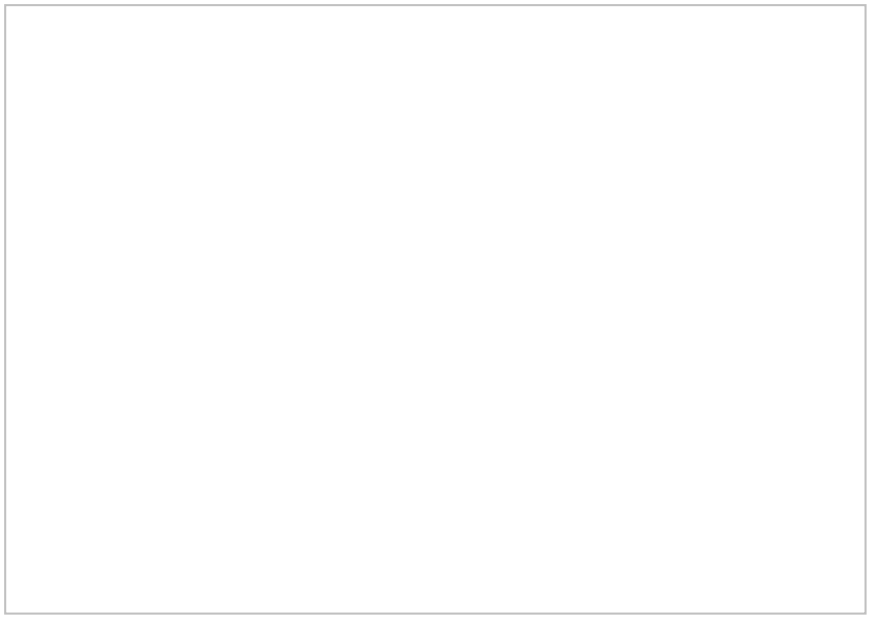
22. Giới thiệu Numpy

NumPy là một từ viết tắt của "Numeric Python" hoặc "Numerical Python". Nó là một mô-đun mở rộng mã nguồn mở cho Python, cung cấp các chức năng biên dịch nhanh cho các thao tác toán học và số. Hơn nữa, NumPy làm phong phú ngôn ngữ lập trình Python với các cấu trúc dữ liệu mạnh mẽ để tính toán hiệu quả các mảng và ma trận đa chiều. Việc thực hiện thậm chí là nhằm vào ma trận và mảng khổng lồ. Bên cạnh đó các mô-đun cung cấp một thư viện lớn các chức năng toán học cấp cao để hoạt động trên các ma trận và mảng.

SciPy (Python Khoa học) thường được đề cập trong cùng với NumPy. SciPy mở rộng khả năng của NumPy với các chức năng hữu ích khác cho tối ưu, hồi quy, biến đổi Fourier và nhiều kĩ thuật khác.

Cả NumPy và SciPy thường không được cài đặt mặc định. NumPy phải được cài đặt trước khi cài đặt SciPy. Numpy có thể được tải xuống từ trang web: Http://www.numpy.org, bạn có thể xem lại phần cài đặt, tôi đã hướng dẫn cài đặt cho một gói bất kì.

(Chú thích: Sơ đồ hình ảnh phía bên phải là đồ thị trực quan của ma trận với 14 hàng và 20 cột. Màu đỏ xác định giá trị âm và màu xanh lá cây biểu thị giá trị dương).



NumPy dựa trên hai mô đun Python trước đó về xử lý các mảng. Một trong số đó là Numeric. Numeric cũng giống NumPy như một môđun Python cho hiệu suất cao, tính toán số học, nhưng nó là lỗi thời ngày nay. Một bản tiền nhiệm khác của NumPy là Numarray, là một bản viết lại hoàn chỉnh của Numeric nhưng cũng không được ủng hộ. NumPy là sự sáp nhập của cả hai, nghĩa là nó được xây dựng trên mã của Numeric và các tính năng của Numarray.

Khi chúng tôi nói "Core Python", chúng tôi muốn nói đến Python mà không có bất kỳ mô đun đặc biệt nào, đặc biệt là không có NumPy.

Những lợi thế của Core Python:

Số: Integer, Floating points

Container: list, dictionary đã được tối ưu BigO cho các thao tác tìm kiếm, mở rộng, xóa.

Ưu điểm của việc sử dụng Numpy với Python:

Tính toán theo mảng

Triển khai hiệu quả các mảng đa chiều

Được thiết kế để tính toán khoa học

Ví dụ đơn giản về numpy.

```
import numpy as np
numbers = [1.23,4,56,6676767.1144]
c = np.array(numbers)
print "c = ",c
print "nhan c voi mot so"
print "c*45 = ", c*45
print "De co ket qua tuong tu voi list ta phai lam nhu sau"
print "[i * 45 for i in numbers] = ", [i * 45 for i in numbers]

Output:

c =  [  1.23000000e+00   4.00000000e+00   5.60000000e+01   6.67676711e+06]

nhan c voi mot so

c*45 =  [  5.53500000e+01   1.80000000e+02   2.52000000e+03   3.00454520e+08]

De co ket qua tuong tu voi list ta phai lam nhu sau

[i * 45 for i in numbers] =  [55.35, 180, 2520, 300454520.148]
```

So sánh running time giữa Python List và Numpy Arrays

```
import time
import numpy as np
def pure_python_version():
    t1 = time.time()
    X = range(size_of_vec)
    Y = range(size_of_vec)
    Z = []
    for i in range(len(X)):
        Z.append(X[i] + Y[i])
    return time.time() - t1

def numpy_version():
    t1 = time.clock()
    X = np.arange(size_of_vec)
    Y = np.arange(size_of_vec)
    Z = X + Y
    return time.clock() - t1

size_of_vec = 1000000
t1 = pure_python_version()
t2 = numpy_version()
print(t1, t2)
print("Numpy is in this example " + str(t1/t2) + " faster!")

Output:

(0.17199993133544922, 0.004248119882314093)

Numpy is in this example 40.4884833998 faster!
```

Từ ví dụ trên đã chứng minh tính vượt trội của numpy khi thực hiện các phép toán với mảng và vector.

Kết luận

Numpy là một thư viện toán học phổ biến và mạnh mẽ của Python. Nó 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. Ngoài ra, Python cũng hỗ trợ một thư viện khác để mở rộng thêm các tính năng của Numpy là Scipy với ưu thế về các phép hồi quy hay biến đổi Fourier… **Trong bài tiếp theo chúng tôi sẽ giới thiệu với các bạn chi tiết hơn về một số hàm quan trọng được hỗ trợ trong thư viện Numpy**.