Lab 2. Làm Quen Với Các Kiểu Dữ Liệu:

List, Tuple, Dictionary, Dataframe

1. List:

```
# Khởi tạo một danh sách
my_list = [1, 2, 3, 4, 5]
# Truy cập phần tử trong danh sách
print(my_list[0]) # Output: 1
# Thay đổi giá trị của phần tử
my_list[1] = 10
# Thêm phần tử mới vào danh sách
my_list.append(6)
# Xóa phần tử khỏi danh sách
my_list.remove(3)
# In ra tất cả các phần tử trong danh sách print(my_list)
```

2. Tuple:

```
# Khởi tạo một tuple
my_tuple = (1, 2, 3, 4, 5)
# Truy cập phần tử trong tuple
print(my_tuple[0])
# Output: 1
# Tuple là không thể thay đổi, nhưng có thể nối các tuple
new_tuple = my_tuple + (6, 7)
# In ra tất cả các phần tử trong tuple mới
print(new_tuple)
```

3. numpy array

là một cấu trúc dữ liệu quan trọng trong thư viện NumPy của Python. Nó mang lại hiệu suất cao và khả năng thực hiện các phép toán số học trên mảng một cách linh hoạt. Dưới đây là một số điểm quan trọng:

1. **Hiệu Suất:** Numpy array cung cấp hiệu suất cao hơn so với list trong Python, đặc biệt là khi thực hiện các phép toán số học trên toàn bộ mảng.
2. **Đa Chiều:** Numpy array có thể là mảng đa chiều, cho phép làm việc dễ dàng với ma trận và tensor.
3. **Phép Toán:** Hỗ trợ nhiều phép toán số học và thống kê trực tiếp trên mảng một cách thuận tiện và hiệu quả.
4. **Bộ Nhớ:** Sử dụng ít bộ nhớ hơn so với list, làm cho nó lựa chọn tốt cho xử lý dữ liệu lớn.
5. **Hỗ Trợ Hàm và Phương Thức:** Cung cấp nhiều hàm và phương thức hỗ trợ cho xử lý mảng, từ thống kê đến đại số tuyến tính.

Dưới đây là một ví dụ đơn giản so sánh sử dụng numpy array và list để cộng hai mảng:

```
# Sử dụng numpy array
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
result_np = arr1 + arr2
# Sử dụng list
list1 = [1, 2, 3]
list2 = [4, 5, 6]
result_list = [a + b for a, b in zip(list1, list2)]
print(result_np)
print(result_list)
# Tạo một numpy array
arr = np.array([1, 2, 3, 4, 5])
# Tính tổng các phần tử trong mảng
total_sum = np.sum(arr)
# Tính giá trị trung bình
average = np.mean(arr)
# Tìm giá trị lớn nhất
max_value = np.max(arr)
# Tìm giá trị nhỏ nhất
min_value = np.min(arr)
```

4. Dictionary:

```
# Khởi tạo một từ điển
 my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
# Truy cập giá trị thông qua key print(my_dict['name'])
# Output: John # Thay đổi giá trị của một key
 my_dict['age'] = 26
# Thêm một cặp key-value mới my_dict['gender'] = 'Male'
# Xóa một cặp key-value
del my_dict['city']
# In ra tất cả các cặp key-value trong từ điển
print(my_dict)
```

5. DataFrame (sử dụng thư viện pandas):

```
import pandas as pd
# Khởi tạo một DataFrame từ một dictionary
data = {'Name': ['John', 'Alice', 'Bob'], 'Age': [25, 30, 22], 'City': ['New York',
'San Francisco', 'Los Angeles']}
```

```
df = pd.DataFrame(data)
# In ra DataFrame print(df)
# Truy cập cột theo tên print(df['Name'])
# Thêm một cột mới
df['Occupation'] = ['Engineer', 'Doctor', 'Artist']
# Xóa một cột df.drop('City', axis=1, inplace=True)
# In ra DataFrame sau khi chỉnh sửa
print(df)
```

1. **Thêm cột:**
   - Sử dụng cú pháp `df['new_column'] = values` để thêm một cột mới vào DataFrame. Ví dụ:

     ```
     df['NewColumn'] = df['OldColumn'] * 2
     ```
2. **Xoá cột:**
   - Sử dụng `df.drop('column_name', axis=1, inplace=True)` để xoá một cột. Đối số `axis=1` cho biết bạn muốn xoá theo cột, và `inplace=True` sẽ thay đổi DataFrame gốc. Ví dụ:

     pythonCopy code
     ```
     df.drop('OldColumn', axis=1, inplace=True)
     ```
3. **Chèn cột:**
   - Sử dụng `df.insert(index, 'column_name', values)` để chèn một cột vào vị trí cụ thể trong DataFrame. Ví dụ:

     pythonCopy code
     ```
     df.insert(1, 'NewColumn', [1, 2, 3, 4])
     ```
4. **Ghép các cột theo điều kiện:**
   - Sử dụng `pd.concat([df1, df2], axis=1, join='inner')` để ghép các DataFrame theo cột và điều kiện nhất định. Ví dụ:

     pythonCopy code
     ```
     result = pd.concat([df1, df2], axis=1, join='inner')
     ```
5. **Lọc dữ liệu theo điều kiện trên cột:**
   - Đã thảo luận trong câu trước, sử dụng điều kiện để lọc dữ liệu. Ví dụ:

     pythonCopy code
     ```
     condition = df['Age'] > 25 filtered_df = df[condition]
     ```
6. **Thêm dòng:**
   - Sử dụng `df.loc[index] = values` để thêm một dòng mới vào DataFrame. Ví dụ:

     pythonCopy code
     ```
     df.loc[len(df)] = ['NewPerson', 28, 55000]
     ```
7. **Bớt dòng:**
   - Sử dụng `df.drop(index, inplace=True)` để xoá một dòng khỏi DataFrame. Ví dụ:

     pythonCopy code
     ```
     df.drop(1, inplace=True) # Xoá dòng có index=1
     ```
8. **Chèn dòng:**

- Sử dụng `df.loc[index] = values` hoặc `df.loc[len(df)] = values` để chèn một dòng mới vào DataFrame. Ví dụ:

```
pythonCopy code
df.loc[1] = ['NewPerson', 28, 55000] # Chèn vào index
```