Contents

[1.NGUYÊN TỐ 8](#_Toc182547689)

[2.LÃI SUẤT NGÂN HÀNG 11](#_Toc182547690)

[3.SỐ MAY MẮN 2 15](#_Toc182547691)

[5.CHỮ HOA – CHỮ THƯỜNG 18](#_Toc182547692)

[5.HOÁN VỊ KÝ TỰ 19](#_Toc182547693)

[6.ƯU THẾ NGUYÊN TỐ 21](#_Toc182547694)

[7.ĐẦU CUỐI NGUYÊN TỐ 24](#_Toc182547695)

[8.TỔNG CHỮ SỐ \_ TÍCH CHỮ SỐ 26](#_Toc182547696)

[**1. Mục đích của đoạn code:** 28](#_Toc182547697)

[**3.** **Chi tiết từng phần trong code** 29](#_Toc182547698)

[**# i là vị trí trong list bắt đầu từ 0 ,1 ,2 ,…..len(a)-1** 29](#_Toc182547699)

[**3. Ví dụ minh họa** 29](#_Toc182547700)

[**In kết quả:** 29](#_Toc182547701)

[9.LÀM TRÒN SỐ 30](#_Toc182547702)

[10.LIỆT KÊ SỐ ĐẸP 33](#_Toc182547703)

[11.ƯỚC SỐ CHUNG NGUYÊN TỐ 34](#_Toc182547704)

[12.CHIA HẾT CHO K 36](#_Toc182547705)

[13.SỐ KHÔNG GIẢM 37](#_Toc182547706)

[14. GIẢI MÃ 39](#_Toc182547707)

# 1.NGUYÊN TỐ

Định nghĩa :

* **Nguyên tố cùng nhau (Coprime)** *: 2 số nguyên mà ước số chung lớn nhất (GCD) của chúng bằng 1*

Nghĩa là chúng ko có ước số nào ngoài 1

VD :

gcd (8,15) =1 .là coprime

gcd (8,12)=4 .Không phải là coprime

* **Giải nghĩa phần input -output**

Số 2 đầu tiên : là có 2 trường hợp kiểm tra sau

2 , 3: là số được kiểm tra

**# Trường hợp 1 : n=2**

Các số nhỏ hơn 2 là : 1

Số coprime với 2

Gcd (1,2)=1 .coprime

Số lượng coprime :1 => k=1 và ko phải số nguyên tố => output :No

**# Trường hợp 2 : n=3**

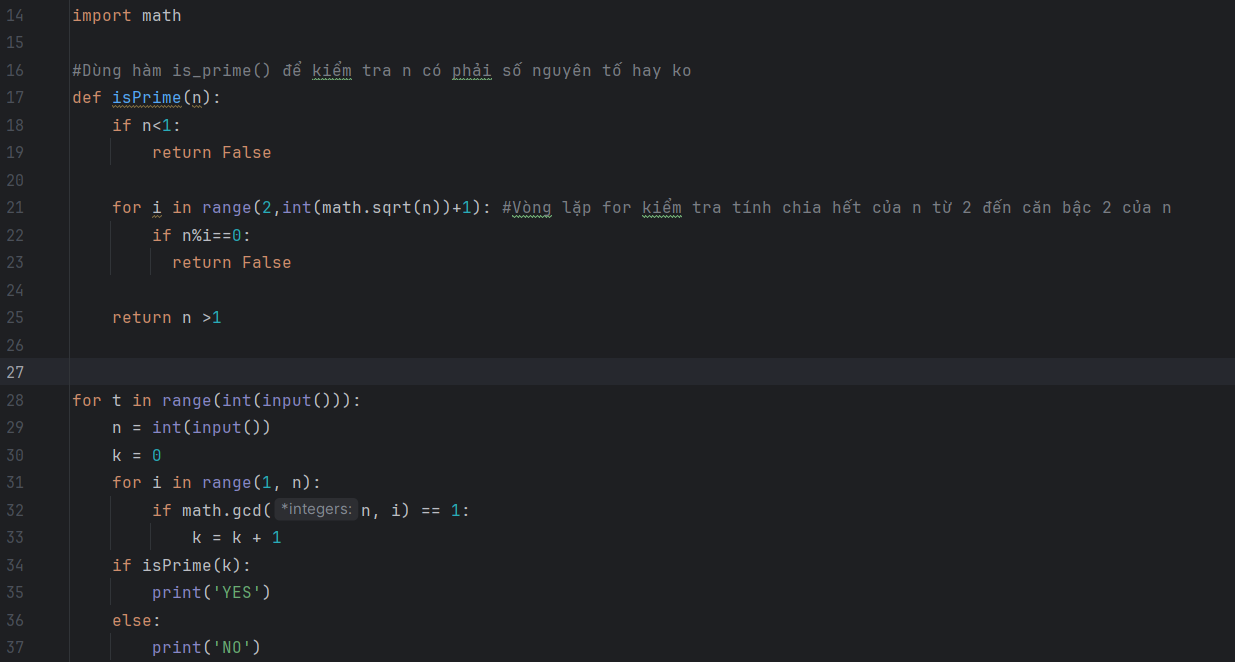
Các số nhỏ hơn 3 là : 1 , 2 ,3

Số coprime với 3

Gcd(1,3) =1 .coprime

Gcd(2,3) =1 .coprime

Số lượng coprime : 2 => k=2 và là số nguyên tố => output : YES



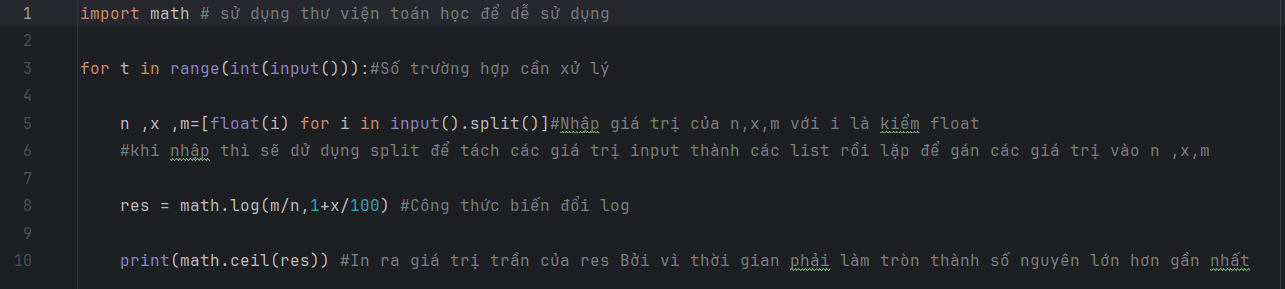
# 2.LÃI SUẤT NGÂN HÀNG

* Công thức Lãi suất kép :



* Yêu cầu input :
* Số đầu tiên là nhập số tình huống cần xử lý
* Nhập giá trị của m (số tiền mong muốn) ; n (số tiền ban đầu) ; x (lãi suất tính theo %) ; res (thời gian )
* Đầu ra output :
* Số năm ( res ) được tính ra
* Xử lý

-Sử dụng thư viện import math : **math.log ()** , **math.ceil ()** (lấy giá trị trần-số nguyên lớn nhất gần nhất )



* Giải thích code

1. **n , x , m =[float(i) for i in input( ).split ( ) ]**
2. **Input()** :Nhập giá trị từ bàn phím của người dùng
3. **.splist()** nó chia chuỗi thành các phần dựa vào khoảng trắng và trả về một danh sách list các chuỗi con

VD : nhập vào 300 4 4000

Trả về [‘300’ ,‘4’ ,’4000’]

1. **[float(i) for I in …]**

**List Comprehension**: Đây là một cách ngắn gọn để tạo một danh sách mới từ một danh sách khác

* **Cách hoạt động**:

**for i in ...** sẽ lặp qua từng phần tử trong danh sách do split() tạo ra.

**float(i)** sẽ chuyển mỗi chuỗi (string) i thành số thực (float).

Kết quả là một danh sách chứa các số thực. Với ví dụ trước, nó sẽ trở thành [1.5, 2.0, 3.3].

**n ,x ,m =…**

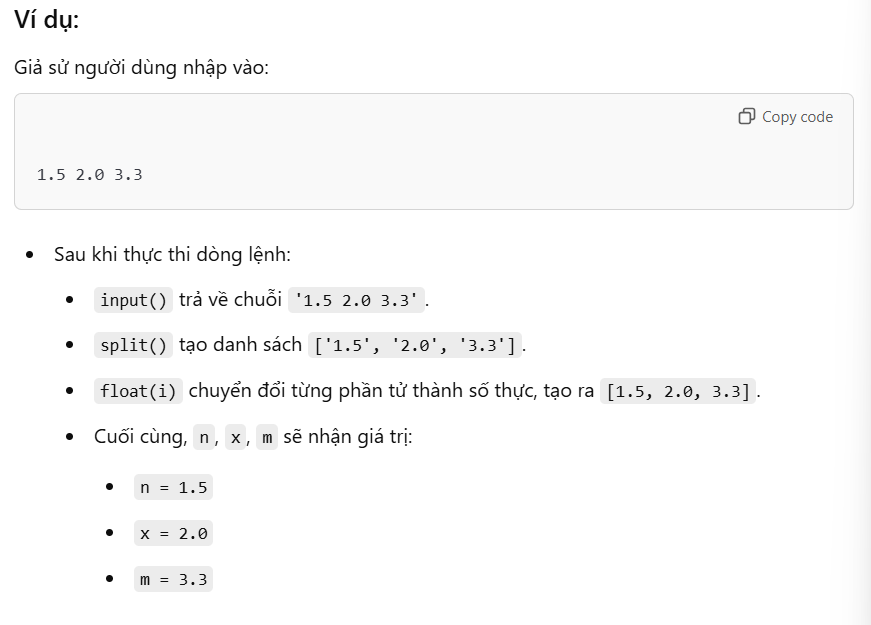
 **Gán giá trị**: Dòng lệnh này sẽ gán từng giá trị trong danh sách mới được tạo ra cho các biến n, x, và m theo thứ tự.

 **Cách hoạt động**:

Giá trị đầu tiên trong danh sách (n) sẽ được gán cho n.

Giá trị thứ hai (x) sẽ được gán cho x.

Giá trị thứ ba (m) sẽ được gán cho m.



# 3.SỐ MAY MẮN 2

* Yêu cầu : Kiểm tra một chuỗi số xem có phải là số may mắn ko (may mắn nếu trong đó chỉa có chữ số 4 và 7 )
* Phân tích input và output

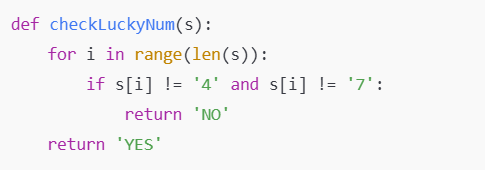
-Chữ số đầu tiên sẽ là số trường hợp cần **for i in range ( int(input())**

**-**Sẽ trả ra kết quả vừa kiểm tra là YES hoặc NO

* Phân tích

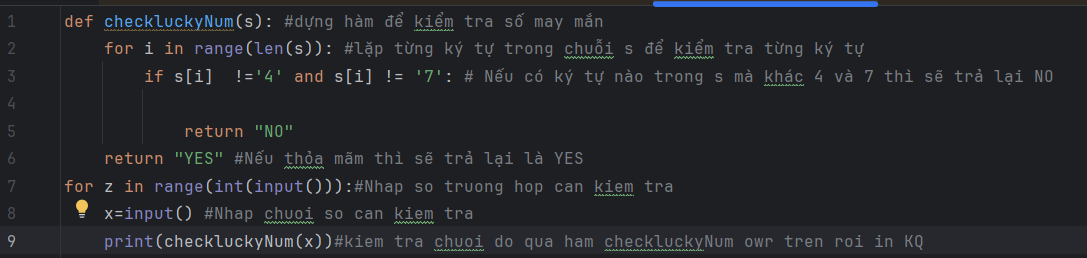
-Xây dựng một hàm **checkLuckyNum(s)** để kiểm tra chuỗi s có chữ số 4 hoặc 7 không

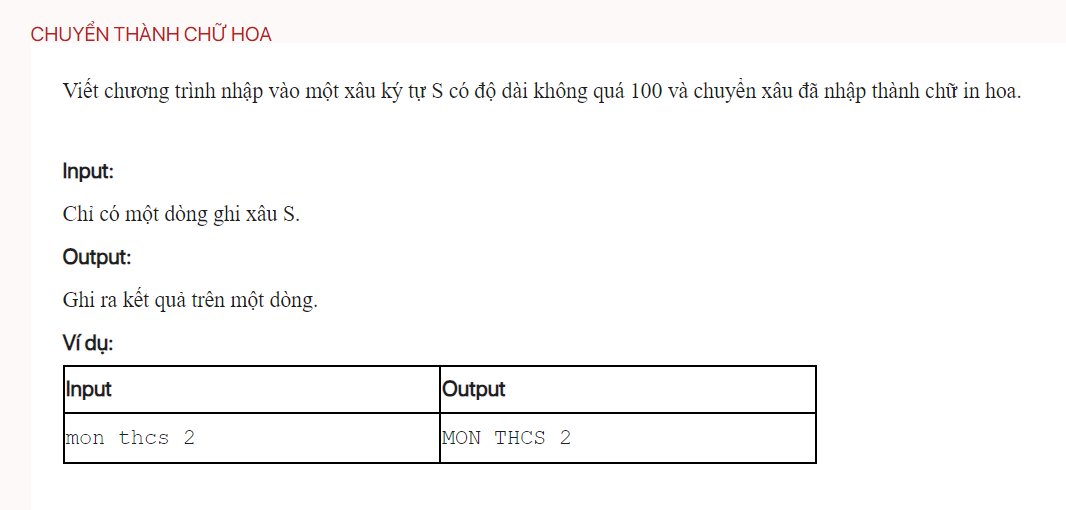
-Cách hoạt động

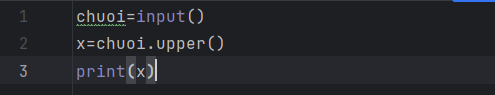


* + 1. **for I in range (len(s))**: Vòng lặp này lặp qua từng chỉ số của chuỗis
    2. **s[i]** : Truy cập ký tự tại vị trí i
    3. **if s[i] != “4” and s[i] !=”7” :** thì hàm trả về NO

**-**and : chỉ trả về “NO”khi cả 2 vế đề thõa mãn điều kiện khác

* + 1. Nếu vòng lặp kết thúc mà không gặp ký tự nào ko hợp lệ thì hàm sẽ trả về YES

4.CHUYỂN THÀNH CHỮ HOA

* Sử dụng **: string.upper()** để chuyển tất cả ký tự thường thành ký tự viết hoa

5.CHỮ HOA – CHỮ THƯỜNG

* Yêu cầu :

-Nhập một sâu ký tự vào và kiểm tra nếu số lượng chữ thường nhiều hơn thì chuyển toàn bộ sâu thành viết thường .Và ngược lại nếu hoa nhiều hơn thì chuyển toàn bộ sâu thành viết hoa

1. Nhập chuỗi từ người dùng

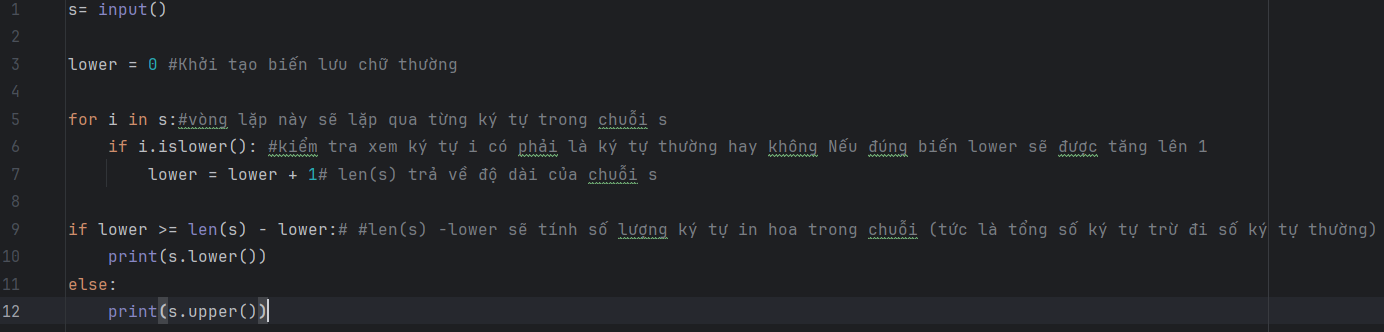
2.Khởi tạo biến đếm

3.Đếm số ký tự thường

4.So sánh số lượng ký tự thường và ký tự in hoa

5.In kết quả

* Phân tích :



# 5.HOÁN VỊ KÝ TỰ

1.Nhập dữ liệu :

-n là chuỗi muốn tìm tất cả hoán vị

2.Mảng đánh dấu :

-Mảng f khởi tạo 1 list với 15 phần tử ,tất cả đều là 0

-Mục đích chính : đóng vai trò như 1 mảng đánh dấu để kiểm tra xem mỗi ký tự của chuỗi n đã được sử dụng trong một hoán vị cụ thể hay chưa

-Danh sách này dùng để đánh dấu các ký tự đã được sử dụng trong hoán vị hiện tại

* Chi tiết
  + f đại diện cho một ký tự trong chuỗi n
  + f[i]=0 cho

# 6.ƯU THẾ NGUYÊN TỐ

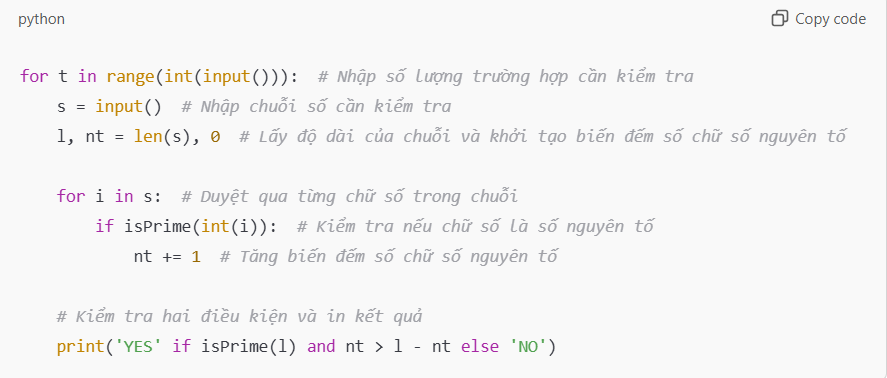
* Yêu cầu : kiểm tra một chuỗi số s xem có thỏa mãn hai điều kiện sau đây hay không:

1. **Độ dài của chuỗi là một số nguyên tố**: Tức là số ký tự trong chuỗi phải là số nguyên tố (lớn hơn 1 và chỉ chia hết cho 1 và chính nó).
2. **Số lượng chữ số nguyên tố trong chuỗi nhiều hơn số lượng chữ số không nguyên tố**:
   * Một chữ số nguyên tố là một chữ số thuộc {2, 3, 5, 7}.
   * Một chữ số không nguyên tố là một chữ số thuộc {0, 1, 4, 6, 8, 9}.
   * Để chuỗi s được coi là "hợp lệ" hoặc "đạt yêu cầu", số lượng chữ số nguyên tố trong chuỗi phải nhiều hơn số lượng chữ số không nguyên tố.

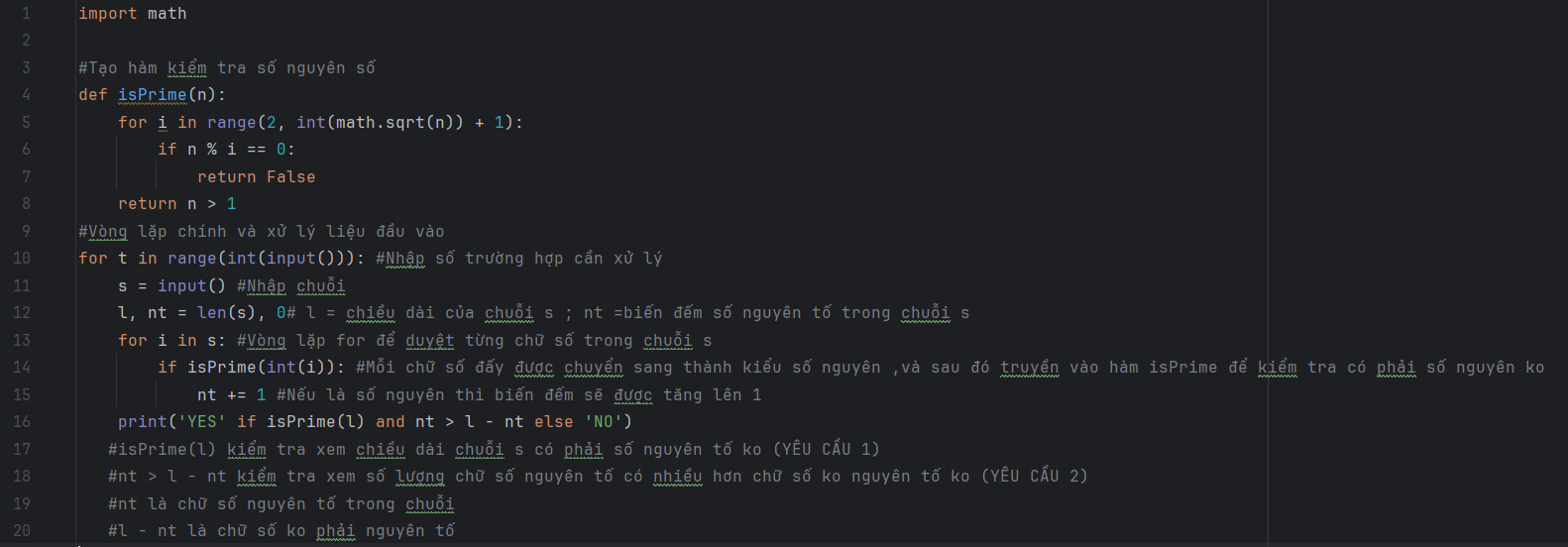
* Phân tích :

1.Xây dựng 1 hàm isPrime (n) để kiểm tra số nguyên số

2.Xây dựng đoạn mã chính



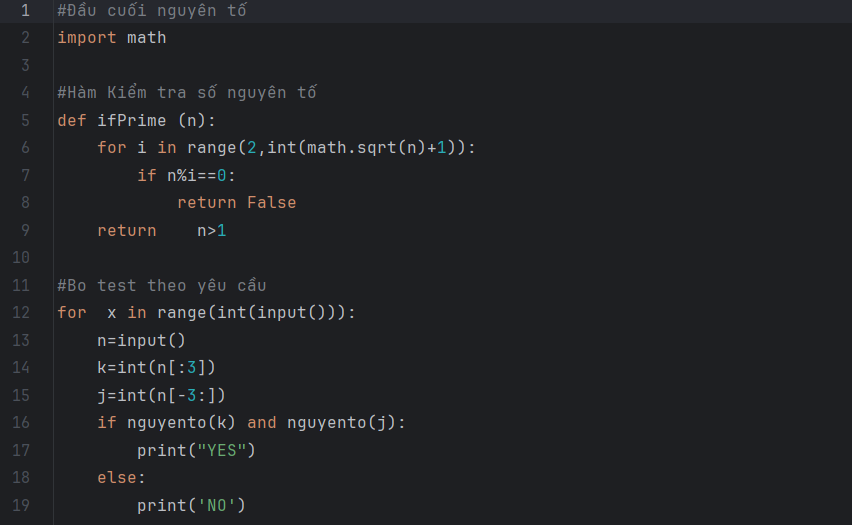
- l,nt =len(s) ,0 tức là :

 + l = len (s) .độ dài của chuỗi s

+ nt = 0 .tạo biến đếm số lượng chữ số nguyên tố trong chuỗi

# 7.ĐẦU CUỐI NGUYÊN TỐ

* Yêu cầu :
  + Kiểm tra xem 3 chữ số đầu và cuối có phải số nguyên tố hay ko
  + Sẽ có một hàm isPrime kiểm tra số nguyên tố
  + Nhập số trường hợp
  + Nhập chuỗi số muốn kiểm tra
  + Trả lại kết quả YES hoặc NO
* Phân tích :



# 8.TỔNG CHỮ SỐ \_ TÍCH CHỮ SỐ

* Yêu cầu :

-Nhập vào số bộ test

-Nhập vào một chuỗi chữ số s (Chuỗi này sẽ ở dạng string )

=> Vậy nên ta sẽ chuyển từng phần tử chủa chuỗi đó thành mỗi một số nguyên và lưu trong 1 list

VD :  
 chuỗi : 1234

Lst =[ 1 , 2 , 3 , 4 ]



+Khi tính tóan sẽ dễ lấy các phần tử ở vị trí chắn hoặc lẻ trong list

#Lưu ý

-Nếu tất cả các chữ số ở vị trí lẻ đề bằng 0 thì tích sẽ bằng 0

-Còn nếu có ít nhất một chữ số khác 0 ở vị trí lẻ thì sẽ nhân các chữ số còn lại đó thành tích

###LƯU Ý NGOÀI LỀ :

* + Phân biệt giữa



for i in range (len (a))

a=lst(……)

-len (a) lấy độ dài của danh sách a

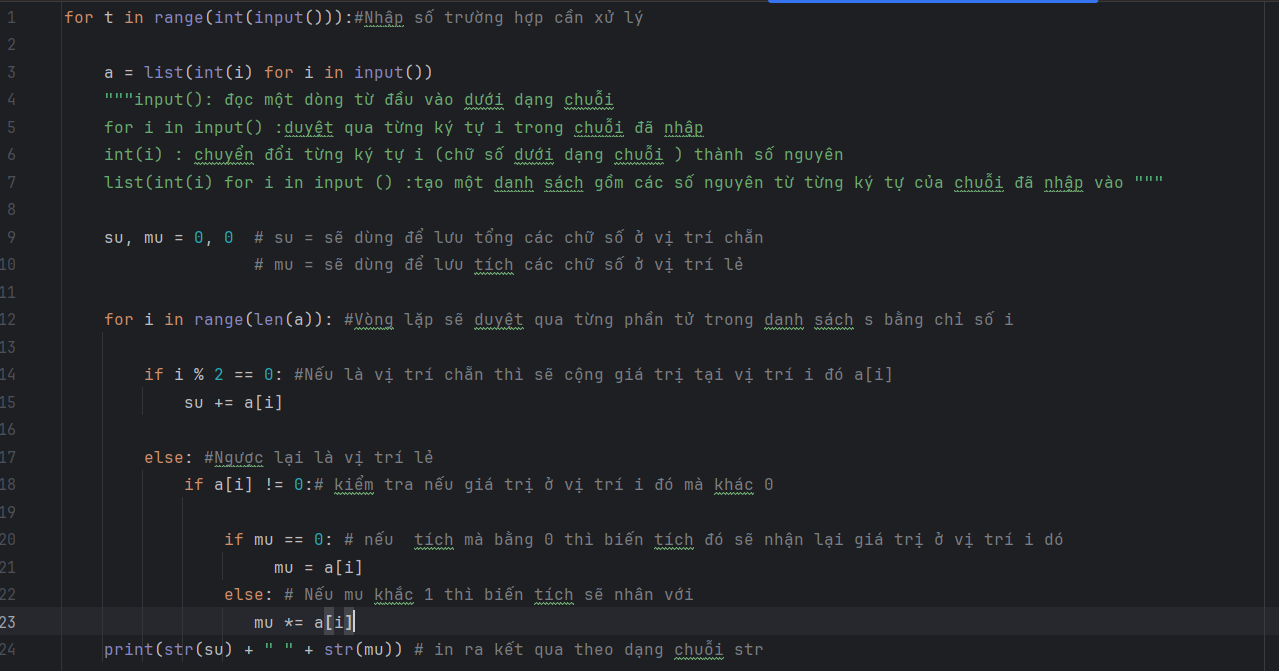
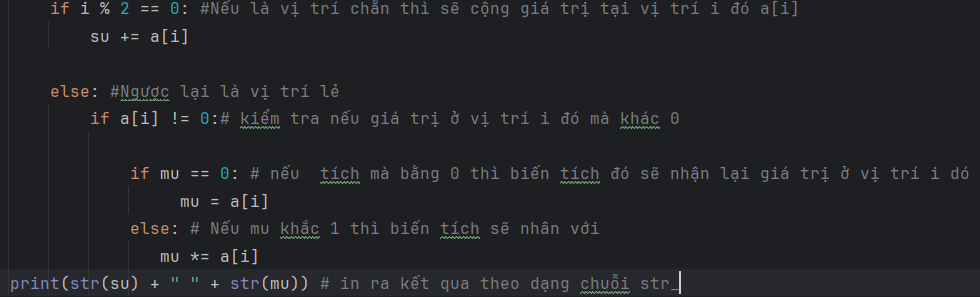
-range(len(a)) tạo một dãy từ : số 0 -> len(a)-1

**print (i)** **print (a[i])**

sẽ in ra số chỉ mục của i sẽ in ra giá trị của phần tử tại vị

( 0 -> n-1) trí i trong danh sách a



* Phân tích :
* Phân tích nhỏ đoạn :

**1. Mục đích của đoạn code:**

Đoạn code này đi qua từng phần tử của danh sách a và:

* Cộng các giá trị ở vị trí **chẵn** vào biến su.
* Tính tích các giá trị **khác 0** ở vị trí **lẻ** vào biến mu.

1. **Chi tiết từng phần trong code**

**# i là vị trí trong list bắt đầu từ 0 ,1 ,2 ,…..len(a)-1**

* **if i % 2 == 0:**
  + Điều kiện này kiểm tra xem chỉ số i của phần tử có phải là **số chẵn** không (0, 2, 4,...).
  + Nếu chỉ số i là số chẵn, giá trị tại a[i] sẽ được **cộng vào su**.
* **else:**
  + Nếu i là số lẻ (1, 3, 5,...):
    - Đoạn code kiểm tra xem giá trị a[i] có **khác 0** không.
      * Nếu có, nó tiếp tục kiểm tra nếu mu đang là **0** (chưa có giá trị tích ban đầu) thì sẽ gán mu bằng a[i].
      * Nếu mu đã có giá trị khác 0, nó sẽ **nhân thêm a[i] vào mu**.
* **Kết quả**: Sau khi duyệt qua toàn bộ danh sách a, chương trình sẽ in ra giá trị của su và mu dưới dạng chuỗi.

**3. Ví dụ minh họa**

Giả sử ta có danh sách a = [1, 2, 3, 4, 5, 6]. Dưới đây là các bước chương trình thực hiện:

* **Bước 1**: Ban đầu, su = 0 và mu = 0.
* **Duyệt qua từng phần tử trong danh sách**:
  + **i = 0 (chẵn)**: a[0] = 1, su = su + a[0] = 0 + 1 = 1
  + **i = 1 (lẻ)**: a[1] = 2, mu = a[1] = 2 (vì mu đang là 0)
  + **i = 2 (chẵn)**: a[2] = 3, su = su + a[2] = 1 + 3 = 4
  + **i = 3 (lẻ)**: a[3] = 4, mu = mu \* a[3] = 2 \* 4 = 8
  + **i = 4 (chẵn)**: a[4] = 5, su = su + a[4] = 4 + 5 = 9
  + **i = 5 (lẻ)**: a[5] = 6, mu = mu \* a[5] = 8 \* 6 = 48
* **Kết quả cuối cùng**:
  + su = 9 (tổng các phần tử ở vị trí chẵn: 1 + 3 + 5)
  + mu = 48 (tích các phần tử khác 0 ở vị trí lẻ: 2 \* 4 \* 6)

**In kết quả:**

Kết quả in ra sẽ là:

Copy code

9 48

# 9.LÀM TRÒN SỐ

* Yêu cầu :
  + Làm tròn các chữ số được nhập vào theo quy tắc

#Quy tắc làm tròn

+ Bắt đầu duyệt từ chữ số cuối cùng bên phải và duyệt hướng về bên trái



+ Nếu chữ số hiện tại >= 5 🡪 thì **chữ số bên trái nó sẽ được tăng lên 1**

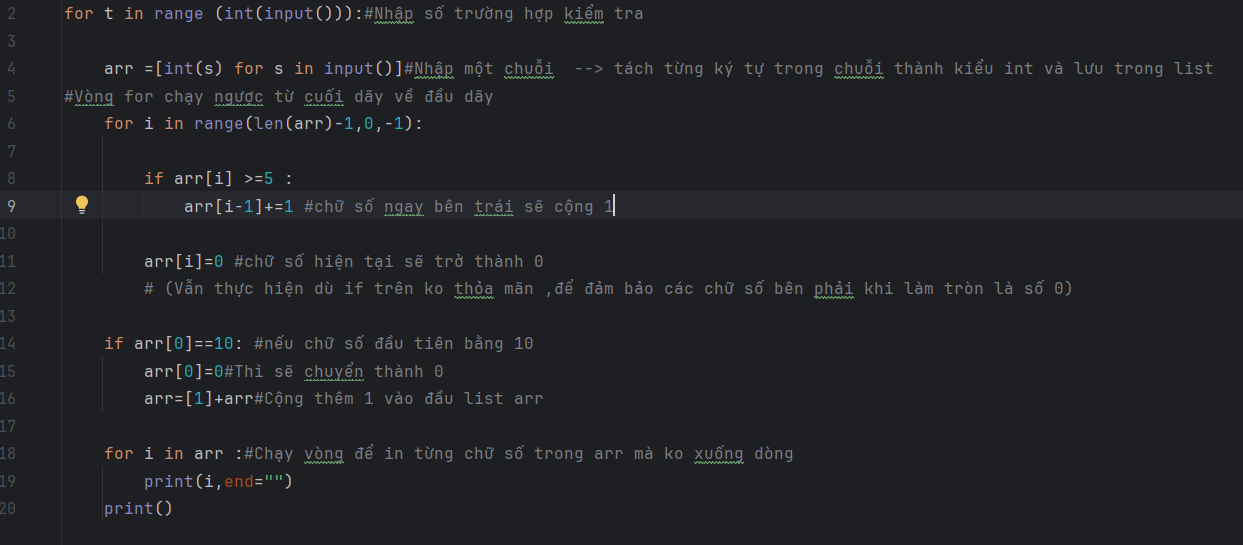
Sau khi làm tròn chữ số hiện tại sẽ được **chuyển thành 0**

+ Tiếp tục làm tròn cho tới khi duyệt qua hết tất cả các chữ số .Đây nghĩa là ta sẽ làm tròn theo từng cặp chữ số từ bên phải sang trái cho đến chữ số đầu tiên

+**Đối với trường hợp Đặc biệt**  Nếu **chữ số đầu tiên trong chuỗi trở thành 10** (đã vượt quá giới hạn) thì ta sẽ đổi chữ số này thành 0 và thêm 1 ở đầu chuỗi

Vd Sau khi là trong [10 ,0 ,0] 🡪chuyển thành [1,0,0,0]





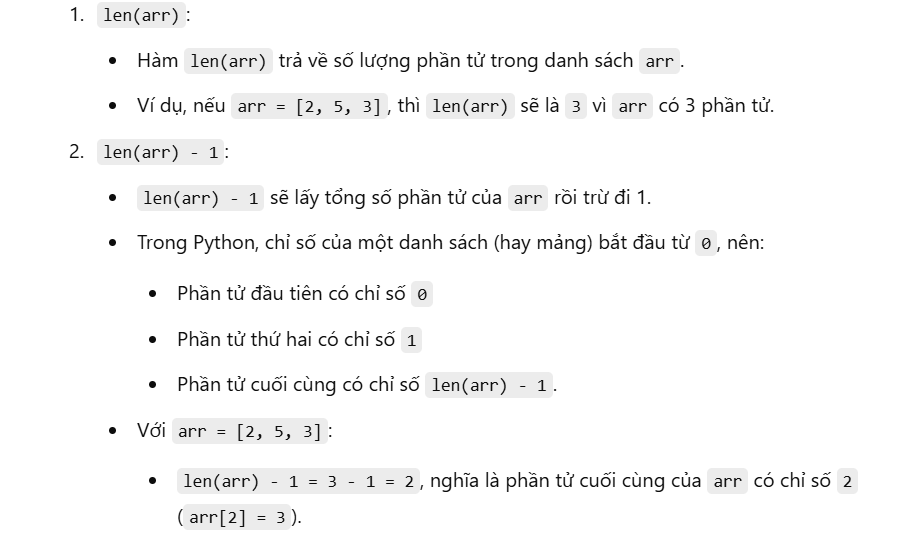
* Phân tích

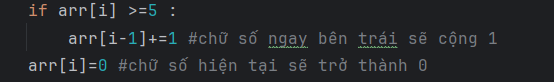
-Đoạn mã làm tròn số theo quy tắc :

for i in range (len(arr)-1 , 0 , -1)

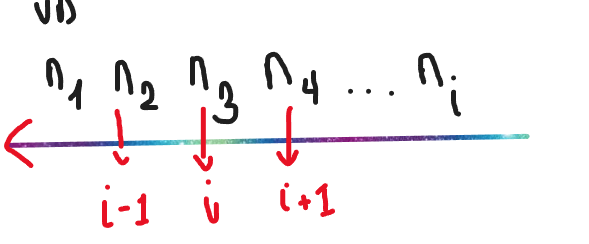


\*len(arr)-1 : là ở cuối





-Nếu là giá trị ở vị trí [i] >=5 thì sẽ cộng 1 vào giá trị của vị trí [i -1]



-Thực hiện chuyển giá trị ở vị trí [i] hiện tại thành 0 dù điều kiện if phía trên sai .dòng này áp dụng cho cả 2 trường hợp



# 10.LIỆT KÊ SỐ ĐẸP

* Yêu cầu

-Tìm và in các số chẵn từ 22 đến n-1 sao cho số đó

+ Có chiều dài chẵn

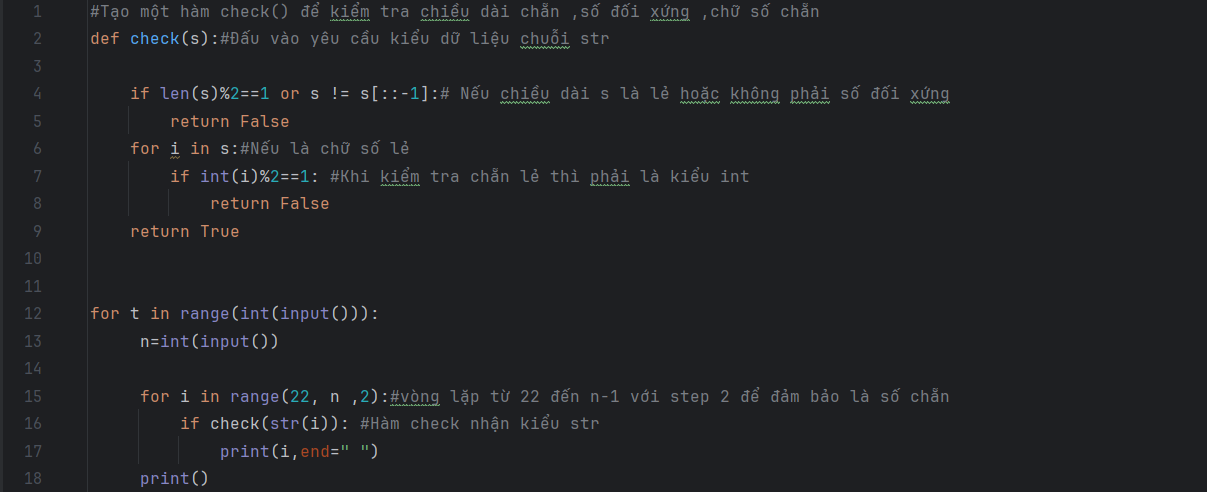


+ Là số đối xứng



+ Chỉ chứa các chữ số chẵn

* Phân tích :
  + Xây dựng một hàm check() để kiểm tra các điều kiện của chữ số
  + Xây dựng vòng lặp chính để nhập dữ liệu vào ,truyền kiểm tra và in ra KQ



# 11.ƯỚC SỐ CHUNG NGUYÊN TỐ

* Yêu cầu:

-nhập vào 2 số nguyên dương A và B

🡪 tìm GCD (A,B)=X

🡪Tổng các chữ số của X là một số Nguyên tố thì trả về YES còn ko phải thì trả về NO

* Phân tích

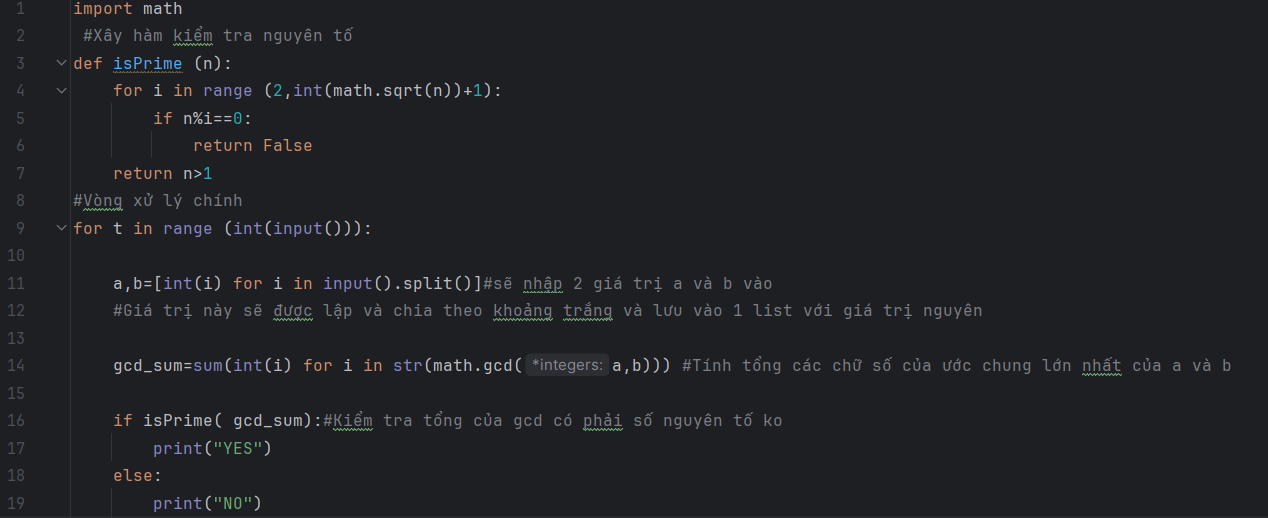
1.Xây dựng hàm kiểm tra nguyên tố

2.Nhập số bộ test

3.Nhập giá trị của 2 số nguyên dương a và b

4.Tính tổng của gcd(a,b)

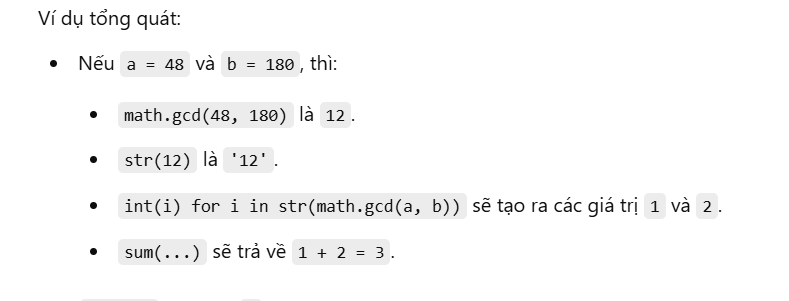
5.Kiểm tra tổng gcd có phải số nguyên tố ko và trả lại kết quả



* Phân tích dòng :sum(int(i) for i in str(math.gcd (a,b)))
  + Math.gcd (a,b) : tính ước chung lớn hất của 2 số a và b
  + Str(math.gcd(a,b) : chuyển giá trị gcd thành chuỗi để có thể lập được qua từng chữ số trong nó

- int(i) for i in …. : Duyệt qua từng l=ký tự trong chuỗi gcd ,chuyển đổi chứ số đó thành kiểu số nguyên để có thể tính tổng

- Sum(…) :tính tổng tất cả các chữ số đã được chuyển



# 12.CHIA HẾT CHO K

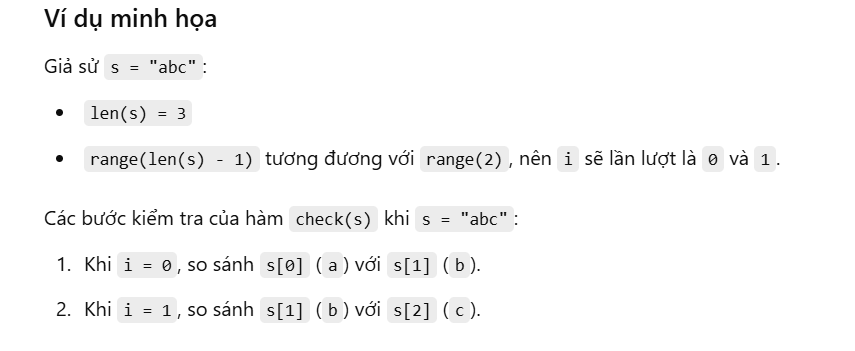
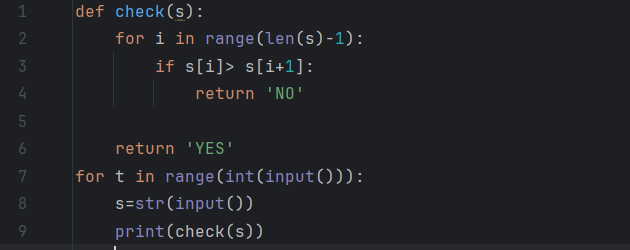
* Yêu cầu :

# 13.SỐ KHÔNG GIẢM

* Yêu cầu :
  + Sẽ kiểm tra cặp số sau với cặp số trước
  + Nếu tồn tại cặp nào số sau mà bé hơn số trường thì sẽ trả về NO
* Phân tích :

-Xây dựng một hàm để kiểm tra

-Trong hàm sẽ có vòng lặp chạy từ 0 đến len(s)-1 vì chỉ kiểm tra đến chữ số trước số cuối cùng (Cặp số cuối ) Nếu để là len(s) thì sẽ bị thiếu 1 chữ số (Mà lúc này chuỗi đã hết )



# 14. GIẢI MÃ

* Yêu cầu

- Xử lý chuỗi đầu vào s theo định dạng Ký tự và số

- Mỗi ký tự trong chuỗi s thì chương trình sẽ lập lại đứng số lần theo số đếm ngay đằng sau

- VD: A3G1 🡪 AAAG

* Phân tích :
  + Xây vòng lập số bộ test
  + Nhập chuỗi s đầu vào với định dạng ký tự đan xen số
  + Khởi tạo một chuỗi rỗng res để lưu kết quả
  + Tạo vòng lặp qua các ký tự và số đếm trong chuỗi s
  + Lặp lại ký tự theo số đếm và nối vào chuỗi res
  + In kết quả
* Chi tiết
  + Vòng lặp : for i in range (0,len(s),2)

+ Vòng sẽ chạy từ 0 đến len(s) -1 với step là 2 ( tăng i lên 2 sau mỗi lần lặp)

+Vòng lặp được thiết kế để duyệt qua từng ký tự và số đếm trong s theo cặp

+Với i là vị trí của ký tự và i + 1 là vị trí của số đém e (số lần ký tự cần lặp lại )

\*VD : s= ‘a3b2’

- len(s) = 4 và khi đó range sẽ chạy từ 0 🡪 4 với step 2 : tạo ra dãy 0 ,2

Khi i=0 ta được ký tự s[0]= ‘a’ và số đếm s[1]= ‘3’

Khi i=2 ta được ký tự s[2]= ‘b’ và số đếm s[3]= ‘2’

* + Lặp lại ký tự theo số đếm và nối vào chuỗi res :

res +=s[i] \* int( s[i+1] )

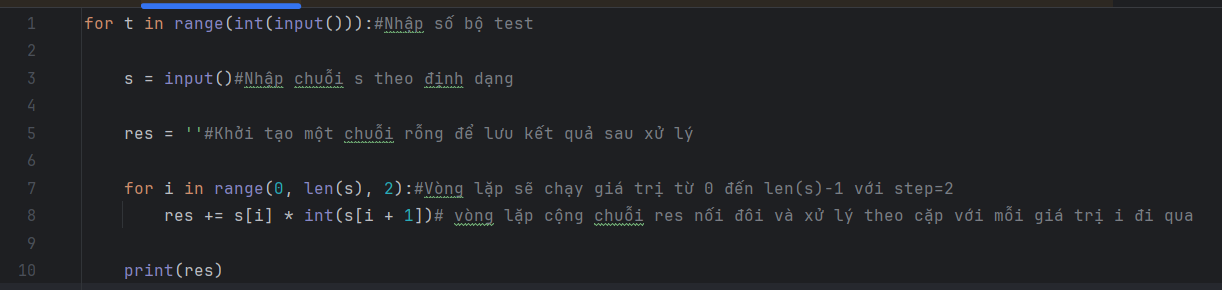
+ s[i] là ký tự cần lặp lại

+s[i+1] là ký tự số biểu thị số lần cần lặp lại ký tự

+int(s[i+1]) để chuyển ký tự “số” thành kiểu chữ số int để có thể sử dụng phép toán nhân

\*VD s= ‘a3’

Khi i = 0 : s[i] = ‘a’ và int (s[i+1]) = 3 🡪 ‘a’ \* 3= ‘aaa’



# 15.MÃ HÓA 1

* Yêu cầu :
  + Ngược lại với bài giải mã trên
  + Chương trình **mã hóa theo dạng Run-Length Encoding (RLE)**, nghĩa là biểu diễn chuỗi bằng cách đếm số lượng ký tự liên tiếp giống nhau, kèm theo ký tự đó.
  + VD:

AAA 🡪 3A

* Phân tích

1.Nhập số lượng bộ test

2.Đọc chuỗi và thêm ký tự đặc biệt (Ký tự kết thúc)

3.Khởi tạo các biến đếm

4.Duyệt qua từng ký tự trong chuỗi

- Kiểm tra ký tự hiện tại :

+ Nếu gặp ký tự giống

+ Nếu gặp ký tự khác

5.Kết thúc

* Chi tiết