TIN HỌC CƠ SỞ
Phần 2: Ngôn ngữ lập trình Python

CHƯƠNG 11.
KIỂU DỮ LIỆU LIST, TUPLE, SET, DICTIONARY

Tin cơ sở (LT): 010100229802 - DHKL16A1HN, - DHKL16A2HN

GV. Cao Diệp Thắng

Mục tiêu chương

Nắm vững:

- Khái niệm, cú pháp khai báo, khởi tạo các kiểu dữ liệu List, Tuple, Set, Dictionary
- Cách viết code với các kiểu dữ liệu List, Tuple, Set, Dictionary

Nội dung

1. Kiểu dữ liệu tuần tự (sequential data type)
2. List (danh sách) trong Python
3. Tuple (hàng)
4. Set (tập hợp)
5. Dictionary (từ điển)

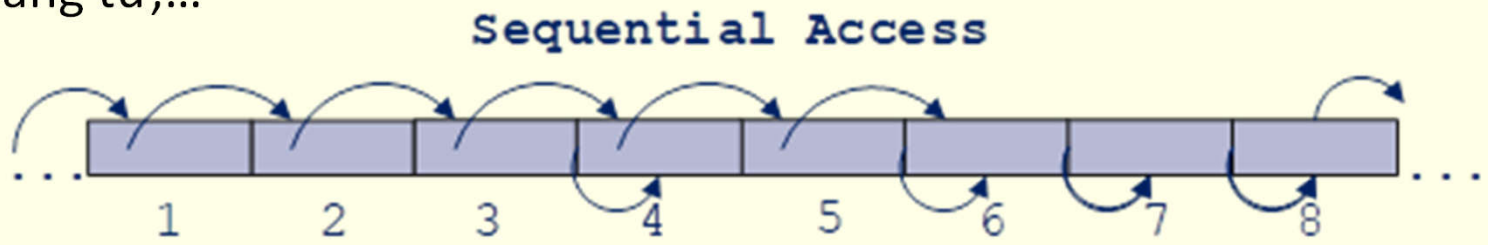
11.1. KIỂU DỮ LIỆU TUẦN TỰ

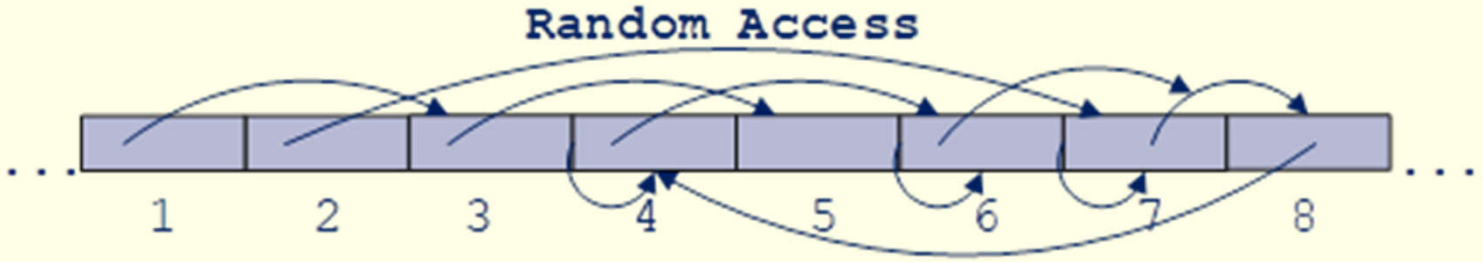
11.1.1. Kiểu dữ liệu tuần tự (sequential data type)

Hai loại đặc trưng cơ bản trong điều khiển

- ❑ Sequential access: truy cập tuần tự.
- ❑ Random access: truy cập ngẫu nhiên

Tuần tự: khá thông dụng trong cuộc sống, chẳng hạn như xếp hàng, xử lý dây chuyền, lưu trữ trong băng từ,…





11.1.2. Kiểu dữ liệu tuần tự trong python

Kiểu dữ liệu tuần tự trong Python là những kiểu dữ liệu chứa bên trong nó các dữ liệu con. Mỗi phần tử trong dãy được gán một số, là vị trí hoặc chỉ mục của nó. Chỉ mục đầu tiên là 0, chỉ mục thứ hai là 1, và …và thường được xử lý bằng cách lấy ra từng phần-tử-một theo thứ tự nào đó (thường là sử dụng vòng lặp for)

- ☐ Có 3 kiểu tuần tự thông dụng là list, tuple và range

11.2. LIST (DANH SÁCH) TRONG PYTHON

11.2.1. Khái niệm:

Danh sách (list) là một loại dữ liệu linh hoạt nhất trong Python. Nó có thể được viết như một danh sách các giá trị được phân cách bởi dấu phẩy giữa các dấu ngoặc vuông.

Danh sách (list) có mối quan hệ với các mảng của các ngôn ngữ lập trình khác như C, C ++ hoặc Java,...nhưng list trong Python **linh hoạt** hơn nhiều so với các cấu trúc dữ liệu kiểu mảng/array "cổ điển".

- ❑ Các mục trong danh sách không nhất thiết phải có cùng kiểu dữ liệu
- ❑ Danh sách (list) có thể mở rộng khi chương trình đang chạy

Ví dụ 11.1.

```
seasons = ['Xuân', 'Hạ', 'Thu', 'Đông']
numbers = [1,2,3,4,5]
animals = ['dog', 'cat', 'birds', 'elephant']
```

11.2.2. Tính chất của List

- List trong Python là cấu trúc dữ liệu cơ bản thuộc nhóm kiểu dữ liệu tuần tự (sequenctial data type)
- List có thể xem là một dãy các đối tượng (một loại mảng đa năng), mỗi phần tử có một vị trí gọi là **index**, phần tử đầu tiên có **index = 0**, phần tử cuối cùng có **index = số phần tử trong list – 1**
- Các phần tử trong list co khả năng lưu giữ các kiểu dữ liệu khác nhau. Tuy nhiên cùng kiểu thì sẽ thuận tiện cho xử lý tính toán.

11.2.3. Tạo list

Cú pháp:


```
<ten_list>=[giatri1, giatri2,...,giatriN]
```

Lưu ý: *Nếu một danh sách (list) lại là một phần tử của một một danh sách (list) khác, thì ta gọi nó là sublist. Tính chất như một list thông thường.*

11.2.3. Tạo list,…

Ví dụ 11.2. Khai báo trực tiếp: liệt kê các phần tử con đặt trong cặp ngoặc vuông ([]), ngăn cách bởi dấu phẩy (,)

```
>>> list1 = [1,2,3,4,5]
>>> list2 = [2.5,7.15,3.14,9.3]
>>> list3 = ['Xuân','Ha', 'Thu', 'Dong']
>>> list4 = ['hello',2022, '10.5', [7,9]]
>>> chuoi = "Xuan, Ha, Thu, Dong"
>>> lst=chuoi.split(",")

>>> print(lst)
['Xuan', ' Ha', ' Thu', ' Dong']
```

Tách chuỗi tạo ra list

Phân cách dấu phẩy ","

Nhận xét: *kiểu chuỗi (str) trong python có thể xem như một list đặc biệt, bên trong gồm toàn các str độ dài 1.*

11.2.4. Truy xuất phần tử trong list

Cú pháp: **tên_list[index],** với index từ 0 đến chiều dài list -1

Ví dụ 11.3.

```
>>> list1 = [1,2,3,4,5]
>>> list2 = [2.5,7.15,3.14,9.3]
>>> list3 = ['Xuân','Ha', 'Thu', 'Dong']
>>> list4 = ['hello',2022, '10.5', [7,9]]
>>> print(list1[0])
1
>>>
>>> print(list2[1])
7.15
>>> print(list3[2])
Thu
```

☐ Cú pháp: **tên_list[start:stop]**

Ví dụ 11.4.

```
>>> numbers = [3,8,5,1,4,2,7,6,9,10]
>>> print("Day so:", numbers)
Day so: [3, 8, 5, 1, 4, 2, 7, 6, 9, 10]
print("\nnumbers[0] =", numbers[0])
print("\nnumbers[2] =", numbers[2])
print("\nnumbers[9] =", numbers[-1])
```

Kết quả:

```
numbers[0] = 3
numbers[2] = 5
numbers[9] = 10
>>> print("\nCac so tu index 0 den can 4:", numbers[0:4])
>>> print("\nCac so tu index 6 den het:", numbers[6:])
```

Kết quả:

```
Cac so tu index 0 den can 4: [3, 8, 5, 1]
Cac so tu index 6 den het: [7, 6, 9, 10]
```

Ví dụ 11.4. (tiếp...)

```
>>> print("\nIn 5 so dau tien :", numbers[0:5])
>>> print("\nIn 5 so cuoi cung :", numbers[-5:])
Kết quả:
In 5 so dau tien : [3, 8, 5, 1, 4]
In 5 so cuoi cung : [2, 7, 6, 9, 10]
```

11.2.5. Cập nhật cho giá trị cho phần tử trong list

Cú pháp: **tên_list[index] = giá_trị**

Ví dụ 11.5

```
>>> list5 = [15,10,23,7,8]
>>> print(list5[1])
10
>>> list5[1] = 100
>>> print(list5[1])
100
>>>
>>> print(list5)
[15, 100, 23, 7, 8]
```

11.2.6. Xóa phần tử trong list

Ví dụ 11.6

```
del(tên_list[index])
```

```
>>> list6 = ['one', 'two', 'three', 'four', 'five']
>>> print("Before:", list6)
Before: ['one', 'two', 'three', 'four', 'five']
>>>
>>> del(list6[1])
>>> print("After:", list6)
After: ['one', 'three', 'four', 'five']
```

11.2.7. Các phương thức cơ bản của List

a. Chiều dài của List (Số phần tử trong list)

Cú pháp: **len(tên_list)**

Ví dụ 11.7

```
>>> list1 = [1,2,3,4,5]
>>> print(len(list1))
5
```

b. Tạo list mới bằng cách nối các list với nhau

Cú pháp **tên_list_mới=tên_list1+tên_list2+tên_list3+…**

Ví dụ 11.8

```
>>> list1 = [1,2,3,4,5]
>>> list2 = [5,6,7,8]
>>> list3 = [9,10]
>>> list4 = list1+list2+list3
>>> print(list4)
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10]
```

a. Tạo list mới bằng cách lặp lại list đã có

Cú pháp:

tên_list_mới=tên_list_cũ*số_lần_lặp

Ví dụ 11.9

```
>>> list_old=["one", "two", "three"]
>>> list_new = list_old*3
>>> print(list_new)
['one','two','three','one',’two','three','one','two','three']
```

Tạo list mới bằng cách copy phần tử từ vị trí index đến cuối list

Cú pháp:

tên_list_mới=tên_list_cũ[index:]

Ví dụ 11.10

```
>>> list_nums = [1,2,3,4,5,6]
>>> list_temp = list_nums[3:]
>>> print(list_temp)
[4, 5, 6]
```

e. Tìm kiếm một giá trị trong list,

Tìm kiếm giá trịtrả về True nếu tìm thấy, ngược lại trả về False.

Cú pháp:

```
giá_trị in tên_list
```

Ví dụ 11.11

```
>>> list_animals = ['ant', 'bear', 'cat', 'dog', 'elephant']
>>> find = "dog" in list_animals
>>> print(find)
True
```

a. Duyệt list

Cú pháp:

```
for item in tên_list:
#xử lý hiển thị hoặc tính toán
```

Ví dụ 11.12

```
>>> list_nums = [10,55,8,12,7,9]
>>> sum = 0
>>> for num in list_nums:
        sum += num
>>> print(sum)
101
```

g. Tìm max/min, tính tổng, trung bình trong list

Tìm **max/min**

Cú pháp:

```
max(tên_list)
min(tên_list)
```

Ví dụ 11.13

```
>>> list_nums = [11,27,30,40,5,68]
>>> print(max(list_nums))
68
>>> print(min(list_nums))
5
```

Tính tổng, tính trung bình của list:

Cú pháp:

```
sum(tên_list)
sum(tên_list)/len(tên_list)
```

Ví dụ 11.14

```
1   ages = [23,25,22,24,23,26]
2   print('\nAges:', ages)
3   s=sum(ages)
4   print('Tổng:',s)
5   m = sum(ages)/len(ages)
6   print('Trung bình: ', round(m,2))
```

Kết quả:

```
Ages: [23, 25, 22, 24, 23, 26]
Tổng: 143
Trung bình:  23.83
```

h. Thêm phần tử vào cuối list

Cú pháp:

```
tên_list.append(element)
```

Ví dụ 11.15

```
1   list_s = [11,27, "one", "ten"]
2   list_s.append("tree")
3   print(list_s)
```

Kết quả:

```
[11, 27, 'one', 'ten', 'tree']
```

Trong Python Shell thực hiện các lệnh sau:

```
>>> ages = []
>>> ages.append(23)
>>> ages.append(21)
>>> ages.append(27)
>>> ages.append(25)
>>> print(ages)
[23, 21, 27, 25]
```

h. Đếm số lần xuất hiện của một phần tử trong danh sách list

Cú pháp:

tên_list.count(element)

Ví dụ 11.16

```
>>> list_nums = [1,4,7,2,5,8,2,9,4,2]
>>> list_nums = [1,4,7,2,5,8,2,9,4,2]
>>> print(list_nums.count(2))
3
>>> print(list_nums.count(4))
2
```

h. Mở rộng list

Cú pháp: **tên_list_muốn_mở_rộng.extend(tên_list)**

Ví dụ 11.17

```
>>> small_list = [1,2,3]
>>>large_list = [4,5,6]
>>>
large_list.extend(small_list)
>>> print(large_list)
[4, 5, 6, 1, 2, 3]
```

Tìm index của

h. Tìm index của phần tử trong list:

Trả về index nhỏ nhất nếu tìm thấy, nếu không tìm thấy sẽ thông báo lỗi

Cú pháp:

tên_list.index(element)

Ví dụ 11.18

```
>>> all_list = ['one','two', 'three', 'abc', 1,2,3]
>>> print(all_list.index("abc"))
3
```

i. Chèn/Thêm một phần tử vào list ở vị trí index

Cú pháp: **tên_list.insert(index, element)**

Ví dụ 11.19

```
>>> all_list = ['one', 'two', 'three', 'abc', 1,2,3]
>>> all_list.insert(2, "elephant")
>>> print(all_list)
['one', 'two', 'elephant', 'three', 'abc', 1, 2, 3]
```

h. Lấy một phần tử ra khỏi list: kết quả trả về là phần tử

Cú pháp:

tên_list.pop([index])

với index = 0 đến chiều dài -1, mặc định không có index thì sẽ lấy phần tử ở cuối list

Ví dụ 11.20

```
>>> all_list = ['one', 'two', 'three', 'abc', 1,2,3]
>>> print(all_list) →['one', 'two', 'three', 'abc', 1, 2, 3]
>>> print(all_list.pop()) → 3
>>> print(all_list)→['one', 'two', 'three', 'abc', 1, 2]
>>> print(all_list.pop())→2
>>> print(all_list)→['one', 'two', 'three', 'abc', 1]
>>> print(all_list.pop(2))→three
>>> print(all_list)→['one', 'two', 'abc', 1]
```

h. Xóa một phần tử ra khỏi list:

Trường hợp nếu không có phần tử cần xóa sẽ thông báo lỗi.

Cú pháp: `Tên_list.remove(element)`

Ví dụ 11.21

```
>>> all_list = ['one', 'two', 'three', 'abc', 1,2,3]
>>> print(all_list)
['one', 'two', 'three', 'abc', 1, 2, 3]
>>>

>>> all_list.remove('abc')
>>> print(all_list)
['one', 'two', 'three', 1, 2, 3]
```

h. Đảo ngược list

Cú pháp:

```
tên_list.reverse()
```

Ví dụ 11.22

```
>>> list_animals=["cat", "elephant", "dog", "bird"]
>>> list_animals.reverse()
>>> print("Reversed list: ", list_animals)
Kết quả
Reversed list:  ['bird', 'dog', 'elephant', 'cat']
```

p. Sắp xếp list theo giá trị tăng dần/giảm dần

Sắp xếp tăng

Cú pháp:

tên_list.sort()

Ví dụ 11.23

```
>>> list_animals = ["cat", "elephant", "dog", "bird"]
>>> print(list_animals)
['cat', 'elephant', 'dog', 'bird']
>>>
>>> list_animals.sort()
>>> print("Sorted list: ", list_animals)
Sorted list:  ['bird', 'cat', 'dog', 'elephant']
>>>
>>> list_nums = [1,4,7,2,5,8]
>>> print(list_nums)
[1, 4, 7, 2, 5, 8]
>>>
>>> list_nums.sort()
>>> print(list_nums)
[1, 2, 4, 5, 7, 8]
```

Sắp xếp giảm

Cú pháp

```
tên_list.sort()
tên_list.reverse()
```

Ví dụ 11.24

```
>>> ages = [23,25,22,24,23,26]
>>> print('\nBefore: ', ages)

Before:  [23, 25, 22, 24, 23, 26]
>>> ages.sort()
>>> ages.reverse()
>>> print('\nAfter reverse: ', ages)
After reverse:  [26, 25, 24, 23, 23, 22]
```

q. Tạo list bằng list-comprehension (bộ suy diễn danh sách)

List-comprehension **là một cú pháp cho phép lập trình viên nhanh chóng tạo ra một biến dữ liệu list mới từ một list cũ hoặc vòng lặp dạng in-line, kết hợp với các điều kiện cho trước**. Về bản chất, lập trình viên hoàn toàn có thể tạo ra một list mới bằng cách sử dụng vòng lặp for/while thông thường

Cú pháp 1: **`newlist=[expression for item in list]`**

Ví dụ 11.25: tạo list mới là các phần tử chữ HOA

```
>>> fruits=['apple','banana','cherry', 'kiwi','mango']
>>> newlist = [x.upper() for x in fruits]
>>> print(newlist)
Kết quả:
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']
```

Cú pháp 2:

```
newlist=[expression for item in list if condition ==True]
```

Ví dụ 11.26

Cú pháp 3

```
>>> fruits=['apple','banana','cherry', 'kiwi','mango']
>>> newlist = [x for x in fruits if "a" in x]
>>> print(newlist)
```

Kết quả:

```
['apple', 'banana', 'mango']
```

```
newlist=[expr1 if condition else expr2 for item in list]
```

Ví dụ 11.27

```
>>> fruits=['apple','banana','cherry', 'kiwi','mango']
>>>newlist=[x if x != "banana" else "orange" for x in fruits]
>>> print(newlist)
```

Kết quả: `['apple', 'orange', 'cherry', 'kiwi', 'mango']`

Ghi nhớ: Một số thao tác thông dụng với list

1. Tạo list
2. Duyệt list
3. Truy cập phần tử theo chỉ số
4. Thay đổi nội dung phần tử
5. Cắt list
6. Thêm phần tử vào list
7. Bỏ phần tử ra khỏi list
8. Tìm kiếm phần tử trong list
9. Sắp xếp các phần tử trong list
10. List hỗn hợp

11.3. LIST CỦA LIST. MẢNG NHIỀU CHIỀU

11.3.1. List của List

Trong mô hình List of List ý nghĩa của chỉ số kép như sau [5]:

`A[i][j]` → **phần tử thứ j của List A[i]**

Mô hình dữ liệu List của List có khuôn dạng như sau:

```
A = [<list1>, <list2>, …,<listN>]
```

Mô tả ma trận trong Python

Giả sử ta có ma trận A (mxn) m dòng n cột:

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{mn} \end{pmatrix}, \text{ ví dụ: } \begin{pmatrix} 1 & 2 & 8 \\ 3 & 0 & 4 \\ 5 & 1 & 6 \end{pmatrix}$$

Ma trận A được mô tả bằng kiểu dữ liệu List của List như sau:

$A=[[a_{11}, a_{12}, \dots a_{1n}], [a_{21}, a_{22}, \dots, a_{2n}], \dots, [a_{m1}, a_{m2}, \dots, a_{mn}]]$

Để biểu thị ma trận A trên màn hình chúng ta sử dụng đoạn chương trình sau:

```
for i in range(m):
    for j in range(n):
        print(A[i][j], end = " ")
    print()
```

Ví dụ 11.28. Yêu cầu nhập từ bàn phím các số tự nhiên m,n sau đó nhập từng phần tử của ma trận A(mxn) và in kết quả ma trận ra màn hình.

```
1   #Nhập ma trận
2   a=[]
3   m=int(input("Nhập số tự nhiên m: "))
4   n=int(input("Nhập số tự nhiên n: "))
5   for i in range(m):
6       print("Chuẩn bị nhập ma trận hàng thứ ", i+1, ": ")
7       b=[]
8       for j in range(n):
9           x=int(input("Nhập phần tử thứ "+ str(j+1)+ ": "))
10          b=b+[x]
11      a.append(b)
12  print("Ma trận A đã nhập xong !")
```

```
13  #In ma trận A ra màn hình
14  for i in range(m):
15      for j in range(n):
16          print(a[i][j], end = " ")
17      print()
```

Kết quả:

```
Nhập số tự nhiên m: 3
Nhập số tự nhiên n: 3
Chuẩn bị nhập ma trận hàng thứ  1 :
Nhập phần tử thứ 1: 1
Nhập phần tử thứ 2: 2
Nhập phần tử thứ 3: 8
Chuẩn bị nhập ma trận hàng thứ  2 :
Nhập phần tử thứ 1: 3
Nhập phần tử thứ 2: 0
Nhập phần tử thứ 3: 4
Chuẩn bị nhập ma trận hàng thứ  3 :
Nhập phần tử thứ 1: 5
Nhập phần tử thứ 2: 1
Nhập phần tử thứ 3: 6
Ma trận A đã nhập xong !
1 2 8
3 0 4
5 1 6
```

Ma trận và mảng đa chiều trong python

Mặc dù Python không có kiểu dữ liệu ma trận, nhưng nó cho phép người lập trình triển khai các ma trận bằng cách sử dụng tập hợp danh sách lồng nhau. Danh sách có thể được tạo nếu lập trình viên đặt tất cả các phần tử được đặt trong dấu ngoặc vuông và phân tách từng phần tử bằng dấu phẩy.

Ví dụ 11.29: tạo một ma trận 3x4 với số hàng là 3 và số cột là 4. Chúng ta sẽ coi mỗi phần tử là một hàng và các giá trị trong mỗi hàng là số cột tương ứng.

```
val = [[10,101,90,95],
  [11,102,85,100],
[12,103,90,95]]
```

11.3.2. Thực hiện tạo các ma trận trong Python

Có thể tạo ma trận m x n bằng cách tạo một tập hợp các phần tử (giả sử là m) trước và sau đó làm cho mỗi phần tử được liên kết với một danh sách khác gồm n phần tử.

Ví dụ 11.30

```
1 >>> m = 3
2 >>> n=4
3 >>> val = [0]*m
4 >>> for j in range(m):
5         val[j] = [0]*n
6 >>> print(val)
```

Kết quả: `[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]`

Phần tử thứ 1 Phần tử thứ 2 Phần tử thứ 3

Giải thích: Đoạn mã trên cho phép chúng ta tạo ra một ma trận gồm 3 hàng 4 cột, hoặc tạo ra một danh sách có 3 phần tử, trong đó mỗi phần tử là một danh sách khác gồm 4 phần tử.

11.3.3. Truy cập vào các phần tử trong ma trận

Cú pháp: **val**[row_no][col_no]

Trong đó: row_no là chỉ số hàng, col_no là chỉ số cột

Ví dụ 11.31

Giả sử có ma trận 3x4 như sau: $\begin{pmatrix} 10 & 101 & 90 & 95 \\ 11 & 102 & 85 & 100 \\ 12 & 103 & 90 & 95 \end{pmatrix}$

Viết chương trình thực hiện các nội dung sau:

- Biểu diễn các phần tử của ma trận trong một list,
- in giá trị các phần tử ở dòng thứ 3,
- in giá trị phần tử nằm ở dòng 1 cột 2

```
1 >>>val = [[10,101,90,95],[11,102,85,100],[12,103,90,95]]
2 >>>print(val[2])
3 [12, 103, 90, 95]
4 >>>print(val[0][1])
5 101
```

Ở dòng thứ 2, chúng ta truy cập đến hàng thứ 3 (do các chỉ số đánh số từ 0) và dòng thứ 3, chúng ta truy cập đến phần tử thứ 2 (chỉ số 1) trong cột thứ 1 (chỉ số 0).

11.3.4. Thực hiện một số phép toán với ma trận

a. Cộng hai ma trận

Ví dụ 11.32. Cho 2 ma trận X, Y: $X=\begin{pmatrix}12 & 7 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix}$; $Y=\begin{pmatrix}5 & 8 & 1\\ 6 & 7 & 3\\ 4 & 5 & 9\end{pmatrix}$, tính tổng X+Y

```
>>> X=[[12,7,3],
       [4,5,6],
       [7,8,9]]
>>>
>>> Y=[[5,8,1],
       [6,7,3],
       [4,5,9]]
>>> sum_X_Y = [[0,0,0],
               [0,0,0],
               [0,0,0]]
>>> #Duyet qua tung dong:
>>> for i in range(len(X)):
     #Duyet qua tung cot:
     for j in range(len(X[0])):
          sum_X_Y[i][j] = X[i][j] + Y[i][j]
>>> for k in sum_X_Y:
     print(k)
```

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
```

Lưu ý: trong Python không có kiểu ma trận hay mảng mà tất cả làm việc trên danh sách và tập hợp. Do đó, nếu chúng ta cộng trực tiếp hai danh sách X và Y ở đây sẽ đồng nghĩa với việc chúng ta nối hai danh sách và tạo ra một ma trận 6x3 với các hàng của Y được viết đằng sau các hàng của X:

[12, 7, 3]
[4, 5, 6]
[7, 8, 9]
[5, 8, 1]
[6, 7, 3]
[4, 5, 9]

b. Nhân ma trận

Phép nhân giữa hai ma trận X và Y chỉ được xác định khi số cột trong X bằng số hàng của ma trận Y.

Nếu X là ma trận n x m và Y là ma trận m x k thì XY được xác định và có kích thước n x k.

Ví dụ 11.33. Cho 2 ma trận $X=\begin{pmatrix}12 & 7 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix}; Z=\begin{pmatrix}5 & 8 & 1 & 2\\ 6 & 7 & 3 & 0\\ 4 & 5 & 9 & 2\end{pmatrix};$ tính X*Z

```
>>> #Vi du 4.2 tich ma tran
>>> X=[[12,7,3],
       [4,5,6],
       [7,8,9]]
>>> Z=[[5,8,1,2],
       [6,7,3,0],
       [4,5,9,1]]
>>>
>>> tich_X_Z = [[0,0,0,0],
                [0,0,0,0],
                [0,0,0,0]]
>>>
>>> #Duyet qua hàng cua X
>>> for i in range(len(X)):
     #Duyet qua cot cua Z
       for j in range(len(Z[0])):
          #Duyet qua cac hang cua Z
          for k in range(len(Z)):
               tich_X_Z[i][j] += X[i][k]*Z[k][j]


>>> for t in tich_X_Z:
     print(t)
```

c. Tìm ma trận chuyển vị

Nhận xét: Ma trận chuyển vị là sự hoán đổi giữa các hàng và cột. Nó được ký hiệu là X '. Phần tử ở hàng thứ i và cột thứ j trong X sẽ được đặt ở hàng thứ j và cột thứ i trong X '. Vì vậy, nếu X là ma trận 3x2 thì X‘ sẽ là ma trận 2x3.

Ví dụ 11.34. Cho ma trận X = $\begin{pmatrix} 12 & 7 \\ 4 & 5 \\ 3 & 8 \end{pmatrix}$; Tìm ma trận chuyển vị của X

```
>>>#Vi du 4.3. Tim ma tran chuyen vi
>>>X = [[12,7],
         [4,5],
         [3,8]]
>>>mt_CV = [[0,0,0],
             [0,0,0]]
>>>
>>>#Duyet qua cac hang
>>>for i in range(len(X)):
       #Duyet qua cac cot
```

```
>>> for k in mt_CV:
        print(k)
```

Trong ví dụ trên, chúng ta sử dụng các vòng lặp for lồng nhau để lặp qua từng hàng và từng cột. Tại mỗi điểm, chúng ta đặt phần tử **X[i][j]** vào trong phần tử **mt_CV**[j][i].