Numpy Library

Nguồn tham khảo : W3S

numpy/numpy: The fundamental package for scientific computing with Python. (github.com)

> 💡 Numpy là một thư viện làm việc với mảng (array). Trong numpy có các function làm việc với đại số tuyến tính, biến đổi Fourier và ma trận.

Lý do sử dụng: Với `list` thông thường thì khi quá trình xử lý của máy `rất lâu` ⇒ `Numpy` ra đời

I. Numpy basic:

1. **Import numpy**

```
import numpy as np
```

2. **Array with numpy**

- Khai báo mảng với 1 phần tử (0 - D)

```
arr1 = np.array(15)
print(arr2)
#15
```

- Khai báo mảng 1 chiều : 1 - D

```
arr2 = np.array([1,5,7,10])
print(arr2)
#[1 5 7 10]
```

- Khai báo mảng 2 chiều: 2-D

```
arr3 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr3)
```

- Khai báo mảng 3 chiều: 3-D

```
arr4 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr4)
```

💡 **Tổng kết:** Số chiều khi khai báo mảng phụ thuộc vào số `[] mà không cùng lớp.` (Này theo sự tự hiểu của mình) Với ví dụ [[arr1][arr2]] , thì [arr1] cùng lớp với [arr2] nhưng đều khác lớp [] sở hữu nó ⇒ có 2 chiều. Chúng ta có thể dùng `print(arr.ndim)` để check số chiều của mảng

3. **Index trong numpy**

   Tương tự list, với các mảng có số chiều > 1 thì check value = index như sau

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])

#6
```

Thứ tự chiều của mảng sẽ tăng dần từ trái sang phải, bắt đầu là 1 - D.

Với một mảng `arr[a,b,c]` thì `a` là `index` của mảng tại chiều `1`, `b` là `index` của mảng tại chiều thứ `2`, `c` là `index` của mảng tại chiều `3`.

3. **Numpy array Slicing**

   Một số kiến thức tính nâng cao của slicing của numpy

```
#Tìm giá trị tại index= 2 trong 2-D của [0:2] mảng trong 1-D
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 2]) #[3 8]
```

```
#Tìm giá trị từ index [1:4] của 2-D thuộc 1-D có index=1
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4]) #[7 8 9]
```

```
#[::2] = tìm value của mảng từ 0 đến cuối cùng với step là 2
arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[::2])
#[1 3 5 7]
```

4. **Datatype với Numpy**

Numpy có thêm một số datatype mà Python bình thường không có

- `i` - integer
- `b` - boolean
- `u` - unsigned integer
- `f` - float
- `c` - complex float
- `m` - timedelta
- `M` - datetime
- `O` - object
- `S` - string
- `U` - unicode string
- `V` - fixed chunk of memory for other type (void)

```
#datatype
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='S')

print(arr)
print(arr.dtype)

#[b'1' b'2' b'3' b'4'] b: byte
#|S1
```

Sorry phần này hơi khó hiểu mình chưa hiểu kĩ

5. **Copy and View**

- Copy có sở hữu data, còn View thì không
- Copy a array

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```

- View a array

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42

print(arr)
print(x)
```

6. **Shape of array**

   Shape của một array trả về 1 tuple chứa số lượng phần từ = số lượng chiều của array, theo thứ tự từ chiều thứ nhất trở về sau (Chiều thứ nhất có index0). Giá trị tại mỗi chiều gồm số phần tử có trong chiều đó.

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

print(arr.shape)

# (2, 4)
```

7. **Reshape**

> `Số phần tử` trong `reshape()` tương ứng với `số chiều` sẽ tách

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)

print(newarr)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

Result

Tách 1 array thành một array mới với D-1 có 4 phần tử, mỗi phần tử có 3 giá trị (Hoặc chia thành 3 phần tử).

Yêu cầu là các số trong reshape (>0) nhân với nhau = số phần tử ban đầu mà arr có

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(2, 3, 2)

print(newarr)
```

⇒ Chuyển array thành một array mới với D-1 có 2 phần tử, D-2 có 3 phần tử, D-3 có 2 phần tử

> Ta có thể dùng -1 trong phần tử cuối của phần tử để tách tự động

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

newarr = arr.reshape(2, 2, -1)

print(newarr)
```

D-1 có 2 phần tử, D-2 có 2 phần tử , D-3=-1 sẽ tự động chia, D-3=2

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

result

> Reshape từ 1 mảng nhiều chiều thành mảng 1 chiều ta dùng (-1)

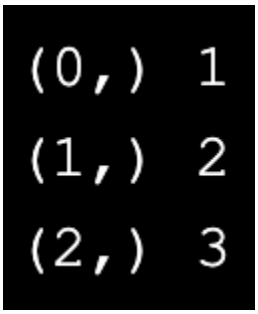
```
arr = np.array([[1, 2, 3], [4, 5, 6]])

newarr = arr.reshape(-1)

print(newarr)

# [1 2 3 4 5 6]
```

8. **NumPy Array Iterating**
   - Liệt kê (iterate) các phẩn tử trong mảng với vòng lặp for
   - Với `nditer()` : chúng ta có thể liệt kê từng phần tử căn bản nhất trong mảng với 1 for

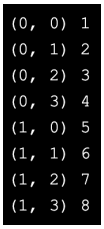
```
import numpy as np

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

for x in np.nditer(arr):
  print(x)
```

   - `ndenumerate()` : tương tự enumerate() thông thường, function này trả về index và value tại index đó. (a,b,c,…,) là index của giá trị đó tại mảng, liên hệ với kiến thức index trong mảng.

```
arr = np.array([1, 2, 3])

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```




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

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```



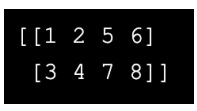
9. **Join Arrays**

> `concatenate():` Kết hợp (Add) nhiều mảng thành 1 mảng khác

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
#[1 2 3 4 5 6]
```

> Nối các mảng nhiều chiều theo trục, axis=0 là nối bth không có trục, axis=1 là 1 trục

```
arr1 = np.array([[1, 2], [3, 4]])

arr2 = np.array([[5, 6], [7, 8]])
```



```
arr = np.concatenate((arr1, arr2), axis=1)

print(arr)
```

axis=1

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

axis=0

> `stack()` : axis max = D(arr) ⇒ result = D+1 với mọi axis

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.stack((arr1, arr2), axis=1)

print(arr)
```

```
[[1 4]
 [2 5]
 [3 6]]
```

```
arr1 = np.array([[1, 2, 5],[4, 3, 3]])

arr2 = np.array([[2, 2, 3],[1, 5, 3]])

arr = np.stack((arr1, arr2))

print(arr)
```

```
[[[1 2 5]
  [4 3 3]]

 [[2 2 3]
  [1 5 3]]]
```

axis=0

```
arr1 = np.array([[1, 2, 5],[4, 3, 3]])

arr2 = np.array([[2, 2, 3],[1, 5, 3]])

arr = np.stack((arr1, arr2),axis=1)

print(arr)
```

```
[[[1 2 5]
  [2 2 3]]

 [[4 3 3]
  [1 5 3]]]
```

axis=1

```
arr1 = np.array([[1, 2, 5],[4, 3, 3]])

arr2 = np.array([[2, 2, 3],[1, 5, 3]])
```

```
arr = np.stack((arr1, arr2),axis=2)

print(arr)
```

```
[[[1 2]
  [2 2]
  [5 3]]

 [[4 1]
  [3 5]
  [3 3]]]
```

axis=2

`hstack()`: stack theo hàng ngang

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.hstack((arr1, arr2))

print(arr) # [1 2 3 4 5 6]
```

`vstack()` : stack theo cột

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.vstack((arr1, arr2))

print(arr)
```

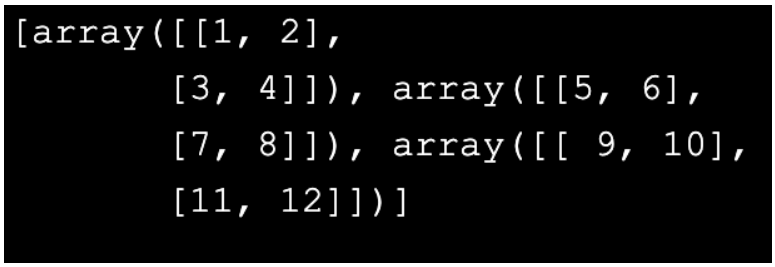
10. **Array split : array_split()**

    Sau khi split thì kết quả trả về sẽ là một list có các phần tử là các array

```
arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 3)
```

```
print(newarr)

#[array([1, 2]), array([3, 4]), array([5, 6])]
```

```
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])

newarr = np.array_split(arr, 3)

print(newarr)
```



11. **Sorting arrays**

`np.sort()` **:** tương tự như list