

KIỂU DỮ LIỆU DANH SÁCH (LIST)

ĐẶT VẤN ĐỀ

- **Bài toán:** Nhập 5 số nguyên từ bàn phím, tính và đưa ra màn hình trung bình cộng của 5 số này.
- **Nhận xét chương trình:**
 - Sử dụng mỗi biến để lưu một giá trị
 - Đòi hỏi nhiều tên biến khác nhau
 - Chương trình dài, phức tạp và có thể gây ra lỗi
 - Lưu 1000 giá trị?
 - **Giải pháp:** kết hợp dữ liệu trong một kiểu dữ liệu đơn? => Danh sách

- **Chương trình:**

```
1 def trung_binh_cong():
2     #Hàm nhập vào 5 số và trả lại trung bình cộng
3     print("Nhập 5 số nguyên: ")
4     a1 = int(input("Số thứ nhất: "))
5     a2 = int(input("Số thứ hai: "))
6     a3 = int(input("Số thứ ba: "))
7     a4 = int(input("Số thứ tư: "))
8     a5 = int(input("Số thứ năm: "))
9
10    return (a1 + a2 + a3 + a4 + a5)/5
11
12 #Chương trình chính
13 print(f"Trung bình cộng của 5 số đã nhập: {trung_binh_cong(): .2f}")
```

NỘI DUNG

- **Khái niệm và một số thao tác cơ bản trên danh sách**
 - Khái niệm và khai báo kiểu dữ liệu danh sách
 - Truy cập đến các phần tử trong danh sách
 - Một số phép toán và hàm trên danh sách
- **Một số thao tác nâng cao trên danh sách**
 - Duyệt danh sách
 - Thêm một phần tử vào danh sách
 - Xóa một phần tử khỏi danh sách
 - Lấy các phần tử liên tiếp trong danh sách
- **Danh sách 2 chiều**

KHÁI NIỆM VÀ KHAI BÁO KIỂU DỮ LIỆU DANH SÁCH

KHÁI NIỆM DANH SÁCH

- **Danh sách:** cấu trúc dữ liệu có thể lưu trữ được nhiều giá trị dưới cùng một tên biến.
- Danh sách được tạo ra (**khởi tạo**) thông qua phép gán giá trị như sau:

```
lst  = [4, 5, 7, 9, 10]  
lst1 = ["Hoa", "Lan", 1, 8]
```

- Các phần tử trong danh sách được chứa trong cặp ngoặc [], hai phần tử cách nhau bởi dấu phẩy.

KHÁI NIỆM DANH SÁCH

- Các phần tử của danh sách có thể có kiểu dữ liệu khác nhau.
- Mỗi phần tử có thể là hằng số, hằng chuỗi kí tự, biến, biểu thức, hoặc một **danh sách khác**

```
fruits = ["banana", "apple", "orange"]  
print(fruits)  
mixed_lst = ["Lan", "Ha noi", [31, 8], 90]  
print(mixed_lst)
```

TRUY XUẤT CÁC PHẦN TỬ CỦA DANH SÁCH

- Mỗi phần tử của danh sách được truy xuất thông qua **chỉ số (index)**
- Có hai cách đánh chỉ số
 - Chỉ số dương: từ **0 (phần tử đầu tiên)** đến $n - 1$
 - Chỉ số âm: từ $-n$ đến **-1 (phần tử cuối cùng)**

Trong đó n là số lượng phần tử trong danh sách
- Ví dụ
 - $a = [10, 'abc', 30, 40, 'cntt', 50, 80]$
 - Chỉ số của a biểu diễn như hình bên
- Truy xuất tới một phần tử:

ten_danh_sach[chi_so]

0	1	2	3	4	5	6
10	'abc'	30	40	'cntt'	50	80
-7	-6	-5	-4	-3	-2	-1

Dãy chỉ số dương

Dãy chỉ số âm

$a[0] = a[-7] = 10$
 $a[4] = a[-3] = 'cntt'$
 $a[6] = a[-1] = 80$

TRUY XUẤT CÁC PHẦN TỬ CỦA DANH SÁCH

- **Chỉ số trong [] có thể là**
 - Một số nguyên: $a[10]$, $a[-5]$,...
 - Một biến kiểu số nguyên: $a[x]$, $a[y]$, $a[i]$,...
 - Một biểu thức có giá trị số nguyên: $a[x + 1]$, $a[y * 5]$,...
 - Một lời gọi hàm có giá trị trả về là một số nguyên: $a[\max(x, y)]$
 - Một phần tử có giá trị số nguyên của một danh sách khác (hoặc cùng danh sách): $a[[b[5]]]$, $a[a[10]]$,...
 - Khi chỉ số có sự tham gia của các biến, giá trị của chỉ số được tính dựa trên giá trị của các biến tại thời điểm tính.
- **Lưu ý:** Giá trị của chỉ số vượt ra ngoài phạm vi chỉ số của danh sách thì sẽ xảy ra lỗi
- Danh sách có n phần tử thì phạm vi chỉ số trong đoạn: $[0, n-1]$ và $[-n, -1]$.

TRUY XUẤT CÁC PHẦN TỬ CỦA DANH SÁCH

```
1 x = 5
2 y = 10
3 a = [2, 4, 6, 'end', 8, 'python', 10, 'why', 12, 20]
```

- Các lệnh ở hình bên cho kết quả là gì?

```
5 print(a[5])
6
7 print(a[x+2])
8
9 print(a[2-y])
10
11 print(a[x + y])
12
13 print(a[a[-3]])
14
15 print(a[a[a[2]]])
16
17 print(a)
```

```
>>> print(a[5])
python
>>> print(a[x+2])
why
>>> print(a[2-y])
6
>>> print(a[x + y])
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: list index out of range
>>> print(a[a[-3]])
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: list indices must be integers or slices, not str
>>> print(a[a[a[2]]])
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
IndexError: list index out of range
>>> print(a)
[2, 4, 6, 'end', 8, 'python', 10, 'why', 12, 20]
```

DUYỆT DANH SÁCH

- Duyệt bằng giá trị
- Duyệt bằng chỉ số

DUYỆT DANH SÁCH

- Duyệt danh sách là thao tác quan trọng khi sử dụng danh sách
- Duyệt là thao tác thăm các phần tử của danh sách theo một thứ tự nào đó
- Có hai cách duyệt danh sách
 - Duyệt trực tiếp qua từng giá trị của phần tử bằng vòng lặp for
 - Duyệt thông qua chỉ số của phần tử bằng vòng lặp for/while

↓	↓	↓	↓	↓	↓	↓
10	'abc'	30	40	'cntt'	50	80

DUYỆT QUA TỪNG GIÁ TRỊ CỦA PHẦN TỬ

- Ví dụ:

```
1 lst_a = [2, 4, 6, 'end', 8, 'python', 10, 'why', 12, 20]
2
3 #duyệt theo chiều xuôi từ đầu danh sách
4 for item in lst_a:
5     print(item)
6
7 #duyệt theo chiều ngược từ cuối danh sách
8 for item in reversed(lst_a):
9     print(item)
```

Kết quả

2	20
4	12
6	why
end	10
8	python
python	8
10	end
why	6
12	4
20	2

- Hàm `reversed(lst)` trả về một danh sách có các phần tử có thứ tự ngược với danh sách `lst`
- Sử dụng hàm `reversed()` của Python để duyệt danh sách theo chiều ngược

DUYỆT DANH SÁCH THÔNG QUA CHỈ SỐ CỦA PHẦN TỬ

- Ví dụ:

```
1 lst_a = [2, 4, 6, 'end', 8, 'python', 10, 'why', 12, 20]
2
3 #duyệt theo chiều xuôi
4 for i in range(len(lst_a)):
5     print(f"{i}: {lst_a[i]}")
6
7 #duyệt theo chiều ngược
8 for i in range(len(lst_a)-1, -1, -1):
9     print(f"{i}: {lst_a[i]}")
```

Kết quả

```
0: 2
1: 4
2: 6
3: end
4: 8
5: python
6: 10
7: why
8: 12
9: 20
```

```
9: 20
8: 12
7: why
6: 10
5: python
4: 8
3: end
2: 6
1: 4
0: 2
```

- Hàm len() của Python để lấy số lượng phần tử trong danh sách
- Cách này giống với duyệt mảng 1 chiều trong Pascal/C++

MỘT SỐ TOÁN TỬ VÀ HÀM VỚI DANH SÁCH

- Toán tử $*$, in , $+$
- Một số hàm thông dụng

CÁC PHÉP TOÁN TRÊN DANH SÁCH

Phép toán	Cú pháp	Ý nghĩa
+	<code>lst = lst1 + lst2</code>	Tạo danh sách lst bằng cách ghép lst1 và lst2 với nhau
*	<code>lst = lst1 * n</code>	Tạo danh sách lst bằng cách lặp lại danh sách lst1 n lần
in	<code>x in lst</code>	Kiểm tra phần tử x có trong danh sách lst không, nếu có thì kết quả True, ngược lại False
not in	<code>x not in lst</code>	Ngược với phép toán in
is	<code>lst1 is lst2</code>	Kiểm tra hai danh sách có chung một địa chỉ không
==	<code>lst1 == lst2</code>	Kiểm tra hai danh sách có bằng nhau
=	<code>lst1 = lst2</code>	Phép gán danh sách lst2 cho lst1

VÍ DỤ

```
1 a = [10, 20, 30]
2 b = [40, 70]
3 c = a + b
4 print(c)
5
6 c = c + [90]
7 print(c)
8
9 c = c * 3
10 print(c)
```

```
[10, 20, 30, 40, 70]
[10, 20, 30, 40, 70, 90]
[10, 20, 30, 40, 70, 90, 10, 20, 30, 40, 70, 90, 10, 20, 30, 40, 70, 90]
```

CÁC HÀM LÀM VIỆC VỚI DANH SÁCH

Hàm	Ý nghĩa
list ([sequence])	Tạo một danh sách dựa trên các tham số
len (lst)	Trả về số lượng các phần tử trong danh sách lst
sum (lst)	Trả về tổng các số trong danh sách lst
any (lst)	Trả về True nếu có ít nhất một giá trị Boolean trong danh sách lst là True
all (lst)	Trả về True nếu tất cả các giá trị Boolean trong danh sách lst là True
max (lst)	Trả về giá trị lớn nhất trong danh sách lst
min (lst)	Trả về giá trị nhỏ nhất trong danh sách lst
sorted (lst)	Trả về một danh sách mới đã được sắp xếp
reversed (lst)	Trả về một danh sách có các phần tử có thứ tự ngược với danh sách lst
del (lst[index])	Xóa phần tử ở vị trí index trong danh sách lst

VÍ DỤ

```
>>> A = [6, 3, 9, 24, 5, 8]
>>> max(A)
24
>>> sum(A)
55
>>> B = sorted(A)
>>> B
[3, 5, 6, 8, 9, 24]
>>> A
[6, 3, 9, 24, 5, 8]
>>> del(A[3])
>>> A
[6, 3, 9, 5, 8]
```

```
>>> A = [4, 9, 'sunny', 9, 'cloudy', 20, 8, 'windy']
>>> sum(A)
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> max(A)
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: '>' not supported between instances of 'str' and 'int'
>>> del(A[4])
>>> A
[4, 9, 'sunny', 9, 20, 8, 'windy']
>>> B = sorted(A)
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
TypeError: '<' not supported between instances of 'str' and 'int'
```

THÊM PHẦN TỬ VÀO DANH SÁCH

- Sử dụng phép toán +
- Sử dụng phương thức `append`
- Sử dụng phương thức `extend`

SỬ DỤNG PHƯƠNG THỨC APPEND

- Phương thức append: thêm **một phần tử** mới vào cuối danh sách hiện tại

Cú pháp: `<tên_danh_sách>.append(<phần tử>)`

- Ví dụ:

```
1 a_list = ['a', 'b', 'c', 'd', 'e']
2 a_list.append('f')    # Add 'f'
3 a_list
```

`['a', 'b', 'c', 'd', 'e', 'f']`

```
1 n_list = [10, 20, 30, 40]
2 n_list.append(50)    # Add 50
3 n_list
```

`[10, 20, 30, 40, 50]`

PHƯƠNG THỨC EXTEND

- Phương thức extend: thêm một danh sách hoặc một phần tử vào sau danh sách hiện tại

Cú pháp: <tên_danh_sách>.extend(<danh sách hoặc phần tử>)

- Ví dụ:

```
1 list1 = ['a', 'b', 'c']
2 list2 = [1, 2, 3]
3 list1.extend(list2)
4 list1
```

['a', 'b', 'c', 1, 2, 3]

```
1 list1.extend('d')
2 list1
```

['a', 'b', 'c', 1, 2, 3, 'd']

KHÁC BIỆT GIỮA APPEND VÀ EXTEND

```
1 list1 = [11, 22, 33, 44]
2 list1.append([55, 66])
3 list1
```

[11, 22, 33, 44, [55, 66]]

```
1 list1 = [11, 22, 33, 44]
2 list1.extend([55, 66])
3 list1
```

[11, 22, 33, 44, 55, 66]

XÓA PHẦN TỬ KHỎI DANH SÁCH

- Sử dụng hàm **del**
- Sử dụng phương thức **pop**
- Sử dụng phương thức **remove**

PHƯƠNG THỨC REMOVE/POP

- Phương thức **pop**: Trả lại giá trị phần tử cuối cùng và xóa nó trong danh sách.

```
1 n_list = [10, 20, 30]
2 print(n_list) # print the entire items
```

[10, 20, 30]

```
1 n = n_list.pop()
2 print('n =', n)
3 print('n_list =', n_list)
```

n = 30
n_list = [10, 20]

- Phương thức **remove**: xóa một giá trị trong danh sách

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list)
3
4 n_list.remove(44)
5 print(n_list)
```

[11, 22, 33, 44, 55, 66]
[11, 22, 33, 55, 66]

LỖI XẢY RA KHI SỬ DỤNG PHƯƠNG THỨC REMOVE

- **Lỗi xảy ra** khi xóa phần tử không có giá trị nằm trong danh sách

```
1 n_list = [11, 22, 33, 44, 55, 66]
2 print(n_list)
3
4 n_list.remove(88)
5 print(n_list)
```

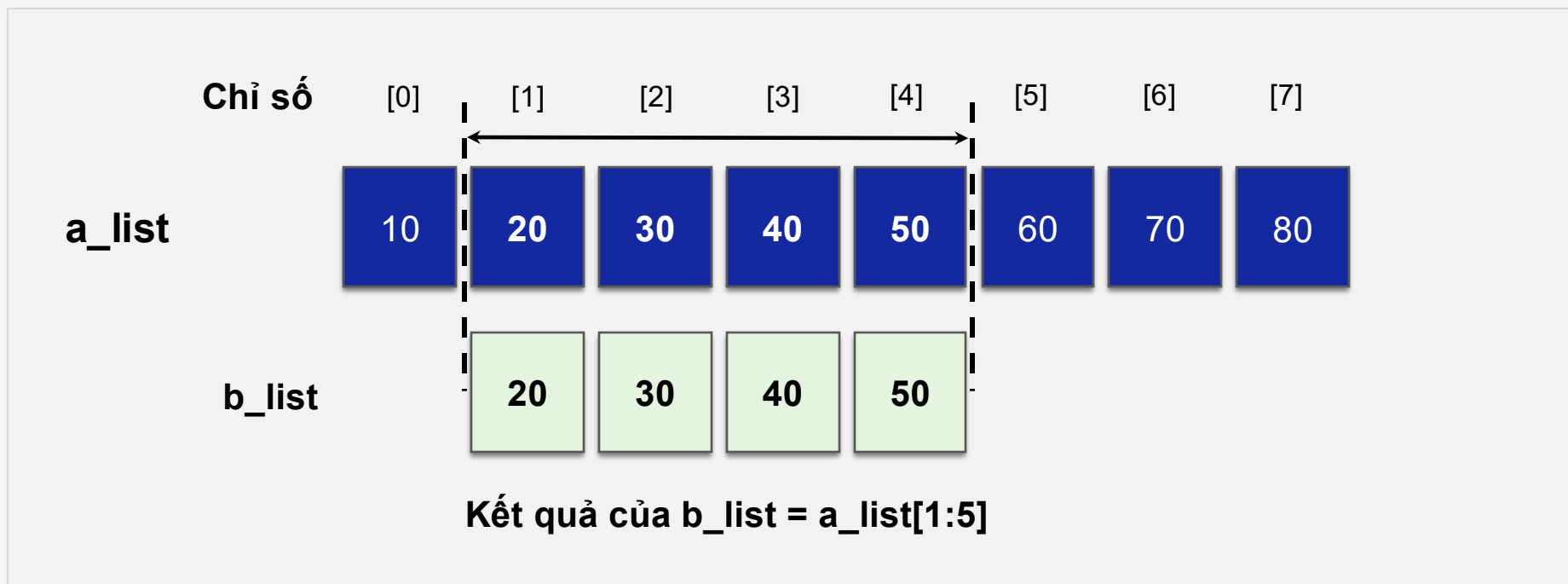
```
[11, 22, 33, 44, 55, 66]
```

```
ValueError                                Traceback (most recent call last)
<ipython-input-46-e5b8ba8b9713> in <module>
      2 print(n_list)
      3
----> 4 n_list.remove(88)
      5 print(n_list)
```

```
ValueError: list.remove(x): x not in list
```

LẤY CÁC PHẦN TỬ LIÊN TIẾP TRONG DANH SÁCH

- **Cú pháp:** <Tên_danh_sách>[start:end:step]
- Trả về danh sách gồm các phần tử từ chỉ số start đến chỉ số end-1 trong Tên_danh_sách.



VÍ DỤ

```
1 a_list = [10, 20, 30, 40, 50, 60, 70, 80]  
2 a_list[1:5]
```

[20, 30, 40, 50]

```
1 a_list[0:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[1:]
```

[20, 30, 40, 50, 60, 70, 80]

```
1 a_list[:5]
```

[10, 20, 30, 40, 50]

```
1 a_list[:]
```

[10, 20, 30, 40, 50, 60, 70, 80]

PHƯƠNG THỨC LẤY CÁC PHẦN TỬ TRONG DANH SÁCH

Cú pháp	Ý nghĩa
<code>a_list[start:end]</code>	Trả về danh sách gồm các phần tử từ start tới end-1
<code>a_list[start:]</code>	Trả về danh sách gồm các phần tử từ start tới cuối danh sách .
<code>a_list[:end]</code>	Trả về danh sách gồm các phần tử từ đầu tới end-1 .
<code>a_list[:]</code>	Trả về toàn bộ danh sách
<code>a_list[start:end:step]</code>	Trả về danh sách gồm các phần tử có chỉ số start, start+step, start+2*step,....start+t*step với $start + t*step < end$

- Python cho phép chỉ số âm và bước nhảy âm.

VÍ DỤ MINH HỌA

Ví dụ 9.1

- **Bài toán:** Nhập danh sách gồm n số nguyên, hiển thị danh sách theo thứ tự xuôi và ngược, tính và hiển thị giá trị trung bình của các phần tử của danh sách.
- **Ví dụ:**

```
Cho số lượng phần tử: 6
A[0] = 10
A[1] = 20
A[2] = 30
A[3] = 40
A[4] = 50
A[5] = 60
Dãy số đã nhập theo chiều xuôi:
10 20 30 40 50 60
Dãy số đã nhập theo chiều ngược:
60 50 40 30 20 10
TBC: 35.00
```

Ví dụ 9.2

- **Bài toán:** Nhập điểm Toán của 10 em học sinh trong lớp, sắp xếp lại danh sách điểm theo chiều không giảm, hiển thị danh sách điểm lên màn hình; tính và hiển thị điểm trung bình lên màn hình.
- **Ví dụ:**

```
Nhập điểm toán cho lớp:  
Học sinh thứ 0: 6.5  
Học sinh thứ 1: 8.5  
Học sinh thứ 2: 4.5  
Học sinh thứ 3: 8  
Học sinh thứ 4: 9.4  
Học sinh thứ 5: 6.3  
Học sinh thứ 6: 6.7  
Học sinh thứ 7: 5.8  
Học sinh thứ 8: 9  
Học sinh thứ 9: 5  
Danh sách điểm Toán đã nhập:  
[4.5, 5.0, 5.8, 6.3, 6.5, 6.7, 8.0, 8.5, 9.0, 9.4]  
Điểm Toán trung bình của lớp: 6.97
```

BÀI TẬP

- **Bài tập 9.1:** Nhập vào một dãy gồm n số nguyên dương, hãy viết chương trình thực hiện các việc sau:
 - Tính tổng các số chẵn chia hết cho 3 có trong dãy
 - Đếm số lượng số nguyên tố có trong dãy
 - Đếm số lượng số chính phương khác nhau có trong dãy
 - Ví dụ:

```
Cho số lượng phần tử: 8
A[0] = 6
A[1] = 13
A[2] = 8
A[3] = 9
A[4] = 13
A[5] = 16
A[6] = 9
A[7] = 19
a. Tổng các số chẵn chia hết cho 3 là: 6
b. Số lượng số nguyên tố trong dãy là: 3
c. Số lượng số chính phương khác nhau là: 2
```

BÀI TẬP

- **Bài tập 9.2:** Cho một danh sách gồm họ tên đầy đủ của một lớp gồm n học sinh ($n < 100$), hãy viết chương trình thực hiện các việc sau:
 - Nhập n và họ tên của n học sinh từ bàn phím (tên học sinh viết tiếng Việt không dấu)
 - Đếm xem có bao nhiêu học sinh trong họ tên có chữ 'h' hoặc 'H'
 - Đếm xem có bao nhiêu học sinh có họ tên "Ngo Bao Chau"
 - Sắp xếp lại danh sách theo chiều giảm dần của họ tên, hiển thị danh sách trước và sau khi sắp xếp lên màn hình
 - Chú ý: Sử dụng hàm để lập trình là một điểm cộng

DANH SÁCH 2 CHIỀU

DANH SÁCH HAI CHIỀU

- **Mỗi phần tử của danh sách là một danh sách khác** (gọi là danh sách con), các danh sách này có thể có số lượng phần tử khác nhau.
- Khi các danh sách con có số lượng phần tử bằng nhau thì tương đương mảng hai chiều/ma trận.
- Ví dụ

```
matrix = [[100, 14, 8, 22, 71],  
          [ 0, 243, 68, 1, 30],  
          [ 90, 21, 7, 67, 112],  
          [115, 200, 70, 150, 8]]
```
- Để truy xuất mỗi phần tử, sử dụng cú pháp: **tên_danh_sách[i][j]**, trong đó **i** là chỉ số phần tử của danh sách (chỉ số dòng), **j** là chỉ số của phần tử của danh sách ở dòng **i** (chỉ số cột)

DANH SÁCH HAI CHIỀU

- Hình dung danh sách hai chiều giống như ma trận ở hình bên, giả sử có n dòng, m cột
- Chỉ số i từ 0 đến $n - 1$
- Chỉ số j từ 0 đến $m - 1$
- Ví dụ: hình bên là danh sách gồm 4 phần tử, mỗi phần tử là một danh sách gồm 5 số nguyên, tương đương với ma trận kích thước 4 x 5.

matrix



	0	1	2	3	4
0	100	14	8	22	71
1	0	243	68	1	30
2	90	21	7	67	112
3	115	200	70	150	8

$$a[0][2] = 8$$

$$a[1][4] = 30$$

$$a[3][3] = 150$$

DUYỆT DANH SÁCH HAI CHIỀU

- Sử dụng **hai vòng lặp for lồng nhau** để duyệt danh sách hai chiều

```
1 list_array = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
2 for i in list_array:
3     for j in i:
4         print(j, end = ' ')
5     print()
```

1 2 3
4 5 6
7 8 9

`[[1, 2, 3], [4, 5, 6], [7, 8, 9]] : list_array`

for i in list_array:

`[1, 2, 3] : được gán tới i`

for j in i

- Duyệt theo từng hàng, từng phần tử** của danh sách:
 - Vòng lặp for ngoài, gán từng hàng của ma trận cho i
 - Vòng lặp for trong, hiển thị mỗi phần tử trên hàng i

DUYỆT DANH SÁCH HAI CHIỀU

- Có thể duyệt khi số lượng phần tử trên các hàng là khác nhau

```
1 list_array = [[1, 2, 3], [4, 5], [7]]
2 for i in list_array:
3     for j in i:
4         print(j, end = ' ')
5     print()
```

```
1 2 3
4 5
7
```

DUYỆT DANH SÁCH HAI CHIỀU

- Duyệt theo chỉ số của các phần tử trong danh sách

```
1 list_array = [[1, 2, 3], [4, 5], [7]]
2 for i in range(len(list_array)):
3     for j in range(len(list_array[i])):
4         print(list_array[i][j], end = ' ')
5     print()
```

```
1 2 3
4 5
7
```

TẠO DANH SÁCH HAI CHIỀU

- Sử dụng hai vòng lặp for lồng nhau, sử dụng phép cộng (+) để thêm các phần tử vào danh sách

```
1 # Tạo danh sách hai chiều
2 n = 3 # số dòng
3 m = 4 # số cột
4 A = []
5
6 # Nhập danh sách
7 for i in range(n):
8     row = []
9     for j in range(m):
10         x = int(input(f"A[{i}][{j}] = "))
11         row = row + [x]
12     A = A + [row]
13
14 # Đưa danh sách ra màn hình dưới dạng ma trận
15 for i in range(n):
16     for j in range(m):
17         print(f"{A[i][j]}", end = " ")
18     print() # xuống dòng
```

```
>>> %Run DS_2D_Tao_+.py
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[0][3] = 4
A[1][0] = 5
A[1][1] = 6
A[1][2] = 7
A[1][3] = 8
A[2][0] = 9
A[2][1] = 10
A[2][2] = 11
A[2][3] = 12
1  2  3  4
5  6  7  8
9  10 11 12
```

TẠO DANH SÁCH HAI CHIỀU

- Sử dụng phương thức `.append()` để tạo danh sách hai chiều

```
1 # Tạo danh sách hai chiều
2 n = 3 # số dòng
3 m = 4 # số cột
4 A = []
5
6 # Nhập danh sách
7 for i in range(n):
8     row = []
9     for j in range(m):
10         x = int(input(f"A[{i}][{j}] = "))
11         row.append(x)
12     A.append(row)
13
14 # Đưa danh sách ra màn hình dưới dạng ma trận
15 for i in range(n):
16     for j in range(m):
17         print(f"{A[i][j]}", end = " ")
18     print() # xuống dòng
```

```
>>> %Run DS_2D_Tao.py
```

```
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[0][3] = 4
A[1][0] = 5
A[1][1] = 6
A[1][2] = 7
A[1][3] = 8
A[2][0] = 9
A[2][1] = 10
A[2][2] = 11
A[2][3] = 12
1 2 3 4
5 6 7 8
9 10 11 12
```

TẠO TỰ ĐỘNG CÁC PHẦN TỬ CỦA DANH SÁCH

- Có thể nhập tự động các phần tử (số) cho danh sách một chiều, hai chiều bằng cách sử dụng các hàm trong thư viện **random**
- Ví dụ

```
1 import random # thư viện tạo số ngẫu nhiên
2
3 # Hàm tạo danh sách 1 chiều
4 def RandomList1D(n):
5     A = []
6
7     for i in range(n):
8         # tạo số ngẫu nhiên trong [-100, 100)
9         x = random.randrange(-100,100)
10        A = A + [x]
11
12    return A
13
14 # Chương trình chính
15 n = int(input("Số phần tử trong danh sách: "))
16
17 list1 = RandomList1D(n)
18
19 print("Danh sách một chiều vừa được tạo ra: ")
20 print(list1)
```

```
>>> %Run -c $EDITOR_CONTENT
```

```
Số phần tử trong danh sách: 5
Danh sách một chiều vừa được tạo ra:
[2, -38, 45, -62, -8]
```

TẠO TỰ ĐỘNG CÁC PHẦN TỬ CỦA DANH SÁCH

```
1 import random # thư viện tạo số ngẫu nhiên
2
3 # Hàm tạo danh sách 2 chiều
4 def RandomList2D(n, m):
5     A = []
6
7     for i in range(n):
8         row = []
9         for j in range(m):
10             x = random.randrange(-100,100)
11             row.append(x)
12         A.append(row)
13
14     return A
15
16 # Chương trình chính
17 n = int(input("Số dòng (n): "))
18 m = int(input("Số cột (m): "))
19
20 matrix = RandomList2D(n, m)
21
22 # Đưa danh sách ra màn hình dưới dạng ma trận
23 for i in range(n):
24     for j in range(m):
25         print(f"{matrix[i][j]}", end = " ")
26     print() # xuống dòng
```

```
>>> %Run RANDOM_2D.py
```

```
Số dòng (n): 3
Số cột (m): 4
-92  -79  -81  73
16   -15  -93  -33
-23  -77   4   61
```

VÍ DỤ MINH HỌA

VÍ DỤ 9.3

- **Bài toán:** Hãy nhập tự động một ma trận vuông bậc n ($n \in [5, 15)$), mỗi phần tử là một số nguyên thuộc $[-100, 100)$, tính và hiển thị tổng của các phần tử trên đường chéo chính và trên đường chéo phụ

- **Ví dụ:**

Ma trận vuông bậc 6:

6 5 8 0 0 3

6 9 8 7 8 3

1 2 4 7 0 8

0 1 0 9 2 2

7 2 8 4 3 4

1 9 6 3 2 9

Tổng trên đường chéo chính: 40

Tổng trên đường chéo phụ: 21

Ma trận vuông bậc 9:

6 7 1 3 5 3 2 5 5

4 8 4 0 9 4 7 4 2

9 0 1 8 6 6 5 0 3

3 1 5 7 0 7 6 8 3

8 5 7 5 5 8 5 1 6

2 5 1 4 8 4 5 2 8

6 1 0 1 5 3 4 2 4

4 1 8 0 4 3 4 5 2

0 3 8 8 8 3 0 4 1

Tổng trên đường chéo chính: 41

Tổng trên đường chéo phụ: 31

BÀI TẬP THỰC HÀNH

BÀI TẬP

- **Bài tập 9.3:** Hãy nhập tự động một ma trận kích thước $n \times m$ ($n, m \in [5, 15)$), mỗi phần tử là một số nguyên thuộc $[-100, 100)$ và thực hiện các việc sau đây
 - Hiển thị ma trận lên màn hình
 - Đếm số lượng số thuộc đoạn $[-20, 20]$ có trong ma trận
 - Đếm số lượng số nguyên dương khác nhau có trong ma trận
 - Tính tích vô hướng của dòng thứ k_1 và k_2 (k_1 và k_2 nhập từ bàn phím)