Câu 1.6: Trình bày các đặc trưng, mục đích của các thư viện numpy, pandas, matplotlib, scikitLearn. Cho các ví dụ minh họa.

* Numpy

Numpy là một thư viện lõi phục vụ cho khoa học máy tính của Python, hỗ trợ cho việc tính toán các mảng nhiều chiều, có kích thước lớn với các hàm đã được tối ưu áp dụng lên các mảng nhiều chiều đó. Numpy đặc biệt hữu ích khi thực hiện các hàm liên quan tới Đại Số Tuyến Tính.

Để cài đặt numpy nếu bạn có Anaconda chỉ cần gõ conda install numpy hoặc sử dụng tools pip pip install numpy.

Sau khi cài đặt xong, trong Python, chúng ta cần khai báo import numpy để có thể bắt đầu sử dụng các hàm của numpy. Vì numpy là thư viện được sử dụng thường xuyên nên nó thường được khai báo gọn lại thành **np** import numpy as np, **np** có thể thay thế bằng các từ khác, tuy nhiên bạn nên đặt là **np** vì các tài liệu hướng dẫn đều ngầm quy ước như thế.

* Array

Một mảng numpy là một lưới các giá trị, và tất cả các giá trị có cùng kiểu giá trị, và được lập chỉ mục bởi một số nguyên không âm, số chiều được gọi là rank của mảng numpy, shape là một tuple các số nguyên đưa ra kích thước của mảng theo mỗi chiều

Có thể khởi tạo numpy arrays từ nested Python lists, và dùng dấu ngoặc vuông để truy cập từng phần tử

|  |
| --- |
| import numpy as np  a = np.array([1, 2, 3]) # Tạo một numpy array với rank = 1  print(type(a))            # Sẽ in ra "<class 'numpy.ndarray'>"  print(a.shape)            # Sẽ in ra "(3,)"  print(a[0], a[1], a[2])   # Sẽ in ra "1 2 3"  a[0] = 5                  # Thay đổi giá trị của 1 phần tử trong mảng  print(a)                  # Sẽ in ra kết quả là "[5, 2, 3]"  b = np.array([[1,2,3],[4,5,6]])    # Tạo một numpy array với rank =2  print(b.shape)                     # In ra "(2, 3)"  print(b[0, 0], b[0, 1], b[1, 0])   # Sẽ in ra "1 2 4" |

Numpy cũng cung cấp rất nhiều hàm để khởi tạo arrays

|  |
| --- |
| import numpy as np  a = np.zeros((2,2))   # Tạo một numpy array với tất cả phẩn tử là 0  print(a)              # "[[ 0.  0.]                        #   [ 0.  0.]]"  b = np.ones((1,2))    # Tạo một numpy array với tất cả phẩn tử là 1  print(b)              # "[[ 1.  1.]]"  c = np.full((2,2), 7)  # Tạo một mảng hằng  print(c)               # "[[ 7.  7.]                         #   [ 7.  7.]]"  d = np.eye(2)         # Tạo một ma trận đơn vị 2 x 2  print(d)              # "[[ 1.  0.]                        #   [ 0.  1.]]"  e = np.random.random((2,2))  # Tạo một mảng với các giá trị ngẫu nhiên  print(e)                     # Có thể là "[[ 0.91940167  0.08143941]                               #             [ 0.68744134  0.87236687]]"  f = np.arange(10) # Tạo 1 numpy array với các phẩn tử từ 0 đến 9  print(f)          # "[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]" |

* Array indexing

Numpy cung cấp một số cách để truy xuất phần tử trong mảng như:

* Slicing: tương tự như list trong python, numpy arrays cũng có thể được cắt.

|  |
| --- |
| import numpy as np    # Khởi tạo numpy array có shape = (3, 4) có giá trị như sau:  # [[ 1 2 3 4]  # [ 5 6 7 8]  # [ 9 10 11 12]]  a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])    # Sử dụng slicing để tạo numpy array b bằng cách lấy 2 hàng đầu tiên  # và cột 1, 2. Như vậy b sẽ có shape = (2, 2):  # [[2 3]  # [6 7]]  b = a[:2, 1:3]    # Một array tạo ra từ slicing sẽ có cùng địa chỉ với array gốc.  # Nếu thay đổi 1 trong 2 thì array còn lại cũng thay đổi.  print(a[0, 1]) # Prints "2"  b[0, 0] = 77 # b[0, 0] ở đây tương đương với a[0, 1]  print(a[0, 1]) # Prints "77" |

Có thể kết hợp việc dùng slicing và dùng chỉ số. Tuy nhiên, các làm đó sẽ cho ra một mảng mới có rank thấp hơn mảng gốc.

|  |
| --- |
| import numpy as np    # Tạo một numpy array có shape (3, 4) với giá trị như sau:  # [[ 1 2 3 4]  # [ 5 6 7 8]  # [ 9 10 11 12]]  a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])    # Hai cách truy cập dữ liệu ở hàng giữa của mảng  # Dùng kết hợp chỉ số và slice -> được array mới có rank thấp hơn,  # Nếu chỉ dùng slice ta sẽ có 1 array mới có cùng rank  # với array gốc  row\_r1 = a[1, :] # Rank 1, hàng thứ 2 của a  row\_r2 = a[1:2, :] # Rank 2, vẫn là hàng thứ 2 của a  print(row\_r1, row\_r1.shape) # Prints "[5 6 7 8] (4,)"  print(row\_r2, row\_r2.shape) # Prints "[[5 6 7 8]] (1, 4)"    # Chúng ta có thể làm tương tự với cột của numpy array:  col\_r1 = a[:, 1]  col\_r2 = a[:, 1:2]  print(col\_r1, col\_r1.shape) # Prints "[ 2 6 10] (3,)"  print(col\_r2, col\_r2.shape) # Prints "[[ 2]  # [ 6]  # [10]] (3, 1)" |

* Integer array indexing: khi bạn truy xuất mảng dùng slicing, kết quả trả về sẽ là mảng con của mảng ban đầu, tuy nhiên sử dụng chỉ số cho phép bạn xây dựng mảng tùy ý một mảng khác.

|  |
| --- |
| import numpy as np    a = np.array([[1,2], [3, 4], [5, 6]])    # Truy xuất mảng dùng chỉ số.  # Kết quả thu được là 1 mảng có shape (3,)  print(a[[0, 1, 2], [0, 1, 0]]) # Prints "[1 4 5]"    # Sẽ thu được kết quả tương đương như trên với cách này:  print(np.array([a[0, 0], a[1, 1], a[2, 0]])) # Prints "[1 4 5]"    # Bạn được phép sử dụng chỉ số mảng để  # truy xuất tới 1 phần tử  # của mảng gốc nhiều hơn 1 lần  print(a[[0, 0], [1, 1]]) # Prints "[2 2]"    # Một cách làm khác tương đương:  print(np.array([a[0, 1], a[0, 1]])) # Prints "[2 2]" |

Một mẹo hữu ích khác dùng chỉ số mảng để chọn và thay đổi phần tử từ mỗi hàng của ma trận

* Pandas
* Matplotlib
* Scikit learn