng hợp

np.sum	np.nansum	Tính toán tổng các phần tử
np.prod	np.nanprod	Tính tích các phần tử
np.mean	np.nanmean	Tính trung bình của các phần tử
np.std	np.nanstd	Tính độ lệch chuẩn
np.var	np.nanvar	Tính phương sai
np.min	np.nanmin	Tìm giá trị nhỏ nhất
np.max	np.nanmax	Tìm giá trị lớn nhất

Các tính năng nâng cao

Danh sách phép toán tổng hợp

np.argmin	np.nanargmin	Tìm chỉ số của giá trị nhỏ nhất
np.argmax	np.nanargmax	Tìm chỉ số của giá trị lớn nhất
np.median	np.nanmedian	Tính trung vị của các phần tử
np.percentile	np.nanpercentile	Tính chỉ số xếp hạng dựa trên xác suất của các phần tử
np.any	N/A	Tồn tại một giá trị đúng
np.all	N/A	Tất cả các phần tử đều đúng

Các tính năng nâng cao

Tích ngoài

Sử dụng phương thức outer đối với bất kỳ phép toán hiệu năng cao nào

Ví dụ

```
1
2
3
4
5
6
7
```

```
import numpy as np
import numpy as np
x = np.arange(1, 6)
print(x)
print(np.add.outer(x, x))
print(np.multiply.outer(x, x))
```

Các tính năng nâng cao

Ví dụ

Dữ liệu về chiều cao của các tổng thống Mỹ được lưu trữ trong tệp president_heights.csv, giá trị của mỗi trường dữ liệu được đặt cách nhau bởi dấu ,. Hãy viết chương trình tính

- Chiều cao trung bình của các tổng thống
- Chiều cao lớn nhất và nhỏ nhất

Các tính năng nâng cao

Ví dụ

```
1   import numpy as np
2   import pandas as ps
3   import seaborn
4   import matplotlib.pyplot as plt
5   def vidu(height):
6       print('Sum height',np.sum(heights))
7       print('Mean height:',np.mean(heights))
8       print('Standart deviation:',np.std(heights))
9       print('Minimun height:',np.min(heights))
10      print('Max height:',np.max(heights))
11      print('argmin height:',np.argmin(heights))
12      print('10th percentile:',np.percentile(heights,10))
13      print('Median:',np.median(heights))
14
```

Các tính năng nâng cao

Ví dụ

```
1           def draw(height):
2               seaborn.set()
3               plt.hist(height)
4               plt.title('Height Distribution of US Presidents')
5               plt.xlabel('height (cm)')
6               plt.ylabel('number')
7
8           if __name__=="__main__":
9               data=ps.read_csv('data\president_heights.csv')
10              heights=np.array(data['height(cm)'])
11              vidu(heights)
12              draw(heights)
13
```

Các phép toán với mảng

Ví dụ

```
1  import numpy as np
2  A=np.array([[11,12,13],[21,22,23],[31,32,33]])
3  print(A)
4  B=np.ones((3,3))
5  print(B)
6  C=A+B
7  print(C)
8  C=A*(B+1)
9  print(C)
10 C=np.power(2,B)
11 print(C)
```

Các phép toán với mảng

Tích vô hướng

Tích vô hướng được định nghĩa:

$$dot(a, b, out = None)$$

- a là một mảng dữ liệu
- b là một mảng dữ liệu

Các phép toán với mảng

Ví dụ

```
1  x=np.array([1,2,3])
2  y=np.array([3,4,5])
3  print("Inner product")
4  print(np.dot(x,y))
5  C=np.dot(A,B)
6  print(C)
7  A = np.array([ [1, 2, 3],
8                 [3, 2, 1] ])
9  B = np.array([ [2, 3, 4, -2],
10                [1, -1, 2, 3],
11                [1, 2, 3, 0] ])
12 if (A.shape(-1)==B.shape(-2)):
13     C=np.dot(A,B)
14     print(C)
```

Các phép toán với mảng

Ví dụ

Giả sử có bốn người, và chúng tôi gọi họ là Lucas, Mia, Leon và Hannah. Mỗi người trong số họ đã mua sôcôla trong số ba lựa chọn với các thương hiệu là A, B và C. Lucas đã mua 100 g nhãn hiệu A, 175 g nhãn hiệu B và 210 nhãn hiệu C. Mia chọn 90 g A, 160 g B và 150 g C. Leon mua 200 g A, 50 g B và 100 g C. Hannah không thích thượng hiệu B và cô ấy dường như là một fan hâm mộ thực sự của thương hiệu C, vì cô ấy đã mua 310 g C và mua 120 g A. Giá bằng Euro của những viên sô cô la này là: A có giá 2,98 trên 100 g, B có giá 3,90 và C chỉ 1,99 Euro. Hãy tính số tiền mỗi người phải trả để mua sôcôla

Các phép toán với mảng

Ví dụ

```
1 import numpy as np
2 numpers=np.array([[100,175,210],[90,160,150],[200,50,100],[310,0,120]])
3 price_of_100_g=np.array([2.98,3.90,1.99])
4 Prices=np.dot(numberpersons,price_of_100_g)
5 Prices_in_Euro=Prices/100;
6 print(Prices_in_Euro);
```

Array Broadcasting

Broadcasting

Là một tập các quy tắc được áp dụng cho các phép toán số học hiệu năng cao trên các mảng có kích thước khác nhau

Ví dụ

Chúng ta muốn cộng thêm một vector hằng vào từng hàng của một ma trận.

Array Broadcasting

Cách 1

```
1  import numpy as np
2  x=np.array([[1,2,3],[2,3,4]])
3  v=np.array([0,1,2])
4  y=np.empty_like(x)
5  print(x.shape[0])
6  for i in range(x.shape[1]):
7  y[:,i]=x[:,i]+v[i]
8  print(y)
9
```

- Ma trận x là rất lớn thì việc thực hiện lặp sẽ rất chậm

Array Broadcasting

Cách 2

```
1
2
3
4
5
6
7
8
9
```

```
import numpy as np

x=np.array([[1,2,3],[2,3,4]])
v=np.array([0,1,2])
vv=np.tile(v,(4,1))
y=np.empty_like(x)
y=x+vv
print(y)
```

- Tạo ma trận vv có kích thước như ma trận y.

Array Broadcasting

Cách 3

```
1
2
3
4
5
6
7
8
```

```
import numpy as np

x=np.array([[1,2,3],[2,3,4]])
v=np.array([0,1,2])
y=np.empty_like(x)
y=x+v
print(y)
```

- Sử dụng quy tắc Broadcasting

Array Broadcasting

Các quy tắc

1. Nếu hai mảng có kích thước số chiều khác nhau, thì hình dạng mảng có kích thước số chiều ít hơn sẽ được thêm chiều vào bên trái
2. Nếu hình dạng của hai mảng không khớp nhau theo một chiều nào đó, mảng có hình dạng một chiều là 1 sẽ được biến đổi để phù hợp với chiều tương ứng của mảng kia
3. Hai mảng sẽ không tương thích nếu sau khi áp dụng các quy tắc 1 và 2 ta thu được hai mảng mới không có cùng số chiều

Array Broadcasting

Các quy tắc

1. Nếu hai mảng có kích thước số chiều khác nhau, thì hình dạng mảng có kích thước số chiều ít hơn sẽ được thêm chiều vào bên trái
2. Nếu hình dạng của hai mảng không khớp nhau theo một chiều nào đó, mảng có hình dạng một chiều là 1 sẽ được biến đổi để phù hợp với chiều tương ứng của mảng kia
3. Hai mảng sẽ không tương thích nếu sau khi áp dụng các quy tắc 1 và 2 ta thu được hai mảng mới không có cùng số chiều

Array Broadcasting

Ví dụ

- array-1: 4 x 3
- array-2: 3

Array Broadcasting

Ví dụ

- array-1: 4 x 3
- array-2: 3
- result-shape: 4 x 3
- array-1: 8
- array-2: 5 x 2 x 8

Array Broadcasting

Ví dụ

- array-1: 4 x 3
- array-2: 3
- result-shape: 4 x 3
- array-1: 8
- array-2: 5 x 2 x 8
- result-shape: 5 x 2 x 8
- array-1: 5 x 2
- array-2: 5 x 4 x 2

Array Broadcasting

Ví dụ

- array-1: 4 x 3
- array-2: 3
- result-shape: 4 x 3
- array-1: 8
- array-2: 5 x 2 x 8
- result-shape: 5 x 2 x 8
- array-1: 5 x 2
- array-2: 5 x 4 x 2
- result-shape: INCOMPATIBLE

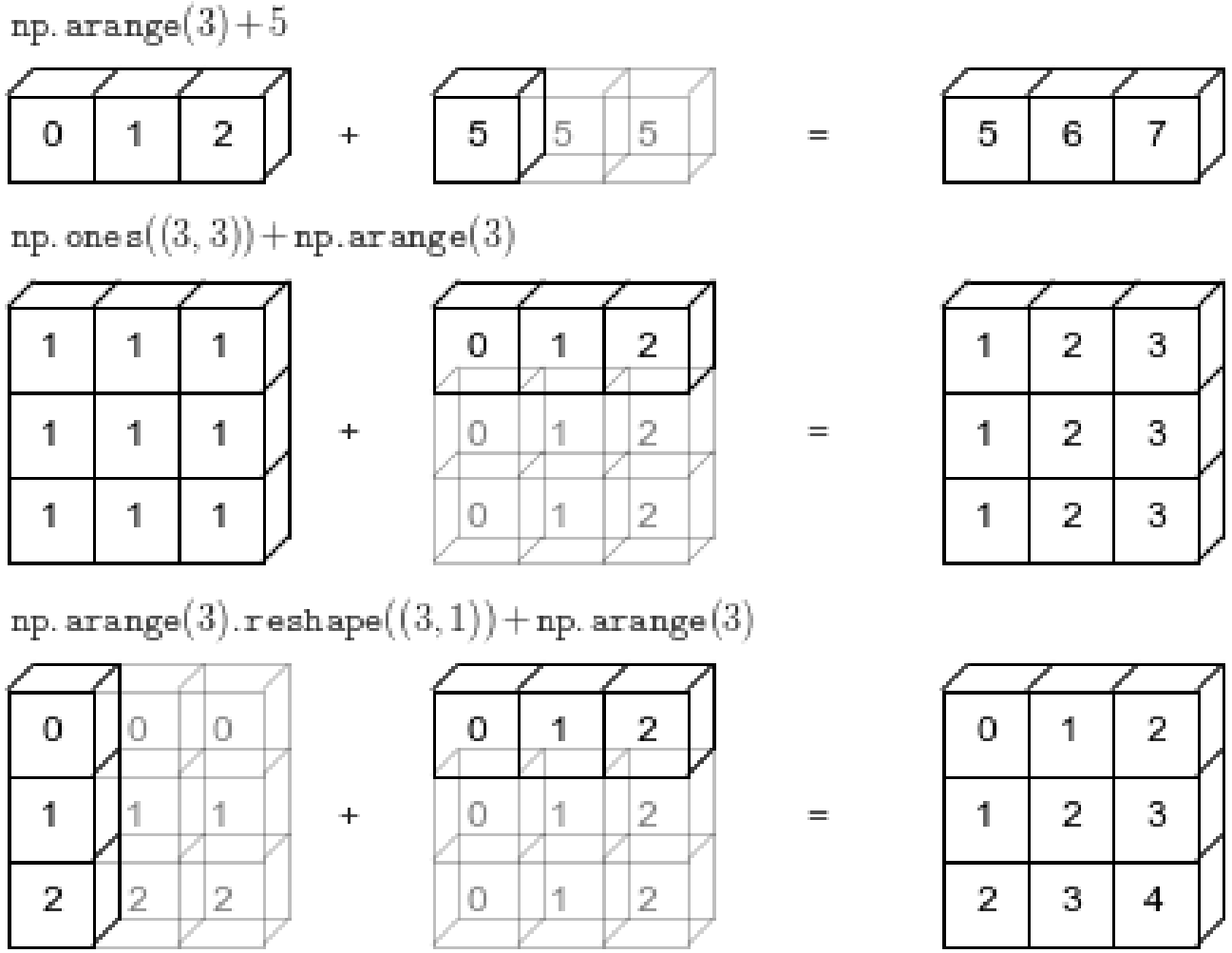
Array Broadcasting

Ví dụ

```
1  import numpy as np
2  a = np.arange(3)
3  print(a+5)
4  M = np.ones((3, 3))
5  print(M.shape,a.shape)
6  print(M+a)
7
8  M=np.arange(3).reshape((3,1))
9  print(M.shape)
10 print(M+a)
11
12 M=np.ones((3,2))
13 print(M.shape)
14 print(M+a)
15
```

Array Broadcasting

Hình 3: Hình ảnh minh họa



Array Broadcasting

Biến đổi kích thước mảng

Nếu bạn muốn biến đổi kích thước cho phù hợp có thể sử dụng phương thức reshape. Tuy nhiên, trong phần này chúng tôi sử dụng np.newaxis

Ví dụ

```
1
2
3
4
5
6
```

```
import numpy as np
M = np.ones((2, 3))
a = np.arange(3)
print(a[:,np.newaxis].shape)
print(M+a[:,np.newaxis])
```

Array Broadcasting

Bài tập 1

Giả sử có 6 sinh viên trong một nhóm, mỗi người đã thực hiện 3 bài kiểm tra; một cách tự nhiên, bạn lưu trữ các điểm số này trong một mảng hai chiều. Để đánh giá học lực của sinh viên chúng ta tính độ lệch giữa điểm đạt được của sinh viên với điểm trung bình đạt được của các sinh viên theo từng môn học xác định.

```
1
2   import numpy as np
3   grades = np.array([[ 0.79,  0.84,  0.84],
4                      [ 0.87,  0.93,  0.78],
5                      [ 0.77,  1.00,  0.87],
6                      [ 0.66,  0.75,  0.82],
7                      [ 0.84,  0.89,  0.76],
8                      [ 0.83,  0.71,  0.85]])
9   print(grades)
```

Array Broadcasting

Cách 1

```
1 mean_exam_scores=np.mean(grades,axis=0)
2 mean_exam_scores=np.round(mean_exam_scores,2)
3 print(mean_exam_scores)
4 score_offset=np.empty_like(grades)
5 print(grades.shape[0])
6 for n in range(grades.shape[1]):
7         score_offset[:,n]=grades[:,n]-mean_exam_scores[n]
8 print(score_offset)
```

Cách 2

```
1 score_offset=grades-mean_exam_scores
2 print(score_offset)
```

Array Broadcasting

Bài tập 2

Tạo một mảng ngẫu nhiên 10.000 điểm 2D bằng cách sử dụng np.random.rand. Tính toán tâm của các điểm là tọa độ x trung bình và tọa độ y trung bình của 10.000 điểm này. Sau đó, tính toán độ lệch tâm của các điểm. Ví dụ: nếu tâm là (0,5,1) và vị trí tuyệt đối của một điểm là (2,3), thì độ lệch tâm của điểm này đơn giản là (2,3) - (0,5,1) = (1,5,2).

Array Broadcasting

Cách 1

```
1 import numpy as np
2 pts=np.random.rand(1000,2)
3 print(pts)
4 center_of_mass=pts.mean(axis=0)
5 relative_pos = pts - center_of_mass
6 print(relative_pos)
```

Chỉ số Boolean

Ví dụ

```
1 import numpy as np
2 A = np.array([4, 7, 3, 4, 2, 8])
3 print(A == 4)
```

Kết quả là một mảng trong đó các phân tử giá trị có kiểu dữ liệu Boolean là kết quả của phép toán so sánh bằng giữa phần tử tương ứng của mảng A với 4

Chỉ số Boolean

Ví dụ

```
1 import numpy as np
2 A = np.array([4, 7, 3, 4, 2, 8])
3 print(A<5)
4 B=np.random.randint(9,size=(3,3))
5 print(B)
6 print(B>5)
```

Fancy Indexing

Fancy Indexing

Lập chỉ mục một mảng C bằng cách sử dụng mặt nạ Boolean của một mảng để chọn các phần tử tương ứng của một mảng khác. Mảng R mới chứa tất cả các phần tử của C trong đó giá trị tương ứng của (A <= 5) là True.

```
1 import numpy as np
2 C = np.array([123,188,190,99,77,88,100])
3 A=np.array([4,7,2,8,6,9,5])
4 R=C[A<=5]
5 print(R)
```

Fancy Indexing

Fancy Indexing

Một mảng được đánh chỉ số bằng một mặt nạ Boolean hoặc một mảng các số nguyên

```
1 import numpy as np
2 C = np.array([123,188,190,99,77,88,100])
3 R=C[[2,3,4,5]]
4 print(R)
```

Fancy Indexing

Bài tập 1

Extract from the array np.array([3,4,6,10,24,89,45,43,46,99,100]) with Boolean masking all the number

1. which are not divisible by 3
2. which are divisible by 5
3. which are divisible by 3 and 5
4. which are divisible by 3 and set them to 42

nonzero and where

nonzero

numpy.nonzero (a) và a.nonzero () trả về các chỉ số của các phần tử khác 0 của a.

```
1 a=np.array([[0,1,2],[0,0,3],[1,2,3]]);
2 print(a);
3 print(np.nonzero(a))
4 print(np.transpose(a.nonzero()))
5 print(a[a.nonzero()])
```

nonzero and where

nonzero

Hàm 'nonzero'có thể sử dụng để nhận được các chỉ số của một mảng thỏa mãn một điều kiện cho trước

```
1 B = np.array([[42,56,89,65],
2               [99,88,42,12],
3               [55,42,17,18]])
4 print(B >= 42)
5 print(np.nonzero(B>=42))
6 print(B[np.nonzero(B>=42)])
```

Bài tập

Bài tập 2

Calculate the prime numbers between 0 and 100 by using a Boolean array.

nonzero and where

count_nonzero

Hàm count_nonzero: Đếm số phần tử khác không trong mảng

```
1 import numpy as np
2 a=np.array([[0,1,2],[0,0,3],[1,2,3]]);
3 print(a);
4 print(np.count_nonzero(a))
```

Sắp xếp dữ liệu

Sắp xếp dọc theo một chiều dữ liệu

Hàm np.sort để sắp xếp dữ liệu

Ví dụ

```
1  import numpy as np
2  a=np.array([[4,3,5],[1,2,1]])
3  #Sap xep theo dong'
4  b=np.sort(a,axis=1)
5  #Sap xep theo cot
6  c=np.sort(a,axis=0)
7  print(b)
8  print(c)
9
```

Sắp xếp dữ liệu

Sắp xếp dữ liệu với các chỉ số

Hàm np.argsort để sắp xếp dữ liệu với các chỉ số

Ví dụ

```
1
2   import numpy as np
3   a = np.array([4, 3, 1, 2])
4   j = np.argsort(a)
5   print(j)
6   print(a[j])
```

Sắp xếp dữ liệu

Sắp xếp một phần

Để tìm k giá trị nhỏ nhất trong mảng. NumPy cung cấp hàm np.partition. Dữ liệu đầu vào của hàm np.partition là một mảng dữ liệu và một số k; kết quả là một mảng mới với các giá trị k nhỏ nhất ở bên trái của phân vùng và các giá trị còn lại ở bên phải, theo thứ tự tùy ý.

```
1
2  import numpy as np
3  a = np.array([4, 3, 1, 2])
4  j = np.argsort(a)
5  print(j)
6  print(a[j])
```

Tính toán cơ bản trong đại số tuyến tính

Sử dụng module numpy.linalg

Tính toán cơ bản trong đại số tuyến tính

numpy.linalg.eig(a)	Tính giá trị riêng và vec tơ riêng của ma trận vuông
numpy.linalg.eigvals(a)	Tính giá trị riêng của ma trận
numpy.linalg.norm(x)	Tính chuẩn của ma trận hoặc vector
numpy.linalg.det(a)	Tính định thức của ma trận a
numpy.linalg.inv(a)	Tính ma trận nghịch đảo của ma trận a
numpy.linalg.pinv(a)	Tính ma trận nghịch đảo suy rộng của ma trận a