BÀI TẬP PYTHON

(Phần cơ bản)

Lê Văn Hạnh
levanhanhvn@gmail.com
THÁNG 9-2019

MỤC LỤC

ĐỀ NGHỊ CÁCH TỔ CHỨC VÀ THỰC HIỆN CHƯƠNG TRÌNH KHI THỰC HÀNH

- Tất cả bài tập trong tài liệu này được đặt chung một project duy nhất với tên *FundamentalsProject*
- Mỗi chương sẽ được đặt riêng trong 1 Package, ví dụ *pakChuong1*, *pakChuong2*, ...
- Trong mỗi pakage lại tạo các sub package cho các phần nhỏ trong chương như *pakXuatNhap_Bien_ToanTu*, *pakIf*, *pakLoopStatement*, ...
- Mỗi bài tập là 1 file ***.py*** riêng được lưu trong sub package tương ứng với tên đề nghị là số thứ tự của bài tập trong chương theo sau 3 ký tự *bt_* như: *bt_01.py*, *bt_02.py*, ...
- Từ sau phần *1.3 Xây dựng hàm* trở đi, sinh viên cần đưa vào các xử lý ngoại lệ cho chương trình.
- Từ phần *1.5 Xây dựng hàm* trở đi, mỗi phần sinh viên sẽ tạo ra 1 module file để chứa các hàm có trong phần thực hành đó.

1 PHẦN CƠ BẢN

1.1. Các lệnh điều khiển cơ bản

1.1.1. Các lệnh xuất nhập - Biến – Toán tử

1.1.1.1. Sử dụng hàm print

1/. Sử dụng lệnh print để lần lượt in ra các hình sau:

a.-

```
* * * * *
* * * * *
* * * * *
```

b.-

```
* * * * *
*       *
* * * * *
```

c.-

```
  * *
*   *
*   *
  * *
```

d.-

```
* * * * *
    * * *
    * * *
    * * *
* * * * *
```

2/. Sử dụng lệnh print để in ra màn hình chữ PYTHON như hình minh họa sau:

```
XXXXX  XX  XX XXXXXX XX  XX  XXXX  XX  XX
XX  XX  X  X    XX   XX  XX XX  XX XXX XX
XXXXX   X  X    XX   XXXXXX XX  XX XXXXXX
XX       XX     XX   XX  XX XX  XX XXX XX
XX       XX     XX   XX  XX  XXXX  XX  XX
```

3/. Chỉ sử dụng 1 lệnh print, yêu cầu in ra màn hình đoạn thơ Quê hương với định dạng như hình minh họa

```
QUÊ HƯƠNG
. . .
Quê hương là chùm khế ngọt
    Cho con trèo hái mỗi ngày
        Quê hương là đường đi học
            Con về rợp bướm vàng bay
. . .
                        Đỗ Trung Quân
```

1.1.1.2. Sử dụng hàm input

4/. Cho nhập 1 ký tự. In ra mã ASCII của ký tự đó.

📖 *Gợi ý*: dùng hàm ord

5/. Cho nhập 2 số nguyên a, b. In ra tổng của 2 số a và b như minh họa sau:

Ví dụ: `a=3, b=5` ⇒ in ra `3 + 5 = 8`

a. Cho nhập 2 số bằng lệnh riêng biệt

b. Cho nhập 2 số bằng một dòng lệnh chung bằng cách sử dụng phối hợp hàm eval với hàm input cho nhập 2 số cách nhau bởi dấu phẩy (,)

```
a, b = eval(input("..."))
```

c. Sử dụng hàm map(), phương thức split của dữ liệu kiểu chuỗi để cho nhập 1 lần nhiều giá trị, các giá trị này được ngăn cách theo chỉ định trước của chương trình (khoảng trắng, dấu phẩy hoặc bất kỳ ký tự nào)

Ví dụ: sử dụng phối hợp 2 hàm map và int với hàm input để cho nhập 2 số cách nhau bởi dấu chấm phẩy:

```
a, b = map(int, input("...").split(';'))
```

6/. Cho nhập số nguyên dương a có giá trị từ 1 đến 9. In ra tổng của a+aa+aaa.

Ví dụ: `a=5` ⇒ in ra `5 + 55 + 555 = 615`

📖 *Gợi ý*: sử dụng phép nối chuỗi các số nguyên (vd: với a=5: `"%s%s" % (a,a) => 55`) và hàm int (đổi kiểu dữ liệu từ chuỗi sang số nguyên).

7/. Cho nhập 2 số nguyên a, b trên cùng 1 dòng (cách nhau bởi khoảng trắng). In ra kết quả của a^b.

1.1.1.3. Sử dụng phương thức format của chuỗi có trong print

8/. Cho nhập chiều dài, chiều rộng và chiều cao của hình khối chữ nhật (theo cm). Tính và xuất kết quả theo ví dụ sau:

```
Nhập chiều dài đáy hình chữ nhật (cm):>? 2.134
Nhập chiều rộng đáy hình chữ nhật (cm):>? 3.4567
Nhập chiều cao hình khối chữ nhật (cm):>? 4.1
Số lượng số lẻ cần hiển thị:>? 2
Diện tích đáy hình chữ nhật = 7.38cm²
Thể tích hình khối= 30.24cm³
```

🏳 *Gợi ý*: để in các ký tự số mũ, sử dụng mã UNICODE lần lượt là **\u00b2** và **\b00b3** trong lệnh `print`.

9/. Cho nhập 3 số nguyên a, b, c. In ra 3 số trên theo thứ tự tăng dần. Ví dụ

```
print('{} {} {}'.format(biến1, biến2, biến3))
```

📖 *Gợi ý*: dùng hàm min, max để tìm số nhỏ và lớn nhất và dùng phương thức format của dữ liệu kiểu chuỗi để xuất kết quả.

1.1.1.4. Tính toán đơn giản

10/. Tạo chương trình yêu cầu người dùng nhập tên và tuổi của họ. Giả sử năm hiện tại là 2020, chương trình in ra màn hình chuỗi (s) thông báo năm mà người đó tròn 100 tuổi.

11/. Có 9 loại tiền 1, 2, 5, 10, 20, 50, 100, 200, 500. Cho nhập số tiền X. Chuyển số tiền X ra các loại tiền sao cho số lượng là ít nhất. In ra số tờ tiền của mỗi loại.

Ví dụ với X=1234, sẽ in ra:

```
So tien 1234 duoc doi thanh:
Loai 500 gom 2 to
Loai 200 gom 1 to
Loai 100 gom 0 to
Loai 50 gom 0 to
Loai 20 gom 1 to
Loai 10 gom 1 to
Loai 5 gom 0 to
Loai 2 gom 2 to
Loai 1 gom 0 to
```

1.1.1.5. Đọc chương trình xác định kết quả

12/. Đọc chương trình xác định kết quả xuất ra của đoạn chương trình dưới đây. Sau khi ghi kết quả vào các ô tương ứng, sinh viên hãy viết và chạy đoạn chương trình này để kiểm tra lại:

a.

Mã lệnh	Kết quả
`X= 5` `print('X=', X)`	
`X-= 1` `print('X=', X)`	
`X+= 3` `print('X=', X)`	
`X= - X` `print('X=', X)`	
`X= True` `print('not X=', not X)`	

b.

Mã lệnh	Kết quả	
	X	Y

`x = True`
`y = False`
`print('x and y :',x and y)`		

`print('x or y :',x or y)`		
`print('not x :',not x)`		
`print('x is y :', x is y)`		
`print('x is not y :', x is not y)`		

c.

Mã lệnh	Kết quả	
	X	Y
X = 1 + 2 print('X=', X)		
Y = X X = X- 1 print('X=', X)		
Y = X X = X* 2 Y = X print('X=', X)		
X = X** 3 Y = X print('X=', X)		
X = X+ 8 Y = X print('X=', X)		
X= X% 7 Y= X print('X=', X)		
X= X// 2 Y= X print('X=', X)		

d.

Mã lệnh	Kết quả		
	x	y	z
x = 10 y = 4 print('x = %d, y = %d'%(x,y))			
z = x==y print('x==y is', z)			
z = x!=y print('x!=y is', z)			
z = x>y print('x>y is', z)			
x = 8 y = 9 print('x = %d, y = %d'%(x,y))			
z = x>=y print('x>=y is', z)			
z = x<y print('x<y is', z)			
z = x<=y print('x<=y is', z)			

e.

Mã lệnh	Kết quả	
	X	Y
x = 15 y = 12 print('Binary of x =', bin(x))		
print('Binary of y =', bin(y))		
print('x & y =', bin(x & y))		
print('x | y =', bin(x | y))		
print('x ^ y =', bin(x ^ y))		
print('~x =', bin(~x))		
print('x << 2 =', bin(x << 2))		
print('y >> 2 =', bin(y >> 2))		

1.1.2. Cấu trúc điều kiện

13/. Cho người dùng nhập 1 số nguyên. Cho biết đó là số chẵn hay số lẻ

a. Mở rộng 1: cho người dùng nhập 1 số nguyên (n). Cho biết n có chia chẵn cho 4 hay không? Nếu không chia chẵn cho 4 thì n có chia chẵn cho 2 hay không?

b. Mở rộng 2: cho người dùng nhập 1 số nguyên n đại diện cho số bị chia và số nguyên m đại diện cho số chia. Cho biết n có chia chẵn cho m hay không?

14/. Thực hiện lại bài tập ***11*** với yêu cầu chỉ in những loại tiền có số tờ lớn hơn 0. (Có 9 loại tiền 1, 2, 5, 10, 20, 50, 100, 200, 500. Cho nhập số tiền X. Chuyển số tiền X ra các loại tiền sao cho số lượng là ít nhất.)

Ví dụ với X=1234, sẽ in ra:

```
So tien 1234 duoc doi thanh:
Loai 500 gom 2 to
Loai 200 gom 1 to
Loai 100 gom 0 to  #không in dòng này vì số lượng =0
Loai 50 gom 0 to   #không in dòng này vì số lượng =0
Loai 20 gom 1 to
Loai 10 gom 1 to
Loai 5 gom 0 to    #không in dòng này vì số lượng =0
```

```
Loai 2 gom 2 to
Loai 1 gom 0 to     #không in dòng này vì số lượng =0
Tong so loai = 5    #in thêm dòng này trong kết quả
```

15/. Cho người dùng nhập điểm (từ 0-100). Nếu điểm nhập có giá trị <0 hoặc >100, chương trình in ra câu báo lỗi '***Chỉ nhận các giá trị từ 0 đến 100***' và kết thúc chương trình. Ngược lại, in ra xếp loại của điểm số đó. Quy định xếp loại cho trong bảng sau:

Điểm số	*Xếp loại*
>=90	A
Từ 80 đến 89	B
Từ 70 đến 79	C
Từ 65 đến 69	D
Dưới 65	E

16/. Viết chương trình cho nhập số dương `n`. Chương trình thực hiện việc trừ n đi số `X` (với `X` là tổng các chữ số thành phần đang có của `n`). Chương trình thực hiện tìm `n` sao cho `n` nhỏ nhất và `n>0`.

Ví dụ: với `n= 37`, sẽ in ra `n=9 là kết quả cần tìm` (vì `37-10=27`, `27-9=18`, `18-9=9`)

Tương tự với `n= 6`, sẽ in ra `n=6 là kết quả cần tìm`

17/. Cho nhập 3 số nguyên a, b trên cùng 1 dòng (cách nhau bởi khoảng trắng). Viết chương trình giải phương trình bậc 1: ax + b = 0.

18/. Cho nhập 3 số nguyên a, b, c trên cùng 1 dòng (cách nhau bởi khoảng trắng). Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$.

```
a,b,c = map(int,input('Nhập 3 số cách nhau bởi khoảng trắng: ').split(' '))
```

19/. Cho nhập 2 số nguyên a, b. Kiểm tra xem a, b, a+b hoặc a*b có chiều dài lớn hơn 80 ký số thì thông báo "overflow", ngược lại in ra tổng, hiệu, tích thương của 2 số a và b. Chú ý cần kiểm tra giá trị của x khi thực hiện phép chia.

20/. Một điểm trong mặt phẳng được biểu diễn bằng 2 tọa độ x và y. Cho nhập tọa độ 3 đỉnh A, B, C của 1 tam giác, cho phập tọa độ điểm P. Xác định xem điểm P nằm trong hay nằm ngoài tam giác.

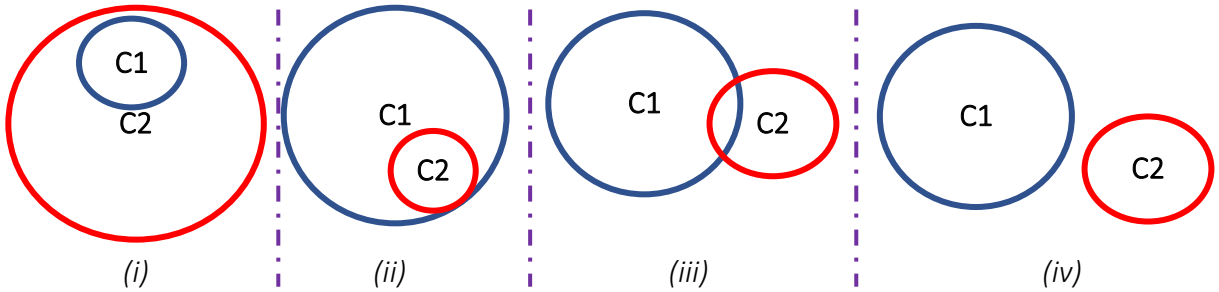
☞ *Gợi ý*:

- Thực hiện tính 3 giá trị `z1, z2, z3` theo công thức sau:

```
z1 = (xB-xA)*(yP-yA)-(yB-yA)*(xP-xA)
z2 = (xC-xB)*(yP-yB)-(yC-yB)*(xP-xB)
z3 = (xA-xC)*(yP-yC)-(yA-yC)*(xP-xC)
```

- Kiểm tra nếu cả 3 giá trị `z1, z2, z3` cùng dương hoặc cùng âm thì điểm P nằm trong tam giác ABC, ngược lại là nằm ngoài.

21/. Mỗi vòng tròn trong mặt phẳng được xác định bởi các yếu tố tâm vòng tròn (x, y) và bán kính (R). Viết chương trình cho nhập thông tin về 2 vòng tròn C1 và C2. Chương trình cho biết C1 và C2 thuộc trường hợp nào trong các trường hợp sau:



(i)- C1 nằm trong C2
(ii)- C2 nằm trong C1
(iii)- C1 và C2 cắt nhau
(iv)- C1 và C2 không cắt nhau (không có phần giao nhau)

22/. Nhập vào 6 số thực a, b, c, d, e, f cách nhau bởi dấu chấm phẩy (;). Giải hệ phương trình sau:

$$\begin{cases} ax+by=c \\ dx+ey=f \end{cases}$$

🏳 Gợi ý:

$$D=\begin{vmatrix} a & b \\ d & e \end{vmatrix} = ae - bd$$

$$Dx=\begin{vmatrix} c & b \\ f & e \end{vmatrix} = ce - bf$$

$$Dy=\begin{vmatrix} a & c \\ d & f \end{vmatrix} = af - dc$$

Nếu D!=0 ⇒ x=Dx/D; y=Dy/D
Ngược lại nếu Dx!=0 hoặc Dy !=0 ⇒ PT vô nghiệm
Ngược lại, ⇒ PT vô định

1.1.3. Cấu trúc lặp

23/. Cho nhập 2 số nguyên a, b (a<b). In ra các số nằm trong khoảng từ a đến b và bỏ qua các số chia chẵn cho 3 thuộc đoạn [a,b].

🏳 *Yêu cầu*: sử dụng lệnh (*statement*) *continue* trong chương trình.

24/. Cho nhập 1 số nguyên n. In ra chuỗi số Fibonacci từ 0 đến <= n.

Ví dụ: với `n= 50`, sẽ in ra `0 1 1 2 3 5 8 13 21 34`

🕮 *Nhắc lại*: chuỗi số Fibonacci luôn bắt đầu từ 2 số 0 và 1. Mỗi số tiếp theo được xác định bằng cách cộng hai số liền trước nó.

25/. Cho nhập số nguyên dương n (11<=n<=100). In ra các số X nằm trong khoảng từ 11 đến n sao cho các ký số có trong X đều giống nhau.

Ví dụ: `n=50` ⇒ in ra `11, 22, 33, 44`

26/. Cho nhập số nguyên dương n. Đếm số lượng chữ số chẵn, số chữ số lẻ có trong n.

Ví dụ: `n=5084` ⇒ in ra `số lượng số lẻ=1, số lượng số chẵn=3`

27/. Tính tổng và tích các chữ số của số nguyên dương n.

Ví dụ: `n=5084` ⇒ in ra `tổng=17, tích=0`

28/. Tìm chữ số lớn nhất có trong số nguyên dương n.

Ví dụ: `n=5084` ⇒ in ra `số lớn nhất=8`

29/. Số may mắn: giả sử người ta cho rằng 1 số gọi là số may mắn nếu chỉ chứa toàn các số 6 hoặc số 8. Viết chương trình cho nhập số nguyên n, xét xem n có là số may mắn hay không?

```
Ví dụ:   n=686⇒ 686 là số may mắn.
         n=68626⇒ 68626 KHÔNG phải số may mắn.
```

30/. Cho nhập số nguyên dương n. Kiểm tra xem các chữ số của n có toàn lẻ (hay toàn chẵn) không?

31/. Hãy kiểm tra các chữ của số nguyên dương n có giảm dần từ trái sang phải hay không?

32/. Cho nhập số nguyên dương n. In ra cách đọc của số n.

`Ví dụ: Nhập n = -105. In ra màn hình: Am Mot khong nam.`

33/. Cho nhập số nguyên dương n (n>=2). Hãy phân tích n thành tích các thừa số nguyên tố.

`Ví dụ: nhập n=1350. In ra` **`1350 = 2 * 3 * 3 * 3 * 5 * 5`**

34/. Cho nhập số nguyên dương n. In ra màn hình cách phân tích n thành thừa số nguyên tố nhưng dưới dạng số mũ.

`Ví dụ: nhập n=1350. In ra` **`1350 = 2 * 3^3 * 5^2`**

35/. Cho nhập n. Tính tổng S (hoặc tích P) trong các trường hợp sau. Yêu cầu: đối với từng bài, thực hiện bằng 2 cách: cách 1: sử dụng lệnh while, cách 2 sử dụng lệnh for và hàm range() :

a. S = 1 + 2 + 3 + . . . + n

b. S = 1 + 3 + 5 + . . . + (2n+1)

c.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

d.

$$S = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n+1}$$

e.

$$S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \ldots + \frac{1}{2n}$$

f.

$$S = 1 + \frac{1}{3} + \frac{1}{5} + \ldots + \frac{1}{2n-1}$$

g. S = (1) + (1+2)+(1+2+3) +(1+2+3+4)+ +(1+2+3+4+5) + . . . +(1+2+3+... +n)

h. P = 1 x 3 x 5 x . . . x (2n-1)

36/. Viết chương trình trò chơi "Bao-Búa-Đinh" cho 2 người chơi. Mỗi người dùng được chọn 1 trong 3 món Bao, búa, đinh. Chương trình cho biết người thắng cuộc theo quy ước:

- Bao thắng Búa
- Búa thắng Đinh
- Đinh thắng Bao

a. *Mở rộng 1*: Sau mỗi lần thông báo kết quả, chương trình cho người dùng được chơi tiếp hay không? Nếu người dùng chọn chơi tiếp thì chương trình cho lặp lại việc chọn món có trong trò chơi.

b. *Mở rộng 2*: Khi kết thúc, chương trình cho biết số lần thắng của mỗi người chơi

37/. Game đoán số. Đầu tiên, chương trình phát sinh 1 số ngẫu nhiên từ 0 đến 9. Sau đó, cho người dùng nhập 1 số nguyên từ 1 đến 9. Nếu người dùng đoán đúng, chương trình kết thúc, ngược lại khi người dùng đoán sai, chương trình cho người dùng biết số mà họ đã đoán thấp hơn hay cao hơn số kết quả và cho họ đoán lại

a. *Mở rộng 1*: game sẽ kết thúc khi người dùng gõ 'exit'.

b. *Mở rộng 2*: sau mỗi lần đoán sai, chương trình thông báo thêm số lần đoán thấp hơn kết quả, số lần đoán cao hơn kết quả.

38/. Cho nhập số nguyên dương n. In ra các số nguyên dương X nhỏ hơn hay bằng n, sao cho X có chứa ít nhất một số 5 ở bất kỳ vị trí nào (hàng chục, hàng đơn vị, hàng trăm, ...). Các số trong kết quả được in ra cách nhau bởi dấu phẩy (,).

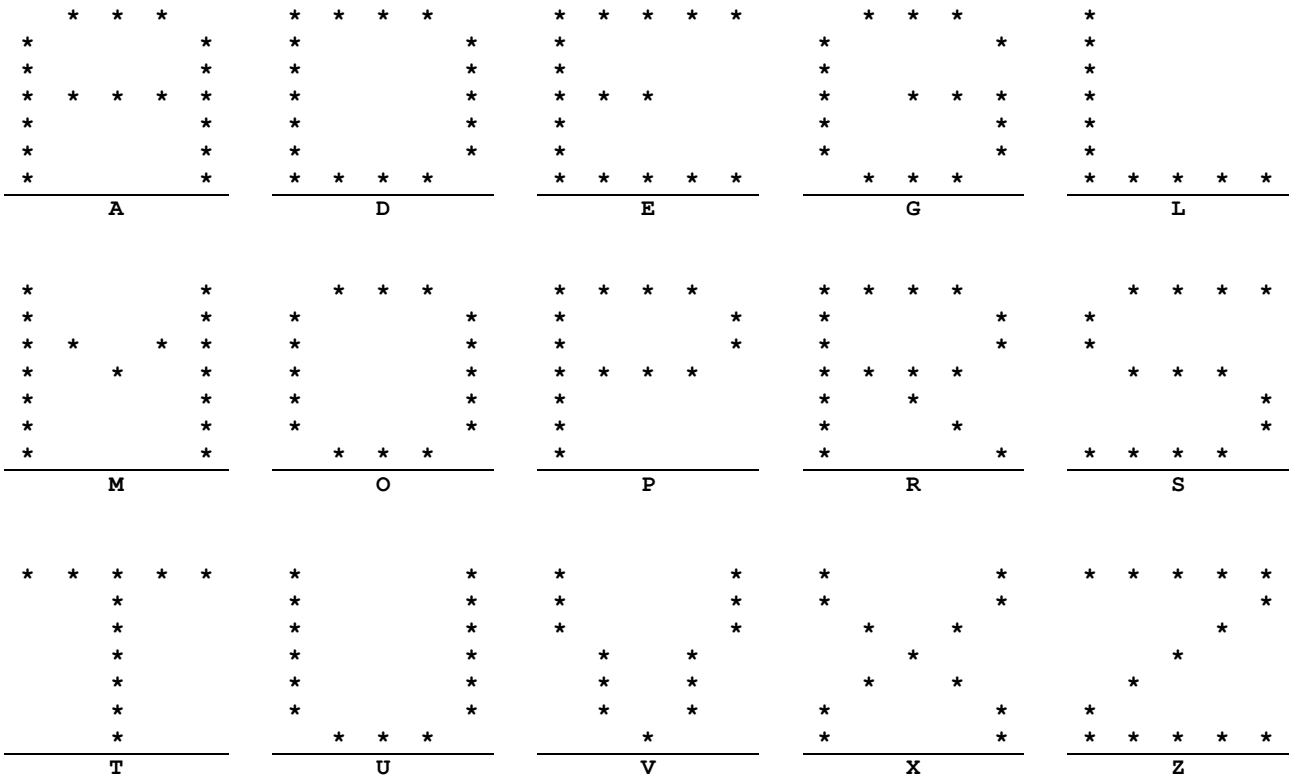
Ví dụ: nhập
```
n= 60
In ra 5, 15, 25, 35, 45, 50
```

39/. Cho nhập số nguyên dương n. In ra các số nguyên dương X nhỏ hơn hay bằng n, sao cho các ký số có trong X đều là số chẵn.

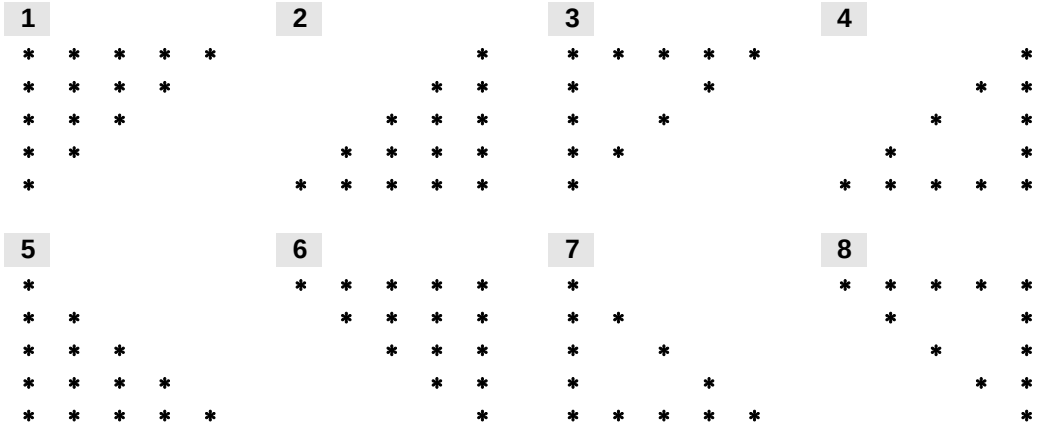
Ví dụ: nhập
```
n= 30
In ra 0, 2, 4, 6, 8, 20, 22, 24, 26, 28
```

40/. Viết chương trình sử dụng lệnh lặp để in lần lượt các ký tự sau ra màn hình: **A, D, E, G, L, M, O, P, R, S, T, U, V, X, Z**. Mỗi ký tự được in trong 1 hình chữ nhật ngang=5 và chiều cao=7.

Minh họa:



41/. Viết chương trình cho người dùng nhập cạnh (n) của tam giác vuông cân và chọn 1 trong 8 hình tam giác cần vẽ dưới đây (hình 1-8). Giả sử với n=5, các hình mà chương trình cần vẽ có dạng như sau:

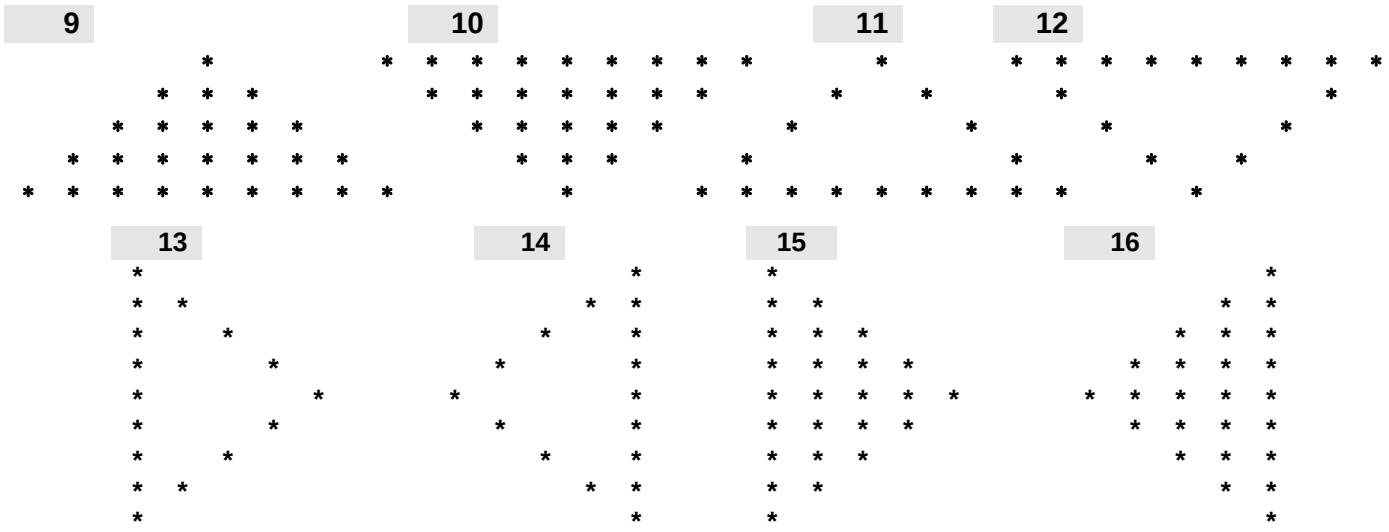


- ***Minh họa***: màn hình menu có dạng như sau:

```
    CHUONG TRINH IN TAM GIAC VUONG CAN
0.- Nhap canh cua tam giac
1.- Hình 1.
2.- Hình 2.
3.- Hình 3.
4.- Hình 4.
5.- Hình 5.
6.- Hình 6.
7.- Hình 7.
8.- Hình 8.
9.- Ket thuc chuong trinh
Chon chuc nang can thuc hien (0-9):
```

- ***Yêu cầu***: tổ chức chương trình thành các hàm chức năng, mỗi hàm vẽ 1 hình

42/. Thực hiện tương tự bài tập trước để tạo menu cho người dùng chọn in mỗi lần 1 hình trong các hình sau (hình 9-16). Giả sử với n=5, các hình mà chương trình cần vẽ có dạng như sau:



43/. Cho nhập 2 số nguyên a, b trên cùng 1 dòng (cách nhau bởi dấu phẩy – ‘,’). In ra các bảng cửu chương từ a đến b (khi a<b) hoặc từ b đến a (khi b<a).

44/. Cho nhập số nguyên dương n. Kiểm tra xem n có phải là số nguyên tố hay không?

45/. Cho nhập số nguyên dương n. Liệt kê các số nguyên tố < n.

46/. Cho nhập số nguyên dương n. Đếm các số nguyên tố <n.

47/. Cho nhập số nguyên dương n Tìm số nguyên tố đầu tiên <n và có giá trị gần với n nhất.

48/. Cho nhập số nguyên dương n. Tìm số nguyên tố đầu tiên >n và có giá trị gần với n nhất.

49/. Cho nhập số nguyên dương n, liệt kê các ước số của n là số nguyên tố.

Ví dụ: Nhập `n=36`. Các ước số của 36 gồm `1,2,3,4,6,9,12,18`

Nhưng chỉ in ra: `các số vừa là ước số của 36, vừa là số nguyên tố: 2,3`

1.1.4. Kiểm tra số do người dùng nhập vào

- ***Mở rộng***: sau khi hoàn tất các bài tập dưới đây, sinh viên cần bổ sung thêm lệnh kiểm tra kiểu của dữ liệu.

50/. Cho người dùng nhập vào 1 số nguyên dương ***n*** thỏa lần lượt các yêu cầu sau đây. Nếu người dùng nhập vào số không đúng yêu cầu thì chương trình cho nhập lại

a. Điều kiện: ***n*>0**. Nếu nhập đúng, chương trình in ra các số chẵn nhỏ hơn ***n***.

b. Điều kiện: **0 <= *n* <= 9**. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.

c. Điều kiện: ***n* <10 hoặc *n* > 50**. Nếu nhập đúng, chương trình in ra ***n*** số ngẫu nhiên.

d. Điều kiện: ***n* bội số của 5**. Nếu nhập đúng, chương trình in ra bảng cửu chương 5.

e. Điều kiện: ***n* nằm trong các số (2, 4, 6, 8, 10)**. Nếu nhập đúng, chương trình cho nhập tiếp 2 số nguyên ***i, j***. Hãy tính và in ra cấp số cộng với số hạng đầu là i, công sai là j và số lượng phần tử của dãy là ***n*** .

51/. Viết chương trình cho nhập 2 số nguyên dương ***n*** và ***m*** sao cho số nhập sau (***m***) phải luôn lớn hơn số nhập trước (***n***). Nếu nhập đúng, in ra các số lẻ nằm trong khoảng từ ***n*** đến ***m***. Khi người dùng nhập sai, thực hiện lần lượt các trường hợp sau:

a. Chỉ yêu cầu nhập lại ***m***.

b. Yêu cầu nhập lại cả ***n*** và ***m***.

52/. Cho nhập 2 số ***a***, ***b*** sao cho: số lớn nhất trong 2 số phải là một số dương và chia hết cho ***7***. Nếu nhập sai phải yêu cầu nhập lại cả 2 số cho đến khi đúng. Ngược lại, khi nhập đúng chương trình sẽ in ra các số chia chẵn cho ***7*** và nằm trong khoảng từ ***a*** đến ***b***.

53/. Cho nhập 2 số nguyên dương ***a*** và ***b*** sao cho ước số chung lớn nhất của a và b phải là bội số của 5. Nếu nhập sai, cho nhập lại cả 2 số ***a*** và ***b***. Nếu nhập đúng, đếm xem từ ***a*** đến ***b*** có bao nhiêu số chia chẵn cho 3.

54/. Nhập vào số nguyên dương ***x*** là lũy thừa của 2. Nếu nhập sai, yêu cầu nhập lại. Khi nhập đúng, in ra số ***x*** dưới dạng **`2^k`**. Với k là số lượng số 2 tham gia vào phép nhân để có được giá trị ***x***).

Ví dụ **`x=32 = 2*2*2*2*2`** => in ra **`2^5`**

1.1.5. Khác

55/. Cho nhập 1 số nguyên dương có giá trị từ 0 đến 27. Tìm các bộ 3 số i, j, k sao cho 0 <= i, j, k <= 9 và i + j + k = n.

☞ *Gợi ý*: để tạo tổ lợp 3 số có giá trị từ 0 đến 9, sử dụng phương thức product của itertools như sau:

```
itertools.product(range(10),range(10),range(10))
```

1.2. Strings

1.2.1. Các phương thức của kiểu dữ liệu chuỗi

56/. Viết chương trình cho người dùng nhập 1 chuỗi (S).

a. Cho nhập tiếp 1 ký tự (c). Đếm xem trong S có bao nhiêu ký tự c (không sử dụng hàm len()).

b. Lần lượt in ra chuỗi S dưới 3 dạng: tất cả là chữ hoa (in), tất cả là chữ thường và tất cả ký tự đầu của mỗi từ trong S là chữ hoa, các ký tự còn lại trong từ là chữ thường.

Ví dụ:
```
S = 'HeLlO PyThoN'
In ra  UPPER: HELLO PYTHON
       Lower: hello python
       Capitalize: Hello Python
```

c. Đếm xem trong S có bao nhiêu ký tự, bao nhiêu ký số.

d. Đếm xem trong S có bao nhiêu ký tự in hoa, ký tự thường, ký số và ký tự loại khác.

e. Đếm và in ra trong S có bao nhiêu nguyên âm (*vowel* - **a,e,i,o,u**) ký tự, bao nhiêu phụ âm (*consonant* – **b,c,d,e,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,y,z**). Lưu ý khi đếm không phân biệt chữ hoa, chữ thường.

Ví dụ: `S='Sai Gon'`

```
Có 3 nguyên âm: a i o
Có 3 phụ âm: S G n
```

Gợi ý: dùng phương thức count của kiểu dữ liệu chuỗi

57/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Cho biết chuỗi đó có là số nguyên (có thể âm hoặc dương) hay không?

Ví dụ: nhập **S = python3 ⇒ chuỗi 'không phải số nguyên**
S = 1273 ⇒ chuỗi là số nguyên
S = -1273 ⇒ chuỗi là số nguyên
S = +1273 ⇒ chuỗi là số nguyên

📖 *Gợi ý*: sử dụng hàm **`all()`** để kiểm tra

58/. Viết chương trình cho nhập số 1 chuỗi (S). Chương trình thực hiện loại bỏ tất cả các nguyên âm (a,e,i,o,u) và là ký tự in thường có trong S.

Ví dụ: `S = "Hello Everybody"` ⇒ xuất ra `Hll Evrybdy`

59/. Viết chương trình cho người dùng nhập 1 chuỗi *S* và 1 từ (*word*). Đếm xem trong S có bao nhiêu từ *word*.

Ví dụ: kết quả sẽ in: `số từ ai là 7` với `word='ai'`

```
Và s='''Chiều chiều trước bến Văn Lâu
      Ai ngồi, ai câu, ai sầu, ai thảm
      Ai thương, ai cảm, ai nhớ, ai trông
      Thuyền ai thấp thoáng ven sông
      Đưa câu mái vẩy chạnh lòng nước non'''
```

60/. Giả sử một chuỗi được gọi là hoàn chỉnh khi:

- Đầu và cuối chuỗi không chứa khoảng trắng (space).
- Giữa các từ chỉ cách nhau bởi 1 khoảng trắng.
- Dấu chấm và dấu phẩy phải đi liền với từ ngay trước mà không được cách bởi khoảng trắng.
- Nếu là bài thơ, các dòng phải được canh thẳng hàng (đều phải xuất phát từ đầu dòng).

Ví dụ: (ký tự ∪ được diễn tả thay cho khoảng trắng)

Chuỗi chưa hoàn chỉnh	*Chuỗi hoàn chỉnh*
∪∪∪∪Quê hương	Quê∪hương
Quê∪hương∪là∪∪∪chùm∪khế∪∪∪ngọt.	Quê∪hương∪là∪chùm∪khế∪ngọt.
∪∪∪∪Cho∪con∪trèo∪hái∪∪∪mỗi∪∪ngày∪∪∪.	Cho∪con∪trèo∪hái∪mỗi∪ngày.
Quê∪∪hương∪là∪∪∪đường∪∪đi∪học∪.	Quê∪hương∪là∪đường∪đi∪học.
∪∪Con∪về∪∪rợp∪bướm∪∪vàng∪bay∪.	Con∪về∪rợp∪bướm∪vàng∪bay.
∪∪Đỗ∪∪∪∪Trung∪Quân∪∪∪∪	Đỗ∪Trung∪Quân

Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả các khoảng trắng thừa.

61/. Viết chương trình cho người dùng nhập 1 chuỗi. Cho biết chuỗi đó có đối xứng (*palindrome*) hay không? Viết chương trình bằng 2 cách:

a. Sử dụng toán tử index để đảo chuỗi

b. Xử lý đảo từng ký tự trong chuỗi ban đầu *X* để có chuỗi *Y*, sau đó so sánh *X* và *Y*.

62/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất S dưới dạng: ký tự ở vị trí lẻ là ký tự thường, tại vị trí chẵn là ký tự in hoa. VD: ThÀnH PhỐ Hồ cHí mInH

63/. Viết chương trình cho người dùng nhập 1 chuỗi (S).

a. Nếu chiều dài chuỗi lớn hơn 4, chương trình thực hiện đảo ngược thứ tự các ký tự trong S, ngược lại khi chiều dài chuỗi nhỏ hơn hay bằng 4, chương trình giữ nguyên chuỗi ban đầu. Ví dụ:

```
S='Python'  ⇒ nohtyP
S='Data'    ⇒ Data
```

b. Gọi số lượng ký tự hoa trong S là *upp*, và gọi số lượng ký tự thường trong S là *low*. Nếu *upp*>=*low*, thực hiện đổi toàn bộ *S* thành chữ hoa, ngược lại, đổi toàn bộ *S* thành chữ thường.

Ví dụ:

```
S='Python'      ⇒ Python
S='PythON'      ⇒ PYTHON
```

64/. Viết chương trình nhập số thực x và số lượng số lẻ dp(DecimalPlaces). Dùng phương thức format của chuỗi để in ra giá trị của x với dp số lẻ.

Ví dụ: `x = 3.1415926` và `dp=4` ⇒ in ra `x = +3.1416`

🕮 *Gợi ý*: để in ra số thực với 2 số lẻ và có dấu (âm hoặc dương), dùng định dạng sau:

```
print("{:+.2f}".format(x))
```

65/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Nhập thêm số lượng chuỗi cần tách (n) và ký tự (c) dùng làm căn cứ cho việc tách chuỗi. Thực hiện tách chuỗi S từ cuối về đầu thành n+1 đoạn dựa vào ký tự c.

Ví dụ: `S='Thành Phố Hồ Chí Minh', c=' '; n=3`

⇒ `xuất ra ['Thành Phố', 'Hồ', 'Chí', 'Minh']`

66/. Viết chương trình cho người dùng nhập 1 chuỗi S và tên của tag T (có trong HTML). Bao chuỗi S bên trong cặp tag T.

Ví dụ: `S='Python', T=i` ⇒ xuất ra `'<i>Python</i>'`

Hay `S='Python', T=b` ⇒ xuất ra `'<b>Python</b>'`

67/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm xem nếu từ '*poor*' đi sau từ '*not*' thì thay đoạn từ '*not*' đến '*poor*' thành duy nhất 1 từ '*good*'. Các trường hợp khác sẽ giữ nguyên chuỗi (S).

Ví dụ: `S= 'The lyrics is **not** that **poor**!'` ⇒ xuất ra `The lyrics is **good**!`

`S= 'The lyrics is **poor**!'` ⇒ xuất ra `The lyrics is **poor**!`

68/. Cho nhập 1 chuỗi S. Chuyển tất cả các khoảng trắng nếu có ra phía đầu chuỗi S.

Ví dụ: (ký tự ∪ được diễn tả thay cho khoảng trắng)

`S= "∪∪∪w3resource∪.∪com∪∪"` ⇒`"∪∪∪∪∪∪∪w3resource.com"`

69/. Viết chương trình cho người dùng nhập 1 chuỗi (S) bao gồm cả ký tự và ký số. Thực hiện tính tổng tất cả các ký số có trong S.

Ví dụ: `S='Python 3.7'` ⇒ `10`

70/. (**) Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tìm đoạn văn bản ngắn nhất trong S1 sao cho đoạn này chứa tất cả các ký tự có trong S2. Nếu không có kết quả trả về chuỗi rỗng.

🏳 *Gợi ý*: sử dụng `Counter` trong `module collections`

Ví dụ: `S1='abcdefgh'; S2='abcabcdgh'` ⇒ `'abcd'`

71/. (**) Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tìm đoạn văn bản giống nhau và dài nhất có trong S1 và S2.

🏳 *Gợi ý*: sử dụng `SequenceMatcher trong module difflib`

Ví dụ: `S1='abcdefgh'; S2='abcabcdgh'` ⇒ `'abcd'`

1.2.2. index

72/. Viết chương trình cho người dùng nhập họ tên (S). Thực hiện tách các phần họ, phần lót và phần tên. In kết quả ra màn hình.

Ví dụ: `S='Nguyễn Thị Minh Khai'`
In ra:
```
Họ: Nguyễn
Lót: Thị Minh
Tên: Khai
```

73/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện hoán vị 2 ký tự đầu và cuối cho nhau.

Ví dụ: `S='Python'` ⇒ `nythoP`
`S='12345'` ⇒ `52341`

74/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả các ký tự tại vị trí có chỉ số là số lẻ.

Ví dụ: **`S='Python'`** ⇒ `pto`
`S='0123456789'` ⇒ `02468`

75/. Viết chương trình cho người dùng nhập 1 chuỗi *S* và số nguyên pos. Xóa ký tự tại vị trí pos của chuỗi (S). Nếu Pos> chiều dài chuỗi S, xem như chuỗi S được giữ nguyên.

Ví dụ: **`S='Python'`**, `pos=0` ⇒ `ython`
`S='Python'`, `pos=3` ⇒ `Pyton`
`S='Python'`, `pos=5` ⇒ `Pytho`

76/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Gọi ký tự đầu tiên của *S* là *c*. Tìm và thay thế tất cả các ký tự *c* có trong chuỗi thành ký tự '$' (không thực hiện đối với ký tự đầu tiên của chuỗi S). Ví dụ: *restart* sẽ được đổi thành *resta**$**t*, tương tự *madam* đổi thành *mada**$***

77/. Viết chương trình cho người dùng nhập 2 chuỗi Svà R. Hoán vị 1 ký tự cuối của R và S cho nhau. Ví dụ: *S='abcd', R='xyz'* sẽ được đổi thành *S='abc**z**', R='xy**d**'*

78/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Nếu chiều dài chuỗi S nhỏ hơn 3, giữ nguyên, ngược lại kiểm tra xem S kết thúc bằng "ing" hay không, nếu chưa có thì thêm ing vào S, ngược lại (khi đã có ing) thì thêm ly vào cuối (S).

Ví dụ: `S='ab'` ⇒ xuất ra `'ab'`
`S='str'` ⇒ xuất ra `'stri**ng**'`
`S='string'` ⇒ xuất ra `'string**ly**'`

79/. Viết chương trình cho người dùng nhập 1 chuỗi S và loại ngoặc định dùng (gồm 4 loại: [, {, (, <), trong đó, người dùng sử dụng ký tự đóng hay mở ngoặc đều được. Chèn chuỗi S vào giữa cặp ngoặc lồng nhau B.

Ví dụ: `S='Python', B = '[' ⇒ xuất ra '[[Python]]'`
Hay `S='Python', B = '}' ⇒ xuất ra '{{Python}}'`
Hay `S='Python', B = '<' ⇒ xuất ra '<<Python>>'`

80/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất S dưới dạng: ký tự đầu và 3 ký tự cuối là chữ thường, còn lại là ký tự hoa. VD: thàNH PHỐ HỒ CHÍ Minh

81/. Viết chương trình cho người dùng nhập 1 chuỗi S (có thể gồm nhiều dòng). Thực hiện đảo các từ trong chuỗi.

Ví dụ: `S = 'Kiến ăn Cá'` ⇒ xuất ra `Cá ăn Kiến`
Hay `S='Chỉ có thuyền mới hiểu\n Biển mênh mông nhường nào'`
⇒ xuất ra `hiểu mới thuyền có Chỉ`

```
nào nhường mông mênh Biển
```

82/. Giả sử giữa 2 người *A* và *B* có quy ước về việc rút ngắn chuỗi ký tự trong văn bản rõ (*plain text*) bằng cách thay thế những ký tự liền kề và giống nhau như sau: ví dụ: `plaintext` chứa chuỗi ký tự '`YYYYY`' sẽ được thay bằng '`#5Y`' trong bản mã (cipher text) hay `999.99` sẽ được thay bằng `#39.#29`.

Viết chương trình Python để khôi phục chuỗi gốc bằng cách nhập chuỗi nén (*cipher text*) với quy tắc này. Lưu ý ký tự # không được xuất hiện trong chuỗi ký tự được khôi phục (*plain text*).

Ví dụ: cipher text là **`XY#6Z1#4023`** sẽ xuất ra plain text là **`XYZZZZZZ1000023`**

Hay cipher text là **`#39+1=1#30`** sẽ xuất ra plain text là **`999+1=1000`**

83/. Viết chương trình cho người dùng nhập 1 chuỗi S chỉ chứa các ký tự từ a-z (ký tự thường). Cho biết S có chứa bao nhiêu ký tự phân biệt.

Ví dụ: `S='abc'` ⇒ `xuất ra 3`
Hay `S='abba'` ⇒ `xuất ra 2`
Hay `S='madam'` ⇒ `xuất ra 3`

1.2.3. enumerate()

84/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Xuất ra vị trí của từng ký tự có trong chuỗi

Ví dụ: `S= "Sài Gòn"`, xuất ra

```
Ký tự S xuất hiện tại vị trì 0
Ký tự à xuất hiện tại vị trì 1
Ký tự i xuất hiện tại vị trì 2
Ký tự   xuất hiện tại vị trì 3
Ký tự G xuất hiện tại vị trì 4
Ký tự ò xuất hiện tại vị trì 5
Ký tự n xuất hiện tại vị trì 6
```

85/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm ký tự đầu tiên xuất hiện nhiều hơn 1 lần

Ví dụ: `S= "Sài Gòn"`, xuất ra `Ký tự S xuất hiện tại vị trì 0`

1.3. try ... except ... finally

86/. Cho biến x=2 và biến y không gán giá trị. Viết chương trình kiểm tra xem biến x và y đã được khai báo và gán giá trị hay chưa?

87/. Cho người dùng nhập 1 số thực. In ra trị tuyệt đối của số đó. Cần xét khả năng người dùng không nhập ký số. Khi đó cần báo lỗi 'Not numeric'

88/. Cho người dùng nhập 1 số nguyên n. In ra bảng cửu chương của n. Nếu n nhập vào không phải là số nguyên thì báo lỗi và yêu cầu nhập lại.

89/. Cho người dùng nhập 2 số nguyên. In ra tổng hiệu tích thương của 2 số đó. Cần xét các khả năng gây lỗi.

📖 *Gợi ý*: sử dụng các exception `ValueError, ZeroDivisionError, Exception`

90/. Viết chương trình cho người dùng nhập số nguyên dương n, chương trình sẽ in ra n số ngẫu nhiên từ 0 đến 100. Nếu người dùng nhập kiểu dữ liệu khác (float, bool, ...) chương trình sẽ hiện ra câu 'Phải nhập số nguyên dương' và cho lặp lại việc nhập n. Chương trình chỉ kết thúc khi người dùng nhập từ 'exit' hoặc nhập số 0 (zero).

1.4. Datetime module

91/. Viết chương trình sử dụng phương thức strftime của đối tượng thuộc class date hoặc datetime để in ra các thông tin như sau:

Thông tin cần hiển thị	*Kết quả hiển thị (ví dụ ngày hiện tại là 18/9/2019)*
Ngày giờ hiện tại	2019-09-18 19:25:58.737442
Năm hiện tại	2019
Tháng hiện tại bằng chữ	September
Tuần hiện tại là tuần thứ mấy trong năm	37
Tuần hiện tại là tuần thứ mấy trong tháng	3
Ngày hiện tại là ngày thứ mấy trong năm	261
Ngày dương lịch hiện tại là ngày	18
Thứ của ngày hiện tại	Wednesday
Giờ phút giây hiện tại	19:25:58

92/. Cho nhập ngày, tháng, năm của 2 ngày. Cho biết số ngày cách nhau giữa 2 ngày.

93/. Cho nhập 1 chuỗi ngày (S) theo dạng **`'Sep 18 2019 2:43PM'`**. Chuyển chuỗi S sang kiểu ngày.

94/. Lấy giá trị khởi đầu (0 giờ, 0 phút, 0 giây) của ngày hiện tại. VD: `2019-09-18 00:00:00`

95/. Sử dụng *datetime.timedelta* để thêm 5 giây vào thời gian hiện tại

96/. Cho nhập tháng (m) và năm dương lịch (y), yêu cầu thực hiện:

a. Cho biết năm y có là năm nhuận hay không?
b. Tháng m của năm y có bao nhiêu ngày?
c. Ngày cuối cùng của tháng m là thứ mấy?
d. In ra lịch của tháng và năm tương ứng. Lưu ý cần kiểm tra giá trị của tháng phải nằm trong khoảng từ 1 đến 12.

```
Input the year : >? 2019
Input the month : >? 8
    August 2019
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

1.5. Xây dựng hàm

97/. Cho nhập 2 số a và b. Lần lượt thực hiện hoán vị 2 biến a, b bằng 2 cách sau:

- Không sử dụng hàm
- Có sử dụng hàm

98/. Viết chương trình cho nhập số nguyên n. In ra các ước số của n và số lượng ước số của n.
Ví dụ: với n=15, chương trình sẽ in ra

```
Các ước số của  15 là: [1, 3, 5, 15]
Số lượng ước số của 15 là: 4
```

99/. Viết chương trình chuyển đổi giữa 3 đơn vị đo lường cm, inch, foot bằng cách tạo menu như minh họa sau. Yêu cầu mỗi chuyển đổi sẽ được tổ chức thành 1 hàm riêng:

```
CHƯƠNG TRÌNH CHUYỂN ĐỔI GIỮA 3 ĐƠN VỊ cm, inch và foot
    1- Từ cm sang inch
    2- Từ cm sang foot
    3- Từ inch sang cm
    4- Từ inch sang foot
    5- Từ foot sang cm
    6- Từ foot sang inch
    0- Kết thúc chương trình
Bạn chọn chức năng (0-6):
```

Biết rằng: centimeter (cm) = Inch / 2.54
inch (in) = cm x 0.3937008
foot (ft) = inches x 12

Sau khi hoàn tất, mở rộng thêm cho các đơn vị yard và mile
yard (yd) = feet x 3
mile (mi) = yards x 1.760

100/. Viết hàm nhận tham số là 1 chuỗi (S). Nếu 2 ký tự đầu của s không phải là 'Is' thì thêm 'Is' vào trước s, ngược lại, nếu đã có thì trả về chuỗi s ban đầu.

101/. Viết hàm nhận tham số là 2 số nguyên. Nếu 1 trong 2 tham số không phải kiểu số nguyên thì sử dụng lệnh *raise* để định nghĩa bổ sung câu báo lỗi là "*Hàm chỉ nhận 2 tham số là số nguyên*" cho *TypeError Exception.*

102/. Mỗi bài tập sau đây (từ câu *a* đến câu *e*) sẽ được viết thành các hàm riêng biệt.

<u>*Yêu cầu*</u>: Cho người dùng nhập 1 số nguyên dương ***n***, kiểm tra nếu n không phải kiểu số nguyên thì cho nhập lại, nếu đúng là kiểu số nguyên thì cho kiểm tra tiếp điều kiện của từng yêu cầu (từ câu *a* đến câu *e*). Nếu tất cả đều hợp lệ, hàm trả về ***n*** để chương trình thực hiện tiếp các yêu cầu còn lại của từng câu.

a. Điều kiện: ***n*>0**. Nếu nhập đúng, chương trình in ra các số chẵn nhỏ hơn ***n***.
b. Điều kiện: **0 <= *n* <= 9**. Nếu nhập đúng, chương trình in ra cách đọc của số vừa nhập.
c. Điều kiện: ***n* <10 hoặc *n* > 50**. Nếu nhập đúng, chương trình in ra ***n*** số ngẫu nhiên.
d. Điều kiện: ***n* bội số của 5**. Nếu nhập đúng, chương trình in ra bảng cửu chương 5.
e. Điều kiện: ***n* nằm trong các số (2, 4, 6, 8, 10)**. Nếu nhập đúng, chương trình cho nhập tiếp 2 số nguyên ***i, j***. Hãy tính và in ra cấp số cộng với số hạng đầu là i, công sai là j và số lượng phần tử của dãy là ***n***.

103/. Viết hàm nhận tham số là 1 số nguyên n. In ra chuỗi số Fibonacci gồm n số.
Ví dụ: với `n= 9`, sẽ in ra `1 1 2 3 5 8 13 21 34`

104/. Tạo 1 list gồm n số nguyên với giá trị ngẫu nhiên từ 0 đến 9. Viết 2 hàm nhận tham số là list vừa tạo, 1 hàm thực hiện vẽ biểu đồ (histogram) ngang và 1 hàm thực hiện vẽ biểu đồ dọc. Biết biểu đồ được vẽ bằng dấu hoa thị (*). Minh họa:

```
n=4
lst=[2,0,7,3]
```

```
*  *
*  *  *  *  *  *  *
*  *  *
```

Biểu đồ ngang

```
*
*
*
*
*  *
*  *  *
*  *  *
```

Biểu đồ dọc

105/. Viết chương trình cho nhập 2 số a và b. Viết hàm tìm ước số chung lớn nhất (greatest common divisor - GCD) của 2 số a và b rồi in kết quả ra màn hình.

106/. Viết chương trình tìm bội số chung nhỏ nhất (LCM - Least Common Multiple hay Lowest Common Multiple) của các số từ 1 đến 10. In ra LCM và thời gian thực hiện việc tìm LCM.

☞Mở rộng: Thực hiện tương tự nhưng tìm LCM cho các số từ 1 đến 100.

107/. Viết chương trình lặp lại nhiều lần việc cho người dùng nhập 1 số nguyên n, xét xem số đó có phải là số nguyên tố hay không? Chương trình chỉ kết thúc khi người dùng nhập n=-1 hoặc nhấn tổ hợp phím Ctrl+C

108/. Nhập vào 3 số nguyên dương a, b, c. Kiểm tra xem 3 số đó có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy cho biết:

- Chu vi, diện tích của tam giác
- Chiều dài mỗi đường cao của tam giác.
- Tam giác đó thuộc loại nào? (vuông cân, cân, vuông, đều, …).

🏳 ***Nhắc lại:***

- Công thức tính diện tích **`s = sqrt(p*(p-a)*(p-b)*(p-c) )`**
- Công thức tính các đường cao: $h_a = 2s/a$, $h_b=2s/b$, $h_c=2s/c$.

(Với p là nữa chu vi của tam giác).

🏳 ***Yêu cầu***: tổ chức chương trình thành các hàm chức năng,mỗi hàm chỉ giải quyết 1 công việc nhất định. Vì dụ: hàm xét tam giác có hợp lệ hay không?, hàm xét tam giác đó có phải là tam giác đều hay không? ...

1.6. Xây dựng hàm đệ quy (Recursion)

109/. Cho nhập số nguyên n. Tính tổng các chữ số có trong n

Ví dụ: n=345 ⇒ tổng = 12

110/. Viết hàm tính giai thừa (factorial number) của một số nguyên dương (n).
Với 1! = 1; 2! = 1 x 2; 3! = 1 x 2 x 3; ... n! = 1 x 2 x 3 x ... x n

111/. Viết hàm nhận tham số là 1 số nguyên dương (n). Tính số hạng thứ n của chuỗi Fibonaci. Ví dụ với n=7 sẽ in ra số 13, n=8 sẽ in ra 21, ...

🏳 Nhắc lại: Chuỗi số Fibonacci: 1 1 2 3 5 8 13 ...

Dãy Fibonaci được Định nghĩa truy hồi như sau:

- $F_0 = F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ nếu n>=2

112/. Viết hàm nhận 2 tham số là 2 số nguyên dương a, b.

a. Tính a^b. Ví dụ: a=2, b=3 ⇒ 2^3=8

b. Tìm ước số chung lớn nhất (greatest common divisor -gcd) của a và b.

Ví dụ: với a=2 và b=7 ⇒ gcd(a,b)=1
với a=6 và b=18 ⇒ gcd(a,b)=6

113/. Viết hàm nhận tham số là 1 số nguyên dương (n). Tính tổng của S trong các trường hợp sau:

a. `S = n+ (n-2) + (n-4) + ...` cho đến khi (n-x)<=0.

Ví dụ: `n=6 ⇒ tổng = 6 + 4 + 2 = 12`
`n=7 ⇒ tổng = 7 + 5 + 3 + 1 = 16`

b.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

c.

$$S = 1 + \frac{1}{2^1} + \frac{1}{2^2} + \ldots + \frac{1}{2^n}$$

Ví dụ: `với n=2 sẽ in ra S=1.75`
`với n = 3 sẽ in ra S = 1.875`

114/. Viết hàm nhận tham số là 1 list.

a. Tính tổng các giá trị có trong List trong trường hợp các phần tử trong List là phần tử đơn. Ví dụ: `[2, 4, 5, 6, 7]`

b. Tính tổng các giá trị có trong List trong trường hợp các phần tử trong List có thể là 1 phần tử đơn hoặc 1 subList. Ví dụ: `[1, 2, [3,4],[5,6]]`

c. In các giá trị có trong List theo chiều từ đầu đến cuối list

d. In các giá trị có trong List theo chiều từ cuối về đầu list

115/. Viết hàm nhận 2 tham số là số nguyên, vơi1 tham số thứ nhất số nguyên hệ thập phân (n) và tham số thứ hai (base) cơ số cần đổi (2,8,16). Thực hiện đổi n sang cơ số base. Kết quả trả về của hàm là chuỗi kết quả.

Ví dụ:
```
n=179, base=2   ⇒ 179 Dec= 10110011 Binary
n=179, base=8   ⇒ 179 Dec= 263 Octal
n=179, base=16  ⇒ 179 Dec= B3 Hexa Decimal
```

116/. Giả sử chuỗi S được gọi là chuỗi tuần tự 4 khi chuỗi được bắt đầu bởi 4 số 1, giá trị từ số thứ 5 trở đi sẽ bằng tổng của 4 số liền trước đó. Ví dụ: S= 1114713254994. . .Viết chương trình cho nhập số nguyên dương n. Hãy in ra số thứ n của dãy chuỗi (S).

Ví dụ:
```
n=5 in ra 4
n=6 in ra 7
n=7 in ra 13
n=8 in ra 25
```

117/. Biết lãi suất vay được tính theo lũy tiến (lời tháng trước được cộng dồn vào vốn vay) và lãi suất 1 tháng là 5%. Cho người dùng nhập số tiền (X) và số tháng cần vay (m). Cho biết số tiền khách hàng phải trả sau thời gian m tháng. Yêu cầu: nếu số tiền vay dưới 1 triệu thì số tiền kết quả sẽ làm tròn đến phần trăm, ngược lại sẽ làm tròn số tiền đến phần ngàn.

1.7. Bitwise Operator

118/. (*) Viết hàm thực hiện phép cộng giữa 2 số nguyên a và b nhưng không sử dụng phép cộng (‘+’) mà chỉ sử dụng các phép toán trên bit.

🏳 ***Gợi ý***

```
Thực hiện lặp khi b!=0
     data= a AND b
     a = a XOR b
     b= data SHIFT LEFT 1
return a
```

1.8. Anonymous function (hàm ẩn danh)

119/. Viết chương trình Python sử dụng lambda để tính cho các trường hợp sau:

a) Hàm nhận 1 đối số là a và trả về giá trị của a+15.

b) Hàm nhận 2 đối số là x và y, trả về tích của x và y.

c) Hàm nhận 1 đối số là n. Cho biết n có là bội số của 13 hoặc 19 hay không?

120/. Viết chương trình Python để tạo một hàm có một đối số và đối số đó sẽ được nhân với một số đã cho.

121/. Viết chương trình sử dụng lambda để tính:

a. Diện tích, chu vi hình tròn với tham số bán kính là r.

📖 *Hướng dẫn*: tính diện tích `s_tron = lambda r: math.pi * math.pow(r,2)`

b. Tính diện tích, chu vi hình chữ nhật với 2 tham số rộng là r và dài là d.

122/. Số Palindrome

📖 *Số Palindrome (hoặc Palindromic)*

- (Theo Wikipedia) Là một số vẫn giữ nguyên giá trị khi các chữ số của nó được đảo ngược. Hay nói cách khác là số đối xứng.
- 30 số thập phân palindrome đầu tiên là: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101 , 111, 121, 131, 141, 151, 161, 171, 181, 191, 202, ...

📖 *Số nguyên tố Palindrome*

- (Theo Wikipedia) Là số nguyên tố viết xuôi hay viết ngược vẫn chỉ cho ra một số. Có vô hạn số Palindrome, nhưng không rõ tập số nguyên tố Palindrome có vô hạn hay không, vì phần nhiều các số loại này là hợp số.
- Các số nguyên tố Palindrome dưới 20000 gồm: 2, 3, 5, 7, 11, 101, 131, 151, 181, 191, 313, 353, 373, 383, 727, 757, 787, 797, 919, 929, 10301, 10501, 10601, 11311, 11411, 12421, 12721, 12821, 13331, 13831, 13931, 14341, 14741, 15451, 15551, 16061, 16361, 16561, 16661, 17471, 17971, 18181, 18481, 19391, 19891, 19991.

Yêu cầu: Sử dụng lambda, viết chương trình

a. In ra các số Palindrome từ 0 đến 100000.
b. In ra các số nguyên tố Palindrome từ 0 đến 100000.

123/. Viết chương trình cho nhập 1 chuỗi S, lần lượt sử dụng lambda để xét các trường hợp sau:
a. Chuỗi S có bắt đầu bằng ký tự **'P'** (in hoa) hay không?
b. Chuỗi S có bắt đầu bằng ký tự **'P'** (in hoa) hoặc **'p'** (in thường) hay không?
📖 *Gợi ý*: sử dụng phương thức **strObject.startswith(strOthers)** để kiểm tra. Phương thức này sẽ trả về kết quả là **True** hoặc **False**.
c. Chuỗi S có bắt đầu bằng 1 nguyên âm hay không?
d. Chuỗi S có phải là chuỗi *palindrome* hay không? Một chuỗi được gọi là *palindrome* khi các ký tự trong chuỗi đối xứng nhau.
Ví dụ: chuỗi sau là chuỗi *palindrome:* `ABLE WAS I ERE I SAW ELBA`

124/. Viết chương trình sử dụng lambda để lần lượt tách ngày, tháng, năm và giờ phút giây của thời điểm hiện tại

Ví dụ: thời điểm hiện tại là `2019-06-08 10:13:35.232478`
Sẽ in ra màn hình:
```
Năm hiện tại: 2019
Tháng hiện tại: 6
Ngày hiện tại: 8
Thời gian hiện tại: 10:13:35.232478
```

Sinh viên tự tìm hiểu thêm các module khác viết chương trình

125/. Viết chương trình sử dụng lambda với đối số là n, với n là số lượng số đầu tiên trong dãy Fibonacci. Hàm trả về 1 list chứa n số trong dãy Fibonacci
📖 *Gợi ý*: sử dụng hàm reduce trong module functools

2 CÁC ĐỐI TƯỢNG DẠNG DANH SÁCH TRONG PYTHON

2.1. Lists

2.1.1. List

1/. Cho trước 1 list như sau:

```
datalist = [1452, 11.23, 1 + 2j, True, 'w3resource',
                        (0, -1), [5, 12], {"class": 'V', "section": 'A'}]
```

Viết chương trình cho biết kiểu dữ liệu của từng thành phần trong list.

Gợi ý: dùng hàm ***type(object)*** *để biết kiểu dữ liệu của object. Các kiểu cần xét trong bài tập này là* ***int, float, complex, bool, list, tuple, dict, str****. Nếu kiểu của dữ liệu không thuộc các loại vừa liệt kê, chương trình in ra Unknown*

2.1.2. Number List

2/. Cho nhập 1 số nguyên n, tạo list L gồm n phần tử với giá trị ngẫu nhiên nguyên dương từ 0 đến 100. Viết hàm nhận tham số là list vừa có, hàm thực hiện in ra các số chẵn có trong list L. Hàm kết thúc khi đã xét hết các giá trị có trong list L hoặc ngay sau khi in giá trị là 99 (giá trị 99 này có thể có trong list khi phát sinh ngẫu nhiên)

3/. Cho nhập 1 số nguyên n, tạo list L gồm n phần tử với giá trị ngẫu nhiên từ 0 đến 100. Viết hàm trả về chuỗi số kết nối từ các số có trong list L. Lưu ý chuỗi kết quả có kèm dấu ngăn cách phần ngàn như minh họa sau:

```
L = [52, 99, 3, 60, 93, 97] ⇒    52.993.609.397
```

4/. Cho nhập n. Xét xem n có phải là số nguyên tố hay không?

Yêu cầu: sử dụng list chứa các ước số của n. Sau đó xét nếu list rỗng thì n là số nguyên tố, ngược lại n không là số nguyên tố. Lần lượt thực hiện tạo list bằng 2 cách: cách thông thường và List Comprehensions.

5/. Tạo 1 danh sách (list), ví dụ:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

a. Viết chương trình in tất cả các phần tử (elements) trong list có giá trị nhỏ hơn 5 ra màn hình.
b. *Mở rộng 1*: Thay vì in trực tiếp các phần tử ra màn hình, chương trình tạo ra 1 list mới chứa các phần tử có giá trị nhỏ hơn 5. Sau đó in tất cả các số có trong list vừa tạo ra màn hình.
c. *Mở rộng 2*: In các giá trị trong list trên cùng 1 dòng.
d. *Mở rộng 3*: Cho người dùng nhập số m. Chương trình in ra các elements trong list có giá trị là bội số của m.

6/. Viết chương trình cho người dùng nhập nhiều lần các số nguyên dương. Việc nhập sẽ kết thúc khi người dùng nhập số âm. Đưa tất cả các số đã nhập (không kể số âm nhập cuối cùng) vào list L. Tính:

a. Tổng các số có trong List
b. Tổng khoảng cách giữa từng cặp số có trong L.
c. Cho nhập số x. Tìm xem x có trong list các số vừa nhập hay không? Nếu có, cho biết x xuất hiện bao nhiêu lần
d. X có lớn hơn tất cả các số có trong list hay không? Nếu không, hãy in ra các số có trong list và lớn hơn x.

Ví dụ: list `L= [1,2,3]in ra 'Tổng khoảng cách giữa các số là: 4'`
Giải thích: vì `|1-2| + |2-3| + |1-3| = 1 + 1 + 2 = 4`

7/. Viết chương trình cho người dùng nhập nhiều lần các số nguyên dương. Sau mỗi lần nhập, chương trình sẽ hỏi người dùng có muốn nhập nữa hay không (Yes / No). Nếu chọn Yes (Y) thì cho người dùng nhập tiếp. Ngược lại nếu chọn No (N), chương trình sẽ thực hiện:

a. In ra các số nguyên tố có trong list.
b. Tính trung bình cộng các số âm, trung bình các số dương
c. Số lớn nhất, số nhỏ nhất
d. Cho biết các số trong list có được sắp xếp tăng dần hay chưa?

8/. Cho nhập 3 số nguyên a, b, c. Cho biết 3 số đó có tạo thành 1 cấp số cộng hay 1 cấp số nhân hoặc không là cả 2 loại này. Nếu là cấp số cộng hay 1 cấp số nhân, hãy in ra số kế tiếp của dãy đó.

🏳 *Giải thích*: Theo Wikipedia,

- Một cấp số cộng (*Arithmetic Progression* hoặc *Arithmetic Sequence*) là một dãy số thoả mãn điều kiện: hai phần tử liên tiếp nhau sai khác nhau một hằng số gọi là công sai (*common difference*). Chẳng hạn, dãy số 3, 5, 7, 9, 11, ... là một cấp số cộng với công sai là 2.
- Một cấp số nhân (*Geometric Progression* hoặc *Geometric Sequence*) là một dãy số thoả mãn điều kiện kể từ số hạng thứ hai, mỗi số hạng đều là tích của số hạng đứng ngay trước nó với một số không đổi. Hằng số này được gọi là công bội (*common ratio*) của cấp số nhân. Chẳng hạn, dãy số 3, 15, 45, 135, ... là một cấp số nhân với công bội là 3.

Ví dụ:

```
Nhập 1,2,3 in ra AP sequence. Next number of the sequence: 4
Nhập 2,6,18 in ra GP sequence. Next number of the sequence:  54.0
Nhập 1,3,2 in ra Not an Arithmetic Progression and Geometric
Progression sequence
```

9/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. Tạo 1 list chứa tất cả các số nguyên i sao cho i<=n và n chia chẵn cho i. In list ra màn hình.

10/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. sắp xếp các số trong n thành từ nhỏ đến lớn (nMin) và từ lớn đến nhỏ (nMax). Tìm hiệu của nMax và nMin.

Ví dụ `nhập n=1423`

`⇒ nMin=1234, nMax=4321, nMax – nMin = 4321-1234 = 3087`

🏳 ***Gợi ý***: Đưa từng số trong n vào list. Sử dụng các phương thức join và sort của dữ liệu chuỗi để tìm nMin và nMax, từ đó suy ra kết quả hiệu

11/. Viết chương trình cho người dùng thực hiện lặp đi lặp lại nhiều lần việc nhập 2 số nguyên sl (số lượng) và tong(tổng). Chương trình tạo ra tổ hợp các số từ 0 đến 9 gồm sl phần tử và tổng các phần tử trong tổ hợp bằng với giá trị tong vừa nhập. In các tổ hợp thỏa điều kiện và số lượng tổ hợp thỏa điều kiện. Chương trình kết thúc khi người dùng nhập sl=0 và tong=0.

Ví dụ

```
nhập sl=2 và tong=7. Kết quả sẽ in ra
        Các tổ hợp gồm  2 phần tử có tổng = 7 là:
        (0, 7) (1, 6) (2, 5) (3, 4)
        Có 4 tổ hợp thỏa điều kiện
```

🏳 ***Gợi ý***: sử dụng ***`itertools.combinations`*** để tạo ra các tổ hợp

12/. Tạo ngẫu nhiên số nguyên n đại diện cho số lượng phần tử sẽ có trong listA (với 5<=n<=20).

- Tạo listA gồm n phần tử với giá trị ngẫu nhiên từ 1 đến 100.
- Tạo listB bằng cách chỉ chọn những số chẵn có trong listA.
- In 2 list ra màn hình.

13/. Viết chương trình tạo 2 danh sách (A, B) với giá trị ngẫu nhiên (có thể trùng nhau) trong khoảng 1-X bằng cách cho người dùng nhập:

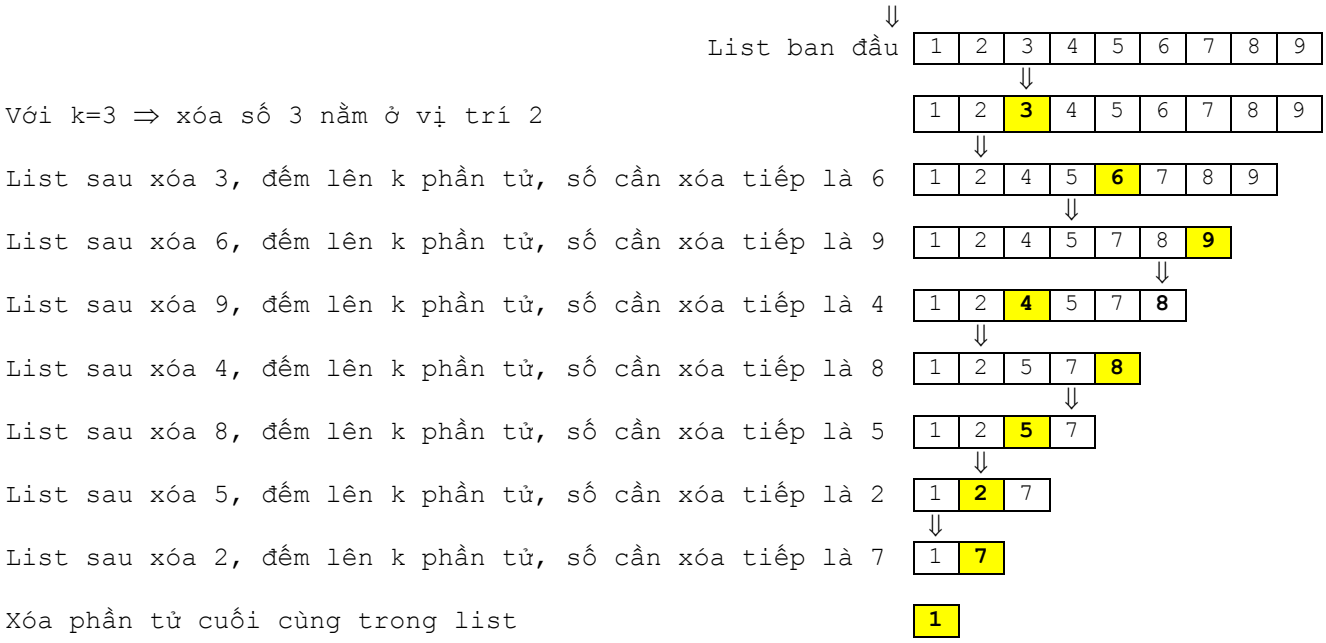
- Giá trị của X.
- Số lượng phần tử có trong từng list nA, nB.
- Range của số ngẫu nhiên sẽ được tạo (Giá trị nhỏ nhất và giá trị lớn nhất)

a. Tạo danh sách Result chứa những số có trong cả A và B.

b. *Mở rộng*: loại đi những số trùng nhau (nếu có) trong Result.

14/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. Tạo 1 list L chứa các số nguyên với giá trị ngẫu nhiên. Viết hàm xóa các phần tử trong L theo thứ tự cách khoảng k cho đến khi L rỗng. Khi xóa số X ra khỏi list L, chương trình sẽ in số X ra màn hình.

Minh họa list gồm n=9 phần tử và khoảng cách k=3:



Kết quả xuất ra màn hình theo thứ tự remove là: **3, 6, 9, 4, 8, 5, 2, 7, 1,**

🏳 *Mở rộng*: Thực hiện xóa list khi cho người dùng nhập giá trị của k.

15/. Cho người dùng nhập chuỗi (S) là 1 địa chỉ IP (IP Address). Thực hiện xóa các số 0 (zero) không có nghĩa.

Ví dụ: `S= '255.024.01.023'` ⇒ sẽ in ra `192.24.1.23`

16/. Cho người dùng nhập chuỗi (S) là 1 chuỗi số hệ nhị phân. Viết chương trình tìm độ dài lớn nhất chứa các số 0 liên tiếp trong S.

Ví dụ: `S= '1110000100000110'` ⇒ sẽ in ra `5`

17/. Lần lượt cho nhập 3 giá trị: chuỗi, số nguyên, số thực vào 3 biến a, b, c. Lần lượt đưa giá trị của 3 biến vào list L. Sau đó sử dụng index để đưa 3 giá trị đang có trong list L vào 3 biến x, y, z bằng 2 cách sau

- Cách 1: dùng 3 dòng lệnh
- Cách 2: dùng 1 dòng lệnh

18/. Viết chương trình cho người dùng nhập 1 số nguyên dương n. Tạo 1 list L chứa các số nguyên với giá trị ngẫu nhiên nằm trong khoảng từ -10 đến +10. Viết hàm tìm các đoạn đầu tiên gồm 3 (gọi là q) phần tử liên tiếp nhau có tổng bằng 0 (gọi là k). Lưu ý có thể có nhiều đoạn cùng có tổng=k

🏳 *Mở rộng*:

(i)- Thực hiện tương tự nhưng cho xuất ra tất cả các đoạn có tổng bằng k.

(ii)- Thực hiện tương tự nhưng cho người dùng nhập giá trị của q và k.

19/. Cho nhập 1 số nguyên n, cho biết số lượng từng ký số có trong n. Lưu ý: chỉ in ra các ký số có xuất hiện. Yêu cầu sử dụng list để lưu biến đếm.

Ví dụ: n=1008 ⇒ in ra
```
Số 1008 gồm:
Số 0 có 2 số
Số 1 có 1 số
Số 8 có 1 số
```

20/. (*) Viết chương trình tạo 1 list L gồm n phần tử có giá trị ngẫu nhiên, với n do người dùng nhập. Viết hàm tạo tất cả các hoán vị (permute) có thể có từ listL.

Ví du: `L=[1, 2, 3]`

⇒ `[[3,2,1], [2,3,1], [2,1,3], [3,1,2], [1,3,2], [1,2,3]]`

☞ *Gợi ý*:

- Đầu tiên tạo `resultList` rỗng
- Lần lượt duyệt từng phần tử X trong list L để tạo một (hoặc nhiều) newList sẽ chứa X và các hoán vị sẽ có với các List đã có trong đó.

  Minh họa: `L=[1, 2, 3]`
  - `resultList =[ [] ]`
  - Xét số đầu tiên (số 1)
  - Tạo 1 hoặc nhiều newList bằng cách kết hợp số 1 với các subList đang có trong resultList ⇒ `resultList =[ [1] ]`
  - Xét số kế tiếp (số 2)
  - Tạo 1 hoặc nhiều newList bằng cách kết hợp số 2 với các subList đang có trong resultList bằng cách chèn số 2 vào các vị trí từ 0 đến len(subList) ⇒ `resultList =[ [2,1], [1,2] ]`
  - Xét số kế tiếp (số 3)
  - Tạo 1 hoặc nhiều newList bằng cách kết hợp số 3 với các subList đang có trong resultList bằng cách chèn số 3 vào các vị trí từ 0 đến len(subList) ⇒
    - Chèn số 3 vào sublist [2,1] (đang có trong resultList) sẽ phát sinh ra được 3 sublist khác là [3,2,1], [2,3,1], [2,1,3].
    - Tương tự, chèn số 3 vào sublist [1,2] (đang có trong resultList) sẽ phát sinh ra được thêm 3 sublist nữa là [3,1,2], [1,3,2], [1,2,3].

21/. Tìm các số nguyên tố nhỏ hơn 1000 bằng giải thuật sàng Erastosthene (giải thuật sàng Erastosthene dùng phương pháp đánh dấu để loại bỏ những số không phải là số nguyên tố. Giải thuật có từ một nhận xét rằng nếu k là số nguyên tố thì các số 2*k, 3*k,... n*k sẽ không là số nguyên tố (vì đã vi phạm định nghĩa về số nguyên tố).

Yêu cầu: sử dụng list để tìm ra các số nguyên tố <1000 dựa trên giải thuật Erastosthene.

22/. (*)Viết chương trình minh họa cho những số <100 về giả thuyết của nhà toán học người Đức - *Christian Goldbach* (1690-1764, trong một bức thư ngày 07/6/1742 gửi cho nhà toán học *Leonhard Euler*): ***mọi số tự nhiên chẵn lớn hơn 2 đều có thể viết được thành tổng của hai số nguyên tố*** (giả thuyết này đã được chỉ ra là đúng tới 4×10^{18}).

Ví dụ: `4 = 2 + 2, 8 = 5 + 3, 20 = 13 + 7`

23/. Cho nhập 1 dãy các hệ số nhị phân (0,1) ngăn cách nhau bởi dấu phẩy. In ra các số nhị phân trong dãy chia chẵn cho 5 (hệ thập phân). Kết quả in chính là dãy nhị phân đã nhập.

Ví dụ: `S= '0000,00101,00110,010100,10101,11010,111100'`

```
        # tương dương với các giá trị thập phân là 0, 5, 6, 20, 21,31,60
```

In ra *0000,00101,010100,111100*

```
                # tương dương với các giá trị thập phân là 0, 5, 20, 60
```

24/. Cho nhập 1 dãy số cách nhau bởi dấu phẩy (s). Chương trình phát sinh ra 1 list từ chuỗi số trên. Ví dụ: `s = '1, 3, 9, 2, 8, 4, 7'`

⇒ `list=[1, 3, 9, 2, 8, 4, 7]`

2.1.3. String List

25/. Tạo trước 1 list chứa tên gọi của các con thú (ví dụ: `['ant', 'bear', 'cat', 'dog', 'elephant', 'fish', 'goat', 'hippo']`). Cho người dùng nhập tên con thú cần tìm, in ra:

- Danh sách các con thú
- Số lượng thú có trong danh sách
- Kết quả tìm kiếm: có hay không có con thú cần tìm.

26/. Cho người dùng nhập 1 danh sách tên các màu cách nhau bởi dấu phẩy (,), ví dụ: Red, Black, Green, White, Orange.

a. Viết chương trình loại bỏ màu đầu tiên có trong list L

b. Viết chương trình sắp xếp danh sách tên các màu theo thứ tự alphabet tăng dần

126/. Viết chương trình cho người dùng nhập 1 chuỗi S. Cho biết từ dài nhất và từ ngắn nhất trong S. Ví dụ:

```
S=‘ Write a Java program to sort an array of given integers using Quick
sort Algorithm.
```

⇒ `từ dài nhất: Algorithm.`

⇒ `từ ngắn nhất: a`

27/. Viết chương trình cho người dùng nhập 1 chuỗi (*S*) và ký tự dùng để phân chia chuỗi (c). Thực hiện phân chia chuỗi S thành các phần sao cho các phần đó không còn chứa các ký tự dùng để phân chia c.

Ví dụ: `S = 'https://www.w3resource.com/python-exercises/string' và c='/'`

Kết quả: `['https:', '', 'www.w3resource.com', 'python-exercises', 'string']`

28/. Viết chương trình tạo cho nhập 1 chuỗi S có chứa 2 từ “Python và “Java”. Chương trình thực hiện hoán đổi 2 từ này cho nhau.

Ví du: `S=’Python is popular than Java’`

`Kết quả in ra: ’ Java is popular than Python’`

🏳 *Gợi ý*:

- Tách các từ trong S vào 1 list.
- Duyệt từ đầu đến cuối list để thay thế từ.

🏳 *Mở rộng:* Cho người dùng nhập chuỗi S và 2 từ tùy ý cần hoán đổi vị trí cho nhau.

29/. Cho nhập chuỗi (S). Đếm xem trong s có bao nhiêu ký tự là nguyên âm (AEIOU) mỗi loại. Lưu ý không phân biệt chữ hoa/thường.

30/. Cho nhập chuỗi (S). In và in ra tất cả các từ cùng có chiều dài lớn nhất có trong chuỗi.

Ví dụ: `S='PHP, Exercises, Backend, Databases'`

`Sẽ in ra:  Exercises  Databases`

31/. Viết chương trình nhập vào năm. In ra tên của năm âm lịch tương ứng.

Ví dụ: nhập năm *2019* in ra *Kỷ Hợi*.

Biết rằng:

CAN	*Giáp*	*At*	*Bính*	*Đinh*	*Mậu*	*Kỷ*	*Canh*	*Tân*	*Nhâm*	*Quý*	
CHI	*Tý*	*Sửu*	*Dần*	*Mão*	*Thìn*	*Tỵ*	*Ngọ*	*Mùi*	*Thân*	*Dậu*	*Tuất*	*Hợi*

32/. Trong mật mã học, mật mã Caesar (*Caesar Cipher*, còn được gọi là mật mã dịch chuyển - *Shift Cipher*, mã của Caesar hoặc dịch chuyển Caesar), là một trong những kỹ thuật mã hóa (*Encryption*) đơn giản và được biết đến rộng rãi nhất. Nó là một loại mật mã thay thế, trong đó mỗi chữ cái trong bản rõ (plain text) được thay thế bằng một chữ cái một số vị trí cố định trong bảng chữ cái dựa trên khóa k (key). Ví dụ: với dịch chuyển trái 3, D sẽ được thay thế bằng A, E sẽ trở thành B, v.v. Phương pháp này được đặt theo tên của Julius Caesar, người đã sử dụng nó trong thư tín riêng của mình.

Thông điệp được mã hóa bằng cách dịch chuyển xoay vòng từng ký tự đi *k* vị trí trong bảng chữ cái. Trường hợp với k=3 gọi là phương pháp *mã hóa Caesar*.

☞ *Minh họa*: Cho *plain text* = $x_1, x_2, x_3, x_4, x_5, x_6$ = 'ATTACK', k=17, chiều dịch chuyển của ký tự là từ trái sang phải và bảng chữ cái các ký tự được mã hóa như hình sau (như vậy, nếu các ký tự ngoài những ký tự này sẽ giữ nguyên mà không thực hiện mã hóa:

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
0	1	2	3	4	5	6	7	8	9	10	11	12
N	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
13	14	15	16	17	18	19	20	21	22	23	24	25

– Thực hiện mã hóa:

$y_1 = x1 + k \bmod 26 = 0 + 17 \bmod 26 = 17 \Rightarrow R$

$y_2 = 19 + 17 \bmod 26 = 10 \Rightarrow K$

$y_3 = 10 \Rightarrow K$

$y_4 = 17 \Rightarrow R$

$y_5 = 2 + 17 \bmod 26 = 19 \Rightarrow T$

$y_6 = 10 + 17 \bmod 26 = 1 \Rightarrow B$

Bản mã hoàn chỉnh (Cipher text) $Y = y_1; y_2; y_3: y_4: y_5; y_6 = RKKRTB$

– Thực hiện giải mã: với k=17 và cipher text (*Y*) = *y*1*; y2; : : : ; y*6 = *RKKRTB*

x1 = y1 - k mod 26 = 17 - 17 mod 26 = 0 ⇒ *A*

x2 = 10 - 17 mod 26 = -7 mod 26 = 19 ⇒ *T*

x3 = 19 ⇒ *T*

x4 = 0 ⇒ *A*

x5 = 19 - 17 mod 26 = 2 ⇒ *C*

x6 = 1 - 17 mod 26 = -16 mod 26= 10 ⇒ *K*

Bản giải mã hoàn chỉnh (bản gốc - plan text) *X = x*1*; x*2*; … ; x*6 = *ATTACK*

☞ *Yêu cầu*: viết chương trình tạo menu với các chức năng sau, trong đó khi thực hiện mã hóa hoặc giải mã, chương trình sẽ yêu cầu nhập khóa ***k***. Lưu ý: xem như văn bản chỉ gồm các ký tự chữ hoa.

a. Nhập chuỗi
b. Xuất chuỗi
c. Mã hóa (*Encryption*)
d. Giải mã (*Decryption*)

🏳 *Mở rộng*:

- Văn bản bao gồm cả ký tự hoa và thường.
- Văn bản bao gồm tất cả các ký tự trong bộ mã ASCII nhưng bỏ qua các mã điều khiển (từ 0-30).
- Cho người dùng chọn chiều dịch chuyển (trái hoặc phải).

33/. Viết chương trình cho nhập 1 chuỗi (S). In ra ký tự xuất hiện nhiều nhất trong S. Lưu ý: có thể có nhiều ký tự cùng đạt nhiều nhất. Yêu cầu:

a. Chỉ in ra 1 ký tự đại diện cho những ký tự có số lần xuất hiện nhiều nhất.

Ví dụ: `S='madam'` ⇒ `in ra 'm'`

b. In ra tất cả các ký tự có số lần xuất hiện là nhiều nhất.

Ví dụ: `S = 'madam'` ⇒ in ra `'ma' hoặc 'am'`

2.1.4. List & Two array dimension

34/. Viết chương trình cho nhập 2 số nguyên m (dòng) và n (cột) của một mảng hai chiều. Chương trình tạo giá trị cho các phần tử trong mảng 2 chiều A theo quy ước A[i][j] = i * j. Với dòng và cột đượctính bắt đầu từ 0.

35/. Cần tạo 1 ma trận vuông cạnh n với giá trị các phần tử (ma trận sẽ có n * n phần tử) do người dùng nhập vào.

- Đầu tiên, cho người dùng nhập 1 số nguyên dương n.
- Tiếp tục cho người dùng nhập n*n số nguyên cho các phần tử.
- Yêu cầu: tính tổng dòng, tổng cột và in ra màn hình ma trận ban đầu, ma trận sau khi thêm tổng dòng, tổng cột.
- *Minh họa*: với n=3

1	2	3
5	6	7
9	10	11

Ma trận ban đầu

1	2	3	**6**
5	6	7	**18**
9	10	11	**30**
15	**18**	**21**	

Ma trận kết quả

- *Mở rộng 1*: giá trị của n*n phần tử được phát sinh ngẫu nhiên trong khoảng từ -10 đến +10.
- *Mở rộng 2*: ma trận không vuông với m dòng và n cột.

36/. (*)Cho người dùng nhập 2 số nguyên dương n và m. Tạo ra 1 ma trận n X m chứa một trong 2 giá trị 0 hoặc -1. Một nhóm các ô chứa giá trị -1 sẽ cùng thuộc một miền liên thông khi các ô trên hoặc dưới hoặc phải hoặc trái của ô đang xét cũng chứa giá trị -1. Viết chương trình xác định số lượng miền liên thông có trong mảng 2 chiều đã cho.

-1	0	-1	-1	-1
0	-1	-1	0	0
0	0	-1	-1	0
0	0	0	0	-1
0	-1	0	-1	0
0	0	-1	0	-1
-1	0	0	0	-1

Hình a: ma trận chỉ chứa giá trị 0 &-1

1	0	2	2	2
0	2	2	0	0
0	0	2	2	0
0	0	0	0	3
0	4	0	5	0
0	0	6	0	7
8	0	0	0	7

Hình b: đánh số thứ tự 8 miền liên thông của mảng a

2.1.5. List comprehensions

37/. Viết chương trình cho nhập 1 list gồm các số âm hoặc dương bất kỳ. Sử dụng lambda để đưa các số dương về đầu list, các số âm về cuối list. Lưu ý không sử dụng phương thức sort.

Ví dụ: với **lst= [-5, 10, -3, -1, 7, 8, 9, 2]**
Kết quả in ra màn hình: **[10, 7, 8, 9, 2, -5, -3, -1]**

38/. Tạo 2 listA, listB với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ:

```
listA = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
listB = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

Viết chương trình trả về ***listAinB*** cho biết những giá trị nào trong ***listA*** có trong ***listB*** và ngược lại tạo 1 ***listBinA*** chứa những giá trị trong listB có trong listA.

a. Mở rộng 1: phát sinh giá trị ngẫu nhiên cho 2 list listA và listB (giá trị trong 2 list có thể trùng nhau.
b. Mở rộng 2: Sao cho mỗi list kết quả (listAinB và listBinA) không chứa giá trị trùng nhau.

39/. Viết chương trình cho nhập số nguyên n. Chương trình tạo ra 1 list L gồm n số nguyên ngẫu nhiên. In ra những số chia hết cho 15 từ list L bằng cách sử dụng lambda.

40/. Viết chương trình cho nhập số nguyên n. Chương trình tạo ra 1 list L gồm n số nguyên âm hoặc dương ngẫu nhiên từ -100 đến +100. Tạo ra list M chỉ chứa những số trong L nhưng có giá trị >0 bằng cách sử dụng lambda.

41/. Viết chương trình cho nhập số nguyên n. Chương trình tạo ra 1 list L gồm n số nguyên ngẫu nhiên từ 1 đến 10. Tính tích các phần tử có trong List bằng cách sử dụng lambda.

42/. Cho người dùng nhập 8 số nguyên. In ra 3 số nguyên lớn nhất trong các số vừa nhập.

Yêu cầu: code sử dụng cho người dùng nhập các số, ử dụng lệnh *input* kèm với *for* theo dạng:

43/. Cho nhập 6 số nguyên liên tiếp cách nhau bởi khoảng trắng. In ra 6 số này với thứ tự giảm dần.

🏳 *Gợi ý*: dùng phương thức *sort()* và *reverse()* của *list*.

44/. Cho nhập 1 số nguyên n, tạo *list L* gồm *n* phần tử với giá trị ngẫu nhiên nguyên dương từ *0* đến *100*. Cho nhập số *k*. Cho biết biến *k* có lớn hơn tất cả các số có trong *L* hay không?

🏳 *Gợi ý*: dùng hàm *all()*.

45/. Tạo 2 listA, listB với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ:

```
x = [1, 2, 3]
y = [5, 10, 15]
```

Viết chương trình tạo ra ***listC*** bằng cách tính tích của từng phần tử trong x với từng phần tử trong y sao cho listC chỉ chứa những phần tử tíchnày nhưng loại đi những số chẵn. Ví dụ: khi lần lượt tính tích của từng phần tử trong x và từng phần tử trong y, ta được: `[5, 10, 15, 10, 20, 30, 15, 30, 45]`. Nhưng kết quả yêu cầu loại bỏ những phần tử chẵn nên listC chỉ còn: `[5, 15, 15, 45]`

46/. Tạo 3 listA, listB, listC với kích thước khác nhau và chứa giá trị tùy ý. Ví dụ: `listA= [1, 2], listB= [1, 2, 3, 4], listC=[3, 4, 2]`

Cho nhập tiếp số nguyên k. Viết chương trình tìm và in ra các tổ hợp gồm 3 phần tử (mỗi phần tử từ một list) có tổng = k.

🏳 ***Gợi ý***:

- Sử dụng `itertools.product` để tạo các tổ hợp
- Sử dụng `Partial function` cho phép sửa một số đối số nhất định của hàm và tạo một hàm mới.

2.1.6. Sử dụng một số module trong việc xử lý trên list

2.1.6.1. Itertools module

47/. Viết chương trình cho nhập 1 chuỗi (S) và số nguyên (n). In ra tất cả các hoán vị (permutations) n phần tử từ các ký tự có trong S.

Ví dụ: với `S= 'xyz' và n=2`

Sẽ in ra: `[('x','x'), ('x','y'), ('x','z'), ('y','x'), ('y','y'), ('y','z'), ('z','x'), ('z','y'), ('z','z')]`

🏳 *Gợi ý*: sử dụng `product` trong `module itertools`

48/. Viết chương trình cho nhập 1 chuỗi (S). Thực hiện xóa tất cả các ký tự liên tiếp trùng nhau trong S.

Ví dụ: với `S= 'xxxyxxByyAAyyy' và n=2`

Sẽ in ra: `S= xyxByAy`

🏳 *Gợi ý*: sử dụng `groupby` trong `module itertools`

2.1.6.2. Collections module

49/. Viết chương trình cho nhập 1 chuỗi (S). Thực hiện tách thành 2 chuỗi S1 và S2, với S1 chỉ chứa các ký tự xuất hiện 1 lần trong S, S2 chứa các ký tự có số lần xuất hiện trong S từ 2 lần trở lên.

Ví dụ: với `S= 'momy and dady'`

Sẽ in ra: `S1= no`

`S2= ␣admy`

50/. Cho 3 list chứa các số nguyên L1, L2, L3. Các số trong mỗi list trên có thể trùng nhau và không được sắp xếp thứ tự.Viết chương trình để so sánh 2 list L1 với L2 và L3 với L2.

Lưu ý: không chuyển đổi list sang kiểu set để thực hiện vì set chỉ loại bỏ các giá trị trùng nhau nhưng không thể đếm được số lượng của từng giá trị trong list.

Ví dụ: cho 3 list sau:

Tên	Thành phần	Đếm và phân loại các giá trị	Kết quả so sánh
L1	[20, 10, 30, 10, 20, 30]	{10: 2, 20: 2, 30: 2}	Khác nhau
L2	[30, 20, 10, 30, 20, 50]	{10:1, 20:2, 30:2, 50:1}	Giống nhau
L3	[10, 20, 20, 30, 30, 50]	{10:1, 20:2, 30:2, 50:1}	Giống nhau

2.2. Tuple

51/. Cho nhập 1 dãy số cách nhau bởi dấu phẩy. Chương trình phát sinh ra 1 tuple từ dãy số trên.

Ví dụ: nhập `'1, 3, 9, 2, 8, 4, 7'` ⇒ `tuple=(1, 3, 9, 2, 8, 4, 7).`

52/. Viết chương trình thực hiện lần lượt các yêu cầu sau:

a. Tạo 2 tuple: *tupleA* chứa 3 số nguyên, *tupleB* chứa 4 số nguyên.

b. Tạo *tupleC* bằng cách kết hợp từ *tupleA* và *tupleB*.

c. Tạo tiếp *tupleD* từ *tupleC* với các phần tử được sắp xếp giảm dần.

d. In ra 2 phần tử đầu và 2 phần tử cuối của *tupleD*.

53/. Lần lượt cho người dùng nhập một số chuỗi bất kỳ. Các chuỗi được nhập sẽ được đưa vào 1 tuple. Việc nhập kết thúc khi người dùng nhập chuỗi rỗng. Yêu cầu:

a. Gọi *n* là số lượng chuỗi vừa nhập. Cho nhập tiếp 1 số nguyên *k* (*–n <= k <n*). In ra giá trị của phần tử có *index=k*.

Ví dụ: `animalTuple=('ant', 'bear', 'cat', 'dog', 'elephant', 'fish', 'goat', 'hippo')`, với `k=-3` hoặc `k= 5` sẽ xuất ra `fish`.

b. Nhập chuỗi cần tìm (sFind). Tìm và đếm số lần xuất hiện của sFind trong tuple.

54/. Viết chương trình thực hiện lần lượt các yêu cầu sau:

a. Lần lượt cho người dùng nhập một số lượng số nguyên bất kỳ. Các số được nhập sẽ được đưa vào 1 *tuple* (*aTuple*). Việc nhập kết thúc khi người dùng nhập giá trị *zero* (*0*).

b. Thực hiện tương tự để tạo thêm 1 *tuple* (*bTuple*).

c. Tạo mới 1 *tuple* (*cTuple*) bằng cách kết hợp 2 *bTuple* và *aTuple*. In ra màn hình các thành phần có trong *cTuple*.

d. Tạo mới 1 *tuple* (*dTuple*) từ *cTuple* với các phần tử được sắp xếp giảm dần. In ra màn hình các thành phần có trong *dTuple*.

e. In các phần tử trong *dTuple* có chỉ số (*index*) là số lẻ

55/. Cho nhập một chuỗi (s), đếm tần suất xuất hiện của mỗi từ có trong (S).

Ví dụ: với nội dung chuỗi nhập như sau:

```
s='''Chiều chiều trước bến Văn Lâu
Ai ngồi, ai câu, ai sầu, ai thẳm
Ai thương, ai cảm, ai nhớ, ai trông
Thuyền ai thấp thoáng ven sông
Đưa câu mái vẩy chạnh lòng nước non'''
```

2.3. Dictionary

2.3.1. Xử lý trên dictionary

56/. Tổ chức chương trình quản lý danh bạ điện thoại (giả sử tên người là không trùng nhau và 1 người chỉ có 1 số điện thoại) bằng đối tượng dictionary, với các cặp **key-value** là **Tên-Số điện thoại**. Tạo hệ thống menu cho chương trình như sau:

```
   CHƯƠNG TRÌNH QUẢN LÝ DANH BẠ ĐIỆN THOẠI
1.- Thêm mới 1 tên cùng số điện thoại
2.- Cập nhật lại số điện thoại thông qua tên
3.- Tìm kiếm số điện thoại thông qua tên
4.- In toàn bộ danh bạ
5.- Xóa 1 tên (cùng số điện thoại)
0.- Kết thúc
      Bạn chọn chức năng:__
```

Minh họa danh bạ điện thoại

Tên	Số điện thoại
Tý	0123456789
Sửu	1234567890
Dần	9876543210

57/. Tương tự như trên, tổ chức chương trình từ điển (giả sử 1 từ tiếng Anh chỉ có 1 từ tiếng Việt tương ứng) bằng đối tượng dictionary, với các cặp **key-value** là **Tên-Số điện thoại**. Tạo hệ thống menu cho chương trình như sau:

```
          CHƯƠNG TRÌNH TỪ ĐIỂN
1.- Thêm mới 1 cặp từ
2.- Cập nhật lại nghĩa tiến Việt
3.- Tìm kiếm từ tiếng Anh
4.- In toàn bộ từ điển
5.- Xóa 1 từ
0.- Kết thúc
    Bạn chọn chức năng:__
```

Minh họa tự điển

Từ Anh	Nghĩa Việt
Cat	Con mèo
Dog	Con chó
Bear	Con gấu

58/. Cho nhập 2 chuỗi (S1 và S2).

a. In ra những ký tự xuất hiện trong cả 2 chuỗi.

Gợi ý: sử dụng class *Counter* trong module *collections* để chuyển mỗi chuỗi vào 1 dict thuộc class Counter. Thực hiện phép và (&) trên 2 dict này để có kết quả.

b. Đếm xem có bao nhiêu ký tự có trong S1 nhưng không có trong S2 và có trong S2 nhưng không có trong S1.

c. In ra những ký tự có trong S1 nhưng không có trong S2 và những ký tự có trong S2 nhưng không có trong S1.

☞ *Gợi ý*: đưa mỗi chuỗi vào 1 dict (dict1 và dict2). Thực hiện dò tìm S1 trên dict2 và tìm S2 trên dict1.

59/. Viết lại chương trình trò chơi "Bao-Búa-Đinh" cho 2 người chơi bằng cách sử dụng Dictionary. Mỗi người dùng được chọn 1 trong 3 món Bao, búa, đinh. Chương trình cho biết người thắng cuộc theo quy ước:

- Bao thắng Búa
- Búa thắng Đinh
- Đinh thắng Bao

60/. Viết chương trình cho người dùng nhập ngày (d) và tháng (m). In ra thứ của ngày tháng vừa nhập ở năm hiện tại.

☞ Gợi ý:

- Khai báo 1 dictionary với nội dung như sau:

```
weekDict = {1:'Thứ Hai',2:'Thứ Ba',3:'Thứ Tư',4:'Thứ Năm',
                                5:'Thứ Sáu', 6:'Thứ Bảy',7:'Chủ nhật'}
```

- Sử dụng hàm *isoweekday* của module *datetime* và dựa vào *dictionary* này để in ra thứ.

61/. (*) Cho trước 1 dictionary như sau:

```
{ '1': 'abc', '2': 'def', '3': 'ghi', '4': 'jkl', '5': 'mno',
                        '6': 'pqrs', '7': 'tuv', '8': 'wxy', '9': 'z' }
```

Viết chương trình cho nhập 1 chuỗi `S` chứa ký số (như `'3'`, `'15'`, `'914'`, ...). In ra tất cả các tổ hợp có được từ việc lấy trong mỗi tổ hợp 1 phần tử. Ví dụ:

Chuỗi nhập	*Kết quả*
'1'	['a', 'b', 'c']
'29'	['dz', 'ez', 'fz']
'112'	['aad', 'aae', 'aaf', 'abd', 'abe', 'abf', 'acd', 'ace', 'acf', 'bad', 'bae', 'baf', 'bbd', 'bbe', 'bbf', 'bcd', 'bce', 'bcf', 'cad', 'cae', 'caf', 'cbd', 'cbe', 'cbf', 'ccd', 'cce', 'ccf']

☞ *Gợi ý*: giả sử chuỗi nhập là `S='27'`

- Đầu tiên tạo `resultList` chứa 1 chuỗi rỗng.
- Sử dụng `tempList` cho các lần phát sinh các tổ hợp sẽ được thực hiện sau đây. Khởi đầu mỗi lần lặp, `tempList` được gán là rỗng. Sau mỗi lần lặp, sẽ gán nội dung có trong `tempList` cho `resultList`.

- Lần lượt duyệt từng chữ số X trong chuỗi (S). Trong mỗi chuỗi số X thường gồm 3 ký tự, mỗi ký tự này sẽ kết hợp với các chuỗi đang có trong resultList để tạo thành các bộ chuỗi mới.

  Minh họa: `S='27'`
  - `resultList = ['']`
  - `tempList=[]`
  - `Xét chữ số đầu tiên (số 2). Chữ số 2 gồm 3 ký tự 'd', 'e', 'f'.`
  - `Lấy mỗi ký tự trong chữ số 2 để nối với chuỗi đang có trong resultList để có các phần tử mới trong ⇒ tempList = [ ''+'d', ''+'e', ''+'f' ] = [ 'd', 'e', 'f' ]`
  - `Gán resultList = tempList`
  - `Gán lại tempList=[]`
  - `Xét số kế tiếp (số 7). Chữ số 7 gồm 3 ký tự 't', 'u', 'v'.`
  - `Tạo 1 hoặc nhiều newList bằng cách kết hợp các ký tự có trong số 7 với các subList đang có trong resultList ⇒ tempList =[ 't'+'d', 't'+'e', 't'+'f', 'u'+'d', 'u'+'e', 'u'+'f', 'v'+'d', 'v'+'e', 'v'+'f' ]= ['dt', 'du', 'dv', 'et', 'eu', 'ev', 'ft', 'fu', 'fv']`

62/. Lần lượt thực hiện bài tập này bằng 2 cách sử dụng tuble và dictionary. Cho nhập một chuỗi (s), cho biết các kết quả sau:

a. Tần suất xuất hiện của mỗi từ
b. Các từ có số lần xuất hiện nhiều nhất
c. Từ dài nhất.

Ví dụ: với `s=`
```
"Chiều chiều trước bến Văn Lâu.
Ai ngồi, ai câu, ai sầu, ai thảm.
Ai thương, ai cảm, ai nhớ, ai trông.
Thuyền ai thấp thoáng ven song.
Đưa câu mái vẩy chạnh lòng nước non."
```

Cho kết quả:

```
YÊU CẦU a: Số lần xuất hiện của các từ (ĐÃ loại những từ trùng nhau):
    {('non', 1), ('sầu', 1), ('trước', 1), ('thấp', 1), ('Văn', 1), ('vẩy', 1), ('chiều', 1), ('ven', 1), ('chạnh', 1), ('cảm', 1), ('nhớ', 1), ('Đưa', 1), ('thảm', 1), ('thương', 1), ('lòng', 1), ('ngồi', 1), ('Thuyền', 1), ('trông', 1), ('mái', 1), ('thoáng', 1), ('ai', 7), ('bến', 1), ('Ai', 2), ('nước', 1), ('Lâu', 1), ('Chiều', 1), ('sông', 1), ('câu', 2)}
YÊU CẦU b: Các từ có số lần xuất hiện nhiều nhất (7 lần) là: ai
YÊU CẦU c: Từ dài nhất là thoáng, Thuyền, thương, gồm 6 ký tự
```

☞ ***Gợi ý cách thực hiện bằng tuple***:

B1. Dựa vào khoảng trắng giữa các từ, tách các từ đưa vào *list1*

B2. Trong các từ vừa tách vẫn còn lẫn các dấu phẩy, dấu chấm (mà các ký tự này là dấu ngắt câu). Hãy tìm cách loại bỏ chúng

B3. Tạo ra 1 list đếm số lần xuất hiện của các từ đưa vào list2

B4. Sử dụng hàm zip với tham số là list1 và list2 để tạo thành các tuble. Đưa tất cả các tuple này vào list3.

B5. Do list3 vẫn còn chứa các tuple có giá trị trùng nhau, nên dùng tiếp hàm set để loại bỏ các tuple có nội dung trùng nhau. Đây là kết quả cần thực hiện

☞ ***Gợi ý sử dụng dictionary***:

B1. Dựa vào khoảng trắng giữa các từ, tách các từ đưa vào *list* ()

B2. Trong các từ vừa tách vẫn còn lẫn ký tự xuống dòng (‘\n’) hoặc các dấu phẩy, dấu chấm trong ngắt câu. Hãy tìm cách loại bỏ chúng

B3. Khai báo 1 dictionary để đếm số lần xuất hiện của các từ.

B4. Để tìm từ dài nhất, sử dụng hàm len(chuỗi của tùng có trong list)

63/. Cho người dùng nhập 1 chuỗi (S). Tìm ký tự đầu tiên trong chuỗi chỉ xuất hiện 1 lần.

Ví dụ:
```
S= 'abcdef'       ⇒ a
S= 'abcabcdef'    ⇒ d
S= 'aabbcc'       ⇒ None
```

64/. Cho nhập 1 chuỗi (S). Tìm ký tự đầu tiên được lặp lại trong S.

Ví dụ:
```
S = "abcabc"  sẽ in ra a
S = "abbcabc" sẽ in ra b
S = "abc"     sẽ in ra None
```

65/. Cho người dùng nhập 1 chuỗi (S). Tìm từ trong S xuất hiện nhiều thứ hai (sau nhiều nhất).

☞ *Gợi ý*: dùng hàm sorted trên dict, sau đó duyệt ngược dict để tìm số lớn thứ nhì.

Ví dụ:
```
S='''Quê hương
        Quê hương là chùm khế ngọt
        Cho con trèo hái mỗi ngày
        Quê hương là đường đi học
        Con về rợp bướm vàng bay
        Đỗ Trung Quân'''
```
Sẽ in ra `('là', 2)  # vì 2 từ 'Quê' = 'hương'= 3`

66/. Cho người dùng nhập 1 chuỗi (S). Thực hiện xóa tất cả những ký tự đi sau và bị trùng (đã xuất hiện trước đó.

Ví dụ:
```
S='madam' ⇒ 'mad'
S='Collections'      ⇒ 'coletins'
```
Gợi ý: sử dụng phương thức join của kiểu dữ liệu chuỗi với tham số là 1 dictionary được tạo từ chuỗi S.

67/. Viết chương trình sử dụng dictionary để quản lý danh bạ điện thoại với các cặp key-value được minh họa như sau:

Name	*Telephone number*
Johnny	0989741258
Katherine	0903852147
Misu	0913753951
Jack	0933753654
…	…

Yêu cầu:

a. Tạo menu có dạng như hình minh họa sau. Sau khi người dùng chọn chức năng, chương trình thực hiện vừa chọn và in lại menu để người dùng tiếp tục sử dụng:
```
        DANH BẠ ĐIỆN THOẠI
1.- Xem danh bạ
2.- Tìm kiếm theo tên
3.- Tìm kiếm theo số điễn thoại
4.- Thêm mới
0.- Kết thúc
Bạn chọn chức băng cần thực hiện (0-4):
```

b. Xem danh bạ: khi được chọn sẽ liệt kê toàn bộ tên và số điện thoại có trong danh bạ.

c. Tìm kiếm theo tên: yêu cầu nhập tên, nếu tìm thấy, liệt kê tên và số điện thoại tương ứng. Nếu không tìm thấy, chương trình thông báo ‘Không tìm thấy <tên>’.

d. Tìm kiếm theo số điện thoại: yêu cầu nhập số điện thoại, nếu tìm thấy, liệt kê tên và số điện thoại tương ứng. Nếu không tìm thấy, chương trình thông báo ‘Không tìm thấy <số điện thoại>’.

e. Thêm mới: cho người dùng nhập tên và số điện thoại để bổ sung vào Danh bạ.
f. Kết thúc chương trình.

68/. Tương tự như trên, viết chương trình sử dụng dictionary để quản lý tự điển Anh Việt với các cặp key là các từ tiếng Anh, value là nghĩa tiếng Việt tương ứng.

2.3.2. Sử dụng một số module trong việc xử lý trên dictionary

69/. Cho 1 tuple mà mỗi thành phần con trong đó cũng có kiểu là tuple, cho biết tên lớp cùng tên sinh viên. Ví dụ:

```
students =  (   ('class_A', 'Tý'),
                ('class_B', 'Suu'),
                ('class_A', 'Dan'),
                ('class_B', 'Mẹo'),
                ('class_B', 'Thìn'),
                ('class_C', 'Tỵ'),
            )
```

Yêu cầu:Đếm số lượng sinh viên của mỗi lớp

70/. Cho trước 1 dictionary (ví dụ `mydict = {'Afghanistan' : 93, 'USA' : 213, 'Algeria' : 213, 'Angola' : 244, 'India' : 355, 'Albania' : 355, 'Andorra' : 376}`). Viết chương trình Python để tạo một thể hiện của OrderedDict dựa trên dictionary đã có. Sắp xếp từ điển trong quá trình tạo và in các thành phần của dictionary theo thứ tự ngược lại.

2.4. Set

71/. Viết chương trình tạo 1 list L gồm n phần tử có giá trị ngẫu nhiên, với n do người dùng nhập. Viết hàm tính tổng các số không trùng nhau có trong list L.

Ví dụ: `L=[1, 2, 2, 2, 3, 5, 5]` ⇒ `tổng =11`

72/. Cho người dùng nhập 1 chuỗi (S). Cho biết S có chứa đầy đủ tất cả các ký tự từ A-Z hay không? Không phân biệt ký tự hoa/thường.

Gợi ý:

- Tạo ra 1 set chứa tất cả các ký tự thường từ a-z bằng 1 trong 2 cách sau:
  - Cách 1: tự khai báo 1 set trong đó liệt kê đầy đủ các ký tự từ a-z
  - Cách 2: sử dụng thuộc tính `ascii_lowercase` trong module `string`.
- Chuyển tất cả các ký tự trong S thành ký tự thường.
- Sử dụng 1 trong các toán tử so sánh (>, >=, <, <=, ==, …) để tập hợp vừa có, nếu bằng nhau sẽ cho ra True, ngược lại sẽ cho False.
- Sử dụng 2 chuỗi sau để kiểm tra:
  - *The quick brown fox jumps over the lazy dog* (đủ các ký tự từ A-Z)
  - *The quick brown fox jumps over the lazy cat* (thiếu 2 ký tự d và g)

73/. Viết chương trình cho nhập số điện thoại (S). In ra các số từ 0 đến 9 không xuất hiện trong số điện thoại vừa nhập.

Ví dụ: `nhập S='0913158020'`

⇒`Trong số điện thoại 0913158020 không chứa các ký số: [4, 6, 7]`

Gợi ý:

- Tạo set1 chứa tất cả các ký số từ 0-9. Tương tự bài tập trước, cũng có 2 cách để tạo set:
  - Cách 1: tự khai báo 1 set trong đó liệt kê đầy đủ các ký số tư 0-9
  - Cách 2: sử dụng thuộc tính `string.digits` trong module `string`.
- Tạo set2 chứa tất cả các ký số có trong số điện thoại.

- Sử dụng phép bù giữa set1 và set2 để tìm kết quả

74/. Cho nhập 1 chuỗi (S). Tìm từ đầu tiên lặp lại trong S.

Ví dụ:
```
S="ab ca bc ab"          sẽ in ra ab
S="ab ca bc ca ab bc"    sẽ in ra ca
S="ab ca bc"             sẽ in ra None
```

🏳 Gợi ý: Tạo set1 rỗng. Lần lượt đưa từng ký tự của S vào trong set, nếu ký tự nào đã có trong set1 thì kết thúc. Sử dụng toán tử **`in`** để kiểm tra ký tự đã có trong set1 hay chưa?

75/. Viết chương trình cho người dùng nhập 2 chuỗi (S1 và S2). Tạo ra chuỗi kết quả bằng cách ghép các ký tự có trong S1 nhưng không có trong S2 và có trong S2 nhưng không có trong S1.

Ví dụ: `S1='abcd'; S2='mncdgh'` ⇒ `'abmngh'`

76/. Viết chương trình cho người dùng nhập 1 chuỗi (S). Tìm đoạn ngắn nhất trong S sao cho đoạn này chứa tất cả các ký tự có trong S.

Ví dụ: `S1 = 'how can i tell her-lobo'=>` in ra: `'w can i tell her-lob'`

🏳 *Gợi ý*: sử dụng `defaultdict` trong `module collections`

77/. Viết chương trình thực hiện việc xử lý trên set như sau:

- Khai báo và khởi tạo *set1*, *set2*
- Cho phép người dùng lần lượt nhập các phần tử số cho *set1* cho đến khi giá trị nhận vào là -1.
- Tương tự, cho phép người dùng lần lượt nhập các phần tử số cho *set2* cho đến khi giá trị nhận vào là -1.

Yêu cầu:

a. Sử dụng lệnh for để in các số có trong *set1* và *set2* ra màn hình với các giá trị cách nhau bởi dấu phẩy (,).
b. Cho biết mỗi *set* có bao nhiêu phần tử, tổng giá trị các phần tử của mỗi *set*.
c. Tìm giá trị lớn nhất, nhỏ nhất của mỗi *set*.
d. Thực hiện set union của *set1* và *set2* và in kết quả.
e. Thực hiện set intersection của *set1* và *set2* và in kết quả.
f. Thực hiện set difference của *set1* với *set2* và in kết quả.
g. Thực hiện set symmetric difference của *set1* với *set2* và in kết quả.
h. Sắp xếp *set1* tăng dần và *set2* giảm dần.

78/. Viết chương trình tạo hệ thống menu với những chức năng như sau:

```
                BÀI TẬP THỰC HÀNH VỀ SET
1.- Nhập giá trị cho set1
2.- Nhập giá trị cho set2
3.- Tính tổng các phần tử có trong mỗi set
4.- Tìm giá trị lớn nhất trong mỗi set
5.- Tìm giá trị nhỏ nhất trong mỗi set
6.- Xóa phần tử có trong set1
7.- Tìm và in ra những phần tử trùng nhau của 2 set
8.- Tìm và in ra những phần tử có trong cả 2 set
9.- In ra những phần tử có trong set1 nhưng không có trong set2
10.- In ra những phần tử có trong set2 nhưng không có trong set1
11.- Sắp xếp set1 giảm dần và set2 tăng dần
0.- Kết thúc chương trình
Bạn chọn chức năng số:_
```

🏳 *Yêu cầu*:

- Chức năng ***1*** và ***2***: khi người dùng nhập số đã có trong set, yêu cầu nhập lại số khác cho đến khi không còn bị trùng.
- Chức năng ***1***: việc nhập kết thúc khi giá trị nhập vào là zero (0).
- Chức năng ***2***: hỏi người dùng cần nhập mấy số, chương trình cho nhập lần lượt từng số.
- Chức năng ***6***: cho người dùng nhập số cần xóa. Tìm và xóa giá trị này, sau đó in ra sét sau khi xóa.
- Chức năng ***8***: loại bỏ những số trùng nhau trong kết quả (nếu có)
- Chương trình chỉ kết thúc khi giá trị nhập là ***0***.
- Nếu giá trị không phải các giá trị từ ***0-11***, chương trình hiện ra câu nhắc người dùng phải nhập đúng.

2.5. Phối hợp các đối tượng dạng danh sách

2.5.1. List & Dictionary

79/. Lần lượt thực hiện các yêu cầu sau:

a. Viết hàm cho người dùng nhập 1 số nguyên n với 1<n<100. Do đó nếu n không phải là số nguyên và nếu n không nằm trong khoảng từ 2->100, chương trình sẽ yêu cầu nhập lại cho đến khi đúng yêu cầu.
b. Viết hàm tạo 1 list L gồm n phần tử với giá trị được phát sinh ngẫu nhiên trong khoảng từ Min=-5 đến Max=+10. Lưu ý các giá trị này có thể là số nguyên hoặc số thực.
c. Viết hàm đếm tần suất xuất hiện của các giá trị có trong list L. Sau đó cho biết số tần suất lớn nhất là bao nhiêu và những giá trị nào có tần suất là nhiều nhất .
d. Viết hàm kiểm tra kiểu dữ liệu của từng phần tử trong list L. Cuối dùng, in ra số lượng phần tử có kiểu dữ liệu là số nguyên, bao nhiêu phần tử có kiểu dữ liệu là số thực.
e. Sắp xếp list L thành list Lsort tăng dần
f. Viết hàm tính trung bình các phần tử trong list L.

80/. Cho một list mà mỗi thành phần trong đó có kiểu là 1 tuple. Ý nghĩa của mỗi tuple là thành phần thứ nhất cho biết tên của lớp học, thành phần thứ 2 cho biết tên của sinh viên.

Viết chương trình để nhóm các tuble thành 1 dictionary với key là tên lớp và value là 1 list chứa tên các sinh viên. In ra kết quả được sắp xếp theo tên lớp

Ví dụ: `studentsList = [ ('classB', 'Sửu'), ('classA', 'Dần'), ('classC', 'Mẹo'), ('classB', 'Thìn') , ('classA', 'Tý'), ('classB', 'Tỵ')]`

Kết quả thực hiện:

```
[('classA', ['Dần', 'Tý']), ('classB', ['Sửu', 'Thìn', 'Tỵ']), ('classC', ['Mẹo'])]
```

2.6. Xây dựng hàm ẩn danh *(Anonymous Function)* cho Iterator object *(sequences types)*

81/. Cho list1 chứa các số nguyên bất kỳ. Sử dụng lambda để:

a. Tạo list2 chứa các số chẵn, list3 chứa các số lẻ. In 2 list này ra màn hình .

Ví dụ `list1 = [-5, 10, -3, -1, 7, 8, 9, 2]`

In ra màn hình:
```
List ban đầu: [-5, 10, -3, -1, 7, 8, 9, 2]
Các số chẵn có trong list: [10, 8, 2]
Các số lẻ có trong list: [-5, -3, -1, 7, 9]
```

b. Tương tự câu a nhưng lần lượt các lambda chỉ tính số lượng số chẵn, số lượng số lẻ. In kết quả thực hiện ra màn hình

🏳 *Gợi ý*: sử dụng hàm filter khi tạo 2 list mới

c. Tạo list2 chứa các giá trị là bình phương của các giá trị có trong list1 và list3 chứa các giá trị là lũy thừa 3 của các giá trị có trong list1. In kết quả ra màn hình

🏳 *Gợi ý* sử dụng hàm map khi tạo 2 list mới

Ví dụ

```
List ban đầu: [-5, 10, -3, -1, 7, 8, 9, 2]
Bình phương các giá trị trong lst: [25, 100, 9, 1, 49, 64, 81, 4]
Lũy thừa 3 các giá trị trong lst: [-125, 1000, -27, -1, 343, 512, 729, 8]
```

d. Tạo list2 chứa các giá trị là số nguyên tố hoặc số chính phương.Với số chính phương *(square number hay perfect square)* là số có căn bậc hai là một số nguyên. *Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.*

🏳 *Gợi ý* viết 2 hàm kiểm tra số nguyên tố và kiểm tra số chính phương. Lambda sẽ sử dụng 2 hàm này trong điều kiện lọc của hàm filter

Ví dụ

```
List ban đầu: [19, 65, 81, 39, 152, 639, 121, 44, 100, 31]
Các số là số nguyên tố hoặc là số chính phương:  [19, 81, 121, 100, 31]
```

82/. Cho 2 list chứa các số nguyên và 2 list này có cùng số lượng phần tử. Sử dụng lambda để tạo ra 1 list mới bằng cách cộng đôi một các số có trong 2 list.

🏳 *Gợi ý*: sử dụng hàm map

Ví dụ: `lst1 = [1, 2, 3], lst2 = [4, 5, 6]=> in ra màn hình[5, 7, 9]`

83/. Cho list1 chứa các chuỗi ký tự. Sử dụng lambda để tạo ra list2 mới thỏa mãn yêu cầu sau:

a. List2 chứa các chuỗi có chiều dài là 6.

🏳 *Gợi ý*: sử dụng hàm filter

Ví dụ: với `List1 = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']`

Sẽ in ra: `['Monday', 'Friday', 'Sunday']`

b. List2 chứa các chuỗi Palindrome. Chuỗi được gọi là Palindrome nếu sau khi đảo ngược các ký tự của nó, ta nhận được chuỗi ban đầu. Ví dụ MADAM.

Ví dụ: với `List1 = ['php', 'www', 'Python', 'abba', 'Java', 'MADAM']`

Sẽ in ra: `['php', 'www', 'abba', 'MADAM']`

84/. Theo Wikipedia, đảo chữ (anagram) là hoán vị các ký tự có trong từ đó (các từ mới này có thể có nghĩa hoặc không có nghĩa). Ví dụ từ eat sẽ có thể được hoán vị thành các từ: tea, eta, tae, aet, ate.

Cho list1 chứa 1 số từ. Viết chương trình yêu cầu người dùng nhập 1 từ (w), sử dụng lambda để tìm các trường hợp đảo ký tự của w có trong list1.

Ví dụ: Các chuỗi có trong list1: `['bcda', 'abce', 'cbda', 'cbea', 'adcb']`

Các đảo chữ của 'abcd' trong list1: `['bcda', 'cbda', 'adcb']`

85/. Viết chương trình cho nhập số lượng sinh viên (n). Cho nhập thông tin của n sinh viên ,biết thông tin của mỗi sinh viên gồm tên và điểm. In ra tên và điểm của tất cả sinh viên có điểm thấp thứ nhì. Nếu có nhiều sinh viên cùng có điểm thấp thứ nhì, chương trình in tên theo thứ tự alphabet.

86/. Cho 1 list mà các thành phần trong đó là 1 tuple. Ví dụ list ban đầu như sau:

```
subjectMarksList = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
```

Sử dụng lambda để sắp xếp list tăng theo điểm. Như vậy sau khi sắp xếp, ***subjectMarksList*** sẽ có dạng như sau:

```
[('Social sciences', 82), ('English', 88), ('Science', 90), ('Maths', 97)]
```

87/. Cho 1 list mà các thành phần trong đó là 1 dictionary. Ví dụ list ban đầu như sau:

```
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]
```

Sử dụng lambda để sắp xếp list tăng theo key là `'color'`. Như vậy sau khi sắp xếp, ***subjectMarksList*** sẽ có dạng như sau:

```
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': '2', 'color': 'Gold'}]
```

3 FILE

3.1. Text file

3.1.1. Đọc file

1/.

- Tạo file .txt: right click vào package, chọn New/File. Đặt tên file là tho.txt. Nội dung file như sau:

```
QUÊ HƯƠNG
...
Quê hương là chùm khế ngọt
Cho con trèo hái mỗi ngày
Quê hương là đường đi học
Con về rợp bướm vàng bay
...
```

a. Viết chương trình đọc và hiển thị nội dung file trên ra màn hình console

b. Viết chương trình đọc và hiển thị nội dung file trên ra màn hình console với số dòng cần đọc do người dùng nhập vào.

🏳 *Hướng dẫn*: import islice trong module itertools để hỗ trợ đọc file theo dòng.

2/. Viết chương trình đọc và in n dòng cuối của file Que_Huong.txt ra màn hình. Với n do người dùng nhập từ bàn phím

🏳 *Hướng dẫn*: import 2 module sys và os.

3/. Viết chương trình đọc nội dung file Que_Huong.txt và đưa vào 1 list. Sau đó in nội dung list L ra màn hình. Lần lượt thực hiện bằng 2 cách:

a. Đọc cả file và đưa vào list 1 lần

b. Đọc từng dòng đưa vào list cho đến khi hết file.

4/. Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về 1 list chứa những từ có chiều dài là dài nhất có trong file.

Minh họa kết quả thực hiện: `['HUONG', 'huong', 'huong', 'duong', 'Thach']`

5/. Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về số dòng có trong file.

Minh họa kết quả thực hiện: `Số lượng dòng có trong file: 8`

6/. Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về tần suất xuất hiện của từng từ có trong file.

Minh họa kết quả thực hiện:

```
Tần suất xuất hiện của các từ: Counter({'...': 2, 'Que': 2, 'huong': 2, 'la': 2, 'QUE': 1, 'HUONG': 1, 'chum': 1, 'khe': 1, 'ngot': 1, 'Cho': 1, 'con': 1, 'treo': 1, 'hai': 1, 'moi': 1, 'ngay': 1, 'duong': 1, 'di': 1, 'hoc': 1, 'Con': 1, 've': 1, 'rop': 1, 'buom': 1, 'vang': 1, 'bay': 1, 'Giap': 1, 'Van': 1, 'Thach': 1})
```

🏳 *Hướng dẫn*: import và sử dụng Counter (trong module collections).

7/. Viết chương trình đọc file text (txt) và đếm số lượng của mỗi ký tự có trong file. Xem như ký tự hoa và thường đều là ký tự thường.

Minh họa nội dung file abc.txt:

abc.txt -
German Unity Day
From Wikipedia, the free encyclopedia
The Day of German Unity (German: Tag der DeutschenEinheit) is the national day of Germany, celebrated on 3 October as a public holiday. It commemorates the anniversary of German reunification in 1990, when the goal of a united Germany that originated in the middle of the 19th century, was fulfilled again. Therefore, the

name addresses neither the re-union nor the union, but the unity of Germany. The Day of German Unity on 3 October has been the German national holiday since 1990, when the reunification was formally completed.

Minh họa kết quả thực hiện:

```
File Name:  abc.txt
Counter({' ': 93, 'E': 64, 'N': 45, 'A': 42, 'T': 40, 'I': 36, 'O': 31, 'R': 29,
'H': 25, 'D': 19, 'M': 17, 'Y': 17, 'L': 15, 'F': 15, 'U': 14, 'C': 13, 'G': 13,
'S': 12, ',': 7, 'B': 6, 'W': 5, '9': 5, '.': 4, 'P': 4, '1': 3, '\n': 2, '0': 2,
'3': 2, ':': 1, '-': 1, 'K': 1, '(': 1, ')': 1, 'V': 1})
```

3.1.2. Ghi file

8/. Viết chương trình ghi nội dung trích dẫn của bài thơ Quê hương vào file Que_Huong.txt. Sau đó, đọc nội dung file đã ghi ra màn hình.

☞ *Hướng dẫn*: import islice trong module itertools để hỗ trợ đọc file theo dòng.

3.1.3. Làm việc file chứa kiểu dữ liệu dạng số

9/. Viết chương trình thực hiện các yêu cầu sau, mỗi yêu cầu được viết thành 1 hàm riêng biệt. Viết chương trình chính lần lượt gọi các hàm đã viết. Giả sử file cần tạo có tên và đường dẫn là D:\Number.txt:

(i). Hàm *NhapN()*: Cho nhập số nguyên n, với nằm trong khoảng từ 2 đến 200. Nếu người dùng nhập sai (nhập ký tự không phải là các ký số, nhập số thực), chương trình sẽ báo lỗi và yêu cầu nhập lại. Khi người dùng nhập đúng, hàm trả về n.

(ii). Hàm *GhiFile(filename, n)*: hàm ghi n số nguyên được phát sinh ngẫu nhiên vào file có tên là filename. Giá trị ngẫu nhiên được phát sinh trong khoảng từ -9 đến +200.

(iii). Hàm *DocFile(filename)*: hàm đọc các giá trị có trong file filename và in ra màn hình.

(iv). Hàm *Tong(filename)*: hàm đọc các giá trị có trong file filename, tính tổng các giá trị đọc được và in ra màn hình.

(v). Hàm *LietKeSoNguyenTo(filename)*: hàm đọc các giá trị có trong file filename, in các số nguyên tố ra màn hình.

(vi). Hàm *LietKeSoChinhPhuong(filename)*: hàm đọc các giá trị có trong file filename, in các số chính phương ra màn hình. Với số chính phương *(square number hay perfect square)* là số có căn bậc hai là một số nguyên. *Ví dụ 9 là số chính phương vì căn bậc hai của 9 là 3.*

(vii). Hàm *LietKeSoHoanThien(filename)*: hàm đọc các giá trị có trong file filename, in các số hoàn thiện ra màn hình. Với số hoàn thiện *(Perfect number)* là số (***n***) có tổng các ước số không kể n bằng chính n. *Ví dụ: 6 là số hoàn thiện vì 1+2+3=6; hay số 28 vì 1+2+4+7+14=28, ...*

(viii). Hàm *LietKeSoArmstrong(filename)*: hàm đọc các giá trị có trong file filename, in các số Armstrong ra màn hình. Với số Armstrong là số có đặc điểm sau: số đó gồm n ký số, tổng các lũy thừa bậc n của các ký số bằng chính số đó. *Ví dụ 153 là một số có 3 ký số, và* $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.

3.1.4. Lấy thông tin về file

10/. Viết chương trình cho nhập tên file, lấy đường dẫn tuyệt đối (absolute file path) của file.

Minh họa kết quả thực hiện:

```
Absolute file path test.txt: /home/students/path_fname
```

11/. Viết hàm đọc nội dung file Que_Huong.txt. Hàm trả về kích thước file tính theo byte.

Minh họa kết quả thực hiện: `File size in bytes:  145`

a. Yêu cầu sử dụng phương thức stat trong module os

b. Yêu cầu sử dụng getsize trong module os.path.

12/. Viết chương trình cho nhập tên file, chương trình trả về thông về ngày giờ tạo lập và ngày giờ hiệu chỉnh cuối cùng của file.

Minh họa kết quả thực hiện:

```
Last modified of test.txt file: Wed Apr 19 11:36:23 2017
Created of test.txt file: Wed Apr 19 11:36:23 2017
```

13/. Viết chương trình lấy thông tin các thuộc tính của file.

Minh họa kết quả thực hiện:

```
File : 8dbacd90-266d-11e7-a9c1-cf681af3cdf1.py
Access time : Fri Apr 21 14:06:03 2017
Modified time: Fri Apr 21 14:06:02 2017
Change time : Fri Apr 21 14:06:02 2017
Size : 304
```

14/. Viết chương trình tìm đường dẫn tuyệt đối của path; path là file hoặc thư mục hay là 1 liên kết (shortcut/link), path có tồn tại hay không? Và có tồn tại 1 liên kết nào đến path hay không?

Minh họa kết quả thực hiện:

```
File : 26cb6a20-266f-11e7-a9c1-cf681af3cdf1.py
Absolute : False
Is File? : True
Is Dir? : False
Is Link? : False
Exists? : True
Link Exists?: True
. . .
Is Dir? : False
Is Link? : False
Exists? : False
Link Exists?: False
```

3.2. CSV file

15/. Viết hàm đọc file Departments.csv và in ra màn hình theo mẫu sau đây, trong đó:

- Tên tiêu đề cột được in canh giữa 2 ký tự '|' ở 2 đầu
- Field dạng chuỗi được in canh trái
- Field dạng số được in canh phải.

Minh họa kết quả thực hiện:

```
-----------------------------------------------------------------------
| department_id  |    department_name    | manager_id | location_id|
 ----------------------------------------------------------------------
|            10  |  Administration       |        200 |        1700|
|            20  |  Marketing            |        201 |        1800|
|            30  |  Purchasing           |        114 |        1700|
...
...
|           270  |  Payroll              |            |        1700|
-----------------------------------------------------------------------
Tổng cộng gồm 27 dòng dữ liệu
```

16/. Viết chương trình gồm 2 hàm sau, với tên file cần tạo mới và cần đọc là Countries.csv:

- Tao_File(TenFile): trong hàm sẽ thực hiện các công việc sau:
  - Khai báo 1 list chứa nội dung các dòng có trong bảng sau.

country_id	country_name	region_id
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2

- Tạo file csv với tên là tham số TenFile được truyền cho hàm. Nội dung của file lấy trong list vừa khai báo ở trên và dấu phân cách được dùng là ký tự dấu thăng ('#'-hash).
- Yêu cầu: Viết 2 hàm Tao_File1 và Tao_File2, trong đó, hàm Tao_File1 ghi toàn bộ list chỉ trong 1 lệnh và hàm Tao_File2 thực hiện ghi mỗi lần một dòng.

- DocFile(TenFile): đọc nội dung file TenFile và in ra kết quả như hình minh họa.

```
------------------------------------------------
| country_id |   country_name    |  region_id   |
 ------------------------------------------------
|         AR |  Argentina        |            2 |
|         AU |  Australia        |            3 |
|         BE |  Belgium          |            1 |
|         BR |  Brazil           |            2 |
|         CA |  Canada           |            2 |
------------------------------------------------
Tổng cộng gồm 5 dòng dữ liệu
```

📖Nội dung file Deparments.csv sử dụng trong phần bài tập

```
department_id,department_name,manager_id,location_id
10,Administration,200,1700
20,Marketing,201,1800
30,Purchasing,114,1700
40,Human Resources,203,2400
50,Shipping,121,1500
60,IT,103,1400
70,Public Relations,204,2700
80,Sales,145,2500
90,Executive,100,1700
100,Finance,108,1700
110,Accounting,205,1700
120,Treasury,,1700
130,Corporate Tax,,1700
140,Control And Credit,,1700
150,Shareholder Services,,1700
160,Benefits,,1700
170,Manufacturing,,1700
180,Construction,,1700
190,Contracting,,1700
200,Operations,,1700
210,IT Support,,1700
220,NOC,,1700
230,IT Helpdesk,,1700
240,Government Sales,,1700
250,Retail Sales,,1700
260,Recruiting,,1700
270,Payroll,,1700
```

3.3. Thư mục và tập tin

17/. Viết chương trình cho nhập tên file kèm ký tự đại diện, thực hiện sắp xếp danh sách các file theo ngày giờ tạo lập.

Minh họa kết quả thực hiện: giả sử yêu cầu liệt kê các file **`"*.txt"`** có trong thư mục hiện tại:

```
result.txt
temp.txt
myfile.txt
mynewtest.txt
mytest.txt
```

```
abc.txt
test.txt
```

18/. Viết chương trình cho nhập tên thư mục (myDict), chương trình sẽ liệt kê các thư mục con có trong myDict được sắp xếp theo thứ tự tăng dần của ngày tạo lập.

19/. Viết chương trình cho nhập tên (đặt là path) thư mục hoặc tên file hay tên đặc biệt khác (như socket, FIFO, device file, ...). Chương trình thực hiện kiểm tra xem path là thư mục, tập tin hoặc tên đặc biệt khác.

🏳 *Hướng dẫn*: sử dụng các phương thức `os.path.isdir` và `os.path.isfile` để kiểm tra

Ví dụ, với file có tên abc.txt, khi thực hiện kiểm tra sẽ cho kết quả là: `It is a normal file`

20/. Viết chương trình để tạo danh sách chứa các tập tin có trong thư mục hiện hành bằng ký tự đại diện.

21/. Viết chương trình cho biết đường dẫn thư mục của người dùng (home directory.

Minh họa kết quả thực hiện: `C:\Users\Administrator`

22/. Viết chương trình cho nhập tên folder cùng đường dẫn (tạm đặt là path), in ra danh sách các file có trong folder đó.

Minh họa kết quả thực hiện: khi `path= '/home/students'`

Sẽ in ra: `['test.txt', '.mysql_history', '.bash_logout', '.bash_history', '.profile', 'abc.py', '.viminfo', 'mynewtest.txt', 'myfile.txt', 'logging_example.out', '.web-term.json', 'abc.txt', '64a57280-272f-11e7-9ce4-832a8e030fef.py', 'exercise.cs', '.bashrc', 'Example.cs', 'myfig.png', 'file.out', 'line.gif', 'mmm.txt\n', 'temp.txt', 'dddd.txt\n', 'sss.dat\n', 'result.txt', 'output.jpg', '26492-1274250701.png', 'mytest.txt']`

3.4. Khác

23/. Viết chương trình thu thập các đối số của dòng lệnh, sau đó cho biết tên của script, số lượng đối số và danh sách các đối số có trong dòng lệnh.

Ví dụ, với script có tên test.py, khi thực hiện dòng lệnh sau:

```
prashanta@server:~$ python test.py arg1 arg2 arg3
```

Sẽ thu được kết quả là

```
This is the name/path of the script: test.py
('Number of arguments:', 4)
('Argument List:', "['test.py', 'arg1', 'arg2', 'arg3']")
```

24/. Viết chương trình cho nhập tên (đặt là path). Phân tách path thành 2 phần: phần 1 gồm tên file (kể cả đường dẫn nếu có), phần 2 gồm phần mở rộng của.

🏳 *Hướng dẫn*: sử dụng các phương thức `os.path.splitext()`

Ví dụ, với các tên được nhập lần lượt là: `['test.txt', 'filename', '/user/system/test.txt', '/', '' ]`

Sẽ cho kết quả:

```
"test.txt"                : ('test', '.txt')
"filename"                : ('filename', '')
"/user/system/test.txt"   : ('/user/system/test', '.txt')
"/"                       : ('/', '')
""                        : ('', '')
```

THAM KHẢO

[1]. https://www.w3resource.com/python-exercises/

[2]. https://www.w3schools.com/python/python_exercises.asp

[3]. https://www.programiz.com/python-programming

[4]. https://quantrimang.com/hon-100-bai-tap-python-co-loi-giai-code-mau-142456