Meta Mind
Explore Knowledge - Open Future

Buổi 5: Numpy trong Khoa học dữ liệu & Final Project 1

AGENDA

Meta Mind
Explore Knowledge - Open Future

- Giới thiệu về Numpy
- Mảng trong Numpy
- Các hàm tính toán cơ bản trong Numpy
- Các hàm thống kê cơ bản trong Numpy
- Final Project Module 1

Meta Mind
Explore Knowledge - Open Future

Giới thiệu về Numpy

GIỚI THIỆU VỀ NUMPY

Import thư viện Numpy

```
import numpy as np
```

- NumPy là một thư viện trong Python.
- NumPy có tên đầy đủ là "**Num**erical **Py**thon".
- NumPy dùng để làm việc với các mảng, vector, ma trận, ...
- NumPy cung cấp các hàm tính toán trên dữ liệu mảng nhiều chiều.

Meta Mind
Explore Knowledge - Open Future

Mảng trong Numpy

MẢNG TRONG NUMPY

1. **Khởi tạo mảng với numpy**

    Khởi tạo thủ công, khởi tạo ngẫu nhiên, từ kiểu dữ liệu khác

2. **Truy cập mảng numpy**

    Slice indices

3. **Thay đổi kích thước của mảng**

    Đổi kích thước mảng

    Chuyển vị chiều

    Nối mảng

1. Khởi tạo mảng với numpy

Meta Mind
Explore Knowledge - Open Future

Có nhiều cách để khởi tạo một mảng numpy.

1. **Thủ công với np.array()**
   - Numpy cho phép khởi tạo mảng với kiểu dữ liệu chỉ định (float, integer, boolean, string, ...) bằng cách thêm tham số *dtype*.
   - Các phần tử cùng một mảng numpy phải đồng nhất về kiểu dữ liệu.

```
float_matrix = np.array([[1, 2],
                         [3, 4]], dtype=np.float64)

print(float_matrix)
print("dtype of matrix: ", float_matrix.dtype)
```

```
[[1. 2.]
 [3. 4.]]
dtype of matrix:  float64
```

   - Có thể đổi kiểu dữ liệu của một mảng numpy với hàm *astype()*.

```
print(float_matrix.dtype)
int_matrix = int_matrix.astype(np.int64) # đổi float64 sang int64
print(int_matrix.dtype)
```

```
float64
int64
```

2. Ngẫu nhiên với np.random

- Với các mảng kích thước lớn, việc nhập tất cả giá trị là không thể. Ta sẽ khởi tạo ngẫu nhiên cho những trường hợp này.
    1. Np.random.randn() - khởi tạo giá trị ngẫu nhiên theo phân phối chuẩn hoá (Normal Distribution)
    2. np.random.normal() - khởi tạo ngẫu nhiên theo phân phối chuẩn (Gaussian Distribution)
    3. np.random.uniform() - khởi tạo ngẫu nhiên theo phân phối đều (Uniform Distribution)
    4. np.random.randint() - khởi tạo ngẫu nhiên các số nguyên.
    5. và còn nhiều hàm random khác nữa

3. Từ kiểu dữ liệu khác

- Có thể khởi tạo mảng numpy từ các kiểu dữ liệu khác như list, tuple, pandas, ... bằng *np.array(<kiểu dữ liệu khác>)* hoặc các hàm được cung cấp cho việc chuyển đổi kiểu dữ liệu.

```
# Từ List Python sang mảng Numpy

my_list = [1, 2, 3, 4, 5]
print("Từ", type(my_list))
print(my_list)

np_array = np.array(my_list)
print("\nSang", type(np_array))
print(np_array)
```

```
Từ <class 'list'>
[1, 2, 3, 4, 5]

Sang <class 'numpy.ndarray'>
[1 2 3 4 5]
```

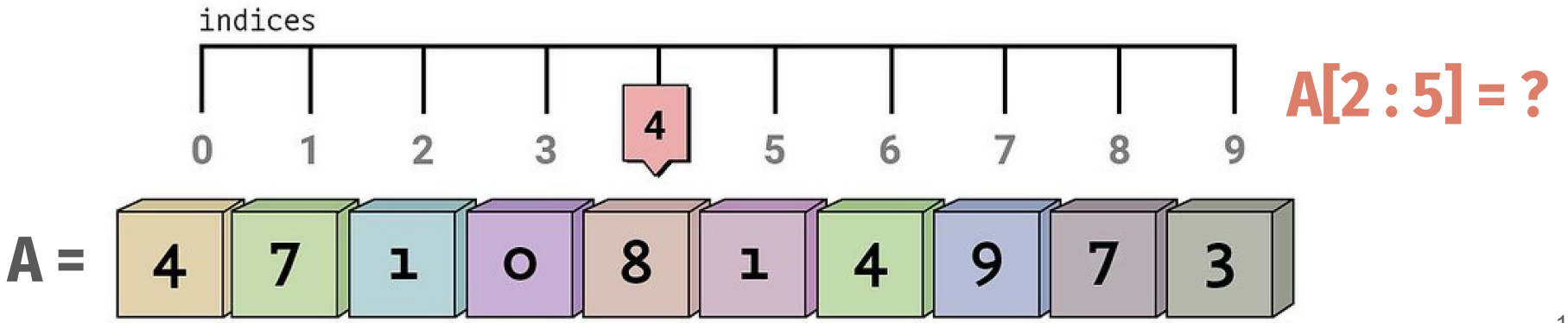
```
my_dataframe = pd.read_csv('/content/sample_data/california_housing_test.csv')
print("Từ",  type(my_dataframe))
print(my_dataframe.head(2))

np_array = my_dataframe.to_numpy()
print("\nSang", type(np_array))
print(np_array[:2])
```

```
Từ <class 'pandas.core.frame.DataFrame'>
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.05     37.37                27.0       3885.0           661.0
1    -118.30     34.26                43.0       1510.0           310.0

   population  households  median_income  median_house_value
0      1537.0       606.0         6.6085            344700.0
1       809.0       277.0         3.5990            176500.0

Sang <class 'numpy.ndarray'>
[[  -122.05       37.37       27.       3885.        661.       1537.
     606.         6.6085 344700.    ]
 [  -118.3        34.26       43.       1510.        310.        809.
     277.         3.599  176500.    ]]
```

2. Truy cập mảng numpy

Meta Mind
Explore Knowledge - Open Future

- Có thể truy cập mảng con theo các chiều của mảng cha dựa vào khai báo indices trên từng chiều với Slice indices.
  - Slice indices giúp rút ngắn việc phải nhập từng indices.
  - Slice indices được ngăn cách bởi dấu (**:**)
  - Phù hợp với các tình huống các vị trí cần truy cập là liên tục nhau.
  - Ví dụ: A[indice_start **:** indice_end **- 1**]

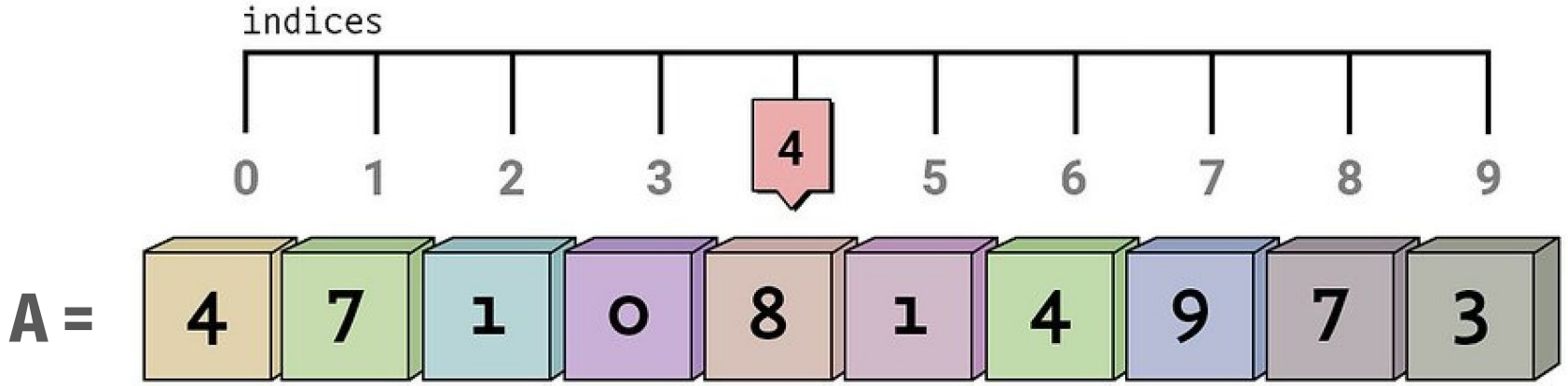


2. Truy cập mảng numpy

Meta Mind
Explore Knowledge - Open Future

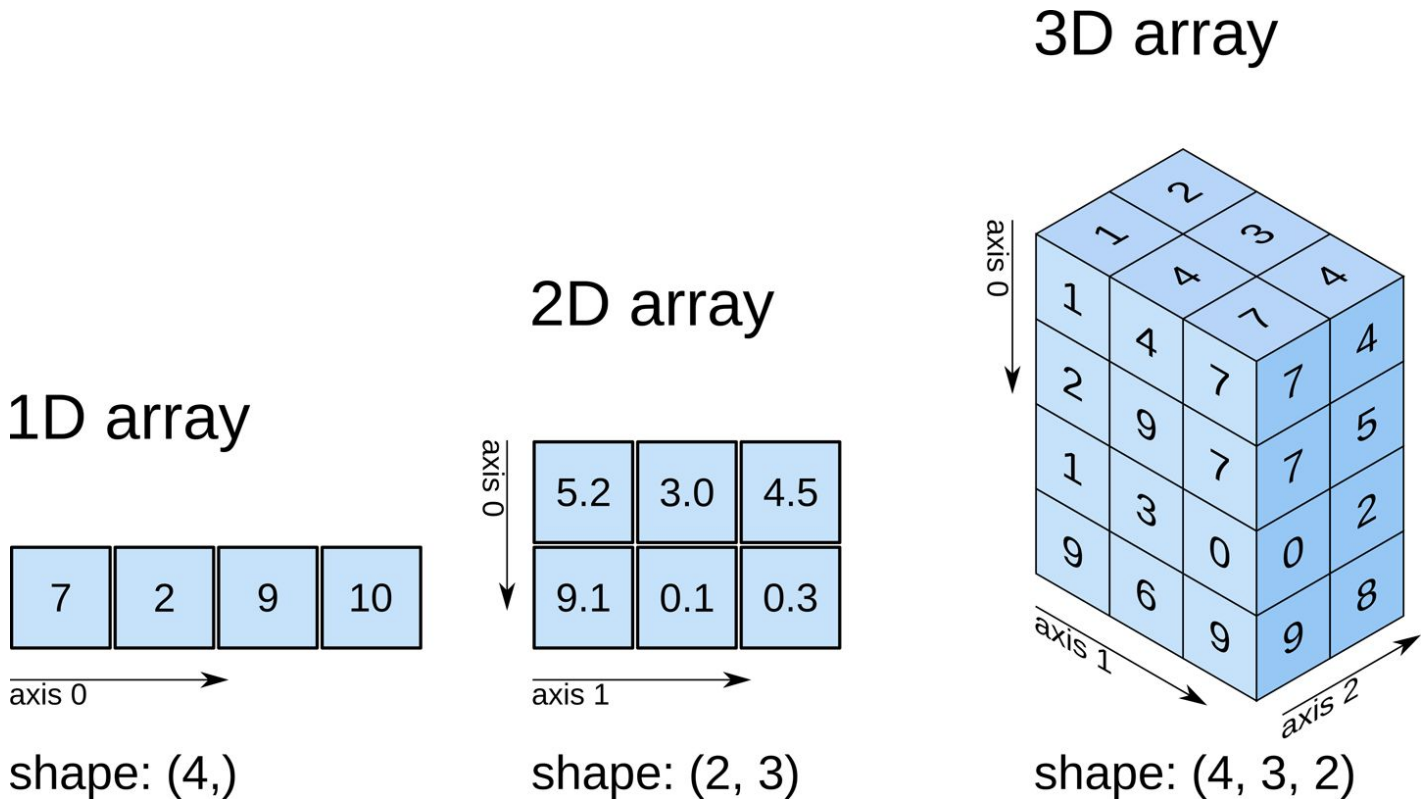
- Nếu để trống *indice_start*, NumPy sẽ hiểu là [0 : *indice_end*].
- Nếu để trống *indice_end*, NumPy sẽ hiểu là [*indice_start* : lấy đến hết mảng].
- Số âm (-) sẽ bắt đầu lấy phần tử từ vị trí cuối cùng đi lên, bắt đầu từ -1.

A[: 3] = ? **A[-2 :] = ?**



2. Truy cập mảng numpy

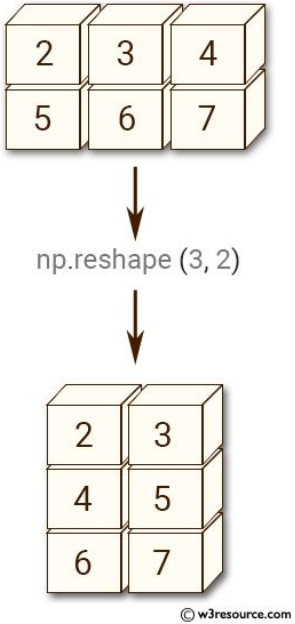
- Mỗi chiều ngăn cách nhau bởi dấu (,), với chiều đầu tiên là axis 0.

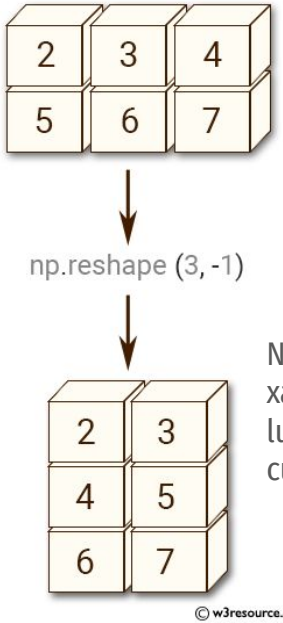


3. Thay đổi kích thước của mảng

Meta Mind
Explore Knowledge - Open Future

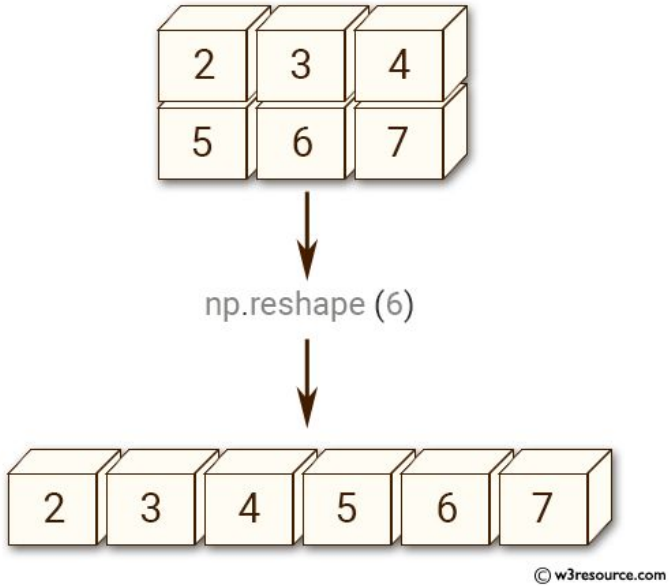
- Có thể thay đổi shape cho mảng bằng hàm *np.reshape()*.
- Hàm này là một thuộc tính sẵn có trong mỗi mảng, có thể sử dụng trực tiếp trên một mảng numpy. (A.reshape())
- Đảm bảo số lượng phần tử mảng sau khi đổi phải bằng mảng ban đầu.





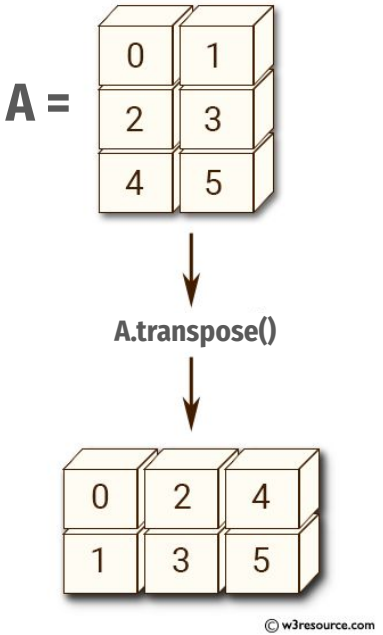
" -1 "

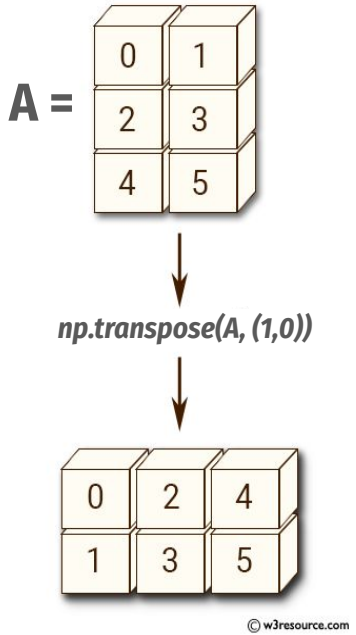
NumPy tự động xác định số lượng phần tử của chiều đó.



3. Chuyển vị các chiều np.transpose()

- Khi xử lý các dữ liệu nhiều chiều, có thể cần đổi vị trí các chiều để phù hợp với mục đích sử dụng (như làm việc giữa các framework khác nhau).
- Thay đổi vị trí các chiều bằng hàm *np.transpose()*.



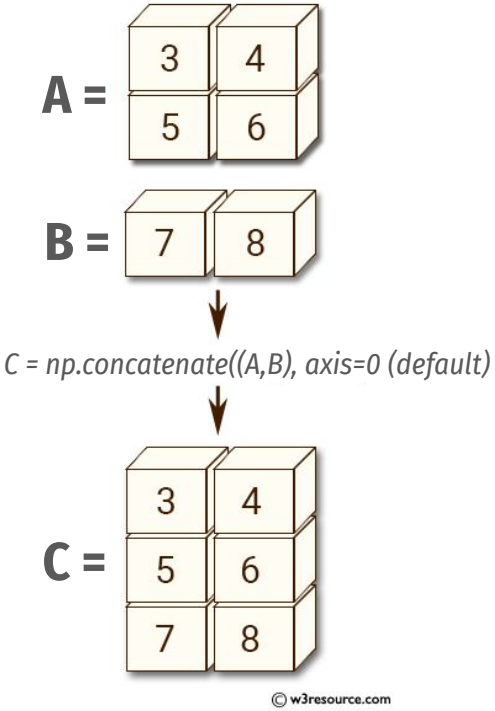


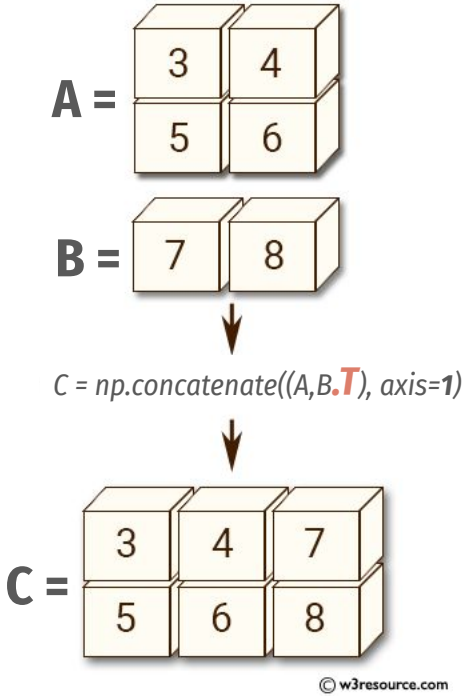
- Có thể thay đổi vị trí các chiều theo thứ tự mong muốn bằng cách thêm tuple chứa vị trí mới.

3. Nối mảng với concatenate() và stack()

Meta Mind
Explore Knowledge - Open Future

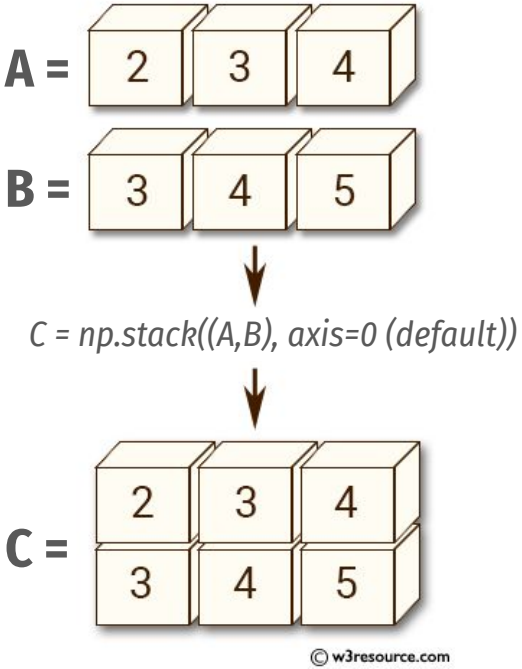
- Khi cần nối các mảng lại với nhau.
- Hàm concatenate() và stack() đều có chức năng giống nhau, nhưng stack() có thể mở rộng số chiều của mảng.

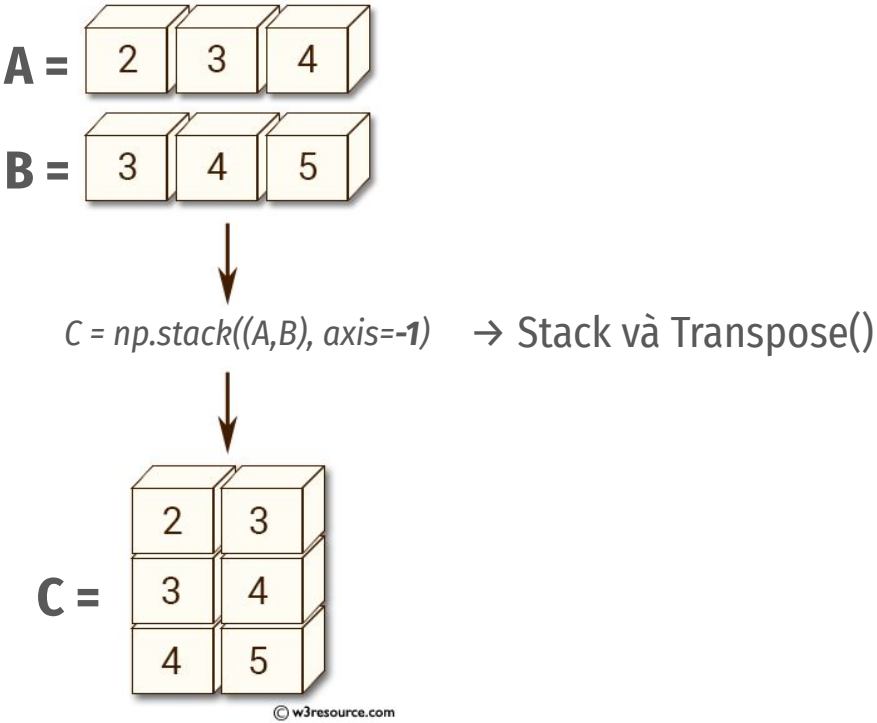




→ Số phần tử của chiều được nối của 2 mảng phải **bằng nhau.**

3. Nối mảng với concatenate() và stack()





Meta Mind
Explore Knowledge - Open Future

Các hàm tính toán cơ bản trong NumPy

Giới thiệu

- NumPy cung cấp một loạt các hàm toán học cho việc thực hiện các phép tính trên các mảng.
- Các loại hàm chính bao gồm:
  - Hàm Lượng Giác
  - Hàm Số Học
  - Hàm Làm Tròn

CÁC HÀM TÍNH TOÁN CƠ BẢN TRONG NUMPY

Meta Mind
Explore Knowledge - Open Future

Hàm Lượng Giác

- Cung cấp các hàm như sine, cosine, tangent, arcsine, arccosine, arctangent,...
- Tính toán các tỉ số lượng giác của một góc.

```
import numpy as np

# Tính sin của một góc
angle_degree = 45
angle_radian = np.radians(angle_degree)  # Chuyển đổi từ độ sang radians
sin_value = np.sin(angle_radian)
print("sin(45 độ) =", sin_value)

# Tính arcsin của một giá trị sin
sin_value = 0.5
angle_radian = np.arcsin(sin_value)  # arcsin trả về góc trong radians
angle_degree = np.degrees(angle_radian)  # Chuyển đổi từ radians sang độ
print("arcsin(0.5) =", angle_degree)
```

```
sin(45 độ) = 0.7071067811865475
arcsin(0.5) = 30.000000000000004
```

Hàm Lượng Giác	Tính toán (theo radians)
sin()	sine của một góc
cos()	cosine của một góc
tan()	tangent của một góc
arcsin()	arcsine (ngược lại của sin)
arccos()	arccosine (ngược lại của cosine)
arctan()	arctangent (ngược lại của tangent)
degrees()	chuyển đổi một góc từ radians sang độ
radians()	chuyển đổi một góc từ độ sang radians

CÁC HÀM TÍNH TOÁN CƠ BẢN TRONG NUMPY

Hàm Số Học

- Cung cấp các hàm như cộng, trừ, nhân, chia, lũy thừa, phần dư.
- Thực hiện các phép toán số học trên các mảng NumPy.

Phép Toán	Hàm Số Học	Toán Tử
Cộng	add()	+
Trừ	subtract()	-
Nhân	multiply()	*
Chia	divide()	/
Lũy Thừa	power()	**
Phần dư	mod()	%

```python
# Tạo hai mảng numpy
a1 = np.array([1, 2, 3, 4])
a2 = np.array([5, 6, 7, 8])

# Cộng hai mảng
sum = np.add(a1, a2)
print("Tổng của hai mảng:", sum)

# Nhân hai mảng
product = np.multiply(a1, a2)
print("Tích của hai mảng:", product)

# Lũy thừa của mảng 1 với mảng 2
power = np.power(a1, a2)
print("Lũy thừa của mảng 1 với mảng 2:", power)

# Phần dư của hai mảng
modulus = np.mod(a1, a2)
print("Phần dư của hai mảng:", modulus)
```

```
Tổng của hai mảng: [ 6  8 10 12]
Tích của hai mảng: [ 5 12 21 32]
Lũy thừa của mảng 1 với mảng 2: [    1    64  2187 65536]
Phần dư của hai mảng: [1 2 3 4]
```

CÁC HÀM TÍNH TOÁN CƠ BẢN TRONG NUMPY

Meta Mind
Explore Knowledge - Open Future

Hàm Làm Tròn

- Cung cấp các hàm làm tròn giá trị trong mảng đến số thập phân cụ thể.

Các Hàm Làm Tròn	Ý nghĩa
round()	làm tròn giá trị đến độ chính xác mong muốn
floor()	làm tròn các giá trị trong mảng xuống gần nhất thành số nguyên nhỏ hơn mỗi phần tử
ceil()	làm tròn các giá trị trong mảng lên gần nhất thành số nguyên lớn hơn mỗi phần tử

```
# Tạo một mảng numpy
a = np.array([1.234, 2.567, 3.891, 4.123])
```

```
# Làm tròn các giá trị trong mảng đến 1 chữ số thập phân
lam_tron = np.round(a, decimals=1)
print("Giá trị đã làm tròn:", lam_tron)
```

```
Giá trị đã làm tròn: [1.2 2.6 3.9 4.1]
```

```
# Làm tròn các giá trị trong mảng xuống gần nhất thành số nguyên nhỏ hơn
xuong = np.floor(a)
print("Giá trị đã làm tròn xuống:", xuong)
```

```
Giá trị đã làm tròn xuống: [1. 2. 3. 4.]
```

```
# Làm tròn các giá trị trong mảng lên gần nhất thành số nguyên lớn hơn
len = np.ceil(a)
print("Giá trị đã làm tròn lên:", len)
```

```
Giá trị đã làm tròn lên: [2. 3. 4. 5.]
```

Meta Mind
Explore Knowledge - Open Future

Các hàm thống kê cơ bản trong NumPy

Giới thiệu

- NumPy cung cấp các hàm mạnh mẽ để thực hiện phân tích thống kê trên các mảng.
- Có 6 hàm thống kê cơ bản:

Hàm	Mô Tả
median()	trả về giá trị trung vị của một mảng
mean()	trả về giá trị trung bình của một mảng
std()	trả về độ lệch chuẩn của một mảng
percentile()	trả về phân vị thứ n của các phần tử trong một mảng
min()	trả về phần tử nhỏ nhất của một mảng
max()	trả về phần tử lớn nhất của một mảng

CÁC HÀM THỐNG KÊ CƠ BẢN TRONG NUMPY

Meta Mind
Explore Knowledge - Open Future

Tính giá trị trung vị (median) của một mảng Numpy:

- Giá trị trung vị của một mảng NumPy là giá trị ở giữa sau khi mảng đã được sắp xếp
- Tính bằng hàm np.median()

Số lượng phần tử lẻ

```
arr_odd = np.array([2, 3, 5, 7, 11, 13, 17])
median_odd = np.median(arr_odd)
print("Giá trị trung vị của mảng là:", median_odd)
```

```
Giá trị trung vị của mảng là: 7.0
```

Số lượng phần tử chẵn

```
arr_even = np.array([2, 3, 5, 7, 11, 13, 17, 19])
median_even = np.median(arr_even)
print("Giá trị trung vị của mảng là:", median_even)
```

```
Giá trị trung vị của mảng là: 9.0
```

- Tính toán giá trị trung vị của mảng 2D: sử dụng tham số axis bên trong hàm np.median() để chỉ định trục theo đó tính toán giá trị trung vị.

  Nếu chỉ định:

  - axis = 0, giá trị trung vị được tính theo trục dọc
  - axis = 1, giá trị trung vị được tính theo trục ngang

  Nếu không sử dụng tham số axis, median sẽ được tính toán trên toàn bộ mảng.

➡ Colab

CÁC HÀM THỐNG KÊ CƠ BẢN TRONG NUMPY

Meta Mind
Explore Knowledge - Open Future

Tính giá trị trung bình (mean) của một mảng Numpy:

- Mean của một mảng là tổng của tất cả các phần tử trong mảng chia cho số lượng phần tử trong mảng.
- Được tính bằng hàm np.mean().

```
# Tạo một mảng numpy
so_nguyen_to = np.array([2, 3, 5, 7, 11, 13, 17])

# Tính toán giá trị trung bình của dãy số nguyên tố
trung_binh_so_nguyen_to = np.mean(so_nguyen_to)

print(trung_binh_so_nguyen_to)
```

```
8.285714285714286
```

- **Trung Bình của Mảng N chiều trong NumPy:** Đối số axis trong hàm np.mean() được sử dụng để chỉ định trục theo đó giá trị trung bình được tính toán. Ví dụ tạo một mảng 2D tên 'arr_2d' :
  - np.mean(arr_2d) - tính toán giá trị trung bình trên toàn bộ mảng.
  - np.mean(arr_2d, axis=0) - tính toán giá trị trung bình theo trục dọc.
  - np.mean(arr_2d, axis=1) - tính toán giá trị trung bình theo trục ngang.

➡ Colab

CÁC HÀM THỐNG KÊ CƠ BẢN TRONG NUMPY

Tính độ lệch chuẩn (standard deviation) của một mảng Numpy:

- Độ lệch chuẩn của một mảng NumPy là một chỉ số đo lường mức độ phân tán của dữ liệu trong mảng.
- Sử dụng hàm np.std().

```
# Tạo một mảng numpy
snt = np.array([2, 3, 5, 7, 11, 13, 17])

# Tính toán độ lệch chuẩn của snt
std_snt = np.std(snt)
print(std_snt)
```

```
5.146823867043378
```

- **Độ lệch chuẩn của Mảng 2D trong NumPy:** Tương tự như trung bình và trung vị, sử dụng tham số axis bên trong np.std() để chỉ định trục theo đó tính toán độ lệch chuẩn. ➡ Colab

CÁC HÀM THỐNG KÊ CƠ BẢN TRONG NUMPY

Meta Mind
Explore Knowledge - Open Future

Tính Phân Vị (Percentile) của một mảng Numpy:

- Là quá trình xác định giá trị mà một phần trăm nhất định của dữ liệu nằm dưới
- Hàm np.percentile() tính phân vị thứ n của một mảng đã cho.

```
# tạo một mảng
array1 = np.array([1, 3, 5, 7, 9, 11, 13, 15, 17, 19])

# tính toán phân vị thứ 25 của mảng
result1 = np.percentile(array1, 25)
print("Phân vị thứ 25:", result1)

# tính toán phân vị thứ 75 của mảng
result2 = np.percentile(array1, 75)
print("Phân vị thứ 75:", result2)
```

```
Phân vị thứ 25: 5.5
Phân vị thứ 75: 14.5
```

Tính Min và Max của một mảng Numpy:

- Hàm np.min() tìm giá trị tối thiểu.
- Hàm np.max() tìm giá trị tối đa.

```
# Tạo một mảng
arr = np.array([3, 8, 2, 6, 1, 9, 5])

# Tính giá trị tối thiểu của mảng
min_value = np.min(arr)
print("Giá trị tối thiểu:", min_value)

# Tính giá trị tối đa của mảng
max_value = np.max(arr)
print("Giá trị tối đa:", max_value)
```

```
Giá trị tối thiểu: 1
Giá trị tối đa: 9
```

Meta Mind
Explore Knowledge - Open Future

Final Project Module 1

FINAL PROJECT MODULE 1

Meta Mind
Explore Knowledge - Open Future

Đề bài: Viết một chương trình để tính toán điểm thi cho nhiều lớp với sĩ số hàng nghìn học sinh. Mục đích của chương trình giúp giảm thời gian chấm điểm. Áp dụng các functions (hàm) khác nhau trong Python để viết chương trình với các tác vụ sau:

- Mở các tập tin văn bản bên ngoài được yêu cầu với exception-handling
- Quét từng dòng của câu trả lời bài thi để tìm dữ liệu hợp lệ và cung cấp báo cáo tương ứng
- Chấm điểm từng bài thi dựa trên tiêu chí đánh giá (rubric) được cung cấp và báo cáo
- Tạo tập tin kết quả được đặt tên thích hợp

Tải Data: Tại đây

Chi tiết đề bài và yêu cầu: Tại đây

Hướng dẫn làm Final Project: Tại đây

Meta Mind
Explore Knowledge - Open Future

Tổng kết

TỔNG KẾT

Meta Mind
Explore Knowledge - Open Future

Hoàn thành Form điểm danh sau (20'): Form điểm danh

Bài tập sau buổi học:

- *Numpy 1D*
- *Numpy 2D*
- *Bài tập Numpy tổng hợp*
- *Final Project 1*

Yêu cầu:

- Đặt tên folder theo mẫu DSMMC_<Buổi học>_<Tên> (Ví dụ: *DSMMC_Buoi5_DangNH*)
- Hoàn thành các bài lab và Final Project lên git cá nhân , sau đó submit link folder git trên Form bài tập
- Hạn nộp: 1 tuần sau buổi học

THANK YOU !