Viết chương trình tính tổng 1 dãy các số nguyên.

Biết rằng số nguyên đầu tiên nhập vào là số lượng các giá trị, còn lại được nhập vào.

Chương trình cần đọc vào 1 giá trị tương ứng với mỗi lệnh scanf.

Dữ liệu nhập vào có thể là

5 100 200 300 400 500

trong đó 5 cho biết là có 5 số sau nó cần được tính tổng.

Dữ liệu in ra phải là:

1500

// Gợi ý giải thuật không dùng kiểu dl mảng

Nhập( n );

sum = 0;

Lặp lại n lần {

Nhập( x );

sum += x;

}

In( sum );

**Input Format**

* Số nguyên đầu tiên chỉ số lượng số cần tính tổng.
* Các số nguyên tiếp theo là các số cần được tính tổng.
* Hai số cách nhau 1 kí tự trắng.

**Constraints**

* Số lượng số cần tính tổng lớn hơn hoặc bằng 0.
* Số lượng các số nhập vào phải bằng số đầu tiên.

**Output Format**

* Số nguyên chỉ tổng các số.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 100 200 300 400 500 | 1500 |

n = int(input())

Sum = 0

for i in range (n):

a = int(input())

Sum += a

print(Sum)

**Đúng hoàn toàn**

n, \*a = map(int, input().split())

s = sum(a)

print(s)

Hoặc  
**n, \*a = map(int, input().split())**

**s = 0**

**for i in a:**

**s += i**

**print(s)**

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 0 | 0 | 0 |  |
|  | 2 5 9 | 14 | \*\*\*Error\*\*\*  Traceback (most recent call last):  File "prog.python3", line 1, in  n = int(input())  ValueError: invalid literal for int() with base 10: '2 5 9' |  |

Nhập vào 3 số thực a, b, c. Hãy kiểm tra xem 3 số thực trên có tạo thành 3 cạnh của tam giác hay ko? Nếu có tính và in ra chu vi, diện tích tam giác. Nếu không thông báo "DL sai"

Input:

- a, b, c cách nhau dấu cách.

Output:

- In ra thông báo "DL Sai" hoặc in ra:

- Dòng 1: "Dien tich tam giac: X"

- Dòng 2: "Chu vi tam giac: Y"

Với X, Y là diện tích, chu vi của tam giác.

Constraints:

- các giá trị là số thực, độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| -459.00 -33.00 -166.00 | DL Sai |

import math

a, b, c = map(float, input().split())

if a + b <= c or b + c <= a or a + c <= b:

print("DL Sai")

else:

p = (a + b + c) / 2

chu\_vi = p \* 2 # Sử dụng chu\_vi thay vì ChuVi

s = math.sqrt(p \* (p - a) \* (p - b) \* (p - c))

dien\_tich = round(s, 2)

chu\_vi\_lam\_tron = round(chu\_vi, 2) # Sử dụng chu\_vi

print("Dien tich tam giac: {:.2f}".format(dien\_tich))

print("Chu vi tam giac:{:.2f}".format(chu\_vi\_lam\_tron))

Nhập một số nguyên chỉ điểm ở thang 10.

In ra xếp loại dựa trên điểm như sau:

* Nếu điểm từ 0~3 in ra **Kem**
* Nếu là 4 in ra **Yeu**
* Nếu từ 5~6 in ra **TB**
* Nếu từ 7~8 in ra **Kha**
* Nếu từ 9~10 in ra **Gioi**
* Nếu điểm không thuộc phạm vi nào bên trên thì in ra **Error**

**Input Format**

* Một số nguyên chỉ điểm (**d**) theo thang 10.

**Constraints**

* 0 <= **d** <= 10

**Output Format**

* In ra một trong các phân loại: **Kem**, **Yeu**, **TB**, **Kha**, **Gioi**
* Hoặc in ra thông báo lỗi: **Error**
* In đúng chữ hoa thường, không in thêm bất kì kí tự nào khác.

**For example:**

| **Input** | **Result** |
| --- | --- |
| -1 | Error |
| 2 | Kem |
| 5 | TB |
| 8 | Kha |
| 10 | Gioi |

diem = int(input())

if diem >= 0 and diem <= 3:

print("Kem")

elif diem == 4:

print("Yeu")

elif diem >= 5 and diem <= 6:

print("TB")

elif diem >= 7 and diem <= 8:

print("Kha")

elif diem >= 9 and diem <= 10:

print("Gioi")

else:

print("Error")

Nhập vào một số nguyên dương **n**. In **n** năm nhuận tiếp theo sau năm nhuận gần nhất là 2016. Năm nhuận là năm thỏa mãn a) hoặc b)

a) Nó lặp lại 4 năm 1 lần, nhưng không lặp lại mỗi 100 năm. (chia hết cho 4 nhưng không chia hết cho 100)

b) Năm nhuận cũng lặp lại mỗi 400 năm. (chia hết cho 400)

**Input Format**

* Một số nguyên dương - **n**.

**Constraints**

* **n** > 0

**Output Format**

* In **n** năm nhuận sau năm 2016 trên 1 dòng, cách nhau 1 dấu cách.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 5 | 5 | 2020 2024 2028 2032 2036 |

n = int(input())

current\_year = 2016

leap\_years = []

while len(leap\_years) < n:

current\_year += 1

if (current\_year % 4 == 0 and current\_year % 100 != 0) or (current\_year % 400 == 0):

leap\_years.append(current\_year)

print(\*leap\_years)

Viết chương trình nhập vào 2 số nguyên a, b. Tính tổng các số lẻ trong đoạn [a, b] hoặc [b, a]

Input:

a b trên cùng một dòng, cách nhau dấu cách.

Ví dụ: 5 12 hoặc 30 -8

Output:

Tổng các số lẻ tìm được. Ví dụ: 30

Constrains:

các biến kiểu nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 1 | 25 |

a, b = map(int, input().split())

nho = min(a,b)

lon = max(a,b)

tong = 0

for i in range(nho,lon+1):

if(i%2!=0):

tong += i

print(tong)

Viết chương trình nhập vào 2 số nguyên a, b. Tính tổng các số chẵn trong đoạn [a, b] hoặc [b, a]

Input:

a b trên cùng một dòng, cách nhau dấu cách.

Ví dụ: 5 12 hoặc 30 -8

Output:

Tổng các số chẵn tìm được. Ví dụ: 30

Constrains:

các biến kiểu nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 1 | 20 |

a, b = map(int, input().split())

nho = min(a,b)

lon = max(a,b)

tong = 0

for i in range(nho,lon+1):

if(i%2==0):

tong += i

print(tong)

Nhập 1 số nguyên dương **n** chỉ độ dài cạnh hình tam tác.

In ra hình tam giác gồm các kí tự \* với độ dài 3 cạnh là **n**, cụ thể như sau:

* Dòng 1: in ra (n-1) kí tự ⎵ và (1) cặp kí tự \*⎵
* Dòng 2: in ra (n-2) kí tự ⎵ và (2) cặp kí tự \*⎵
* ...
* Dòng n: in ra (0) kí tự ⎵ và (n) cặp kí tự \*⎵
* Sau dòng cuối dùng (dòng n) là 1 dòng trống

Với ⎵ là kí tự trắng.

**Input Format**

Một số nguyên dương. Vd: 6

**Constraints**

Không có.

**Output Format**

Như mô trả trên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 | \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \* \* \* |

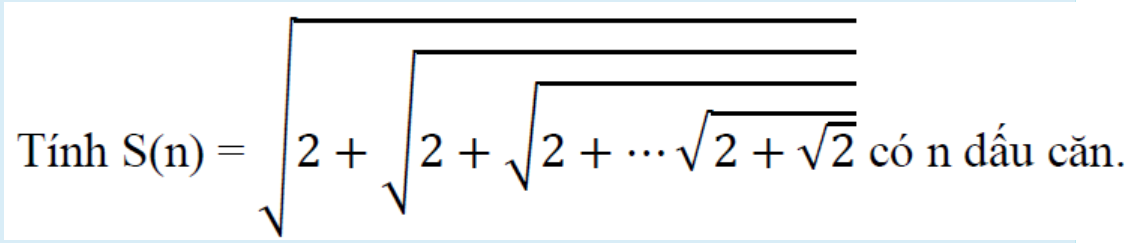
n = int(input())

for i in range(1,n+1):

print(" "\*(n-i), end="")

print("\* "\*i)

Viết chương trình nhập vào số nguyên n.  Tính tổng sau



Input:

n  là số nguyên.

Ví dụ: 5 hoặc -8

Output:

Tổng S(n) như trên với độ chính xác 2 chữ số. Hoặc thông báo "Error." nếu n là số âm.

Constrains:

n kiểu nguyên, s là số thực

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 | 2.00 |

import math

def Tong\_Can(n):

if n<0 : return "Error."

result = 2

for i in range(n):

result = math.sqrt(result + 2)

return "{:.2f}".format(result)

n = int(input())

print(Tong\_Can(n))

Viết chương trình tính tiền phạt khi trả sách muộn cho thư viện theo công thức sau:

* Nếu trả trước hoặc đúng ngày thì tiền phạt là 0
* Nếu trả muộn hơn ngày hẹn trả nhưng trong cùng tháng thì tiền phạt là 5000 \* số ngày trả muộn
* Nếu trả muộn hơn tháng hẹn trả nhưng trong cùng năm thì tiền phạt là 100000 \* số tháng trả muộn
* Nếu trả muộn hơn năm hẹn trả thì số tiền phạt là 1000000

Đầu vào:

* Dòng 1 chứa ba số nguyên là ngày, tháng, năm trả sách
* Dòng 2 chứa ba số nguyên là ngày, tháng, năm hẹn trả sách

Ràng buộc: Dữ liệu đầu vào đảm bảo là ngày tháng năm hợp lệ

Đầu ra: Số tiền phạt

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4 2022  1 4 2022 | 10000 |

# Nhập ngày tháng năm trả sách

ngay\_tra, thang\_tra, nam\_tra = map(int, input().split())

# Nhập ngày tháng năm hẹn trả

ngay\_hen, thang\_hen, nam\_hen = map(int, input().split())

# Tính số ngày trễ

songaytre = 0

if nam\_tra > nam\_hen:

songaytre = 1000000

elif nam\_tra == nam\_hen:

if thang\_tra > thang\_hen:

songaytre = 100000 \* (thang\_tra - thang\_hen)

elif thang\_tra == thang\_hen and ngay\_tra > ngay\_hen:

songaytre = 5000 \* (ngay\_tra - ngay\_hen)

# In kết quả

print(songaytre)

Viết chương trình nhập vào 4 số nguyên a, b, x, y. Tính trung bình cộng các số chẵn trong đoạn [a, b], hoặc [b, a], tính trung bình cộng các số lẻ trong đoạn [x, y], hoặc [y, x].

Input:

- a, b trên cùng một dòng, cách nhau dấu cách.

- x, y trên cùng một dòng, cách nhau dấu cách.

Ví dụ:

5 12

30 -8

Output:

Dòng 1: Trung bình cộng các số chẵn tìm được hoặc "NO" nếu không tính được.

Dòng 1: Trung bình cộng các số lẻ tìm được hoặc "NO" nếu không tính được.

Ví dụ:

30.00

NO

Constrains:

+ các biến a, b, x, y kiểu nguyên.

+ trung bình cộng có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 1  88 88 | 5.00  NO |

a, b = map(int, input().split())

x, y = map(int, input().split())

tong\_chan = 0

tong\_le = 0

dem\_chan = 0

dem\_le = 0

if a < b:

for i in range(a, b+1):

if i % 2 == 0:

tong\_chan += i

dem\_chan += 1

else:

for i in range(b, a+1):

if i % 2 == 0:

tong\_chan += i

dem\_chan += 1

if x < y:

for i in range(x, y+1):

if i % 2 != 0:

tong\_le += i

dem\_le += 1

else:

for i in range(y, x+1):

if i % 2 != 0:

tong\_le += i

dem\_le += 1

if dem\_chan > 0:

print("{:.2f}".format(tong\_chan/dem\_chan))

else:

print("NO")

if dem\_le > 0:

print("{:.2f}".format(tong\_le/dem\_le))

else:

print("NO")

**Hoặc**  
a, b = map(int, input().split())

x, y = map(int, input().split())

MIN\_ab = min(a,b)

MAX\_ab = max(a,b)

MIN\_xy = min(x,y)

MAX\_xy = max(x,y)

tong\_chan\_ab = 0

so\_chan\_ab = 0

tong\_le\_xy = 0

so\_le\_xy = 0

for i in range(MIN\_ab,MAX\_ab+1):

if i % 2 == 0:

tong\_chan\_ab += i

so\_chan\_ab += 1

for j in range(MIN\_xy,MAX\_xy+1):

if i % 2 != 1:

tong\_le\_xy += i

so\_le\_xy += 1

#In ra

if so\_chan\_ab > 0:

print("{:.2f}".format(tong\_chan\_ab/so\_chan\_ab))

else:

print("NO")

if so\_le\_xy > 0:

print("{:.2f}".format(tong\_le\_xy/so\_le\_xy))

else:

print("NO")

Giả sử lịch học của một bạn sinh viên như sau: thứ 2, 3 học lý thuyết tại giảng đường, thứ 4, 5, 6 học thực hành tại phòng máy tính, riêng tối thứ 4 học bù môn Lập trình Python, thứ 7 học ngoại ngữ tại trung tâm ngoại ngữ, chủ nhật được nghỉ.

Bạn hãy viết chương trình thực hiện công việc sau:

Nhập một số nguyên chỉ thứ trong tuần, số 2 ứng với thứ 2, số 3 ứng với thứ 3, ..., số 8 ứng với chủ nhật. In ra lịch học tương ứng của bạn sv trên.

Nếu người dùng nhập các số không đúng quy định thì in ra thông báo "Du lieu sai."

**Input Format**

* Một số nguyên chỉ thứ trong tuần.

**Constraints**

* Số nhập vào là số nguyên

**Output Format**

* In ra thông báo lịch học "Hoc ly thuyet.", "Hoc thuc hanh.", "Hoc ngoai ngu.", "Toi nay hoc bu LT Python.", "Nghi.", "Du lieu sai." như yêu cầu của đề bài.
* Nếu có nhiều thông báo thì mỗi thông báo trên một dòng, thông báo nào đến trước thì ở dòng trên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| -1 | Du lieu sai. |
| 2 | Hoc ly thuyet. |
| 5 | Hoc thuc hanh. |

Giả sử lịch học của một bạn sinh viên như sau: thứ 2, 3 học lý thuyết tại giảng đường, thứ 4, 5, 6 học thực hành tại phòng máy tính, riêng tối thứ 4 học bù môn Lập trình Python, thứ 7 học ngoại ngữ tại trung tâm ngoại ngữ, chủ nhật được nghỉ.

Bạn hãy viết chương trình thực hiện công việc sau:

Nhập một số nguyên chỉ thứ trong tuần, số 2 ứng với thứ 2, số 3 ứng với thứ 3, ..., số 8 ứng với chủ nhật. In ra lịch học tương ứng của bạn sv trên.

Nếu người dùng nhập các số không đúng quy định thì in ra thông báo "Du lieu sai."

**Input Format**

* Một số nguyên chỉ thứ trong tuần.

**Constraints**

* Số nhập vào là số nguyên

**Output Format**

* In ra thông báo lịch học "Hoc ly thuyet.", "Hoc thuc hanh.", "Hoc ngoai ngu.", "Toi nay hoc bu LT Python.", "Nghi.", "Du lieu sai." như yêu cầu của đề bài.
* Nếu có nhiều thông báo thì mỗi thông báo trên một dòng, thông báo nào đến trước thì ở dòng trên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| -1 | Du lieu sai. |
| 2 | Hoc ly thuyet. |
| 5 | Hoc thuc hanh. |

n = int(input())

if n<=1:

print("Du lieu sai.")

elif n == 2 or n == 3:

print("Hoc ly thuyet.")

elif n == 4 or n == 5 or n == 6:

print("Hoc thuc hanh.")

if n == 4:

print("Toi nay hoc bu LT Python.")

elif n == 7:

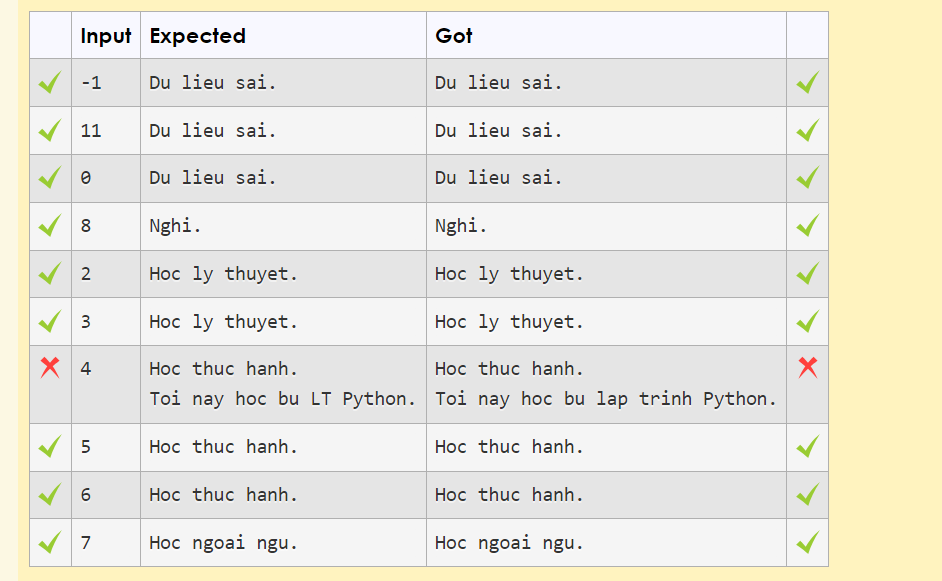
print("Hoc ngoai ngu.")

elif n == 8:

print("Nghi.")

else:

print("Du lieu sai.")



Nhập vào 3 số thực a, b, c. Hãy kiểm tra xem 3 số thực trên có tạo thành 3 cạnh của tam giác hay ko? Nếu có tính và in ra chu vi, diện tích tam giác. Nếu không thông báo "DL sai"

Input:

- a, b, c cách nhau dấu cách.

Output:

- In ra thông báo "DL Sai" hoặc in ra:

- Dòng 1: "Dien tich tam giac: X"

- Dòng 2: "Chu vi tam giac: Y"

Với X, Y là diện tích, chu vi của tam giác.

Constraints:

- các giá trị là số thực, độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| -459.00 -33.00 -166.00 | DL Sai |

import math

a, b, c = map(float, input().split())

if a+b<=c or b+c<=a or a+c<=b:

print("DL Sai")

else:

p = (a+b+c)/2

chu\_vi = p\*2

dien\_tich = math.sqrt(p\*(p-a)\*(p-b)\*(p-c))

print("Dien tich tam giac: {:.2f}".format(dien\_tich))

print("Chu vi tam giac: {:.2f}".format(chu\_vi))

Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm ước số chung lớn nhất của 3 số trên nếu chúng là các số nguyên dương. Nếu không, thông báo "DL sai."

Input: a, b, c là số nguyên cách nhau dấu cách.

Output: Ước số chung lớn nhất của a, b, c, hoặc thông báo "DL sai."

Constrains: các số kiểu int

**For example:**

| **Input** | **Result** |
| --- | --- |
| 18 34 86 | 2 |

import math

a, b, c = map(int, input().split())

if a<=0 or b<=0 or c<=0:

print("DL sai.")

else:

UCLN\_2\_so = math.gcd(a,b)

UCLN\_3\_so = math.gcd(UCLN\_2\_so,c)

print(UCLN\_3\_so)

Viết chương trình nhập vào 2 số nguyên a, b. In ra các số nguyên tố trong đoạn [a, b] hoặc [b, a]. Nếu trong đoạn này không có số nguyên tố thì in ra thông báo "Khong co". Biết số nguyên tố là số nguyên dương >=2 và chỉ chia hết cho 1 và chính nó.

Input: a, b là số nguyên cách nhau dấu cách.

Output: các số nguyên tố tìm được viết trên một dòng, cách nhau dấu cách. Hoặc thông báo "Khong co".

Constrains: các số kiểu int

Ví dụ 1:

+ Input

a= -30, b=-9

+ Output

Khong co

Ví dụ 2:

+ Input

a= 0, b=9

+ Output

2 3 5 7

Ví dụ 2:

+ Input

a= 9, b=0

+ Output

7 5 3 2

**For example:**

| **Input** | **Result** |
| --- | --- |
| -9 -33 | Khong co |
| 28 18 | 23 19 |

import math

def SNT(n):

if n == 2:

return 1

if n % 2 == 0 or n < 2:

return 0

else:

for i in range (3, int(math.sqrt(n)) + 1, 2):

if n%i == 0: return 0

return 1

def KT(a, b):

dem = 0

if(a>b):

c = a

a = b

b = c

for i in range(a,b+1):

if SNT(i) == 1:

dem += 1

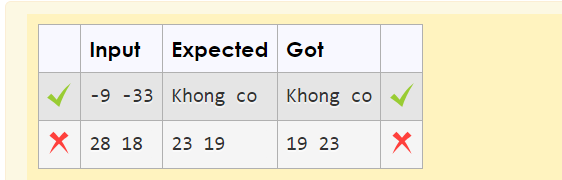
print(i, end=" ")

if dem == 0:

print("Khong co")

a, b = map(int, input().split())

KT(a, b)



Viết chương trình nhập vào 2 số nguyên a, b. Tính trung bình cộng các số chẵn trong đoạn [a, b] hoặc [b, a].

Input:

a b trên cùng một dòng, cách nhau dấu cách.

Ví dụ: 5 12 hoặc 30 -8

Output:

Trung bình cộng các số chẵn tìm được với độ chính xác 2 chữ số thập phân. Ví dụ: 30.00

Constrains:

+ các biến a, b kiểu nguyên.

+ trung bình cộng có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 1 | 5.00 |

a, b = map(int, input().split())

if a>b:

a,b = b,a

sum = 0

count = 0

for i in range (a,b+1):

if i % 2 == 0:

sum += i

count += 1

avg = sum / count

print("{:.2f}".format(avg))

Viết chương trình nhập vào một biểu thức dạng x op y� op �, trong đó:

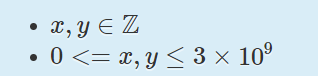
* x,y�,� là hai số nguyên
* opop là một trong năm phép toán: +, -, \*, /, % (chia lấy dư)

và tính giá trị của biểu thức đó. Nếu không thể tính giá trị của biểu thức thì in ra "error"

+ Đầu vào: một dòng chứa biểu thức cần tính giá trị

+ Đầu ra: Giá trị của biểu thức với độ chính xác là hai chữ số sau dấu phẩy

+ Ràng buộc:



**For example:**

| **Input** | **Result** |
| --- | --- |
| 1+2 | 3.00 |

import re

expression = input()

pattern = r"(\d+)([\+\-\\*\/])(\d+)"

match = re.match(pattern, expression)

if match:

x = int(match.group(1))

op = match.group(2)

y = int(match.group(3))

if op == '+':

z = x + y

print("{:.2f}".format(z))

elif op == '-':

z = x - y

print("{:.2f}".format(z))

elif op == '\*':

z = x \* y

print("{:.2f}".format(z))

elif op == '/':

z = x / y

print("{:.2f}".format(z))

elif op == '%':

z = x % y

print("{:.2f}".format(z))

else:

print("error")

Viết hàm nhận đầu vào là số nguyên h khác 0. In ra hình vuông rỗng với cạnh bằng các ký tự \* theo quy tắc sau.

Ví dụ: h=5 thì hình vuông cần vẽ là

\*\*\*\*\*

\*    \*

\*    \*

\*    \*

\*\*\*\*\*

Viết chương trình nhập vào 3 số nguyên x, y, z. Dùng hàm trên để vẽ các hình vuông với chiều cao tương ứng với |x|, |y|, |z|.

Input: 3 số nguyên x, y, z cách nhau dấu cách.

Output:

Dòng 1 in số nguyên x

Từ dòng 2 in ra hình vuông với cạnh |x|

Tiếp theo đến y, z

Constrains: các dữ liệu là kiểu số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 -8 1 | 6  \*\*\*\*\*\*  \* \*  \* \*  \* \*  \* \*  \*\*\*\*\*\*  -8  \*\*\*\*\*\*\*\*  \* \*  \* \*  \* \*  \* \*  \* \*  \* \*  \*\*\*\*\*\*\*\*  1  \* |

def ve\_hinh\_vuong(h):

for i in range (h):

if i == 0 or i == h-1:

print('\*'\*h)

else:

print('\*' + ' ' \* (h-2) + '\*')

x, y, z = map(int, input().split())

print(x)

ve\_hinh\_vuong(abs(x))

print(y)

ve\_hinh\_vuong(abs(y))

print(z)

ve\_hinh\_vuong(abs(z))

**Bạn hãy viết hàm: Kiểm tra các chữ số của một số nguyên dương n có toàn là số chẵn không.**

**Viết chương trình nhập vào ba số nguyên x, y, z. In ra "Yes" nếu các chữ số của x (hoặc y, z) toàn là số chẵn. In ra "No" nếu các chữ số của x (hoặc y, z) có ít nhất một số lẻ. In ra "Error" nếu x (hoặc y, z) là số âm. Thông báo trên cùng một dòng, cách nhau dấu cách.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output: No Error Yes**

**Example 2:**

**+ Input: 190 172 111**

**+ Output: No No No**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | No Error Yes |

def Kiem\_Tra\_Chan(n):

for chan in str(n):

if int(chan) % 2 != 0:

return False

return True

x , y , z = map(int, input().split())

results = []

if x < 0:

results.append("Error")

elif Kiem\_Tra\_Chan(x):

results.append("Yes")

else:

results.append("No")

if y < 0:

results.append("Error")

elif Kiem\_Tra\_Chan(y):

results.append("Yes")

else:

results.append("No")

if z < 0:

results.append("Error")

elif Kiem\_Tra\_Chan(z):

results.append("Yes")

else:

results.append("No")

print(" ".join(results))

**Bạn hãy viết hàm:**

**1) Đếm số chữ số của một số nguyên dương n.**

**2) TÍnh tổng các chữ số của số nguyên dương n.**

**Viết chương trình nhập vào một số n. Nếu n là số âm thì in ra thông báo "Du lieu sai.". Ngược lại in ra số chữ số và tổng các chữ số của nó trên cùng một dòng, cách nhau dấu cách.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190**

**+ Output: 3 10**

**Example 2:**

**+ Input: -190**

**+ Output: Du lieu sai.**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 | 3 10 |

def Get\_Sum(n):

Sum = 0

for digit in str(n):

Sum += int(digit)

return Sum

n = int(input())

if n <= 0:

print("Du lieu sai.")

else:

print(len(str(n)), Get\_Sum(n))

**Bạn hãy viết hàm: Tìm chữ số lớn nhất của một số nguyên dương n.**

**Viết chương trình nhập vào ba số nguyên x, y, z. In ra các chữ số lớn nhất tìm được của x, y, z. In ra "Error" nếu x (hoặc y, z) là số âm. Thông báo trên cùng một dòng, cách nhau dấu cách.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output: 9 Error 8**

**Example 2:**

**+ Input: 190 172 115**

**+ Output: 9 7 5**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | 9 Error 8 |

def Tim\_Max(n):

if n <= 0: return "Error"

max\_digit = 0

while n > 0:

digit = n % 10

if digit > max\_digit:

max\_digit = digit

n //= 10

return str(max\_digit)

x, y ,z = map(int, input().split())

print(Tim\_Max(x), Tim\_Max(y), Tim\_Max(z))

Cho một dãy số tự nhiên a và một số nguyên b. Viết chương trình tìm số lớn nhất chia hết cho b.

* Input: Mỗi test bao gồm hai dòng. Dòng đầu tiên là dãy a, các phần tử cách nhau bởi một khoảng trắng. Dòng thứ hai chứa số b.
* Output: Số lớn nhất chia hết cho b, hoặc 0 (nếu không tìm thấy số nào chia hết cho b)

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 3 8 4 9 2 10  2 | 10 |

def Find\_Max\_Nums\_Divisible(numbers, divisor):

max\_nums = None

for num in numbers:

if num % divisor == 0:

if max\_nums is None or num > max\_nums:

max\_nums = num

return max\_nums if max\_nums is not None else 0

a = list(map(int, input().split()))

b = int(input())

result = Find\_Max\_Nums\_Divisible(a,b)

print(result)

**Hoặc**

def find\_max(n,div):

max\_div = 0

for num in n:

if num%div==0:

if num>max\_div:

max\_div = num

return max\_div

a = list(map(int, input().split()))

b = int(input())

c = find\_max(a,b)

print(c)

Một số được gọi là số mạnh mẽ nếu nó chia hết cho một số nguyên tố và chia hết cho bình phương của một số nguyên tố. Ví dụ, 6363 là số mạnh mẽ vì nó chia hết cho 77 và 3232. Hãy viết chương trình kiểm tra xem một số có phải số mạnh mẽ hay không?

* Đầu vào: một số a
* Đầu ra: true nếu a là số mạnh mẽ, false nếu ngược lại
* Ràng buộc: a là số tự nhiên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 63 | true |
| 10 | false |

**ĐÚNG hoàn toàn**

import math

def SNT(n):

if n < 2: return False

for i in range(2, int(math.sqrt(n))+1):

if n%i == 0: return False

return True

def StrongNums(n):

if n<=1: return "false"

maxFactor = int(math.sqrt(n))

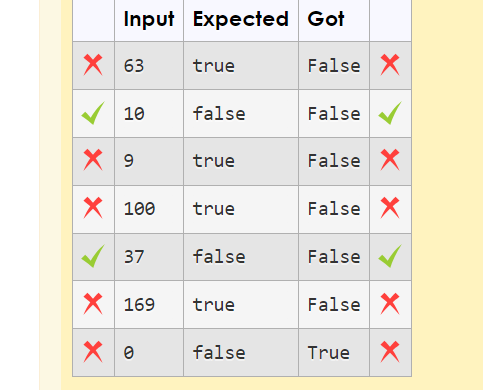
for i in range(2,maxFactor+1):

if n%i==0:

if SNT(i) and n%(i\*i)==0:

return "true"

return "false"

n = int(input())

print(StrongNums(n))

Cho một dãy số nguyên. Hãy chia dãy thành hai phần: phần bên trái gồm các số âm sắp xếp theo thứ tự giảm dần, phần bên phải gồm các số không âm sắp xếp theo thứ tự tăng dần.

* Đầu vào: Dãy các số nguyên cách nhau bởi dấu cách
* Đầu ra: Dãy sau khi đã sắp xếp

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 3 2 -1 -4 0 | -1 -4 0 2 3 5 |

def Sort\_Array(arr):

negative = [x for x in arr if x < 0]

positive = [x for x in arr if x >= 0]

negative.sort(reverse=True)

positive.sort()

return negative + positive

a = list(map(int, input().split()))

result = Sort\_Array(a)

print(" ".join(map(str, result)))

Phần tử đặc biệt trong một ma trận là phần tử xuất hiện trên tất cả các hàng và tất cả các cột. Hãy viết chương trình tìm các phần tử đặc biệt trong một ma trận.

* Đầu vào:
  + Dòng đầu tiên gồm hai số n, m cách nhau bởi dấu cách là số hàng và số cột của ma trận
  + n dòng tiếp theo, mỗi dòng chứa m số là các phần tử trong một hàng, cách nhau bởi dấu cách
* Đầu ra: các phần tử đặc biệt tìm được, cách nhau bởi dấu cách
* Ràng buộc: n, m là các số nguyên dương.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4  0 1 2 3  1 2 4 1  2 6 1 2 | 1 2 |

def find\_special\_elements(matrix):

if not matrix or not matrix[0]:

return []

rows, cols = len(matrix), len(matrix[0])

row\_set = set(matrix[0])

for i in range(1, rows):

row\_set &= set(matrix[i])

col\_set = set()

for j in range(cols):

col\_elements = set(matrix[i][j] for i in range(rows))

col\_set |= col\_elements

return list(row\_set & col\_set)

n, m = map(int, input().split()) # Nhập số hàng và số cột

matrix = [list(map(int, input().split())) for \_ in range(n)] # Nhập dữ liệu cho mỗi hàng

special\_elements = find\_special\_elements(matrix)

print(" ".join(map(str, special\_elements)))

Cho hai số m, n. Hãy loại bỏ một chữ số của m để tạo thành một số p nhỏ nhất chia hết cho n.

* Input: hai số nguyên dương m, n cách hau một dấu cách
* Output: số p tìm được, nếu không tìm được in ra "impossible"
* Ràng buộc:
  + 10 < m < 1000000
  + 1 < n < 100000

**For example:**

| **Input** | **Result** |
| --- | --- |
| 248 12 | 24 |
| 208 4 | 8 |
| 229 23 | impossible |

def remove\_digit\_to\_get\_smallest\_p(m, n):

m\_str = str(m) # Chuyển số m thành chuỗi để dễ dàng xử lý từng chữ số

for i in range(len(m\_str)):

p\_str = m\_str[:i] + m\_str[i+1:] # Loại bỏ chữ số thứ i

p = int(p\_str)

if p % n == 0:

return p

return "impossible"

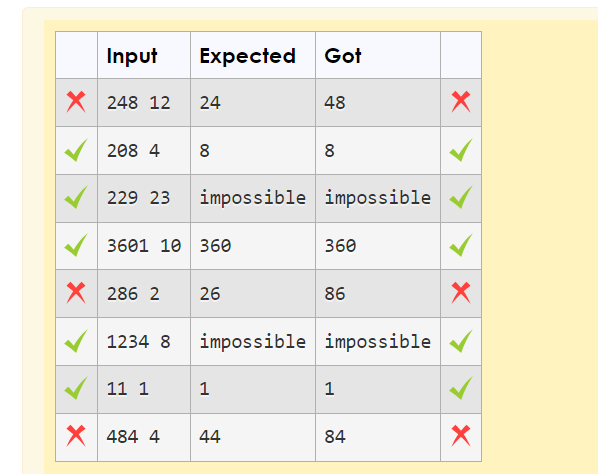
# Đọc input

m, n = map(int, input().split())

# Tìm kết quả và in ra

result = remove\_digit\_to\_get\_smallest\_p(m, n)

print(result)



Nhập ma trận vuông **a** có kích thước **n**x**n**.

Tìm và in ra giá trị lớn nhất trên đường chéo chính của ma trận.

**Input Format**

Dữ liệu nhập vào trên **n+1** dòng.

* Dòng 1: kích thước (số hàng=số cột) ma trận vuông - **n**.
* **n** dòng còn lại: các dòng của ma trận, mỗi dòng chứa các **a[i][j]** cách nhau 1 dấu cách.

**Constraints**

* **n**, **a[i][j]**: số nguyên
* 1 <= **n**<= 100

**Output Format**

* In ra giá trị lớn nhất trên đường chéo chính.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  37 38 -9  -23 19 -17  7 38 -18 | 37 |

n = int(input())

a = []

for i in range(n):

row = list(map(int, input().split()))

a.append(row)

max\_a = a[0][0]

for i in range(n):

if a[i][i] > max\_a:

max\_a = a[i][i]

print(max\_a)

Phần tử đặc biệt trong một ma trận là phần tử xuất hiện trên tất cả các hàng và tất cả các cột. Hãy viết chương trình tìm các phần tử đặc biệt trong một ma trận.

* Đầu vào:
  + Dòng đầu tiên gồm hai số n, m cách nhau bởi dấu cách là số hàng và số cột của ma trận
  + n dòng tiếp theo, mỗi dòng chứa m số là các phần tử trong một hàng, cách nhau bởi dấu cách
* Đầu ra: các phần tử đặc biệt tìm được, cách nhau bởi dấu cách
* Ràng buộc: n, m là các số nguyên dương.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4  0 1 2 3  1 2 4 1  2 6 1 2 | 1 2 |

#Khai bao ham

def find(a):

#Kiem tra ma tran rong hoac khong co cot thi tra ve list rong

if not a or not a[0]:

return []

#Lay so hang so cot cua ma tran

rows, cols = len(a), len(a[0])

#tap hop cac phan tu hang dau tien

row\_set = set(a[0])

#Duyet tu phan tu thu 2 lay giao cua row\_set va phan tu moi hang

for i in range(1,rows):

row\_set &= set(a[i])

#Khoi tao col\_set la tap rong

col\_set = set()

#Duyet tu cot dau tien den co cuoi dung, them phan tu moi cot vao col\_set

for j in range(cols):

col\_elements = set(a[i][j] for i in range(rows))

col\_set |= col\_elements

return list(row\_set & col\_set)

n, m = map(int, input().split())

a = [list(map(int, input().split())) for \_ in range(n)]

r = find(a)

print(" ".join(map(str, r)))

Viết hàm kiểm tra một ma trận vuông có đối xứng qua đường chéo chính không.

Viết chương trình nhập vào hai ma trận vuông cấp N1, N2. Kiểm tra xem các ma trận có đối xứng qua đường chéo chính ko. In kết quả ra màn hình.

Input:

- Dòng thứ nhất nhập vào N1 (kích thước của ma trận thứ nhất)

- Các dòng tiếp theo nhập vào các phần tử của ma trận thứ nhất.

- Dòng tiếp theo nhập vào N2 (kích thước của ma trận thứ nhất)

- Các dòng tiếp theo nhập vào các phần tử của ma trận thứ hai.

Output:

- Dòng thứ nhất in ra thông báo "Ma tran 1: Co/Khong doi xung"

- Dòng thứ nhất in ra thông báo "Ma tran 2: Co/Khong doi xung"

Constrains: 1<=N, N2<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6  5 0 5 11 35 61  28 31 12 9 35 5  16 57 5 55 23 34  4 33 27 14 54 61  12 0 34 61 6 23  5 50 50 51 11 34  3  5 0 7  0 11 2  7 2 6 | Ma tran 1: Khong doi xung  Ma tran 2: Co doi xung |

n = int(input())

a = [[0] \* 100 for \_ in range(100)]

for i in range(n):

a[i] = list(map(int, input().split()))

m = int(input())

b = [[0] \* 100 for \_ in range(100)]

for i in range(m):

b[i] = list(map(int, input().split()))

flag1, flag2 = 0, 0

# Kiểm tra ma trận a

for i in range(n):

for j in range(n):

if a[j][i] != a[i][j]:

flag1 += 1

# Kiểm tra ma trận b

for i in range(m):

for j in range(m):

if b[j][i] != b[i][j]:

flag2 += 1

# Xuất kết quả

if n > 0:

if flag1 != 0:

print("Ma tran 1: Khong doi xung")

else:

print("Ma tran 1: Co doi xung")

if m > 0:

if flag2 != 0:

print("Ma tran 2: Khong doi xung")

else:

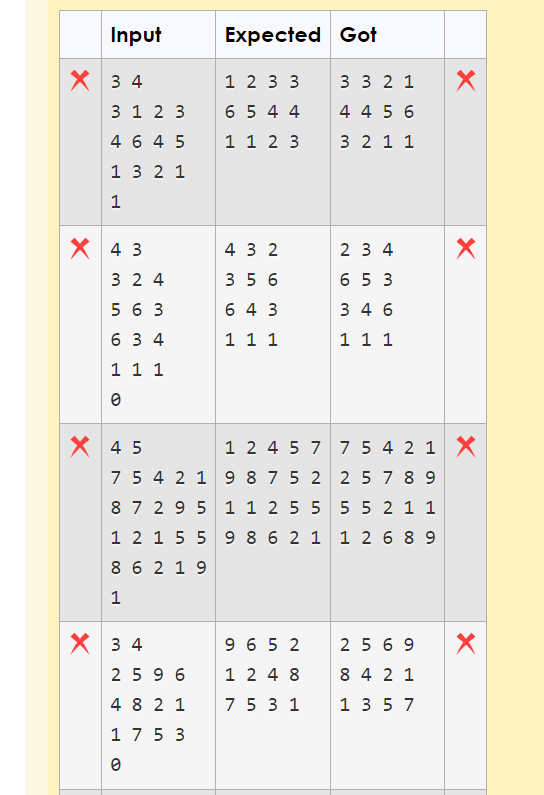
print("Ma tran 2: Co doi xung")

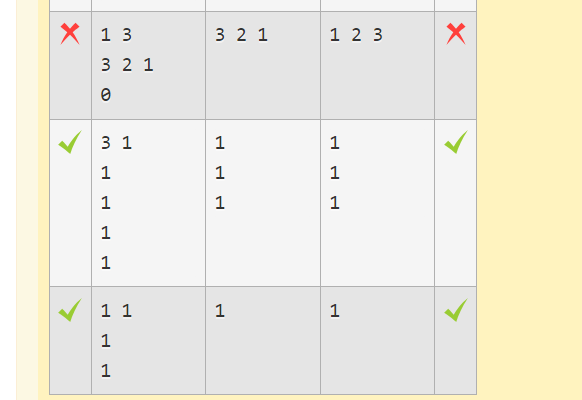
Cho một ma trận, hãy sắp xếp các phần tử trong từng hàng theo thứ tự tăng, giảm xen kẽ theo thứ tự từ trái qua phải. Ví dụ, nếu hàng thứ nhất sắp xếp theo chiều tăng dần thì hàng thứ hai sắp xếp theo chiều giảm dần.

* Đầu vào:
  + Dòng đầu tiên là hai số m, n đại diện cho số hàng và số cột của ma trận, cách nhau bởi dấu cách
  + m dòng tiếp theo, mỗi dòng gồm n số là các phần tử trên một hàng cách nhau bởi dấu cách
  + Dòng cuối cùng là một số nhị phân chỉ ra cách sắp xếp hàng đầu tiên. 0 là giảm dần, 1 là tăng dần.
* Đầu ra: ma trận đã sắp xếp, in ra trên m dòng, mỗi dòng gồm n phần tử cách nhau bởi dấu cách.
* Ràng buộc: các phần tử trong mảng là số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3 4  3 1 2 3  4 6 4 5  1 3 2 1  1 | 1 2 3 3  6 5 4 4  1 1 2 3 |
| 4 3  3 2 4  5 6 3  6 3 4  1 1 1  0 | 4 3 2  3 5 6  6 4 3  1 1 1 |





**ĐÚNG HẾT**def sort\_matrix(matrix, order):

for i in range(len(matrix)):

matrix[i] = sorted(matrix[i], reverse=(1-order[i%2]))

return matrix

#Nhap so hang so cot

n,m = map(int, input().split())

#Khoi tao ma tran

matrix = []

for \_ in range(n):

row = list(map(int, input().split()))

matrix.append(row)

#Nhap thu tu sap xep

order = int(input())

#Sap xep ma tran theo thu tu xen ke

sorted\_matrix = sort\_matrix(matrix, [order, 1-order])

#In ra

for row in sorted\_matrix:

print(" ".join(map(str, row)))

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số nguyên. Sắp xếp từng hàng của ma trận tăng dần. In kết quả ra màn hình.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output:

- Dòng thứ nhất in ra n và m cách nhau dấu cách

- Dòng tiếp theo in ra ma trận kết quả

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 6  17 23 9 3 0 9  12 2 21 12 23 7  12 2 2 13 7 6  7 24 22 12 11 20  4 5 7 24 2 11  15 23 20 22 20 24 | 6 6  0 3 9 9 17 23  2 7 12 12 21 23  2 2 6 7 12 13  7 11 12 20 22 24  2 4 5 7 11 24  15 20 20 22 23 24 |

def nhap(a, n, m):

for i in range(n):

a[i] = list(map(int, input().split()))

def sx(a, n, m):

for i in range(n):

a[i].sort()

def in\_ma\_tran(a, n, m):

print(n, m)

for row in a:

print(\*row)

n, m = map(int, input().split())

a = [[0]\*m for \_ in range(n)]

nhap(a, n, m)

sx(a, n, m)

in\_ma\_tran(a, n, m)

Một xâu kí tự được gọi là đối xứng nếu viết theo chiều ngược lại ta vẫn được xâu đó. Cho một xâu kí tự, hãy tìm xâu con đối xứng dài nhất.

* Đầu vào: xâu s
* Đầu ra: xâu con đối xứng dài nhất của s, nếu có nhiều xâu như vậy, hãy in ra xâu xuất hiện đầu tiên theo thứ tự từ trái sang phải.
* Ràng buộc: xâu s có ít nhất một kí tự

**For example:**

| **Input** | **Result** |
| --- | --- |
| shissi white | issi |
| easy | e |

def Chuoi\_dx(s):

n = len(s)

longest = ""

for i in range(n):

for j in range (i, n):

substring = s[i:j+1]

if substring == substring[::-1] and len(substring) > len(longest):

longest = substring

return longest

s = str(input())

result = Chuoi\_dx(s)

print(result)

Một số được gọi là số mạnh mẽ nếu nó chia hết cho một số nguyên tố và chia hết cho bình phương của một số nguyên tố. Ví dụ, 6363 là số mạnh mẽ vì nó chia hết cho 77 và 3232. Hãy viết chương trình kiểm tra xem một số có phải số mạnh mẽ hay không?

* Đầu vào: một số a
* Đầu ra: true nếu a là số mạnh mẽ, false nếu ngược lại
* Ràng buộc: a là số tự nhiên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 63 | true |
| 10 | false |

import math

def isStrongNumber(n):

# Kiểm tra nếu n <= 1 thì trả về False

if n <= 1:

return "false"

# Tạo max ước số căn bậc 2 của n

maxFactor = int(math.sqrt(n))

# Duyệt từ 2 tới maxFactor

for i in range(2, maxFactor+1):

if n % i == 0:

# Nếu tìm thấy ước số là số nguyên tố

if isPrime(i):

# và bình phương của ước số cũng chia hết cho n

# thì n là số mạnh mẽ

if n % (i\*i) == 0:

return "true"

# Không tìm thấy ước thỏa mãn điều kiện

return "false"

def isPrime(x):

if x < 2:

return "false"

for i in range(2, int(math.sqrt(x))+1):

if x % i == 0:

return "false"

return "true"

n = int(input())

print(isStrongNumber(n))

Viết chương trình nhập vào một danh sách gồm n nhân viên. Tìm nhân viên có lương lớn nhất. Nếu có nhiều người có lương lớn nhất bằng nhau thì kết quả là người ở đầu danh sách. Biết thông tin của một nhân viên gồm: tên, mã nv, hệ số lương, phụ cấp

Input:

+ Dòng thứ nhất nhập vào n là số nhân viên

+ Dòng tiếp theo nhập vào thông tin của các nhân viên theo thứ tự tên, mã, hệ số lương, phụ cấp.

Output:

+ Dòng thứ nhất in ra thông báo "Nhan vien co luong lon nhat"

+ Dòng thứ hai in ra thông tin của nhân viên tìm được, các thông tin cách nhau dấu cách. Thứ tự mã nhân viên, tên nhân viên, hệ số lương, phụ cấp, lương tháng in ở cuối mỗi dòng. Lương tháng được tính bằng hệ số lương \* 2000000 + phụ cấp.

Constrains: 1<=n<=200, mã nhân viên, phụ cấp là các số nguyên dương, tên nhân viên không chứa dấu cách, các thông tin còn lại là số thực có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  Anh 78 2.34 1000000  Vinh 5 4.32 250000  Nhung 61 2.67 700000  Trang 27 4.32 250000 | Nhan vien co luong lon nhat  5 Vinh 4.32 250000 8890000.00 |

class nv:

def \_\_init\_\_(self, ten, ma, hsl, pc):

self.tennv = ten

self.manv = ma

self.hsl = hsl

self.pc = pc

def lg(self):

return self.hsl \* 2000000 + self.pc

# Input

n = int(input())

m = []

for i in range(n):

infor = input().split()

ten = infor[0]

ma = int(infor[1])

hsl = float(infor[2])

pc = int(infor[3])

t = nv(ten, ma, hsl, pc)

m.append(t)

# Output the employee with the highest salary

max\_salary\_employee = max(m, key=lambda x: x.lg())

print("Nhan vien co luong lon nhat")

print(max\_salary\_employee.manv, max\_salary\_employee.tennv, "{:.2f}".format(max\_salary\_employee.hsl),

max\_salary\_employee.pc, "{:.2f}".format(max\_salary\_employee.lg()))

Dãy số tăng là dãy số mà trong đó mọi số đứng sau đều lớn hơn số đứng trước.

Viết hàm kiểm tra một mảng số thực gồm n phần tử có tăng hay ko? Hàm nhận 2 tham số là mảng a và số phần tử. Hàm trả về 0 nếu dãy a ko tăng, 1 nếu ngược lại.

Viết chương trình (hàm main) nhập mảng **a** gồm **n** số thực và mảng b gồm m số thực. Với **n, m** là các số nguyên dương nhập từ bàn phím.

Kiểm tra phần tử mảng a, b có phải là một dãy số tăng hay không?

**Input Format**

Dữ liệu vào nằm trên 4 dòng:

* Dòng 1: một số nguyên dương chỉ số phần từ mảng a.
* Dòng 2: các phần tử mảng a, số thực, cách nhau bởi dấu cách.
* Dòng 3: một số nguyên dương chỉ số phần từ mảng b.
* Dòng 4: các phần tử mảng b, số thực, cách nhau bởi dấu cách.

**Constraints**

* Mảng nhập vào có ít nhất 1 phần tử và nhiều nhất là 500 phần tử. (1<=n, m<=500)
* Kiểu các phần tử mảng là số thực

**Output Format**

**Hai dòng: thông báo kết quả của mảng a và b tương ứng**

* Nếu dãy số tăng, in ra **TANG**
* Ngược lại, in ra **KHONG\_TANG**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  2.00 4.50 6.12 8.30 10.99  5  2.00 4.50 6.15 10.95 8.00 | TANG  KHONG\_TANG |
| 5  2.00 4.50 6.15 10.95 8.00  5  2.00 4.50 6.12 8.30 10.99 | KHONG\_TANG  TANG |

def kiem\_tra\_tang(mang, n):

for i in range(1, n):

if mang[i] <= mang[i - 1]:

return "KHONG\_TANG"

return "TANG"

def main():

# Nhập số phần tử và mảng a

n = int(input())

mang\_a = list(map(float, input().split()))

# Nhập số phần tử và mảng b

m = int(input())

mang\_b = list(map(float, input().split()))

# Kiểm tra và in kết quả

ket\_qua\_a = kiem\_tra\_tang(mang\_a, n)

ket\_qua\_b = kiem\_tra\_tang(mang\_b, m)

print(ket\_qua\_a)

print(ket\_qua\_b)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tổng 2 ma trận đó và in kết quả ra màn hình. Nếu không tính được tổng 2 ma trận thì in ra thông báo "Du lieu vao sai"

Input:

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

Output:

+ Nếu tính được tổng:

- Dòng thứ nhất in ra thông báo "Ma tran tong"

- Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tổng:

- In ra thông báo "Du lieu vao sai"

Constranins: 1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 8  9 0 19 24 3 8 12 14  5 20 6 2 11 16 20 17  2 8  2 11 16 4 2 3 17 7  21 16 18 20 22 1 21 13 | Ma tran tong  11 11 35 28 5 11 29 21  26 36 24 22 33 17 41 30 |
| 5 6  10 6 16 1 7 1  12 17 21 10 16 23  12 9 9 4 22 7  23 0 13 23 17 4  9 5 8 0 23 4  6 3  23 6 0  13 16 5  5 17 12  11 2 24  3 2 18  21 23 15 | Du lieu vao sai |

n1, m1 = map(int, input().split())

x = []

for i in range(n1):

row = list(map(int, input().split()))

x.append(row)

n2, m2 = map(int, input().split())

y = []

for i in range(n2):

row = list(map(int, input().split()))

y.append(row)

if n1 != n2 or m1 != m2:

print("Du lieu vao sai")

else:

s = [[0] \* m1 for \_ in range(n1)]

for i in range(n1):

for j in range(m1):

s[i][j] = x[i][j] + y[i][j]

print("Ma tran tong")

for i in range(n1):

for j in range(m1):

print(s[i][j], end=" ")

print()

Viết chương trình nhập vào danh sách gồm n hình tròn trong mặt phẳng. Sắp xếp giảm dần danh sách các hình tròn theo diện tích. In danh sách đã sắp xếp ra màn hình. Biết thông tin của hình tròn gồm mã hình tròn, bán kính, tọa độ tâm (x, y).

Input:

+ Dòng thứ nhất nhập vào số nguyên dương N

+ Dòng tiếp theo nhập vào thông tin của từng hình tròn. Thứ tự nhập vào là mã, tâm x, tâm y, bán kính,

Output:

+ Dòng thứ nhất in ra thông báo "Danh sach hinh tron da sap xep: N". Với N là số lượng các hình tròn.

+ Các dòng tiếp theo in ra thông tin các hình tròn. Thứ tự in ra là mã, tọa độ x, tọa độ y, bán kính. Các thông tin cách nhau dấu cách. Giá trị số thực có độ chính xác 3 chữ số thập phân.

Constrains:

+ 1<=N<=1000

+ bán kính là số thực, tọa độ tâm, mã hình tròn là các số nguyên

+ Số pi =3.14159

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  78 62 64 19.667  5 81 27 15.333  61 95 42 30.667  27 91 4 12.333 | Danh sach hinh tron da sap xep: 4  61 95 42 30.667  78 62 64 19.667  5 81 27 15.333  27 91 4 12.333 |

class HT:

def \_\_init\_\_(self, ma, x, y, r):

self.ma = ma

self.x = x

self.y = y

self.r = r

def DienTich(self):

return self.r \* 3.14159

n = int(input())

m = []

for i in range(n):

infor = input().split()

ma = int(infor[0])

x = int(infor[1])

y = int(infor[2])

r = float(infor[3])

t = HT(ma, x, y, r)

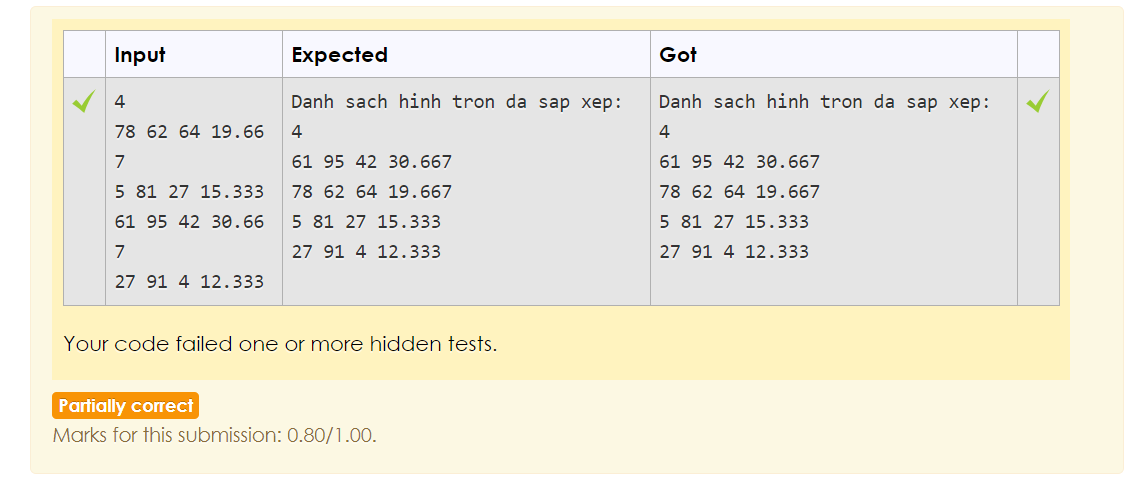
m.append(t)

m.sort(key=lambda ht: ht.DienTich(), reverse=True)

print("Danh sach hinh tron da sap xep:",n)

for hinhtron in m:

print('%d %d %d %.3f'%(hinhtron.ma,hinhtron.x,hinhtron.y,hinhtron.r))



Viết chương trình nhập vào một danh sách gồm n nhân viên. Tìm nhân viên có lương lớn nhất. Nếu có nhiều người có lương lớn nhất bằng nhau thì kết quả là người ở đầu danh sách. Biết thông tin của một nhân viên gồm: tên, mã nv, hệ số lương, phụ cấp

Input:

+ Dòng thứ nhất nhập vào n là số nhân viên

+ Dòng tiếp theo nhập vào thông tin của các nhân viên theo thứ tự tên, mã, hệ số lương, phụ cấp.

Output:

+ Dòng thứ nhất in ra thông báo "Nhan vien co luong lon nhat"

+ Dòng thứ hai in ra thông tin của nhân viên tìm được, các thông tin cách nhau dấu cách. Thứ tự mã nhân viên, tên nhân viên, hệ số lương, phụ cấp, lương tháng in ở cuối mỗi dòng. Lương tháng được tính bằng hệ số lương \* 2000000 + phụ cấp.

Constrains: 1<=n<=200, mã nhân viên, phụ cấp là các số nguyên dương, tên nhân viên không chứa dấu cách, các thông tin còn lại là số thực có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  Anh 78 2.34 1000000  Vinh 5 4.32 250000  Nhung 61 2.67 700000  Trang 27 4.32 250000 | Nhan vien co luong lon nhat  5 Vinh 4.32 250000 8890000.00 |

class NV:

def \_\_init\_\_(self, ten, ma, hsl, pc):

self.ten = ten

self.ma = ma

self.hsl = hsl

self.pc = pc

def Luong(self):

return self.hsl \* 2000000 + self.pc

n = int(input())

m = []

for i in range(n):

infor = input().split()

ten = infor[0]

ma = int(infor[1])

hsl = float(infor[2])

pc = int(infor[3])

t = NV(ten, ma, hsl, pc)

m.append(t)

nv\_luong\_max = max(m, key=lambda nv:nv.Luong())

print("Nhan vien co luong lon nhat")

print(nv\_luong\_max.ma, nv\_luong\_max.ten, format(nv\_luong\_max.hsl, ".2f"), nv\_luong\_max.pc, format(nv\_luong\_max.Luong(), ".2f"))

Công ty điện lực HP quản lý 3 loại khách hàng. Khách hàng là doanh nghiệp, khách hàng là hộ kinh doanh và khách hàng thông thường. Mỗi khách hàng được công ty lưu trữ với các thông tin sau: mã khách hàng, tên khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ.

+ tên  hàng không chứa dấu cách,

+ mã khách hàng là chuỗi ký tự, ký tự đầu là các chữ cái hoa D: cho doanh nghiệp, hoặc K: cho hộ kinh doanh, hoặc H: cho khách hàng thường.

Viết chương trình nhập vào một danh sách gồm n khách hàng.

In thông tin khách hàng là hộ kinh doanh phải trả nhiều tiền nhất. Nếu có nhiều khách hàng phải trả nhiều tiền nhất thì in ra khách hàng có số thứ tự đầu tiên tìm thấy. Nếu ko có khách hàng là hộ kinh doanh trong danh sách thì in ra thông báo "Khong co khach hang la ho kinh doanh."

Input:

+ Dòng thứ nhất nhập vào n là số khách hàng

+ Dòng tiếp theo nhập vào thông tin của các khách hàng theo thứ tự tên, mã số khách hàng, chỉ số đầu kỳ, chỉ số cuối kỳ.

Output:

+ In ra tên, mã khách, hàng chỉ số đầu kỳ, chỉ số cuối kỳ, lượng điện tiêu thụ, số tiền phải trả của khách hàng tìm được hoặc thông báo "Khong co khach hang la ho kinh doanh." Các thông tin in ra cách nhau dấu cách.

Constrains:

+ Lượng điện tiêu thụ = chỉ số cuối kỳ - chỉ số đầu kỳ.

+ Đơn giá là 1000 đồng/1kw cho khách hàng thông thường (với khách hàng là doanh nghiệp thì tăng đơn giá 5%, với khách hàng là hộ kinh doanh thì tăng đơn giá 7%):

+ 1<=n<=200,

+ các thông tin còn lại là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  ` | Trang K99 43 643 600 642000 |

class KhachHang:

def \_\_init\_\_(self, ten, ma\_kh, chi\_so\_dau\_ky, chi\_so\_cuoi\_ky):

self.ten = ten

self.ma\_kh = ma\_kh

self.chi\_so\_dau\_ky = chi\_so\_dau\_ky

self.chi\_so\_cuoi\_ky = chi\_so\_cuoi\_ky

def luong\_dien\_tieu\_thu(self):

return self.chi\_so\_cuoi\_ky - self.chi\_so\_dau\_ky

def tinh\_tien\_phai\_tra(self):

don\_gia\_thuong = 1000

if self.ma\_kh[0] == 'D':

don\_gia\_thuong \*= 1.05

elif self.ma\_kh[0] == 'K':

don\_gia\_thuong \*= 1.07

return int(self.luong\_dien\_tieu\_thu() \* don\_gia\_thuong)

def main():

n = int(input())

danh\_sach\_khach\_hang = []

for \_ in range(n):

thong\_tin\_kh = input().split()

ten = thong\_tin\_kh[0]

ma\_kh = thong\_tin\_kh[1]

chi\_so\_dau\_ky = int(thong\_tin\_kh[2])

chi\_so\_cuoi\_ky = int(thong\_tin\_kh[3])

khach\_hang = KhachHang(ten, ma\_kh, chi\_so\_dau\_ky, chi\_so\_cuoi\_ky)

danh\_sach\_khach\_hang.append(khach\_hang)

khach\_hang\_ho\_kinh\_doanh = [kh for kh in danh\_sach\_khach\_hang if kh.ma\_kh[0] == 'K']

if not khach\_hang\_ho\_kinh\_doanh:

print("Khong co khach hang la ho kinh doanh.")

else:

khach\_hang\_ho\_kinh\_doanh.sort(key=lambda x: x.tinh\_tien\_phai\_tra(), reverse=True)

khach\_hang\_max\_tien = khach\_hang\_ho\_kinh\_doanh[0]

print("{} {} {} {} {} {}".format(

khach\_hang\_max\_tien.ten,

khach\_hang\_max\_tien.ma\_kh,

khach\_hang\_max\_tien.chi\_so\_dau\_ky,

khach\_hang\_max\_tien.chi\_so\_cuoi\_ky,

khach\_hang\_max\_tien.luong\_dien\_tieu\_thu(),

khach\_hang\_max\_tien.tinh\_tien\_phai\_tra()

))

if \_\_name\_\_ == "\_\_main\_\_":

main()

Viết chương trình nhập vào một danh sách gồm n thí sinh thi đại học. In thông tin thí sinh thủ khoa (thi đỗ và có tổng điểm cao nhất) ra màn hình.

   +Nếu có nhiều thủ khoa thì in ra thủ khoa có số thứ tự nhỏ nhất trong danh sách sách.

   + Nếu không có thủ khoa (mọi thí sinh đều trượt) thì thông báo "Khong co thi sinh thi do"

   + Biết thông tin của một thí sinh gồm: tên, số báo danh, điểm môn 1, điểm môn 2, điểm môn 3.

   + Thí sinh thi đỗ khi tổng điểm >= 15.00 và điểm các môn >=1.00.

Input:

+ Dòng thứ nhất nhập vào n là số thí sinh

+ Dòng tiếp theo nhập vào thông tin của các thí sinh theo thứ tự tên, số báo danh, điểm môn 1, điểm môn 2, điểm môn 3.

Output:

Nếu tìm được thủ khoa:

+ Dòng thứ nhất in ra thông báo "So thu tu cua thu khoa: X", X là số thự tự của thí sinh đỗ thủ khoa tìm được theo yêu cầu.

+ Dòng thứ hai in ra thông tin của thí sinh thủ khoa tìm được. Các thông tin cách nhau dấu cách (thứ tự: số báo danh, tên, điểm môn 1, điểm môn 2, điểm môn 3), tổng điểm in ở cuối mỗi dòng.

Nếu KHÔNG tìm được thủ khoa:

+ In ra thông báo "Khong co thi sinh thi do"

Constrains: 1<=n<=200, điểm các môn và tổng điểm có độ chính xác 2 chữ số thập phân, số báo danh là số nguyên dương, tên thí sinh không chứa dấu cách.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  Huy 87 5.00 3.00 3.00  Trang 76 9.00 9.00 9.00  Lan 2 3.00 2.00 3.00  Anh 25 8.00 9.00 10.00  Van 35 0.00 10.00 9.00 | So thu tu cua thu khoa: 1  76 Trang 9.00 9.00 9.00 27.00 |

class ThiSinh:

def \_\_init\_\_(self, ten, sbd, d1, d2, d3):

self.ten = ten

self.sbd = sbd

self.d1 = d1

self.d2 = d2

self.d3 = d2

def Tong\_Diem(self):

return self.d1 + self.d2 + self.d3

n = int(input())

danh\_sach = []

for \_ in range(n):

thong\_tin = input().split()

ten = thong\_tin[0]

sbd = thong\_tin[1]

d1 = float(thong\_tin[2])

d2 = float(thong\_tin[3])

d3 = float(thong\_tin[4])

thi\_sinh = ThiSinh(ten,sbd,d1,d2,d3)

danh\_sach.append(thi\_sinh)

#Kiem tra thi sinh thi do

thi\_sinh\_do = [thi\_sinh for thi\_sinh in danh\_sach if thi\_sinh.Tong\_Diem()>=15 and thi\_sinh.d1 >= 1 and thi\_sinh.d2 >= 1 and thi\_sinh.d3 >= 1]

if not thi\_sinh\_do:

print("Khong co thi sinh thi do")

else:

thu\_khoa = max(danh\_sach,key=lambda sv:sv.Tong\_Diem())

STT = danh\_sach.index(thu\_khoa)

print("So thu tu cua thu khoa:",STT)

print("{} {} {:.2f} {:.2f} {:.2f} {}".format(

thu\_khoa.sbd,

thu\_khoa.ten,

thu\_khoa.d1,

thu\_khoa.d2,

thu\_khoa.d3,

thu\_khoa.Tong\_Diem()

))