Chương 0 - Beautiful, Idiomatic Python

0.1) number separation

In [32]:

```
num1 =  10_000_000_000
num2 = 100_000_000

total = num1 + num2

print( f"{total}")
print( f"{total:,}")
```

```
10100000000
10,100,000,000
```

0.2) context manager

In [38]:

```
#-----------BAD WAY----------
f = open('song-xuan-quynh.txt', 'r', encoding='utf8')
file_contents = f.read()
f.close()

words = file_contents.split()
word_count = len(words)

print(word_count)

#-----------GOOD WAY: with context manager, don't have to close file manually
with open('song-xuan-quynh.txt', 'r', encoding='utf8') as f:
    file_contents = f.read()

words = file_contents.split()
word_count = len(words)

print(word_count)
```

```
191
191
```

0.3) enumerate

In [39]:

```
names = ['Corey', 'Chris', 'Dave', 'Travis']

#-----------BAD WAY-------------
index = 1
for name in names:
    print(index, name)
    index += 1

#-----------GOOD WAY-------------
for index, name in enumerate(names, start=1):
    print(index, name)
```

```
1 Corey
2 Chris
3 Dave
4 Travis
1 Corey
2 Chris
3 Dave
4 Travis
```

0.4) zip, izip, create dict from zip

In [67]:

```
names = ['Peter Parker', 'Clark Kent', 'Wade Wilson', 'Bruce Wayne']
heroes = ['Spiderman', 'Superman', 'Deadpool', 'Batman']
universes = ['Marvel', 'DC', 'Marvel', 'DC']

#----------BAD WAY-------------
for i in range(len(names)):
    name = names[i]
    hero = heroes[i]
    universe = universes[i]
    print(name, hero, universe)
print()

#----------GOOD WAY-------------
for name, hero, universe in zip(names, heroes, universes):
    print(f"{name} is actually {hero} from {universe}")
print()

#-----CONSTRUCT A DICT FROM PAIRS-----
d = dict(zip(names, heroes))
print(d)

d = dict(enumerate(names))
print(d)
```

```
Peter Parker Spiderman Marvel
Clark Kent Superman DC
Wade Wilson Deadpool Marvel
Bruce Wayne Batman DC

Peter Parker is actually Spiderman from Marvel
Clark Kent is actually Superman from DC
Wade Wilson is actually Deadpool from Marvel
Bruce Wayne is actually Batman from DC

{'Peter Parker': 'Spiderman', 'Clark Kent': 'Superman', 'Wade Wilson': 'Deadpool', 'Bruce Wayne': 'B
atman'}
{0: 'Peter Parker', 1: 'Clark Kent', 2: 'Wade Wilson', 3: 'Bruce Wayne'}
```

0.5) unpack and use all values in a tuple

In [47]:

```
a, b = (1, 2)
print(a)
print(b)
print()

# unpack and use only one value in a tuple
a, _ = (3, 4)
print(a)
print()

# unpack the rest to one variable
a, b, *c = (5, 6, 7, 8, 9)
print(a)
print(b)
print(c)
print()

# unpack
a, b, *_ = (10, 11, 12, 13, 14)
print(a)
print(b)
print()
```

```
1
2

3

5
6
[7, 8, 9]

10
11
```

0.6) input secret information

In [49]:

```
from getpass import getpass

username = input( "Username: " )
username = getpass( "Password: " )

print( "Logged in" )
```

```
Username: vimaru
Password: ········
Logged in
```

0.7) list comprehension

In [50]:

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

squares = [num*num for num in my_list]

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

0.8) dict's get

In [68]:

```
ages = {
    'Mary': 31,
    'Jonathan': 28
}

#-----------THE BAD WAY------------
if 'Nick' in ages:
    age = ages['Nick']
else:
    age = 'Unknown'
print(f"Nick is {age} years old")

#-----------THE GOOD WAY------------
age = ages.get('Nick', 'Unknown')
print(f"Nick is {age} years old")
```

```
Nick is Unknown years old
Nick is Unknown years old
```

0.9) custom sort order

In [55]:

```
colors = ['red', 'green', 'blue', 'yellow']

print('Sorted by value')
print(sorted(colors))

print('\nSorted by len')
print(sorted(colors, key=len))
```

```
Sorted by value
['blue', 'green', 'red', 'yellow']

Sorted by len
['red', 'blue', 'green', 'yellow']
```

0.10) distinguishing multiple exit points in loops

In [69]:

```
def find(seq, target):
    found = False
    for i, value in enumerate(seq):
        if value == target:
            found = True
            break
    if not found:
        return -1
    return i

def better_find(seq, target):
    for i, value in enumerate(seq):
        if value == target:
            break
    else:
        return -1
    return i

l = [i for i in range(5)]
print(find(l, 0))
print(better_find(l, 4))
```

```
0
4
```

0.11) looping over dictionary keys

In [58]:

```
d = {'matthew': 'blue', 'rachel': 'green', 'raymond': 'red'}

#--------THE BAD WAY-----------
#RuntimeError: dictionary changed size during iteration
#for k in d.keys():
#       if k.startswith('r'):
#               del d[k]

#--------THE GOOD WAY-----------
d = {k:d[k] for k in d if not k.startswith('r')}
print(d)
```

```
{'matthew': 'blue'}
```

0.12) counting, grouping with dict

In [70]:

```
colors = ['red', 'green', 'red', 'blue', 'green', 'red']

#------COUNTING WITH DICT------
d = {}
for color in colors:
    if color not in d:
        d[color] = 0
    d[color] += 1

print(d)

#------better counting with dict------
d = {}
for color in colors:
    d[color] = d.get(color, 0) + 1

print(d)
print()

#-----GROUPING NAMES BY THEIR LEN
names = ['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith', 'charlie']

#-----the bad way-----
d = {}
for name in names:
    key = len(name)
    if key not in d:
        d[key] = []
    d[key].append(name)

print(d)

#-----the better way-----
d = {}
for name in names:
    key = len(name)
    d.setdefault(key, []).append(name)

print(d)
```

```
{'red': 3, 'green': 2, 'blue': 1}
{'red': 3, 'green': 2, 'blue': 1}

{7: ['raymond', 'matthew', 'melissa', 'charlie'], 6: ['rachel', 'judith'], 5: ['roger', 'betty']}
{7: ['raymond', 'matthew', 'melissa', 'charlie'], 6: ['rachel', 'judith'], 5: ['roger', 'betty']}
```

0.14) UPDATING MULTIPLE STATE VARIABLES

In [72]:

```
#-----the bad way-----
def fibonacci(n):
    x = 0
    y = 1
    for i in range(n):
        print(x, end=' ')
        t = y
        y = x + y
        x = t

#-----the better way-----
def fibonacci_better(n):
    x, y = 0, 1
    for i in range(n):
        print(x, end=' ')
        x, y = y, x+y

fibonacci(10)
print()
fibonacci_better(10)
```

```
0 1 1 2 3 5 8 13 21 34
0 1 1 2 3 5 8 13 21 34
```

0.15) CONCATENATING STRINGS & UPDATING SEQUENCES

In [73]:

```
from collections import deque
names = ['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith', 'charlie']

# CONCATENATING STRINGS: the bad way-----
s = names[0]
for name in names[1:]:
    s += ', ' + name
print(s)

# CONCATENATING STRINGS: the better way-----
s = ', '.join(names)
print(s)
print()

# UPDATING SEQUENCES: the bad way-----
names = ['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith', 'charlie']
del names[0]
names.pop()
names.insert(0, 'mark')
print(names)

# UPDATING SEQUENCES: the better way-----
names = deque(['raymond', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith', 'charlie'])
del names[0]
names.popleft()
names.appendleft('mark')
print(names)
```

```
raymond, rachel, matthew, roger, betty, melissa, judith, charlie
raymond, rachel, matthew, roger, betty, melissa, judith, charlie

['mark', 'rachel', 'matthew', 'roger', 'betty', 'melissa', 'judith']
deque(['mark', 'matthew', 'roger', 'betty', 'melissa', 'judith', 'charlie'])
```

Chương 1 và 2

Câu 1) Giải và biện luận phương trình bậc nhất hai ẩn.

$$\begin{cases} a_1x+b_1y=c_1 \\ a_2x+b_2y=c_2 \end{cases}$$

Các hệ số của hệ phương trình là các số thực được nhập từ bàn phím. In kết quả ra màn hình theo quy tắc: nếu có nghiệm thì các giá trị nghiệm in trên một dòng cách nhau dấu cách, nếu vô số nghiệm thì in ra màn hình thông báo "Vo so nghiem", nếu vô nghiệm thì in ra màn hình thông báo "Vo nghiem".

In [74]:

```
a1 = float(input( 'a1=' ))
b1 = float(input( 'b1=' ))
c1 = float(input( 'c1=' ))

a2 = float(input( 'a2=' ))
b2 = float(input( 'b2=' ))
c2 = float(input( 'c2=' ))

T = a1*b2 - b1*a2;    Tx = c1*b2 - b1*c2;    Ty = a1*c2 - c1*a2;

if( T == 0 ):
    if( Tx == 0 ):
        print( '\nHệ pt có vô số nghiệm.' )
    else:
        print( '\nHệ pt vô nghiệm.' )

else:
    print( '\nHệ pt có nghiệm: ' )
    print( 'x = %f, y = %f' %(Tx/T, Ty/T ) )
```

```
a1=1
b1=2
c1=3
a2=4
b2=5
c2=6

Hệ pt có nghiệm:
x = -1.000000, y = 2.000000
```

In [ ]:

```
```

Câu 2) Viết chương trình tính tổng s

Viết chương trình tính gần đúng giá trị của hàm số e^x theo công thức:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

. Giá trị được tính cho tới khi $\left|\frac{x^n}{n!}\right| < ep$, với ep và x là các số thực nhập từ bàn phím, $(0 < ep < 1)$. In kết quả ra màn hình.

In [75]:

```
from math import *

def giaithua(n):
    gt=1
    for i in range (1,n+1,1):
        gt*=i
    return gt

s = 1.0
n = 1
ep = float( input( 'ep: ' ) )
x = float( input( 'x: ' ) )

while( abs(x**n) / giaithua(n) >= ep ):
    s += abs(x**n) / giaithua(n)
    n += 1

print(s)
print(str(exp(x)) + " (dùng hàm có sẵn)")
```

```
ep: 0.000001
x: 1
2.7182815255731922
2.718281828459045 (dùng hàm có sẵn)
```

Câu 3

Câu 4) Giải và biện luận phương trình bậc 2

$ax^{2}+bx+c=0$, với a, b, c là các số nguyên được nhập từ bàn phím.

In [76]:

```
from math import sqrt

a = float( input( "a = " ) )
b = float( input( "b = " ) )
c = float( input( "c = " ) )
delta = b*b - 4*a*c

if( delta <0 ):
    print( "PT VO NGHIEM" )

elif(delta ==0):
    print( "PT CO NGHIEM DUY NHAT x = %f" %(-1.0*b/(2*a)) )

else:
    x1 = (-b+sqrt(delta)) / (2*a)
    x2 = (-b-sqrt(delta)) / (2*a)
    print( "x1 = %f, x2 = %f" %(x1, x2) )
```

```
a = 2
b = -5
c = 3
x1 = 1.500000, x2 = 1.000000
```

Câu 5) Kiểm tra tam giác.

Nhập từ bàn phím các số thực x1, x2, x3, y1, y2, y3. Kiểm tra các đỉnh có toạ độ (x1, y1), (x2, y2), (x3, y3) có lập thành một tam giác không. Nếu đúng hãy tính: Chu vi, diện tích của tam giác đó. Nếu sai thông báo cho người dùng biết dữ liệu nhập vào bị sai.

In [4]:

```
from math import sqrt

x1 = float( input( 'Nhập x1: ' ) )
y1 = float( input( 'Nhập y1: ' ) )
x2 = float( input( 'Nhập x2: ' ) )
y2 = float( input( 'Nhập y2: ' ) )
x3 = float( input( 'Nhập x3: ' ) )
y3 = float( input( 'Nhập y3: ' ) )

a = sqrt( (x2-x1)**2 + (y2-y1)**2 )
b = sqrt( (x2-x3)**2 + (y2-y3)**2 )
c = sqrt( (x1-x3)**2 + (y1-y3)**2 )

if( (a+b>c) and (a+c>b) and (b+c>a) ):
    print('\nBa điểm trên tạo thành một tam giác.')
    cv = a + b + c;
    p = cv/2;
    s = sqrt( p*(p-a)*(p-b)*(p-c) );
    print( 'Chu vi: %f; diện tích: %f' %(cv, s) )
else:
    print('Ba điểm trên không tạo thành một tam giác.')
```

```
Nhập x1: 1
Nhập y1: 1
Nhập x2: 2
Nhập y2: 2
Nhập x3: 1
Nhập y3: 2

Ba điểm trên tạo thành một tam giác.
Chu vi: 3.414214; diện tích: 0.500000
```

Câu 6) Tính tiền đi taxi từ số km đã được nhập vào, biết:

- 1 km đầu giá 15000đ
- Từ km thứ 2 đến km thứ 5 giá 13500đ
- Từ km thứ 6 trở đi giá 11000đ
- Nếu đi hơn 120km sẽ được giảm 10% trên tổng số tiền.

In [5]:

```
km = int( input( 'Nhập số km đã đi: ' ) )

if( km<1 ):
    print( 'Số km nhập vào không hợp lệ.' )

else:
    if( km>=1 and km<=5 ):
        tien = 15000 + (km-1)*13500

    elif( km>=6 and km<=120 ):
        tien = 1*15000 + 4*13500 + (km-5)*11000

    else:
        tien = 1*15000 + 4*13500 + (km-5)*11000
        tien = tien*0.9

    print('Số tiền phải trả: %d' %(tien) )
```

```
Nhập số km đã đi: 200
Số tiền phải trả: 1992600
```

Chương 3

Câu 7

Viết hàm tính ước số chung lớn nhất và bội số chung nhỏ nhất của hai số nguyên dương a,b

In [8]:

```
def ucln(a, b):
    if( a<=0 or b<=0 ):
        return 0

    while a!=b:
        if a>b:
            a-=b
        else:
            b-=a

    return a

# test cases
a =  [7, 3, 12, 17, 17, 0]
b =  [8, 9, 24, 11, 11, 121122]
kq = [2, 3, 12, 1,  1,  0]

# unit test
for i in range( len(a) ):
    if ucln(a[i], b[i]) != kq[i]:
        print( f'Sai với bộ dữ liệu {i}' )
```

```
Sai với bộ dữ liệu 0
```

Câu 8

Viết hàm để nhận biết một số nguyên dương có phải là số nguyên tố hay không. Số nguyên tố là số chỉ chia hết cho 1 và chính nó.

In [78]:

```
def is_prime(n):
    if n<=1 :
        return False
    for  i in range (2 , int(n/2)+1):
        if (n%i==0):
            return False
    return True

# test cases
n  = [1,     2,    4 ,    5 ,   99,   107,  10005]
kq = [False, True, False, True, False,True, False]

# unit test
for i in range( len(n) ):
    if( is_prime( n[i] ) != kq[i]):
        print( 'Sai với bộ dữ liệu ' + str(i) )
```

In [ ]:

```

```

Chương 4 và 5

Câu 12

Phát sinh ngẫu nhiên toạ độ hai véc tơ X = (x1, x2,..., xn), Y = (y1, y2,...yn), trong đó n là số tự nhiên nhỏ hơn 15 được nhập từ bàn phím.

Tính độ dài véctơ X theo công thức ||X|| = $\sqrt{x1^2 + x2^2 + x3^2 + ... + xn^2}$

Tìm tổng véctơ Z = X + Y (Z=(z1, z2, ...zn)) theo công thức zi = xi+yi (i = 1..n).

In [74]:

```
from math import sqrt
import random

n = int(input('Nhập n: ' ))
while n > 15 or n < 1:
    n = int(input('Nhập n: ' ))

#-----the C way-----
x=[]
y=[]
z=[]

do_dai_x = 0
for i in range (n):
    x.append( random.randint(1,30) )
    y.append( random.randint(1,30) )
    z.append( x[i]+y[i] )
    do_dai_x += x[i]**2

do_dai_x = sqrt(do_dai_x)
print("\nVector X")
print(x)
print("\nVector Y")
print(y)
print("\nVector Z=X+Y")
print(z)
print(f"\nDo dai vector x: {do_dai_x}")

#-----the Python way-----
x = [random.randint(1,30) for i in range(n)]
y = [random.randint(1,30) for i in range(n)]
z = [xi+yi for xi,yi in zip(x, y)]
do_dai_x = sqrt(sum([xi*xi for xi in x]))

print("\nVector X")
print(x)
print("\nVector Y")
print(y)
print("\nVector Z=X+Y")
print(z)
print(f"\nDo dai vector x: {do_dai_x}")
```

```
Nhập n: 5

Vector X
[5, 25, 8, 22, 10]

Vector Y
[19, 10, 11, 8, 18]

Vector Z=X+Y
[24, 35, 19, 30, 28]

Do dai vector x: 36.02776706930364

Vector X
[8, 18, 21, 6, 17]

Vector Y
[25, 4, 23, 1, 1]

Vector Z=X+Y
[33, 22, 44, 7, 18]

Do dai vector x: 33.97057550292606
```

Câu 13

Viết chương trình nhập vào một mảng gồm n (n<100) số nguyên. Hãy in ra:

a) Các số nguyên khác nhau trong mảng

b) Số nguyên xuất hiện nhiều lần nhất trong mảng

In [28]:

```
import random

n = int(input("Nhập số phần tử: "))
while n<1 or n>100:
    n = int(input("Nhập số phần tử: "))

#-----the C way-----
a = []
for i in range(n):
    #ai = int(input(f'a[{i}]='))
    ai = random.randint(0, 20)
    a.append(ai)

# a)
b = []
for ai in a:
    if not ai in b:
        b.append(ai)

print('List A')
print(a)
print('\nCác số khác nhau trong A')
print(b)

# b)
d = {}
for ai in a:
    if not ai in d.keys():
        d[ai] = 1
    else:
        d[ai] += 1

print('\nSố lần xuất hiện của từng phần tử')
print(d)

max_count = 0
for bi in b:
    if max_count < d[bi]:
        max_count = d[bi]
        pt_xhn = bi    # pt xuất hiện nhiều

print('\nPhần tử xuất hiện nhiều nhất')
print(pt_xhn)

#-----the PYthon way-----
a = [random.randint(0, 20) for i in range(n)]

# a)
b = set(a)
print('\nList A')
print(a)
print('\nCác số khác nhau trong A')
print(b)

# b)
d = {}
for ai in a:
    d[ai] = d.get(ai, 0) + 1
sorted_d = {k: v for k, v in sorted(d.items(), key=lambda item: item[1])}

print('\nSố lần xuất hiện của từng phần tử')
print(sorted_d)

print('\nPhần tử xuất hiện nhiều nhất')
print(list(sorted_d)[-1])
```

```
Nhập số phần tử: 5
List A
[7, 17, 19, 16, 12]

Các số khác nhau trong A
[7, 17, 19, 16, 12]

Số lần xuất hiện của từng phần tử
{7: 1, 17: 1, 19: 1, 16: 1, 12: 1}

Phần tử xuất hiện nhiều nhất
7

List A
[20, 19, 17, 19, 14]

Các số khác nhau trong A
{17, 19, 20, 14}

Số lần xuất hiện của từng phần tử
{20: 1, 17: 1, 14: 1, 19: 2}

Phần tử xuất hiện nhiều nhất
19
```

Câu 14)

Nhập vào một dãy số gồm n phần tử nguyên, n<100. In dãy số ban đầu và các số thuộc đoạn [a, b] ra màn hình. Trong đó: a, b là hai số nguyên nhập từ bàn phím.

In [19]:

```python
import random

n = int(input("Nhập số phần tử: "))
while n<1 or n>100:
    n = int(input("Nhập số phần tử: "))

lst = [random.randint(0, 20) for i in range(n)]
print('\nRandom list A')
print(lst)

a = int(input("\nNhập a: "))
b = int(input("Nhập b: "))

print(f"\nCác pt thuộc đoạn [{a}, {b}]")

#-----the C way-----
for i in range(n):
    if a <= lst[i] <= b:
        print(lst[i], end=' ')
print()

#-----the Python way-----
b = [ai for ai in lst if a<=ai<=b]
print(b)
```

```
Nhập số phần tử: 5

Random list A
[9, 19, 5, 10, 1]

Nhập a: 1
Nhập b: 10

Các pt thuộc đoạn [1, 10]
9 5 10 1
[9, 5, 10, 1]
```

Câu 16

Nhập vào một ma trận gồm n hàng, m cột (n và m <100). In ma trận đó ra màn hình. In ra phần tử lớn nhất của ma trận.

In [2]:

```
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))

while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

matrix = [[random.randint(0, 20) for j in range(m)] for i in range(n)]

print('\nMa trận ngẫu nhiên')
for row in matrix:
    print(row, end=' => ')
    print(max(row))

print('\nPhần tử lớn nhất của ma trận')

# -----the C way
max_value = matrix[0][0]
for i in range(n):
    for j in range(m):
        if max_value < matrix[i][j]:
            max_value = matrix[i][j]
print(max_value)

# -----the Python way
max_value = max([max(row) for row in matrix])
print(max_value)

# -----the Python way 2
max_value = max(map(max, matrix))
print(max_value)
```

```
Nhập số hàng: 5
Nhập số cột: 6

Ma trận ngẫu nhiên
[11, 4, 6, 13, 9, 12] => 13
[12, 5, 7, 10, 3, 12] => 12
[1, 7, 10, 15, 13, 13] => 15
[0, 14, 9, 7, 10, 20] => 20
[2, 19, 8, 10, 12, 4] => 19

Phần tử lớn nhất của ma trận
20
20
20
```

Câu 18

Nhập vào một ma trận gồm n hàng, m cột (n và m <100). In ra các phần tử lớn nhất trong mỗi hàng của ma trận

In [3]:

```python
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))

while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

matrix = [[random.randint(0, 20) for j in range(m)] for i in range(n)]

print('\nMa trận ngẫu nhiên')
for row in matrix:
    print(row)
print()

#-----the C way
for i in range(n):
    max_row_i = matrix[i][0]
    for j in range(1, m):
        if max_row_i < matrix[i][j]:
            max_row_i = matrix[i][j]
    print(f"Pt lớn nhất hàng {i} là {max_row_i}")
print()

#-----the Python way
for i in range(n):
    max_row_i = max(matrix[i])
    print(f"Pt lớn nhất hàng {i} là {max_row_i}")
```

```
Nhập số hàng: 5
Nhập số cột: 6

Ma trận ngẫu nhiên
[2, 1, 18, 2, 14, 2]
[10, 3, 6, 8, 16, 0]
[20, 4, 1, 10, 5, 11]
[13, 20, 17, 14, 0, 1]
[4, 18, 9, 14, 10, 18]

Pt lớn nhất hàng 0 là 18
Pt lớn nhất hàng 1 là 16
Pt lớn nhất hàng 2 là 20
Pt lớn nhất hàng 3 là 20
Pt lớn nhất hàng 4 là 18

Pt lớn nhất hàng 0 là 18
Pt lớn nhất hàng 1 là 16
Pt lớn nhất hàng 2 là 20
Pt lớn nhất hàng 3 là 20
Pt lớn nhất hàng 4 là 18
```

Câu 19

Nhập vào một mảng ma trận gồm n hàng, m cột (n và m <100). In ra vị trí và giá trị của phần tử nhỏ nhất trong ma trận (nếu có nhiều hơn một phần tử nhỏ nhất thì chỉ cần in ra một trong số đó).

In [81]:

```
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))

while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

matrix = [[random.randint(0, 20) for j in range(m)] for i in range(n)]

print('\nMa trận ngẫu nhiên')
for row in matrix:
    print(row)
print()

#-----the C way
min_i = 0
min_j = 0
min_value = matrix[min_i][min_j]

for i in range(n):
    for j in range(m):
        if min_value > matrix[i][j]:
            min_i = i
            min_j = j
            min_value = matrix[min_i][min_j]

print(f"Pt nhỏ nhất là {min_value}, tại vị trí ({min_i}, {min_j})")

#-----the Python way
min_value = min([min(row) for row in matrix])
for i in range(n):
    if min_value in matrix[i]:
        min_i = i
        min_j = matrix[i].index(min_value)
        break
print(f"Pt nhỏ nhất là {min_value}, tại vị trí ({min_i}, {min_j})")

#-----the Python way 2
min_value = min([min(row) for row in matrix])
min_i, min_j = [(index, row.index(min_value)) for index, row in enumerate(matrix) if min_value in row][0]
print(f"Pt nhỏ nhất là {min_value}, tại vị trí ({min_i}, {min_j})")
```

```
Nhập số hàng: 4
Nhập số cột: 6

Ma trận ngẫu nhiên
[2, 13, 16, 13, 20, 16]
[12, 12, 11, 11, 8, 16]
[11, 12, 19, 2, 20, 11]
[9, 9, 5, 19, 20, 17]

Pt nhỏ nhất là 2, tại vị trí (0, 0)
Pt nhỏ nhất là 2, tại vị trí (0, 0)
Pt nhỏ nhất là 2, tại vị trí (0, 0)
```

Câu 21

Nhập vào từ bàn phím một ma trận vuông n.n phần tử, n<100. In ma trận đó ra màn hình và ghi kết quả vào file “ketqua.txt”.

In [34]:

```
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))
while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

matrix = [[random.randint(0, 20) for j in range(m)] for i in range(n)]

print('\nMa trận ngẫu nhiên')
for row in matrix:
    print(row)
print()

#-----the C way
output = open('ketqua.txt', 'w')
for row in matrix:
    output.write(str(row) + '\n')
output.close()

#-----the Python way: context manager
with open('ketqua.txt', 'w') as output:
    for row in matrix:
        output.write(str(row) + '\n')
# không cần close()
```

```
Nhập số hàng: 3
Nhập số cột: 5

Ma trận ngẫu nhiên
[4, 3, 4, 3, 4]
[7, 18, 1, 3, 6]
[6, 4, 10, 0, 14]
```

Câu 22

Nhập vào từ bàn phím một ma trận vuông n.n phần tử, n<100. In ra vị trí và giá trị của phần tử lớn nhất trên đường chéo chính.

In [5]:

```
import random

n = int(input("Nhập số hàng (cột) của ma trận vuông: "))
while n<1 or n>100:
    n = int(input("Nhập số hàng (cột) của ma trận vuông: "))

#matrix = [[int(input(f"a[{i}][{j}]=")) for j in range(n)] for i in range(n)]
matrix = [[random.randint(0, 20) for j in range(n)] for i in range(n)]

print('\nMa trận vuông ngẫu nhiên')
for row in matrix:
    print(row)
print()

#-----the C way
max_ij = 0
max_value = matrix[max_ij][max_ij]

for i in range(n):
    if max_value < matrix[i][i]:
        max_ij = i
        max_value = matrix[max_ij][max_ij]

print(f"Pt lớn nhất là {max_value}, tại vị trí ({max_ij}, {max_ij})")

#-----the Python way
dcc = [matrix[i][i] for i in range(n)]    # các pt trên đường chéo chính
max_value = max(dcc)
max_ij = dcc.index(max_value)

print(f"Pt lớn nhất là {max_value}, tại vị trí ({max_ij}, {max_ij})")
```

```
Nhập số hàng (cột) của ma trận vuông: 5

Ma trận vuông ngẫu nhiên
[3, 14, 12, 4, 1]
[5, 17, 12, 1, 3]
[19, 8, 9, 17, 14]
[9, 4, 4, 13, 19]
[20, 17, 7, 13, 10]

Pt lớn nhất là 17, tại vị trí (1, 1)
Pt lớn nhất là 17, tại vị trí (1, 1)
```

Câu 23

Nhập vào từ bàn phím một ma trận vuông n.n phần tử, n<100. Sắp xếp các cột của ma trận theo chiều giảm dần.

In [6]:

```
import random

n = int(input("Nhập số hàng (cột) của ma trận vuông: "))

while n<1 or n>100:
    n = int(input("Nhập số hàng (cột) của ma trận vuông: "))

#matrix = [[int(input(f"a[{i}][{j}]")) for j in range(n)] for i in range(n)]
matrix = [[random.randint(0, 20) for j in range(n)] for i in range(n)]
matrix2 = [row[:] for row in matrix]    # copy to matrix2 for Python way

print('\nMa trận vuông ngẫu nhiên')
for row in matrix:
    print(row)

print('\nMa trận vuông sau khi sắp xếp cột giảm')

#-----the C way
for j in range(n):
    for i in range(n-1):
        for k in range(i+1, n):
            if matrix[i][j] < matrix[k][j]:
                tmp = matrix[i][j]
                matrix[i][j] = matrix[k][j]
                matrix[k][j] = tmp
for row in matrix:
    print(row)
print()

#-----the Python way: dùng matrix2
transposed = [list(a_tuple) for a_tuple in zip(*matrix2)]  # ma trận chuyển vị của matrix2
for row in transposed:                                     # sắp xếp các hàng của transpose
    row.sort(reverse = True)
matrix2 = [list(a_tuple) for a_tuple in zip(*transposed)]  # chuyển vị transposed lại
for row in matrix2:
    print(row)
```

```
Nhập số hàng (cột) của ma trận vuông: 5

Ma trận vuông ngẫu nhiên
[16, 2, 2, 3, 9]
[14, 19, 14, 2, 1]
[1, 19, 3, 15, 4]
[7, 6, 7, 19, 5]
[20, 7, 0, 9, 10]

Ma trận vuông sau khi sắp xếp cột giảm
[20, 19, 14, 19, 10]
[16, 19, 7, 15, 9]
[14, 7, 3, 9, 5]
[7, 6, 2, 3, 4]
[1, 2, 0, 2, 1]

[20, 19, 14, 19, 10]
[16, 19, 7, 15, 9]
[14, 7, 3, 9, 5]
[7, 6, 2, 3, 4]
[1, 2, 0, 2, 1]
```

Câu 25

Nhập vào từ bàn phím một ma trận vuông n.n phần tử, n<100. Sắp xếp các hàng của ma trận theo chiều tăng dần.

In [10]:

```
import random

n = int(input("Nhập số hàng (cột) của ma trận vuông: "))
while n<1 or n>100:
    n = int(input("Nhập số hàng (cột) của ma trận vuông: "))

#matrix = [[int(input(f"a[{i}][{j}]")) for j in range(n)] for i in range(n)]
matrix = [[random.randint(0, 20) for j in range(n)] for i in range(n)]

print('\nMa trận vuông ngẫu nhiên')
for row in matrix:
    print(row)

print('\nMa trận vuông sau khi sắp xếp hàng tăng')

#-----the C way
for i in range(n):
    for j in range(n-1):
        for k in range(j+1, n):
            if matrix[i][j] > matrix[i][k]:
                tmp = matrix[i][j]
                matrix[i][j] = matrix[i][k]
                matrix[i][k] = tmp
for row in matrix:
    print(row)
print()

#-----the Python way
for row in matrix:
    row.sort(reverse = False)
    print(row)
```

```
Nhập số hàng (cột) của ma trận vuông: 5

Ma trận vuông ngẫu nhiên
[8, 0, 20, 2, 18]
[8, 5, 7, 9, 20]
[3, 2, 1, 19, 5]
[4, 9, 20, 6, 11]
[5, 12, 9, 20, 15]

Ma trận vuông sau khi sắp xếp hàng tăng
[0, 2, 8, 18, 20]
[5, 7, 8, 9, 20]
[1, 2, 3, 5, 19]
[4, 6, 9, 11, 20]
[5, 9, 12, 15, 20]

[0, 2, 8, 18, 20]
[5, 7, 8, 9, 20]
[1, 2, 3, 5, 19]
[4, 6, 9, 11, 20]
[5, 9, 12, 15, 20]
```

Câu 28

Nhập vào từ bàn phím một ma trận các số thực gồm n hàng, m-1 cột (n và m<100). Tính tổng các phần tử trong từng hàng và lưu vào cột cuối cùng.

In [18]:

```
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))
while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

#matrix = [[int(input(f"a[{i}][{j}]")) for j in range(m-1)] for i in range(n)]
matrix = [[random.randint(0, 20) for j in range(m-1)] for i in range(n)]
matrix2 = [row[:] for row in matrix]    # copy to matrix2 for Python way

print(f'\nMa trận ngẫu nhiên {n}x{m-1}')
for row in matrix:
    print(row)

print(f'\nMa trận sau khi thêm cột tổng {n}x{m}')

#-----the C way
for i in range(n):
    sum_row_i = 0
    for j in range(m-1):
        sum_row_i += matrix[i][j]
    matrix[i].append(sum_row_i)

for row in matrix:
    print(row)
print()

#-----the Python way
for row in matrix2:
    row.append(sum(row))

for row in matrix2:
    print(row)
```

```
Nhập số hàng: 3
Nhập số cột: 5

Ma trận ngẫu nhiên 3x4
[18, 19, 3, 16]
[5, 1, 9, 7]
[4, 7, 19, 13]

Ma trận sau khi thêm cột tổng 3x5
[18, 19, 3, 16, 56]
[5, 1, 9, 7, 22]
[4, 7, 19, 13, 43]

[18, 19, 3, 16, 56]
[5, 1, 9, 7, 22]
[4, 7, 19, 13, 43]
```

Câu 29

Nhập vào từ bàn phím một ma trận các số thực gồm n hàng, m-1 cột (n và m<100). Tìm hàng có tổng lớn nhất.

In [82]:

```
import random

n = int(input("Nhập số hàng: "))
m = int(input("Nhập số cột: "))
while (n<1 or n>100) or (m<1 or m>100):
    n = int(input("Nhập số hàng: "))
    m = int(input("Nhập số cột: "))

#matrix = [[int(input(f"a[{i}][{j}]")) for j in range(m-1)] for i in range(n)]
matrix = [[random.randint(0, 20) for j in range(m-1)] for i in range(n)]

print(f'\nMa trận ngẫu nhiên {n}x{m-1}')
for row in matrix:
    print(row)
print()

#-----the C way
i_max = 0
sum_max = sum(matrix[i_max])
for i in range(1, n):
    sum_row_i = sum(matrix[i])
    if sum_max < sum_row_i:
        sum_max = sum_row_i
        i_max = i
print(f'Hàng {i_max} có tổng lớn nhất, tổng là {sum_max}')

#-----the Python way
sums_of_rows = [sum(row) for row in matrix]
sum_max = max(sums_of_rows)
i_max = sums_of_rows.index(sum_max)
print(f'Hàng {i_max} có tổng lớn nhất, tổng là {sum_max}')
```

```
Nhập số hàng: 3
Nhập số cột: 5

Ma trận ngẫu nhiên 3x4
[7, 14, 4, 11]
[3, 6, 5, 0]
[4, 0, 1, 6]

Hàng 0 có tổng lớn nhất, tổng là 36
Hàng 0 có tổng lớn nhất, tổng là 36
```

Câu 30

Viết chương trình phát sinh ngẫu nhiên mảng a gồm n số nguyên (n<100). In dãy số ra màn hình theo quy tắc các số được in trên một dòng cách nhau dấu phẩy.

In [22]:

```
import random

n = int(input("Nhập số phần tử: "))
while (n<1 or n>100):
    n = int(input("Nhập số phần tử: "))

a = [random.randint(0, 20) for i in range(n)]
print('\nMảng ngẫu nhiên')
print(a)

print('\nIn các số trên 1 dòng, cách nhau dấu phẩy')
for ai in a[:-1]:
    print(ai, end=',')
print(a[-1])
```

```
Nhập số phần tử: 5

Mảng ngẫu nhiên
[13, 5, 10, 4, 8]

In các số trên 1 dòng, cách nhau dấu phẩy
13,5,10,4,8
```

Câu 31

Viết chương trình phát sinh ngẫu nhiên mảng a gồm n số nguyên (n<100). Tính $T=\sqrt{a[0]^2 + a[0]^2 + ... + a[0]^2}$

In [23]:

```
from math import sqrt
from random import randint

n = int(input("Nhập số phần tử: "))
while (n<1 or n>100):
    n = int(input("Nhập số phần tử: "))

a = [randint(0, 20) for i in range(n)]
print('\nMảng ngẫu nhiên')
print(a)

#----the C way
T = 0
for ai in a:
    T += ai**2
T = sqrt(T)
print(f'\nT = {T}')

#----the Python way
T = sqrt(sum([ai*ai for ai in a]))
print(f'\nT = {T}')
```

```
Nhập số phần tử: 5

Mảng ngẫu nhiên
[5, 9, 17, 13, 0]

T = 23.748684174075834

T = 23.748684174075834
```

Câu 32

Viết chương trình phát sinh ngẫu nhiên mảng a gồm n số nguyên (n<100). Tìm vị trí và giá trị phần tử lớn nhất trong dãy (nếu có nhiều phần tử lớn nhất thì lấy phần tử đầu tiên).

In [24]:

```
import random

n = int(input('Nhập số pt n: '))
while n<1 or n>99:
    n = int(input('Nhập số pt n: '))

a = [randint(0, 20) for i in range(n)]
print('\nList ngẫu nhiên')
print(a)

#-----the C way
max_i = 0
max_value = a[max_i]
for i in range(1, n):
    if max_value < a[i]:
        max_i = i
        max_value = a[max_i]
print(f'\nPt lớn nhất là {max_value}, nằm tại vị trí {max_i}')

#-----the Python way
max_value = max(a)
max_i = a.index(max_value)
print(f'\nPt lớn nhất là {max_value}, nằm tại vị trí {max_i}')
```

```
Nhập số pt n: 5

List ngẫu nhiên
[8, 4, 9, 1, 20]

Pt lớn nhất là 20, nằm tại vị trí 4

Pt lớn nhất là 20, nằm tại vị trí 4
```

Câu 33

Viết chương trình phát sinh ngẫu nhiên mảng a gồm n số nguyên (n<100). Tìm số nguyên xuất hiện nhiều nhất trong mảng a.

In [25]:

```
import random

n = int(input('Nhập số pt n: '))
while n<1 or n>99:
    n = int(input('Nhập số pt n: '))

a = [randint(0, 10) for i in range(n)]
print('\nList ngẫu nhiên')
print(a)

#-----the C way: sv tự viết

#-----the Python way
d = {}
for ai in a:
    d[ai] = d.get(ai, 0) + 1

sorted_d = {k: v for k, v in sorted(d.items(), key=lambda item: item[1])}
print('\nThống kê số lần xuất hiện các pt')
print(sorted_d)
print(f'\nPt xuất hiện nhiều nhất là {list(sorted_d)[-1]}')
```

```
Nhập số pt n: 5

List ngẫu nhiên
[10, 5, 2, 4, 5]

Thống kê số lần xuất hiện các pt
{10: 1, 2: 1, 4: 1, 5: 2}

Pt xuất hiện nhiều nhất là 5
```

Câu 35

Viết chương trình nhập một xâu ký tự từ bàn phím. Kiểm tra xem trong xâu có bao nhiêu chữ cái viết hoa, bao nhiêu chữ cái viết thường, bao nhiêu chữ số. Hiển thị các kết quả ra màn hình.

In [26]:

```
s = input('Nhập một xâu kí tự: ')

#-----the C way: sv tự viết

#-----the Python way
upper_list = [ch for ch in s if 'A'<=ch<='Z']
lower_list = [ch for ch in s if 'a'<=ch<='z']
digit_list = [ch for ch in s if '0'<=ch<='9']

print(f'\nSố kí tự hoa: {len(upper_list)}')
print(f'Số kí tự thường: {len(lower_list)}')
print(f'Số kí số: {len(digit_list)}')
```

```
Nhập một xâu kí tự: Vo Van Thuong

Số kí tự hoa: 3
Số kí tự thường: 8
Số kí số: 0
```

Câu 36 (30 điểm):

Đếm số lần xuất hiện các ký tự trong xâu. Ví dụ xâu s="nngeunuee1n", kết quả là 'n': 4, 'g':1, 'u':2, 'e' : 3, '1': 1.

In [27]:

```
s = input('Nhập một xâu kí tự: ')

#-----the C way: sv tự viết

#-----the Python way
d = {}
for ch in s:
    d[ch] = d.get(ch, 0) + 1

print('\nSố lần xuất hiện các kí tự')
for ch, c in d.items():
    print(f'{ch}:{c}')
```

```
Nhập một xâu kí tự: Hello Vimaru

Số lần xuất hiện các kí tự
H:1
e:1
l:2
o:1
 :1
V:1
i:1
m:1
a:1
r:1
u:1
```

Câu 37 (30 điểm):

Nhập vào từ bàn phím một xâu ký tự. Kiểm tra xem trong xâu có bao nhiêu chữ cái x, với x là một chữ cái nhập từ bàn phím.

In [28]:

```
s = input('Nhập một xâu kí tự: ')
ch = input('Nhập một kí tự: ')[0]

#-----the C way: sv tự viết

#-----the Python way
print(f"\n'{ch}' xuất hiện {s.count(ch)} lần trong {s}")
```

```
Nhập một xâu kí tự: Võ Văn Thưởng
Nhập một kí tự: V

'V' xuất hiện 2 lần trong Võ Văn Thưởng
```

Câu 45 (10 điểm):

Xây dựng lớp sách (tên, số trang, giá tiền) với các phương thức khởi tạo, nhập, xuất. Viết chương trình nhập vào một mảng n (n<100) quyển sách, sắp xếp mảng theo chiều giảm dần của giá tiền trung bình của một trang sách và in kết quả vào file “sach.txt”.

Câu 46 (10 điểm):

Xây dựng lớp sách (tên, số trang, giá tiền) với các phương thức khởi tạo, nhập, xuất. Viết chương trình nhập vào một mảng n (n<100) quyển sách, sắp xếp mảng theo chiều tăng dần của tên sách trung bình của một trang sách và in kết quả vào file “sach.txt”.

Câu 47 (10 điểm):

Đọc một mảng n (n<100) quyển sách (tên, số trang, giá tiền) từ file “sach.txt” các dữ liệu ghi trên các dòng khác nhau. Ghi vào file “ketqua.txt” các cuốn sách có giá tiền >100000 và số trang <200.

Câu 48 (20 điểm):

Nhập vào một mảng n (n < 100) số nguyên sinh ngẫu nhiên. In mảng vừa nhập ra màn hình và ghi mảng đó vào file mang.txt.

Câu 49 (20 điểm):

Viết chương trình đọc một mảng các số nguyên từ file “mang.txt”. Sắp xếp mảng giảm dần và ghi kết quả vào file “ketqua.txt”

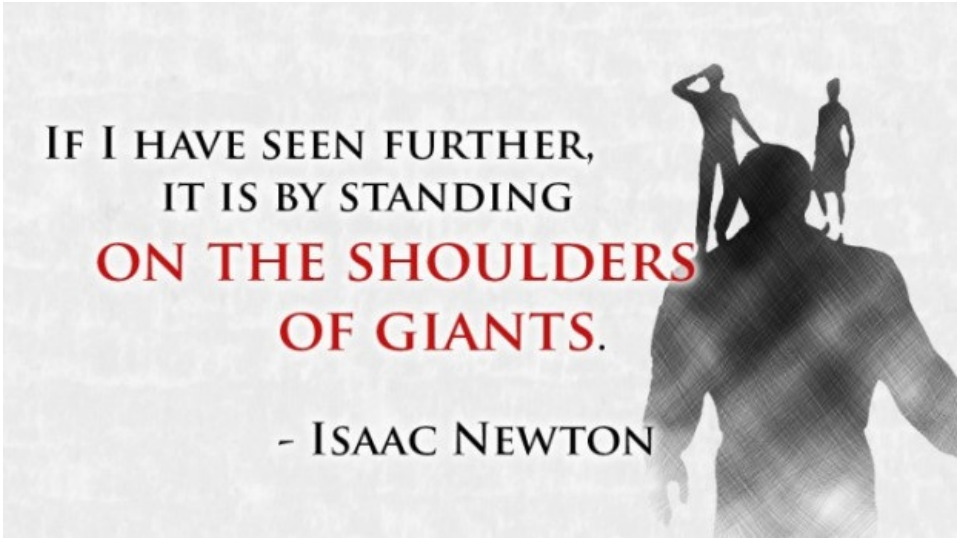
Câu 50 (20 điểm):

Viết chương trình đọc một ma trận các số nguyên từ file “matran.txt”. Sắp xếp ma trận theo chiều tăng dần của các cột và ghi kết quả vào file “ketqua.txt”

In [ ]:

```
```

Chương 6 - Standing on the shoulders of giants



Hãy phát triển các ý tưởng mới dựa trên những chương trình có sẵn.

6.1) Tell me when my girlfriend comes home

Nguồn: https://youtu.be/imN-vhi5ZWQ (https://youtu.be/imN-vhi5ZWQ)

It sniffs the network for DHCP traffic, when her phone joins the wifi network outside it uses the say command to let me knows.

In [16]:

```
import sys
import subprocess
import os
from decouple import config

IP_NETWORK = config('IP_NETWORK')
IP_DEVICE = config('IP_DEVICE')

proc = subprocess.Popen(['ping', IP_NETWORK], stdout=subprocess.PIPE)

while True:
    line = proc.stdout.readline()

    if not line:
        break
    connected_ip = line.decode('utf-8').split()[2]
    connected_ip = connected_ip[1:-1]    # remove left ( and right )

    if connected_ip == IP_DEVICE:
        subprocess.Popen('say', 'Girlfriend just connected to the network!')
```

6.2) Auto like posts on facebook

Nguồn: [1] https://hailiangblog.business.blog/2019/12/10/tu-tao-chuong-trinh-auto-like-facebook-bang-selenium-phan-1-gioi-thieu-selenium-va-dang-nhap-vao-facebook/ (https://hailiangblog.business.blog/2019/12/10/tu-tao-chuong-trinh-auto-like-facebook-bang-selenium-phan-1-gioi-thieu-selenium-va-dang-nhap-vao-facebook/)

[2] https://nguyenvanhieu.vn/tu-dong-dang-nhap-facebook-voi-selenium/ (https://nguyenvanhieu.vn/tu-dong-dang-nhap-facebook-voi-selenium/)

[3] https://github.com/justdvl/facebook-auto-liker (https://github.com/justdvl/facebook-auto-liker)

[4] https://github.com/puffyboa/facebook-auto-like (https://github.com/puffyboa/facebook-auto-like)

In [ ]:

```
from selenium import webdriver
from decouple import config

import time

# The way to locate the button or thing you want on a website in chrome is
# by pressing cmd+shift+c on mac or ctrl+shift+c on windows
# then simply use your mouse to find the infor on the element that your're looking for
# copy the full xpath to that element

options = webdriver.ChromeOptions()
driver = webdriver.Chrome("path/to/your/chromedriver", chrome_options=options)
driver.get(config('SITE'))

button = driver.find_element_by_xpath('/html/body/dive[3]/div/div/div[2]/span/button')
button.click()

# to ensure that the next page has loaded we wait for two seconds
time.sleep(2)
```

6.3) Image scraping and Image Identity

Dua vao cac chuong trinh ve Image scraping and Image Identity, hay viet chuong trinh tu dong thu thap hinh anh ve mot nguoi noi tieng ma ban ham mo. Co the them cac dieu kien nhu: anh chi co mot so nguoi (1,2,3...) nhat dinh.

Image scraping

[1] https://rubikscode.net/2019/12/02/scraping-images-with-python/ (https://rubikscode.net/2019/12/02/scraping-images-with-python/)

[2] https://stackoverflow.com/questions/22701833/scraping-images-using-beautiful-soup (https://stackoverflow.com/questions/22701833/scraping-images-using-beautiful-soup)

[3] https://youtu.be/RkaHdOje-cI (https://youtu.be/RkaHdOje-cI) (WEB SCRAPING Baby Yoda Pictures with Python, Beautiful Soup and Requests (Tutorial))

[4] https://youtu.be/t2k5Lsbpj8Y (https://youtu.be/t2k5Lsbpj8Y) (Download Images With Python Automatically - Python Web Scraping Tutorial)

[5] https://youtu.be/m_agcM_ds1c (https://youtu.be/m_agcM_ds1c) (Introduction to Web Scraping (Python) - Lesson 04 (Download Images))

Image Identity

In [ ]:

```
# Identify and draw box on David
# https://github.com/ageitgey/face_recognition/blob/master/examples/identify_and_draw_boxes_on_faces.py

import face_recognition
import numpy as np
from PIL import Image, ImageDraw

# Load a sample picture and learn how to recognize it.
known_image = face_recognition.load_image_file("./img/mayweather.jpg")
encoding = face_recognition.face_encodings(known_image)[0]

# Load an image with unknown faces
unknown_image = face_recognition.load_image_file("./img/pacquiao-vs-mayweather.jpg")

# Find all the faces and face encodings in the unknown image
face_locations = face_recognition.face_locations(unknown_image)
face_encodings = face_recognition.face_encodings(unknown_image, face_locations)

# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library
# See http://pillow.readthedocs.io/ for more about PIL/Pillow
pil_image = Image.fromarray(unknown_image)

# Create a Pillow ImageDraw Draw instance to draw with
draw = ImageDraw.Draw(pil_image)

# Loop through each face found in the unknown image
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):

        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces([encoding], face_encoding)

        # Use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance([encoding], face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:

                # Draw a box around the face using the Pillow module
                draw.rectangle(((left -20, top -20), (right +20, bottom +20)), outline=(0, 255, 0), width=20)

# Remove the drawing library from memory as per the Pillow docs
del draw

# Display the resulting image
pil_image.show()

# You can also save a copy of the new image to disk if you want by uncommenting this line
# pil_image.save("image_with_boxes.jpg")
```

6.4) Data scraping

Nguồn: [1] https://youtu.be/hkF_oIm3lU4 (https://youtu.be/hkF_oIm3lU4) (Lập Trình Data Science 4.0 Cơ bản Tự Học Cho Người Mới Bắt Đầu | Python Phân Tích Điểm Thi 2020)

[2] https://youtu.be/XQgXKtPSzUI (https://youtu.be/XQgXKtPSzUI) (Intro to Web Scraping with Python and Beautiful Soup)

Hay viet chuong trinh tu dong thu thap du lieu (vd du lieu sinh vien, diem) tu website (vd. cac truong DH o Viet Nam).

6.5) Password detectors

Top 10 commons passwords in 2019:

1. 123456
2. 123456789
3. qwerty
4. password
5. 1234567
6. 12345678
7. 12345
8. iloveyou
9. 111111
10. 123123

[1] https://youtu.be/M9OPVXtnBu8 (https://youtu.be/M9OPVXtnBu8) (Create A Brute Force Password Cracker With Python)

[2] https://youtu.be/RL3_RV-v5Os (https://youtu.be/RL3_RV-v5Os) (How To Make A Simple Brute Force Script In Python)

Viet chuong trinh tu dong do mat khau email, hoctructuyen cua sinh vien truong DHHH va gui email canh bao neu mat khau qua yeu.

Luu y: Nghiem cam cac hanh vi gay hai.