LAB 3: Thư viện Numpy

Numpy (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python. 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 "core Python" đơn thuần.

a. Cài đặt thư viện Numpy

Gõ lệnh: `pip install numpy`

b. Các thao tác với Numpy

- ***Khai báo thư viện:***

```
import numpy as np
```

- ***Khởi tạo mảng:***

```
#Khởi tạo mảng một chiều với kiểu dữ liệu các phần tử là Integer
Arr1 = np.array([1,3,4,5,6], dtype = int)
# Khởi tạo mảng hai chiều
Arr2 = np.array([(4,5,6), (1,2,3)], dtype = int)
# Khởi tạo mảng ba chiều
Arr3 = np.array(([(2,4,0,6), (4,7,5,6)],
                 [(0,3,2,1), (9,4,5,6)],
                 [(5,8,6,4), (1,4,6,8)]), dtype = int)
```

- ***Khởi tạo với các hàm có sẵn***

- `np.zeros((3,4),dtype = int)`: Tạo mảng hai chiều các phần tử 0 với kích thước 3x4.
- `np.ones((2,3,4),dtype = int)`: Tạo mảng 3 chiều các phần tử 1 với kích thước 2x3x4.
- `np.arange(1,7,2)`: Tạo mảng với các phần tử từ 1 - 6 với bước nhảy là 2.
- `np.full((2,3),5)`: Tạo mảng 2 chiều các phần tử 5 với kích thước 2x3.
- `np.eye(4,dtype=int)`: Tạo ma trận đơn vị với kích thước là 4x4.
- `np.random.random((2,3))`: Tạo ma trận các phần tử ngẫu nhiên với kích thước 2x3.

c. Thao tác với mảng

- **dtype**: Kiểu dữ liệu của phần tử trong mảng.
- **shape**: Kích thước của mảng.
- **size**: Số phần tử trong mảng.
- **ndim**: Số chiều của mảng.

Ví dụ:

```
print("Kiểu dữ liệu của phần tử trong mảng:", arr2.dtype)
print("Kích thước của mảng:", arr2.shape)
print("Số phần tử trong mảng:", arr2.size)
print("Số chiều của mảng:", arr2.ndim)
```

- ***Truy cập phần tử trong mảng***

Các phần tử trong mảng được đánh số từ 0 trở đi

arr[i]:Truy cập tới phần tử thứ i của mảng 1 chiều.
arr1[i,j]:Truy cập tới phần tử hàng i, cột j của mảng 2 chiều.
arr2[n,i,j]:Truy cập tới phần tử chiều n, hàng i, cột j của mảng 3 chiều.
arr[a:b]:Truy cập tới các phần tử từ a đến b-1 trong mảng 1 chiều.
arr1[:,:i]:Truy cập tới phần tử từ cột 0 đến cột i-1, của tất cả các hàng trong mảng 2 chiều.

Ví dụ:

```
print("arr[2]=", arr[2])
print("arr1[1:2]=", arr1[1,2])
print("arr2[1,2,3]=", arr2[1,1,3])
print("arr[0:3]=", arr[0:3])
print("arr1[:,:1]=", arr1[:,:2])
```

- ***Các hàm thống kê***

arr.max() hoặc np.max(arr): *Lấy giá trị lớn nhất của mảng arr.*
arr.min() hoặc np.min(arr): *Lấy giá trị nhỏ nhất của mảng arr.*
arr.sum() hoặc np.sum(arr): *Tổng tất cả các phần tử trong mảng arr.*
arr.mean() hoặc np.mean(arr): *Trung bình cộng của tất cả các phần tử trong mảng arr.*
np.median(arr): *Trả về giá trị trung vị của mảng arr.*

THỰC HÀNH

1. Tạo một ma trận 4x4 toàn các giá trị False

```python
import numpy as np
a = np.full((4, 4), False, dtype = bool)
print(a)
```

2. Cho một dãy số nguyên 100 phần tử, hãy tách lấy tất cả những phần tử lẻ cho vào một mảng

3. Cho một dãy số tự nhiên 20 phần tử, hãy thay thế tất cả những phần tử lẻ bằng số -1

4. Hai mảng a và b có cùng số dòng, hãy ghép chúng theo các dòng thành mảng c, các cột của a rồi đến các cột của b

5. Mảng a và b có cùng số cột, hãy ghép chúng theo các cột thành mảng c, các dòng của a rồi đến của b

6. Cho một mảng a, hãy in ra tất cả những phần tử trong khoảng từ 5 đến 10

7. Tạo ra một mảng số thực có 1000 phần tử, các phần tử nằm trong khoảng từ -0.5 đến <0.5

8. Tạo ra một ma trận 3x5 gồm các số ngẫu nhiên từ 0 đến nhỏ hơn 10, tính và in ra số lớn nhất trên mỗi dòng của ma trận

9. Nhập mảng a và b có 10 phần tử, tính khoảng cách euclid giữa a và b
Hướng dẫn:
dist = sqrt((a1-b1)² + (a2-b2)² + (a3-b3)²...)
