

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙠🟅🙢



**ĐỒ ÁN HỌC PHẦN**

**Nhập môn lập trình Python cho phân tích**

**TÌM HIỂU THƯ VIỆN NUMPY**

**VIẾT DEMO MINH HỌA**

**Giảng viên hướng dẫn: Nguyễn Thanh Sơn**

**Sinh viên thực hiện:**

**Nguyễn Thái Hải 19110356**

**Trần Kiện Khang 19110375**

**Nguyễn Minh Luân 19110395**

**Lê Thị Minh Nguyệt 19110413**

**Nguyễn Văn Phú 19110430**

**Trần Anh Tiến 19110471**

*Thành phố Hồ Chí Minh – Tháng 5/2021*

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

**Giảng viên giảng dạy: Nguyễn Thanh Sơn**

**TÊN HỌC PHẦN: Nhập môn lập trình Python cho phân tích**

**MÃ SỐ LỚP HP: IPPA233277\_02**

**Tên đề tài: Tìm hiểu thư viện Numpy. Viết demo minh họa.**

**Ngôn ngữ lập trình sử dụng: Python**

**Tên đăng nhập và Mật khẩu (password) (nếu có):**

**Tập dữ liệu thực nghiệm (nếu có):**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá:………….(…………………)**

**Ngày……./……../2021**

**Giảng viên: Nguyễn Thanh Sơn**

**Ký tên**

**TP.HCM, tháng 5 năm 2021**

# **LỜI CẢM ƠN**

Lời đầu tiên, nhóm chúng em xin gửi lời cám ơn sâu sắc nhất đến thầy Nguyễn Thanh Sơn.

Trong quá trình tìm hiểu và học tập môn Nhập môn lập trình Python cho phân tích chúng em đã nhận được sự giảng dạy và hướng dẫn tận tình, tâm huyết từ thầy, giúp chúng em tích lũy thêm rất nhiều kiến thức. Từ những kiến thức thầy đã truyền đạt chúng em xin trình bày lại những gì mình đã tìm hiểu về Thư viện Numpy trong python.

Tuy nhiên, kiến thức về môn học của chúng em vẫn còn những hạn chế nhất định và không tránh khỏi những thiếu sót trong quá trình hoàn thành bài báo cáo này. Chúng em mong nhận được lời góp ý, nhận xét đến từ cô và các bạn để đề tài báo cáo của chúng em được hoàn thiện hơn.

Chúc em kính chúc thầy sức khỏe, hạnh phúc và thành công hơn nữa trong sự nghiệp trồng người để tiếp tục dìu dắt nhiều thế hệ sinh viên đến những bến bờ tri thức.

Chúng em xin chân thành cảm ơn thầy!

**MỤC LỤC**

Tìm hiểu thư viện Numpy

[**LỜI CẢM ƠN** 3](#_Toc72244048)

[**I.** **GIỚI THIỆU THƯ VIỆN NUMPY** 5](#_Toc72244049)

[**II.** **MỘT SỐ HÀM CƠ BẢN CỦA NUMPY** 5](#_Toc72244050)

[**TÍNH NĂNG SẮP XẾP** 5](#_Toc72244051)

[**1.** **Numpy.sort()** 5](#_Toc72244052)

[**2.** **numpy.argsort() :** 7](#_Toc72244053)

[**3.** **numpy.lexsort()** 8](#_Toc72244054)

[**BROADCAST – NUMPY** 8](#_Toc72244055)

[**III.** **HÀM CƠ BẢN TẠO MẢNG VÀ MA TRẬN** 10](#_Toc72244056)

[**1.** **Tạo ma trận ngẫu nhiên** 10](#_Toc72244057)

[**2.** **Tạo mảng theo số phần tử** 11](#_Toc72244058)

[**3.** **Tạo ma trận cùng giá trị** 12](#_Toc72244059)

[**4.** **Tạo mảng theo khoảng cách** 12](#_Toc72244060)

[**5.** **Thông số ma trận** 13](#_Toc72244061)

[**6.** **Ma trận với kiểu dáng copy** 13](#_Toc72244062)

[**7.** **Ma trận đơn vị** 13](#_Toc72244063)

[**8.** **Sao chép một ma trận** 14](#_Toc72244064)

[**IV.** **CÁC PHÉP TOÁN TRONG MA TRẬN** 14](#_Toc72244065)

[**CÁC PHÉP TOÁN CƠ BẢN** 14](#_Toc72244066)

[**1.** **Cộng trận với một số** 14](#_Toc72244067)

[**2.** **Trừ ma trận với một số** 15](#_Toc72244068)

[**3.** **Nhân ma trận với một số** 15](#_Toc72244069)

[**4.** **Chia ma trận với một số** 15](#_Toc72244070)

[**5.** **Chia ma trận lấy nguyên** 15](#_Toc72244071)

[**6.** **Lũy thừa ma trận** 15](#_Toc72244072)

[**PHÉP TOÁN LIÊN QUAN ĐẾN ĐẠI SỐ TUYẾN TÍNH** 15](#_Toc72244073)

[**1.** **Nhân hai ma trận** 15](#_Toc72244074)

[**2.** **Tạo ma trận đường chéo** 16](#_Toc72244075)

[**3.** **Lấy vector đường chéo** 16](#_Toc72244076)

[**4.** **Tính định thức** 16](#_Toc72244077)

[**5.** **Reorganizing Arrays** 17](#_Toc72244078)

[**6.** **Lấy rank của ma trận (số hàng khác 0 khi biến đổi về ma trận bậc thang** 17](#_Toc72244079)

[**7.** **Tìm số nghiệm của hệ phương trình tuyến tính** 17](#_Toc72244080)

[**V.** **CODE DEMO MINH HỌA CÁI MÁY TÍNH** 18](#_Toc72244081)

# **GIỚI THIỆU 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.

**Cài đặt thư viện Numpy**

Mở Command Prompt và gõ lệnh: pip install numpy

**Khai báo thư viện khi sử dụng**

import numpy as np

# **MỘT SỐ HÀM CƠ BẢN CỦA NUMPY**

## **TÍNH NĂNG SẮP XẾP**

### **Numpy.sort()**

Hàm sort () trả về một bản sao đã được sắp xếp của mảng đầu vào

numpy.sort(a, axis, kind, order)

Trong đó :

1. a : Mảng được sắp xếp
2. axis : Trục dọc theo mảng sẽ được sắp xếp. Nếu không có, mảng sẽ được làm phẳng, sắp xếp trên trục cuối cùng
3. kind : Mặc định là quicksort
4. order : Nếu mảng chứa các trường, thứ tự các trường sẽ được sắp xếp

**Ví dụ :**

import numpy as np

a = np.array([[1,5],[3,2]])

print ('Array a: ')

print (a)

print ('\n')

print ('Applying sort() function:')

print (np.sort(a))

#print ('\n')

print ('Sort along axis 0:' )

print (np.sort(a, axis = 0))

print ('\n')

# Order parameter in sort function

dt = np.dtype([('name', 'S10'),('age', int)])

a = np.array([("Phu",21),("Nguyet",25),("Tien", 17), ("Luan",27)], dtype = dt)

print ('Array a: ')

print (a)

print ('\n')

print ('Order by name:')

print (np.sort(a, order = 'name'))

print ('Order by age:')

print (np.sort(a, order = 'age'))

Kết quả :

Array a:

[[1 5]

 [3 2]]

Applying sort() function:

[[1 5]

 [2 3]]

Sort along axis 0:

[[1 2]

 [3 5]]

Array a:

[(b'Phu', 21) (b'Nguyet', 25) (b'Tien', 17) (b'Luan', 27)]

Order by name:

[(b'Luan', 27) (b'Nguyet', 25) (b'Phu', 21) (b'Tien', 17)]

Order by age:

[(b'Tien', 17) (b'Phu', 21) (b'Nguyet', 25) (b'Luan', 27)]

### **numpy.argsort() :**

Hàm numpy.argsort () thực hiện sắp xếp gián tiếp trên mảng đầu vào, dọc theo trục đã cho và sử dụng kiểu sắp xếp cụ thể để trả về mảng chỉ số dữ liệu. Mảng này được sử dụng để xây dựng mảng đã sắp xếp

**Ví dụ :**

import numpy as np

x = np.array([3, 1, 2, 4, 5])

print ('Array x: ')

print (x)

print ('\n')

print ('Applying argsort() to x:')

y = (np.argsort(x) )

print (y)

print ('\n')

print ('Reconstruct original array in sorted order:')

print (x[y])

print ('\n'**)**

**Kết quả :**

Array x:

[3 1 2 4 5]

Applying argsort() to x:

[1 2 0 3 4]

Reconstruct original array in sorted order:

[1 2 3 4 5]

### **numpy.lexsort()**

Hàm thực hiện sắp xếp gián tiếp bằng cách sử dụng một chuỗi khóa. Các khóa có thể được xem như một cột trong bảng tính. Hàm trả về một mảng chỉ số, sử dụng dữ liệu đã sắp xếp để lấy dữ liệu. Lưu ý rằng khóa cuối cùng sẽ là khóa chính

**Ví dụ :**

import numpy as np

name = ('Luan','Khang','Tien','Nguyet')

age = ('17', '21', '11.', '5')

ind = np.lexsort((age,name))

print ('Applying lexsort() function:')

print (ind)

print ('Use this index to get sorted data:')

print ([name[i] + ", " + age[i] for i in ind])

Kết quả :

Applying lexsort() function:

[1 0 3 2]

Use this index to get sorted data:

['Khang, 21', 'Luan, 17', 'Nguyet, 5', 'Tien, 11.']

## **BROADCAST – NUMPY**

Thuật ngữ broadcast đề cập đến khả năng của NumPy để xử lý các mảng có hình dạng khác nhau trong các phép toán số học. Các phép toán số học trên mảng thường được thực hiện trên các phần tử tương ứng. Nếu hai mảng có hình dạng hoàn toàn giống nhau thì các thao tác này được thực hiện trơn tru.

**Ví dụ:**

import numpy as np

a = np.array([1,2,3,4])

b = np.array([10,20,30,40])

c = a \* b

print c

**Kết quả :**

[10   40   90   160]

Nếu kích thước của hai mảng **không giống nhau**, thì không thể thực hiện các thao tác giữa phần tử với phần tử. Tuy nhiên, các thao tác trên các mảng có hình dạng không giống nhau vẫn có thể thực hiện được trong NumPy, vì khả năng broadcasting. Mảng nhỏ hơn được phát tới kích thước của mảng lớn hơn để chúng có hình dạng tương thích.

**Có thể broadcasting nếu các quy tắc sau được thỏa mãn:**

* Mảng có ndim nhỏ hơn mảng kia được thêm chữ '1' trong shape.
* Kích thước trong mỗi kích thước của shape đầu ra là tối đa của kích thước đầu vào trong kích thước đó.
* Một đầu vào có thể được sử dụng trong tính toán, nếu kích thước của nó trong một thứ nguyên cụ thể khớp với kích thước đầu ra hoặc giá trị của nó chính xác là 1.
* Nếu đầu vào có kích thước thứ nguyên là 1, thì mục nhập dữ liệu đầu tiên trong thứ nguyên đó sẽ được sử dụng cho tất cả các phép tính dọc theo thứ nguyên đó.

**Một tập hợp các mảng được cho là có thể broacast nếu các quy tắc trên tạo ra kết quả hợp lệ và một trong những điều sau là đúng:**

* Các mảng có hình dạng giống hệt nhau.
* Mảng có cùng số kích thước và độ dài của mỗi thứ nguyên là độ dài chung hoặc 1.
* Mảng có quá ít kích thước có thể có hình dạng của nó được thêm vào trước với kích thước có độ dài 1, do đó thuộc tính đã nêu ở trên là đúng.

**Ví dụ:**

import numpy as np

#Mang a size 4x3

a = np.array([[0.0,0.0,0.0],[10.0,10.0,10.0],[20.0,20.0,20.0],[30.0,30.0,30.0]])

#Mang b

b = np.array([1.0,2.0,3.0])

print (a+b)

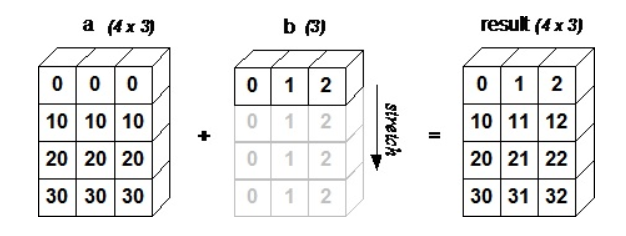
**Kết quả:**

[[ 1.  2.  3.]

 [11. 12. 13.]

 [21. 22. 23.]

 [31. 32. 33.]]



# **HÀM CƠ BẢN TẠO MẢNG VÀ MA TRẬN**

## **Tạo ma trận ngẫu nhiên**

|  |
| --- |
| def TaoMaTranNgauNhien():  D=int(input("Nhap so dong cua ma tran: "))  C=int(input("Nhap so cot cua ma tran: "))  s=int(input("Nhap seed: "))  A=np.random.seed(s)  A=np.random.random([D,C])  print(A) |

***Giải thích:***

*#giúp cho khi gọi random thì đều ra các con số ngẫu nhiên như nhau*

*#VD=np.random.seed(1)*

*# random(size) tạo ra một mảng với các số ngẫu nhiên có giá trị từ 0.0 đến 1.0*

*VD=np.random.random()*

*#normal(loc=0.0,scale=1.0,size=None) vẽ mẫu ngẫu nhiên từ phân phối chuẩn với loc là giá trị*

*#trung bình và scale là độ lệch chuẩn*

*VD=np.random.normal(1,2,(2,3))*

*print(VD)*

**Ví dụ:**

|  |
| --- |
| #Ma trận random với dot nomal  #np.random.normal(loc=0.0, scale=1.0, size=None)  #loc là số trung tâm ( float), sacle là độ lệch chuẩn (float), size là kiểu ma trận  #Tạo một ma trận 2x3 với các số xung quanh 5, độ lệch chuẩn là 0.5  np.random.normal(loc = 5.0, scale = 0.5 ,size = (2,3)) |

**Kết quả:**

array([[4.93827499, 4.79755991, 5.39900885],

[5.3570059 , 4.70813973, 4.60530852]])

|  |
| --- |
| #Ma trận random với dot rand  #np.random,rand(d0,d1,d2,...dn) Ví dụ : d0=2,d1=3 là ma trận 2x3  #Trả về một mảng vô hướng với độ lệch chuẩn [0,1)  np.random.rand(2,3)  #nếu dot randn thì độ lệch chuẩn là ngẫu nhiên |

**Kết quả:**

array([[0.97383859, 0.25841026, 0.78402286],

[0.92507325, 0.98875593, 0.213005 ]])

|  |
| --- |
| #Ma trận random với dot ranint  #np.random.randint(low, high=None, size=None, dtype=’l’), nếu truyền 1 số thì nó sẽ hiểu là high  #low và hight là giá trị thấp nhất và cao nhất phần tử có thể nhận được trong ma trận  np.random.randint(1,5, size = (2,3),dtype = 'int') |

**Kết quả:**

array([[2, 1, 3],

[3, 3, 4]])

## **Tạo mảng theo số phần tử**

|  |
| --- |
| def TaoMangTheoSoPhanTu():  Bd=int(input("So nho nhat: "))  Kt=int(input("So lon nhat: "))  Pt=int(input("Nhap so phan tu: "))  A=np.linspace(Bd,Kt,Pt)  print(A) |

***Giải thích:***

*#linspace(start,stop,num=50,endpoint=True,retstep=False,dtype=None,axis=0) tạo một*

*#list với số phần tử quy định, khoảng cách bằng nhau giữa các phần tử, mặc định là Float*

*H=np.linspace(0,10,5)*

## **Tạo ma trận cùng giá trị**

|  |
| --- |
| def TaoMaTranCungGiaTri():  D=int(input("Nhap so dong cua ma tran: "))  C=int(input("Nhap so cot cua ma tran: "))  Gt=float(input("Gia tri: "))  if Gt%1==0:  Gt=int(Gt);  A=np.full([D,C],Gt)  print(A) |

***Giải thích:***

*#full(shape,fill\_value,dtype=None,order='C') tạo 1 mảng với giá trị fill\_value*

*G=np.full([3,5],6.9)*

## **Tạo mảng theo khoảng cách**

|  |
| --- |
| def TaoMangTheoKhoangCach():  Bd=int(input("So nho nhat: "))  Kt=int(input("So lon nhat: "))  Kc=float(input("Khoảng cách: "))  if Kc%1==0:  Kc=int(Kc);  A=np.arange(Bd,Kt,Kc)  print(A) |

***Giải thích:***

*# arange(start,stop,step,dtype=None)*

*F=np.arange(0,20,3)*

## **Thông số ma trận**

|  |
| --- |
| def ThongSo\_MaTran(A):  X=A.shape  print("So dong:",X[0])  print("So cot:",X[1])  print("So chieu",A.ndim**)** |

***Giải thích:***

*B.shape # trả về số ( dòng , cột) của array*

*B.ndim #trả về số chiều của list*

*B.dtype #trả về kiểu dữ liệu của các phần tử trong list*

*B.size #tổng số thành phần trong numpy*

*#tạo numpy array, bằng cách sử dụng các hàm có sẵn của numpy cung cấp*

*# zeros(shape,dtype=float,order='C') shape kích thước, dtype kiểu dữ liệu (mặc định float)*

*#công dụng tạo 1 mảng(list ???) các số 0*

*D=np.zeros([5,4],dtype = int )*

*# zeros(shape,dtype=None,order='C') shape kích thước, dtype kiểu dữ liệu (mặc định None)*

*#công dụng tạo 1 mảng(list ???) các số 1*

*E=np.ones([3,2])*

## **Ma trận với kiểu dáng copy**

|  |
| --- |
| a = np.array([[1,2,3],[4,5,6]]) # 2 hàng 3 cột  print(a)  np.full\_like(a , 4) # tạo ra ma trận có 2 hàng 3 cột như a với toàn giá trị 4 |

## **Ma trận đơn vị**

|  |
| --- |
| np.identity(3, dtype='int') |

**Kết quả:**

array([[1, 0, 0],

[0, 1, 0],

[0, 0, 1]])

## **Sao chép một ma trận**

|  |
| --- |
| a = np.array([[1,2,3],[4,5,6]])  b = a  b[0,2] = 4  print(a)  print(b) |

**Kết quả:**

[[1 2 4]

[4 5 6]]

[[1 2 4]

[4 5 6]]

|  |
| --- |
| #Ở ví dụ trên nếu gán b = a thì cả 2 biến đều sử dụng chung vùng nhớ  #Nếu thay đổi bất cứ giá trị nào của một trong 2, thì biến còn lại cũng thay đổi như vậy  #Để tạo ra một biến độc lập thì làm như sau  a = np.array([[1,2,3],[4,5,6]])  b = a.copy() # như vậy sẽ tạo ra matrix b độc lập vùng nhớ  b[0,2] = 4  print(a)  print(b) |

**Kết quả:**

[[1 2 3]

[4 5 6]]

[[1 2 4]

[4 5 6]]

# **CÁC PHÉP TOÁN TRONG MA TRẬN**

## **CÁC PHÉP TOÁN CƠ BẢN**

### **Cộng trận với một số**

|  |
| --- |
| arrr = np.array([[1,2,3,4],[5,6,7,8]])  print(arrr+2) |

Kết quả:

[[ 3 4 5 6]

[ 7 8 9 10]]

### **Trừ ma trận với một số**

|  |
| --- |
| print(arrr-2) #dung lai ma tran o tren |

Kết quả:

[[-1 0 1 2]

[ 3 4 5 6]]

### **Nhân ma trận với một số**

|  |
| --- |
| print(arrr\*2) #dung lai ma tran o tren |

Kết quả:

[[ 2 4 6 8]

[10 12 14 16]]

### **Chia ma trận với một số**

|  |
| --- |
| print(arrr/2) #dung lai ma tran o tren |

Kết quả:

[[0.5 1. 1.5 2. ]

[2.5 3. 3.5 4. ]]

### **Chia ma trận lấy nguyên**

|  |
| --- |
| print(arrr//2) #dung lai ma tran o tren |

Kết quả:

[[0 1 1 2]

[2 3 3 4]]

### **Lũy thừa ma trận**

|  |
| --- |
| print(arrr\*\*2) #dung lai ma tran o tren |

Kết quả:

[[ 1 4 9 16]

[25 36 49 64]]

## **PHÉP TOÁN LIÊN QUAN ĐẾN ĐẠI SỐ TUYẾN TÍNH**

### **Nhân hai ma trận**

|  |
| --- |
| # Nhân 2 ma trận A(mxn) và B(sxr) lại với nhau  # Quy tắc nhân : Nếu AxB thì n = s ; Nếu BxA thì r = m  A = np.random.randint(7,size = (2,3),dtype ='int')  print(A)  B = np.random.randint(5,size = (3,4),dtype ='int')  print(B)  np.matmul(A,B) # (2,3)x(3,4) = (2,4) |

Kết quả:

[[2 2 5]

[3 6 1]]

[[1 3 1 4]

[4 4 1 1]

[2 1 4 3]]

array([[20, 19, 24, 25],

[29, 34, 13, 21]])

### **Tạo ma trận đường chéo**

|  |
| --- |
| # np.diagflat(v, k) với v là vecto ma trận, k là đường chéo ( đường chéo chính k = 0 lá gtri mặc định )  diagmatrix = np.diagflat([[1,2],[3,4]],0) # hoặc diagmatrix = np.diagflat([1,2,3,4],0)  print(diagmatrix) |

Kết quả:

[[1 0 0 0]

[0 2 0 0]

[0 0 3 0]

[0 0 0 4]]

### **Lấy vector đường chéo**

|  |
| --- |
| # np.diag(m , k) với m là ma trận cần lấy đường chéo, k là đường chéo  vector1 = np.diag(diagmatrix)  print(vector1) |

Kết quả:

[1 2 3 4]

### **Tính định thức**

|  |
| --- |
| matrix = np.array([[0,2,3],[0,0,6],[9,0,0]])  print(np.linalg.det(matrix)) |

Kết quả:

108.00000000000003

### **Reorganizing Arrays**

|  |
| --- |
| before = np.array([[1,2,3,4],[5,6,7,8]])  # Vị dụ một ma trận có k phần tửm ở đây là 8  phần tử  before  after = before.reshape(4,2)  # định dạng lại cấu hình dảm bảo k phần tử reshape(n0,n1,...nn) with n0 x n1 x....x nn = k  after |

Kết quả:

array([[1, 2],

[3, 4],

[5, 6],

[7, 8]])

### **Lấy rank của ma trận (số hàng khác 0 khi biến đổi về ma trận bậc thang**

|  |
| --- |
| def RankValue(A):  return np.linalg.matrix\_rank(A) |

### **Tìm số nghiệm của hệ phương trình tuyến tính**

|  |
| --- |
| def SolutionLinearEquations(A, B):  C = np.append(A, B.T, axis=1) # Gọi C là ma trận bổ sung  # Nếu Hạng của Ma Trận Hệ Số = Hạng của Ma Trận Bổ Sung  # Xét tiếp , nếu số ẩn (số cột) = rank thì có nghiệm duy nhất  # nếu số ẩn (số cột) > rank thì có vô số nghiệm  if(RankValue(A) == RankValue(C)):  if(RankValue(A) == len(A[0])):  print("He PT co nghiem duy nhat")  x = np.linalg.solve(A, B.T)  print(x)  elif (RankValue(A) < len(A[0])):  print("He PT co vo so nghiem")  # Nếu Hạng của Ma Trận Hệ Số < Hạng của Ma Trận Bổ Sung thì pt vô nghiệm  elif(RankValue(A) < RankValue(C)):  print("He PT vo nghiem") |

# **CODE DEMO MINH HỌA CÁI MÁY TÍNH**

|  |
| --- |
| import numpy as np  print("Input first matrix")  R1 = int(input("Enter the number of rows:"))  C1 = int(input("Enter the number of columns:"))  print("Enter the entries in a single line (separated by space): ")  # User input of entries in a  # single line separated by space  entries1 = list(map(int, input().split()))  A = np.array(entries1).reshape(R1, C1)  print("Input second matrix")  R2 = int(input("Enter the number of rows:"))  C2 = int(input("Enter the number of columns:"))  entries2 = list(map(int, input().split()))  B = np.array(entries2).reshape(R2, C2)  '''print("lua chon cac phuong thuc tinh toan:")  print("Cong 2 ma tran - Press 1")  print("Tru 2 ma tran - Press 2")  print("Nhan 2 ma tran - Press 3")  print("Khoi tao ma tran chuyen vi - Press 4")  print("Tinh dinh thuc ma tran vuong - Press 5")  print("Ma tran nghich dao - Press 6")  print("Ma tran nhan voi mot so - Press 7")  print("Thoat - Press 0")'''  choice= 100  while(choice!=0):  print("lua chon cac phuong thuc tinh toan:")  print("Cong 2 ma tran - Press 1")  print("Tru 2 ma tran - Press 2")  print("Nhan 2 ma tran - Press 3")  print("Khoi tao ma tran chuyen vi - Press 4")  print("Tinh dinh thuc ma tran vuong - Press 5")  print("Ma tran nghich dao - Press 6")  print("Ma tran nhan voi mot so - Press 7")  print("Ma tran chia voi mot so - Press 8")  print("Hang cua ma tran - Press 9")  print("Luy thua ma tran - Press 10")  print("Thoat - Press 0")  choice=int(input())  if(choice == 1):  print("Ket qua ma tran:\n",A+B)  elif(choice ==2):  print("Ket qua ma tran:\n",A-B)  elif(choice ==3):  print("Ket qua ma tran:\n",A@B)  elif(choice ==4):  print("Ket qua ma tran chuyen vi:\n",A.T)  elif(choice == 5):  print("Dinh thuc cua ma tran vuong la:\n",np.linalg.det(A))  elif(choice == 6):  print("Ma tran nghich dao:\n",np.linalg.inv(A))  elif(choice == 7):  num=int(input("Ma tran nhan voi mot so, nhap so vao:\n "))  print(A\*num)  elif(choice == 8):  num=int(input("Ma tran chia voi mot so, nhap so vao:\n "))  print(A/num)  elif(choice == 9):  print("Hang cua ma tran la: ",np.linalg.matrix\_rank(A))  elif(choice == 10):  num=int(input("Luy thua cua ma tran, nhap so vao: "))  print(np.linalg.matrix\_power(A, num))  else:  print("Tat may tinh") |