GIỚI THIỆU VỀ NUMPY

Thư viện Numpy được sử dụng cho phân tích dữ liệu kiểu số hoặc kiểu dữ liệu tính toán. Nó có bộ nhớ dữ liệu tốt và hiệu quả hơn rất nhiều so với thư viện pandas. Ngoài ra, nó còn có một bộ các phương thức phong phú để phân tích dữ liệu kiểu số như kiểu chuỗi số, kiểu dữ liệu mảng đa chiều như ma trận, ...

Cấu trúc: Trong phần này, chúng ta sẽ thảo luận về các chủ đề sau:

- Numpy là gì?
- Mảng trong Numpy
- Tạo ra mảng Numpy
- Tạo ra mảng Numpy bằng cách sử dụng list và tuple trong Python
- Tạo ra mảng bằng cách sử dụng chuỗi số
- Chỉ mục và kĩ thuật slicing trong mảng Numpy
- Kiểu dữ liệu trong Numpy
  - Lấy kiểu dữ liệu và kho chứa thông tin bộ nhớ của mảng Numpy
  - Tạo ra mảng Numpy cùng với định nghĩa kiểu dữ liệu
  - Cấu trúc kiểu dữ liệu và lưu kiểu
- Cách thao tác hình dạng trong mảng Numpy
- Chèn thêm và xóa bớt các phần tử mảng
- Thêm gia và phân chia các mảng Numpy
- Chức năng trạng thái trong Numypy
- Vận dụng kiểu số trong Numpy
- Sắp xếp trong Numpy
- Lưu dữ liệu vào tệp
- Đọc dữ liệu từ tệp

Mục tiêu: Sau khi hoàn thành các nội dung phần này, học viên đạt:

- Hiểu được khái niệm và ứng dụng của Numpy trong phân tích dữ liệu kiểu số.
- Biết cách tạo ra mảng 1 chiều và mảng đa chiều.
- Hiểu các chức năng của Numpy để nhập dữ liệu từ tệp vào một mảng.
- Thực hiện các thao tác mảng và phân tích dữ liệu bằng cách sử dụng các chức năng và phương thức trong Numpy.
- Ghi lại hoặc xuất ra dữ liệu của mảng đa chiều vào một tệp ngoài.

1. Numpy

Numpy (Số trong Python) là một thư viện được sử dụng rộng rãi trong Python cho mục đích khoa học tính toán; nó có hiệu quả bộ nhớ tốt và nhanh. Nó là mảng đa chiều và một bộ sưu tập chức năng phong phú để thực hiện quá trình phân tích dữ liệu.

2. Mảng trong Numpy

Trong Numpy, mảng đa chiều đồng nhất là loại mảng cơ bản nhất. Mảng Numpy có thể chỉ chứa các kiểu phần tử giống nhau (ví dụ như mảng các số nguyên). Trong mảng numpy, mảng đa chiều liên quan đến trục.

Ví dụ, một mảng 1 chiều ([1,2,3]) sẽ có một trục với 3 phần tử.

Mảng trong numpy có 2 loại trục: trục axis=0 là tập hợp các phần tử của mảng và trục axis=1 là tập hợp bên trong các phần tử của mảng.

3. Tạo ra mảng Numpy

Trong Numpy, có thể tạo ra mảng đa chiều bằng cách sử dụng hàm **array()**;

Ngoài ra, có thể tạo ra một mảng Numpy bằng cách từ một list hoặc tuple trong hàm **array()**.

Dưới đây là một ví dụ:

```
import numpy as np

array1_1D = np.array([1,2,3,4,5])
array2_1D = np.array((2,3,4,5))

print("Output")
print("array1_1D:", array1_1D)
print("array2_1D:", array2_1D)
```

```
Output:
array1_1D: [1 2 3 4 5]
array2_1D: [2 3 4 5]
```

Hoặc có thể sử dụng **dtype** trong hàm **array()** để đổi kiểu dữ liệu trong mảng:

```
import numpy as np

array1_1D = np.array([1,3,5], dtype='complex')

print("Output:")
print("array1_1D:", array1_1D)
```

```
Output:
array1_1D: [1.+0.j 3.+0.j 5.+0.j]
```

Đối với các mảng đa chiều thì cũng có thể khai báo tương tự như vậy.

Trong Numpy, có thể tạo ra các mảng có tính chất đặc biệt, ví dụ như một mảng chỉ có số 0, và mảng rỗng,...

- Mảng chỉ có số 0

```
import numpy as np
array_zeros = np.zeros((2,3))
```

```
print("Output:")
array_zeros
```

```
Output:

array([[0., 0., 0.],
       [0., 0., 0.]])
```

- Mảng rỗng

```
import numpy as np
array_empty = np.empty((3,3))
print("Output:")
array_empty
```

```
Output:

array([[0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
       [0.00000000e+000, 0.00000000e+000, 5.88926250e-321],
       [9.34608431e-307, 1.42410974e-306, 2.56761491e-312]])
```

4. Tạo ra các mảng Numpy bằng cách sử dụng list và tuple trong Python

Tạo ra mảng Numpy bằng cách sử dụng list hoặc tuple đang có sẵn, sử dụng chức năng như hàm **array()**:

```
import numpy as np

t1=[1,2,3]
print("\n1-D Array from Python list:\n:", np.asarray(t1,dtype='int16'))

t2=[[1,2,3],[2,3,4]]
print("\n2-D Array from Python list:\n:", np.asarray(t2,dtype='int16'))
```

```
1-D Array from Python list:
 [1 2 3]

2-D Array from Python list:
 [[1 2 3]
 [2 3 4]]
```

Có thể khai báo mảng 1 và đa chiều đối với tuple tương tự như chúng ta đã làm với list.

5. Tạo ra mảng sử dụng chuỗi số

Tạo ra một mảng từ một dãy số; Numpy có hàm arange(start, stop, step). Với sự hỗ trợ của hàm này, có thể tạo ra một mảng. Hãy xem ví dụ bên dưới:

```
import numpy as np
np.arange(1,10,2)
```

```
array([1, 3, 5, 7, 9])
```

6. Chỉ mục và kĩ thuật slicing trong mảng Numpy

Có thể sử dụng kĩ thuật slicing như sau:

```
import numpy as np
a= np.arange(10,20,2)
```

```
print("Input Array:\na =", a)
print("\nSlicing Examples:")

print("\nExample#1 : a[:] =", a[:])
print("\nExample#2 : a[0:2] =", a[0:2])
print("\nExample#3 : a[:3] =", a[:3])
print("\nExample#4 : a[3:] =", a[3:])
print("\nExample#5 : a[-3:-1] =", a[-3:-1])
```

```
Input Array:
a = [10 12 14 16 18]

Slicing Examples:

Example#1 : a[:] = [10 12 14 16 18]

Example#2 : a[0:2] = [10 12]

Example#3 : a[:3] = [10 12 14]

Example#4 : a[3:] = [16 18]

Example#4 : a[-3:-1] = [14 16]
```

Đối với mảng 2 chiều thì có ví dụ sau:

```
import numpy as np
b = np.arange(20)
b.resize(4,6)
print("Input multi-dimensional Array:\n b=", b)

print("\n Slicing Examples:")
print("\nExamples#1 : b[0] =", b[0])
print("\nExamples#2 : b[0:2] =", b[0:2])
print("\nExamples#3 : b[0:2, 0:2] =", b[0:2, 0:2])
print("\nExamples#4 : b[2:, 4:] =", b[2:, 4:])
```

```
Input multi-diemnstional Array:
 b= [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19  0  0  0  0]]

 Slicing Examples:

Example#1 : b[0] = [0 1 2 3 4 5]

Example#2 : b[0:2] = [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

Example#3 : b[0:2, 0:2] = [[0 1]
 [6 7]]

Example#4 : b[2:, 4:] =  [[16 17]
 [ 0  0]]
```

7. Kiểu dữ liệu trong Numpy

Kiểu dữ liệu	Mã code	Mô tả
int	i	số nguyên có dấu
uint	u	số nguyên không dấu
float	f	số thực
bool	?	đúng hoặc sai
complex	c	số phức
string	S	

unicode string	U	
datetime	m	
timedelta	M	

8. Cách thao tác hình dạng trong mảng Numpy

Chức năng/Phương thức	Mô tả
reshape()	trả lại giá trị mảng với hình dạng nhất định mà không sửa đổi dữ liệu
flat()	làm nhẵn mảng rồi trả lại các phần tử của các vị trí trong mảng
flatten()	trả lại giá trị mảng một chiều được cóp lại của mảng nhập vào
ravel()	trả lại giá trị mảng một chiều của mảng nhập vào
transpose()	chuyển đổi các trục
resize()	giống như reshape(), nhưng resize định dạng mảng nhập vào khi được áp dụng, hoặc định dạng mảng có liên quan

9. Chèn thêm và xóa bớt các phần tử mảng

Có các hàm để chèn và xóa như dưới đây:

- **numpy.append()**

```
num_array = np.array([1,2,3,45,6])

print("\nExisting Array:\n",num_array)

new_num_array = np.append(num_array,[10])
print("\nNew array with appended element:\n",new_num_array)
```

```
Existing Array:
 [ 1  2  3 45  6]

New array with appended element:
 [ 1  2  3 45  6 10]
```

Khi làm việc với mảng đa chiều thì sẽ như sau:

```
a= np.array([[1,2,3],[4,5,6]])
print("\nExisting Array:\n",num_array)

a1 = np.append(a,[7,8,9])
print("\nNew array with appended element (without axis):\n",a1)

a2 = np.append(a,[[7,8,9]],axis=0)
print("\nNew array with appended element (without axis):\n",a2)

a3 = np.append(a,[[0],[1]],axis=1)
```

```
print("\nNew array with appended element (without axis):\n",a3)
```

```
Existing Array:
 [[1 2 3]
 [4 5 6]]

New array with appended element (without axis):
 [1 2 3 4 5 6 7 8 9]

New array with appended element (without axis):
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

New array with appended element (without axis):
 [[1 2 3 0]
 [4 5 6 1]]
```

- **numpy.insert()**

```
num_array = mp.array([1,2,3,45,6])
print("\nExisting Array:\n",num_array)

new_num_array = np.insert(num_array,1,[10])
print("\nNew array with inserted element:\n",new_num_array)
```

```
Existing Array:
 [ 1  2  3 45  6]

New arry with inserted element:
 [ 1 10  2  3 45  6]
```

- **numpy.delete()**

```
num_array = mp.array([1,2,3,45,6])
print("\nExisting Array:\n",num_array)

new_num_array = np.delte(num_array,1)
print("\nNew array with inserted element:\n",new_num_array)
```

```
Existing Array:
 [ 1  2  3 45  6]

New arry with inserted element:
 [ 1  3 45  6]
```

10. Nối và phân chia các mảng Numpy

Để nối hoặc phân chia mảng, có các hàm để thực hiện như sau:

- **numpy.concatenate():**

```
array_1 = np.array([[1,2],[3,4]])
array_2 = np.array([[5,6],[7,8]])
print("\nFirst array :\n",array_1)
print("\nSecond array :\n",array_2)

array_concat = np.concatenate((array_1,array_2))
print("\nExample#1 : Concatenated Array with default axis :\n",array_concat)

array_concat = np.concatenate((array_1,array_2),axis=1)
print("\nExample#1 : Concatenated Array with axis=1 :\n",array_concat)
```

```
First array :
 [[1 2]
 [3 4]]

Second array :
 [[5 6]
 [7 8]]

Example#1 :Concated Array with default axis :
 [[1 2]
 [3 4]
 [5 6]
 [7 8]]

Example#2 :Concated Array with axis=1 :
 [[1 2 5 6]
 [3 4 7 8]]
```

- **numpy.hstack():**

```
array_1 = np.array([[1,2],[3,4]])
array_2 = np.array([[5,6],[7,8]])
print("\nFirst array :\n",array_1)
print("\nSecond array :\n",array_2)

array_hstacked = np.hstack((array_1,array_2))
print("\n: Concatenated Array with hstack() :\n",array_hstacked)
```

- **numpy.vstack():** Hàm này chúng ta sử dụng giống như hàm concatenate() với axis=1.
- **numpy.split():**

```
import numpy as np

a=np.array([1,2,3,4,5,6,7,8])
print("\nInput array Before split()\n",a)

print("\nOutput array Before split()\n",np.split(a,4)
```

```
Input array Before split()
 [1 2 3 4 5 6 7 8]

Output array Aefore split()
 [array([1, 2]), array([3, 4]), array([5, 6]), array([7, 8])]
```

- **numpy.hsplit():**

```
array_input = np.array([(1,2,3,4),(6,7,8,9)])

array_1, array_2 = np.hsplit(array_input, 2)

print("\nArray Before hsplit()\n",array_input)
print("\nHorizontal split array 1\n:",array_1)
print("\nHorizontal split array 2\n:",array_2)
```

```
Array Before hsplit()
 [[1 2 3 4]
 [6 7 8 9]]

Horizontal split array 1
: [[1 2]
 [6 7]]

Horizontal split array 2
: [[3 4]
 [8 9]]
```

- **numpy.vsplit():**

```
array_input = np.array([(1,2,3,4,10,11),(6,7,8,9,10,11)])
```

```
array_1,array_2 = np.vsplit(array_input,2)
print("\nArray Before vsplit()\n",array_input)
print("\nvertically split array 1\n:",array_1)
print("\nvertically split array 2\n:",array_2)
```

```
Array Before vsplit()
 [[ 1  2  3  4 10 11]
 [ 6  7  8  9 10 11]]

vertically splited array 1: [[ 1  2  3  4 10 11]]

vertically splited  array 2: [[ 6  7  8  9 10 11]]
```

11. Hàm trạng thái trong Numpy

Để hiểu hơn về dữ liệu và cách sử dụng nó, cần có trợ giúp của các phương thức thông tin trạng thái như số trung bình, trung vị,... Trong thư viện Numpy cũng có nhiều chức năng để lấy được thông tin trạng thái của dữ liệu. Dưới đây là một số các chức năng quan trọng mà mọi người thường sử dụng:

- **numpy.amin() và numpy.amax():** đây là hai hàm một cái lấy số lớn nhất trong mảng và lấy số bé nhất trong mảng, có thể áp dụng nó để lấy mảng có tổng giá trị lớn nhất và bé nhất trong mảng đa chiều.
- **numpy.mean() và numpy.average():** cả hai hàm này đều dùng để tính số trung bình của các giá trị trong mảng.
- **numpy.median():** hàm này dùng để lấy số có ở trong mảng với giá trị ở giữa.
- **numpy.std():** hàm này dùng để tính độ lệch chuẩn của các số có trong mảng.
- **numpy.var():** hàm này dùng để tính phương sai của các số có trong mảng.
- **numpy.percentile():** hàm này dùng để tính phần trăm của các số có trong mảng.

12. Vận dụng kiểu số trong Numpy

Dưới đây là một số chức năng giúp chúng ta có thể vận dụng thư viện Numpy tốt hơn:

- **Add, substract, multiply và divide:**

```
import numpy as np
arr1 = np.arange(12, dtype = np.float_).reshape(3,4)
arr2 = np.array([1,2,3,4]], dtype = np.float_)

print("First array – arr1:\n",arr1)
print("Second array – arr1:\n",arr2)

result_add = np.add(arr1,arr2)
result_substract = np.substract(arr1,arr2)
result_multiply = np.multiply(arr1,arr2)
result_divide = np.divide(arr1,arr2)

print("\nResult after adding arr1 with arr2:\n"),result_add)
print("\nResult after substracting arr1 with arr2:\n"),result_substract)
print("\nResult after multiplying arr1 with arr2:\n"),result_multiply)
print("\nResult after dividing arr1 with arr2:\n"),result_divide)
```

```
First array - arr1:
 [[ 0.  1.  2.  3.]
 [ 4.  5.  6.  7.]
 [ 8.  9. 10. 11.]]
Second array - arr2:
 [[1. 2. 3. 4.]]

Result after adding arr1 with arr2 (arr1+arr2) :
 [[ 1.  3.  5.  7.]
 [ 5.  7.  9. 11.]
 [ 9. 11. 13. 15.]]

Result after subtracting arr2 from arr1 (arr1-arr2)
 [[-1. -1. -1. -1.]
 [ 3.  3.  3.  3.]
 [ 7.  7.  7.  7.]]

Result after multiplying arr1 with arr2 (arr1*arr2):
 [[ 0.  2.  6. 12.]
 [ 4. 10. 18. 28.]
 [ 8. 18. 30. 44.]]

Result after dividing arr1 by arr2 (arr1/arr2) :
 [[0.         0.5        0.66666667 0.75      ]
 [4.         2.5        2.         1.75      ]
 [8.         4.5        3.33333333 2.75      ]]
```

- **numpy.power():**

```
import numpy as np
arr1 = np.arange(12, dtype = np.float_).reshape(3,4)
print("Input array -> arr1:\n",arr1)

power_2_array = np.power(arr1,2)
print("Output array:\n",power_2_array)
```

```
Input array -> arr1:
 [[ 0.  1.  2.  3.]
 [ 4.  5.  6.  7.]
 [ 8.  9. 10. 11.]]
Onput array :
 [[  0.   1.   4.   9.]
 [ 16.  25.  36.  49.]
 [ 64.  81. 100. 121.]]
```

- **numpy.mod():**

```
import numpy as np
arr1 = np.array([10,23,21,11])
arr2 = np.array([1,2,3,4])

print("First array – arr1:\n",arr1)
print("Second array – arr2:\n",arr2)

out_arr = np.mod(arr1,arr2)
print("Output array after mod(arr1,arr2):\n",out_arr)
```

```
First array - arr1:
 [10 23 21 11]
Second array - arr2:
 [1 2 3 4]
Output array after mod(arr1,arr2):
 [0 1 0 3]
```

- **numpy.reciprocal():**

```
import numpy as np
input_arr1 = np.array([[0.5,3,0.8],[4,1,7]])
input_arr2 = np.array([[2,3,4],[4,1,7]])
```

```
print("Input array – arr1:\n",input_arr1)
print("Reciprocal of arr1:\n",np.reciprocal(input_arr1))

print("Input array – arr2:\n",input_arr2)
print("Reciprocal of arr2:\n",np.reciprocal(input_arr2))
```

```
Input array - arr1:
 [[0.5 3.  0.8]
 [4.  1.  7. ]]
Reciprocal of arr1:
 [[2.         0.33333333 1.25      ]
 [0.25       1.         0.14285714]]

Input array - arr2:
 [[2 3 4]
 [4 1 7]]
Reciprocal of arr2:
 [[0 0 0]
 [0 1 0]]
```

13. Sắp xếp trong Numpy

Dưới đây là ví dụ về hàm **numpy.sort()** trong numpy:

```
a = np.array([[11,14],[13,11]])

print("Input Array is:\n",a)

out1=np.sort(a)
print("\nExample#1: Output sorted array when no axis defined means default axis:\n",out1)

out2 = np.sort(a,axis=None)
print("\nExample#2: Output sorted array when axis=None:\n",out2)

out3 = np.sort(a,axis=0)
print("\nExample#3: Output sorted array when axis=0:\n",out3)
```

```
Input Array is :
 [[11 14]
 [13 11]]

Example#1 :Output sorted array when no axis defined means defalut axis:
 [[11 14]
 [11 13]]

Example#2 :Output sorted array when axis=None:
 [11 11 13 14]

Example#3 :Output sorted array when axis=0 :
 [[11 11]
 [13 14]]
```

Dưới đây là một ví dụ về nó:

```
import numpy as np
emp_dt = np.dtype([('emp_name', 'S10'),('dept_no', int)])

arr_input = np.array([("Garima", 101), ("Sudesh", 101), ("Indra", 102),
                      ("Chitra", 103), ("Aparna", 101)], dtype = emp_dt)

print("Input array is:\n",arr_input)

out_arr = np.sort(arr_input, order = 'emp_name')
```

```
print("\nOutput sorted array order by emp_name field\n:",out_arr)
```

```
Input array is :
 [(b'Garima', 101) (b'Sudesh', 101) (b'Indra', 102) (b'Chitra', 103)
 (b'Aparna', 101)]

Outptut sorted array order by emp_name field
: [(b'Aparna', 101) (b'Chitra', 103) (b'Garima', 101) (b'Indra', 102)
 (b'Sudesh', 101)]
```

14. Ghi dữ liệu vào tệp

Có thể dùng các hàm sau để lưu dữ liệu vào tệp:

- **numpy.save():**

```
import numpy as np
emp_dt = np.dtype([('emp_name', 'U10'), ('dept_no', 'int')])
arr_input = np.array([("Ankit", 1010), ("Ravi", 101), ("John", 102),
                      ("Sam", 103)], dtype = emp_dt)
print("Input array is:\n",arr_input)

np.save("data/emp_dt.npy",arr_input)
print("\nfile {} been written using function np.save()!!".format('data/emp_dt.npy'))
```

```
Input array is :
 [('Ankit', 101) ('Ravi', 101) ('John', 102) ('Sam', 103)]

file data/emp_dt.npy been written using fucntion np.save()!!
```

- **numpy.savetxt():**

```
import numpy as np
emp_dt = np.dtype([('emp_name', 'U10'), ('dept_no', 'int')])
arr_input = np.array([("Ankit", 1010), ("Ravi", 101), ("John", 102),
                      ("Sam", 103)], dtype = emp_dt)
print("Input array is:\n",arr_input)

np.savetxt()("data/emp_dt.csv",arr_input,fmt = '%s', delimiter=',', header='emp_name,
dept_no')
print("\nfile {} been written using function np.savetxt()!!".format('data/emp_dt.csv))

num_array = np.array([[1,2,3,4], [5,6,7,8], [3,4,5,6]])

print("\nInput array is:\n",num_array)
np.savetxt("data/num_array.csv",num_array,fmt = '%d', delimiter =',')
print("\nfile {} been written using function np.savetxt()!!".format('data/num_array.csv))
```

```
Input array is :
 [('Ankit', 101) ('Ravi', 101) ('John', 102) ('Sam', 103)]

file data/emp_dt.csv been written using fucntion np.savetxt()!!

Input array is :
 [[1 2 3 4]
 [5 6 7 8]
 [3 4 5 6]]

file data/num_array.csv been written using fucntion np.savetxt()!!
```

15. Đọc dữ liệu từ tệp

Chúng ta có thể sử dụng các hàm sau đây để đọc dữ liệu từ tệp:

- **numpy.load():**

```
import numpy as np

array_read1 = np.load("data/emp_dt.npy")

print("Array read from file (data/emp_dt.npy) using np.load():\n",arrat_read1)
```

```
Array read from file (data/emp_dt.npy) using np.load() :
 [('Ankit', 101) ('Ravi', 101) ('John', 102) ('Sam', 103)]
```

- **numpy.loadtxt():**

```
emp_dt = np.dtype([('emp_name', 'U10'), ('dept_no', 'int')])

array_read2 = np.loadtxt("data/emp_dt.csv), delimiter=',', skiprows=1, dtype= emp_dt)
print("\nArray read from file (data/emp_dt.csv) using np.loadtxt():\n",array_read2)

array_read3 = np.loadtxt("data/num_array.csv), delimiter=',', skiprows=1, dtype= int)
print("\nArray read from file (data/emp_dt.csv) using np.loadtxt():\n",array_read3)
```

```
Array read from file (data/emp_dt.csv) using np.loadtxt() :
 [('Ankit', 101) ('Ravi', 101) ('John', 102) ('Sam', 103)]

Array read from file (data/num_array.csv) using np.loadtxt() :
 [[1 2 3 4]
 [5 6 7 8]
 [3 4 5 6]]
```

16. Kết luận

Trong phầnnày, các tính chất và chức năng của thư viện Numpy đã được giới thiệu. Chúng ta cũng đã được hiểu và thực hành thông qua các ví dụ về Numpy, cách để sử dụng nó và các chức năng và phương thức hữu dụng khác mà chúng ta cần để phân tích dữ liệu kiểu số.