Bài tập python

I.Bài tập Dictionary

1.Đếm số lần xuất hiện của các từ trong một câu: Tạo một dictionary, với key là từ và value là số lần xuất hiện. Duyệt qua từng từ trong câu và cập nhật dictionary.

In [ ]:
```
p = "hello , hello python , hello . python is great world"

clean  = p.replace(",", "").replace(".", "")

def count_word(clean):
    list_p = clean.lower().split()
    count = {}

    for i in list_p:
        if i in count and i.isalnum() == True:
            count[i] +=1
        else:
            count[i] = 1
    return count

print(count_word(clean))
```

2.Tạo một danh bạ điện thoại: Mỗi người là một entry trong dictionary, với key là tên và value là số điện thoại. Thực hiện các chức năng: thêm, xóa, tìm kiếm liên lạc.

In [ ]:
```
phonebook = {
    "Nguyễn Văn A": "0987654321",
    "Trần Thị B": "0123456789",
    "Lê Văn C": "0912345678",
    "Phạm Thị D": "0345678912",
    "Vũ Văn E": "0876543210",
    "Hoàng Thị F": "0967891234",
    "Đặng Văn G": "0789123456",
    "Bùi Thị H": "0234567890",
    "Ngô Văn I": "0901234567",
    "Đỗ Thị K": "0678912345"
}

def add_contact(name, phone):
    phonebook[name] = phone
    print("Added name: ", name)

def delete_contact(name):
    if name in phonebook:
        del phonebook[name]
        print("Đã xóa: ", name)
```

```
    else:
        print("Ko tìm thấy: " , name)

def search_contact(name):
    if name in phonebook:
        print("Phone: " , phonebook[i])
    else:
        print ("Name không tồn tại: ", name)


while True:
  print("1. Thêm liên lạc")
  print("2. Xóa liên lạc")
  print("3. Tìm kiếm liên lạc")
  print("4. Xuất ")
  print("5. Thoát")

  choice = input("\n Chọn chức năng: ")

  if choice == '1':
    name = input("Nhập tên: ")
    phone = input("Nhập số điện thoại: ")
    add_contact(name, phone)
  elif choice == '2':
    name = input("Nhập tên cần xóa: ")
    delete_contact(name)
  elif choice == '3':
    name = input("Nhập tên cần tìm: ")
    search_contact(name)
  elif choice == '4':
    print ("Phone book: " , phonebook)
  elif choice == '5':
      break
  else:
    print("Lựa chọn không hợp lệ.")
```

3.Đảo ngược một dictionary: Tạo một dictionary mới, với key và value đổi chỗ cho nhau.

In [3]:
```
phonebook = {
    "Nguyễn Văn A": "1",
    "Trần Thị B": "2",
    "Lê Văn C": "3",
    "Phạm Thị D": "4",
    "Vũ Văn E": "5",
    "Hoàng Thị F": "6",
    "Đặng Văn G": "7",
    "Bùi Thị H": "8",
    "Ngô Văn I": "9",
    "Đỗ Thị K": "10"
}

reversed_dict = {value: key for key, value in phonebook.items()}

print(reversed_dict)
```

```
{'1': 'Nguyễn Văn A', '2': 'Trần Thị B', '3': 'Lê Văn C', '4': 'Phạm Thị
D', '5': 'Vũ Văn E', '6': 'Hoàng Thị F', '7': 'Đặng Văn G', '8': 'Bùi Th
ị H', '9': 'Ngô Văn I', '10': 'Đỗ Thị K'}
```

4.Tìm giá trị lớn nhất/nhỏ nhất trong một dictionary: Xét cả trường hợp key là số và value là số.

In [11]:
```
my_dict = {'a': 10, 'b': 3, 'c': 7, 'd': 20, 1: 2, 2: 7, 3: 9}

print(max(my_dict.values()))
print(min(my_dict.values()))
```

```
20
```

5.Kiểm tra xem hai dictionary có giống nhau hay không: Cả về số lượng phần tử, khóa và giá trị.

In [13]:
```
dict1 = {'a': 10, 'b': 3, 'c': 7}
dict2 = {'a': 10, 'b': 4, 'c': 7}
dict3 = {'a': 10, 'c': 7, 'b': 3}

print(dict1 == dict2)  # False
print(dict1 == dict3)  # True (thứ tự khóa không quan trọng trong dict)
```

```
False
True
```

6.Ghép hai dictionary:
Nếu có khóa trùng nhau, ưu tiên giá trị của dictionary sau.

In [22]:
```
dict1 = {'a': 10, 'b': 3, 'c': 7}
dict2 = {'a': 10, 'b': 4, 'c': 7}

dict3 = dict1 | dict2
print(dict3)
```

```
{'a': 10, 'b': 4, 'c': 7}
```

7.Tạo một dictionary chứa các số nguyên tố từ 1 đến n: Key là số nguyên tố, value là True.

In [30]:
```
def prime_dictionary (n):
    primes = [True]*(n-1)
    num = 2

    while num*num <=n:
        if primes[num] == True:
            #Đánh dấu bội của num => ko là số nguyên tố
            for i in range (num*num , n+1 , num ):
                primes[i] = false

    prime_dict = {i: True for i in range(2, n+1) if primes [i]}
    return prime_dict

# call func
n = int (input("Enter n: " ))
```

```
result = create_prime_dict(n)
print(result)
```

```
{2: True, 3: True, 5: True, 7: True, 11: True, 13: True, 17: True, 19: T
rue, 23: True, 29: True, 31: True, 37: True, 41: True, 43: True, 47: Tru
e, 53: True}
```

II.Bài tập tổng hợp

Bài 1.

Cho một số nguyên, in "YES" nếu chữ số cuối của nó là 7 và in "NO" nếu không. Ví dụ: 127 → YES 333 → NO

In [ ]:
```
num_a = int(input("Nhập số nguyên"))
print(num_a);
if (num_a %10 ==7):
    print("YES")
else:
     print("NO")
```

Bài 2.

Cho biết tọa độ của ba điểm A, B, C trên một đoạn thẳng. In khoảng cách từ điểm A đến điểm gần nó nhất. Ví dụ: 10 35 30 Kết quả: 20

In [ ]:
```
point_a = int(input("Input a:"))
point_b = int(input("Input b:"))
point_c = int(input("Input c:"))

distance = (point_c - point_a) - (point_b - point_a)

if (distance < 0):
    print("min distance C", -distance)
elif (distance > 0):
    print("min distance B", distance)
else:
    print("min distance A = C", distance)
```

Bài 3.

Viết chương trình tính tổng các số trong một danh sách Ví dụ: Input: [2, 5, 8, 10, 12] Output: Sum = 37

In [ ]:
```
my_list = list(map(int, input("Nhập các phần tử (cách nhau bởi dấu cách
print(my_list)

result = sum(my_list)
print("sum = " result)
```

Bài 4.

Viết game đoán số may mắn (guess the number) ** Máy tính nghĩ một số random từ 1 cho đến 15 và hỏi bạn đoán. Máy tính sẽ nói cho bạn khi bạn đoán sai là số may mắn là phải lớn hơn hoặc nhỏ hơn. Bạn sẽ chiến thắng nếu bạn đoán đúng số đó trong 5 lượt chơi.

****Gợi ý:**

```
import random
'# random number from 1 to 15 <br>
random_number = random.randint(1, 15)
```

Máy tính sẽ tạo một số ngẫu nhiên từ 1 cho đến 15

Input/Output:

Tôi đang nghĩ một số giữa 1 và 15. Bạn hãy đoán số may mắn đó giúp tôi.

Số may mắn đó là: 10

Số bạn đoán phải nhỏ hơn 10

Số may mắn đó là: 2

Số bạn đoán phải lớn hơn: 2

Số may mắn đó là: 4

Chúc mừng bạn đã chiến thắng. Bạn đã đoán đúng sau 2 lượt chơi.

- Nếu sau 5 lượt chơi bạn không đoán được thì kết thúc chương trình và in ra

Bạn đã không may mắn, đây là số 4

In [ ]:
```
import random

# 5 lần chơi
count = 5
# random tu 1 - 15
random_number = random.randint(1, 15)
print("hidden", random_number)

for i in range(count):
    # Nhập số đoán
    guess = int(input("Nhập số đoán:"))
    # Đoán số == random
    if guess == random_number:
        # Xuất nhận xét thắng cuộc
        print(f"Bạn đã đúng sau {i+1} lượt chơi")
        break
    elif guess < random_number:
        print("Nhỏ hơn số thực tế")
    elif guess > random_number:
        print("Lớn hơn số thực tế")

 if i == count-1: # Xuất nhận xét thua
     print("Thua cuộc")
```

Bài 5.

Một cửa hàng sẽ giảm giá 10% nếu tổng chi phí mua hàng lớn hơn 10.000 Người dùng về số lượng từ bàn phím Giả sử, một đơn vị mặt hàng sẽ có giá 100 đồng. In tổng chi phí hóa đơn cho người dùng. **Ví dụ:** Input: 120 Output: 10800 Input: 20 Output: 2000

In [19]:
```
num = int(input ("Nhập số sp: ") )
print("num:", num)
# sum = num*100
sum = num*100
print("sum:", sum)
#sum > 10.000  reduce 10% => sum*(90/100)
if (sum > 10000 ):
    sum = sum*0.9
    print("Total payment: ", f"{sum:,.2f}")
else:
    print("Total payment: ", f"{sum:,.2f}")
```

```
num: 1000
sum: 100000
Total payment:  9,000,000.00
```

Bài 6.

Viết chương trình in ra tất cả các số chẵn trong một danh sách

Ví dụ:

Input: [2, 5, 8, 10, 12]

Output: 2, 8, 10, 12

In [2]:
```
arr = list(filter(lambda x: x%2 == 0,map(int,(input("Nhap arr").split()
print (arr)
```

```
[2, 4, 6]
```

Bài 7.

Viết một chương trình tìm ra số lớn nhất của một danh sách mà không sử dụng hàm max()

Ví dụ:

Input: [2, 5, 8, 10, 12]

Output: 12

In [25]:
```
arr = list(map(int,input("Nhập mảng:").split()));
print(arr)
temp_max= arr[0]

for i in arr:
    if temp_max <= i:
        temp_max = i
print("Max is:", temp_max)
```

```
[1, 2, 5, 2, 5, 2, 4, 5]
Max is: 5
```

Bài 8.

Cho một danh sách các số, hãy tìm và in tất cả các phần tử lớn hơn phần tử trước đó.

Ví dụ1:

Input: 1 5 2 4 3

Output: 5 4

Ví dụ 2:

Input: 5 5 5 5 5

Output: 5 5 5 5

In [54]:
```
# Hàm tìm & in phần tử lớn hơn
def find_larger_ele(arr_num):
    large_ele = arr_num[0]
    for i in arr_num[1:]: # bat dau tu phan tu thu 2
        if i >= large_ele:
            print (i, end =" ")
        large_ele = i
# Input
arr_num = list(map(int, input ("Nhap mang: ").split()))
print(arr_num)
find_larger_ele(arr_num);
```

```
[2, 3, 52, 4, 5, 5, 3]
3 52 5 5
```

Bài 9.

Viết một chương trình xóa tất cả phần tử lặp lại (trùng lặp) ra khỏi danh sách.

Ví dụ:

Input: [1, 3, 5, 6, 3, 5, 6, 1]

Output: [1, 3, 5, 6]

In [34]:
```
arr = list (map(int , input("Nhap mang:").split()))
arr = list(dict.fromkeys(arr))
print(arr)
```

```
[2, 3, 4, 5, 6]
```

Bài 10.

Get first, second best scores from the list.

List may contain duplicates.

Input: [86,86,85,85,85,83,23,45,84,1,2,0]

Output: should get 86, 85

In [45]:
```
arr = list (map(int , input("Nhap mang:").split()))
arr = list(dict.fromkeys(arr))
max_val_1 = max(arr)
tem_arr = arr.remove(max_val_1)
max_val_2 = max(arr)
print(arr)
print("Top 1:", max_val_1)
print("Top 2:",max_val_2)
```

```
[4, 5, 2, 3]
Top 1: 23
Top 2: 5
```

Bài 11.

Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in min(v1, v2) and max(v1, v2) functions return the smaller or larger of two values.
big_diff([10, 3, 5, 6]) → 7
big_diff([7, 2, 10, 9]) → 8
big_diff([2, 10, 7, 2]) → 8

In [ ]:
```
def big_diff(arr_num):
    max_num = max(arr_num)
    min_num = min (arr_num)
    return max_num - min_num

arr_num = list(map(int, input("Nhap mang:").split()))

diff = big_diff(arr_num)

print (diff)
```

Bài 12.

Viết một chương trình in ra các số chia hết cho 7 nhưng không chia hết cho 5 nằm trong khoảng 100 cho đến 1000 (tính cả 100 và 1000).
Kết quả: In trên một dòng và cách nhau bởi dấu phẩy.

In [53]:
```
arr_num = list(map(int, input("Nhap mang:").split()))
print(arr_num)
result = list(filter(lambda x: x%7 == 0 and  x%5 != 0 and x>= 100 and x
print (result)
```

```
[1, 777, 112, 133, 500, 847]
[777, 112, 133, 847]
```

Bài 13.

Viết một chương trình in tất cả các số nguyên tố nhỏ hơn n. Với n là số nguyên dương nhập từ bàn phím.

Ví dụ:

n = 12

Kết quả:

2 3 5 7 11

In [67]:
```
import math

# function
# Prime: 2=>n
def is_prime (num, n):
    if num <2:
        return False
    elif num >= 2 and num < n:
        for i in range (2, int(math.sqrt(num)+1 )):
            # num = a*b, Both a & b not > sqrt(num) because a*b > n
            if num % i == 0:
                return False
        return True

# call func
n = int(input("Nhập n:"));
# Tạo mảng tăng dần từ 1 đến n
arr_num = list(range(1, n + 1))

result = list (filter(lambda x: is_prime (x, n) == True, arr_num))

print ("Prime list is: ",result)
```

```
Prime list is:  [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Bài 16.

Ask the user to enter a new password. Ask them to enter it again. If the two passwords match, display “Thank you”. If the letters are correct but in the wrong case, display the message “They must be in the same case”, otherwise display the message “Incorrect”

In [ ]:
```
arr_num1 = list(map(str, input("Enter password 1'st: ").split()))
print(arr_num1)
arr_num2= list(map(str, input("Enter password 2'st: ").split()))
print(arr_num2)


if (arr_num1 == arr_num2):
    print("Thank you")
else:
     print("Incorrect")
```

Bài 17.

Viết một chương trình nhập số nguyên dương n và tính tổng của các chữ số của số n. Ví dụ: Nếu người dùng nhập 3141 thì chương trình của bạn nên hiển thị 3 + 1 + 4 + 1 = 9

In [9]:
```
num = int(input("Enter num: "))

temp_arr = []

while num > 0:
    temp_arr.append(num%10)
    num = num//10

print (sum(temp_arr))
```

```
10
```

Bài 18.

Ứng dụng chuyển đổi nhiệt độ từ độ C sang F, chuyển đổi từ kg sang pao(lb), diện tích, thể tích, tốc độ, thời gian (Easy)
https://www.metric-conversions.org/vi/trong-luong/kilogam-sang-pao.htm
Ý tưởng:
Xây dựng một chương trình gồm nhiều chức năng trên, người dùng chọn:
chức năng 1: Chuyển đổi độ C sang độ F
chức năng 2: Chuyển đổi độ F sang độ C
chức năng 3: Chuyển đổi từ kg sang pao(lb)
chức năng 4: Chuyển đổi từ pao(lb) sang kg
chức năng 5: Chuyển đổi từ mét sang feet
chức năng 6: Chuyển đổi từ feet sang meet
chức năng 7: Thoát chương trình

In [24]:
```
def celsius_to_fahrenheit(c):
    return (c * 9/5) + 32

def fahrenheit_to_celsius(f):
    return (f - 32) * 5/9

def kg_to_lb(kg):
    return kg * 2.20462

def lb_to_kg(lb):
    return lb / 2.20462

def meters_to_feet(m):
    return m * 3.28084

def feet_to_meters(ft):
    return ft / 3.28084

def menu():
```

```
    print("\nChương trình chuyển đổi đơn vị:")
    print("1. Chuyển đổi độ C sang độ F")
    print("2. Chuyển đổi độ F sang độ C")
    print("3. Chuyển đổi từ kg sang pao (lb)")
    print("4. Chuyển đổi từ pao (lb) sang kg")
    print("5. Chuyển đổi từ mét sang feet")
    print("6. Chuyển đổi từ feet sang mét")
    print("7. Thoát chương trình")

while True:
    menu()
    choice = input("Chọn chức năng (1-7): ")

    if choice == '1':
        c = float(input("Nhập nhiệt độ (°C): "))
        print(f"{c}°C = {celsius_to_fahrenheit(c)}°F")

    elif choice == '2':
        f = float(input("Nhập nhiệt độ (°F): "))
        print(f"{f}°F = {fahrenheit_to_celsius(f)}°C")

    elif choice == '3':
        kg = float(input("Nhập khối lượng (kg): "))
        print(f"{kg} kg = {kg_to_lb(kg)} lb")

    elif choice == '4':
        lb = float(input("Nhập khối lượng (lb): "))
        print(f"{lb} lb = {lb_to_kg(lb)} kg")

    elif choice == '5':
        m = float(input("Nhập chiều dài (mét): "))
        print(f"{m} mét = {meters_to_feet(m)} feet")

    elif choice == '6':
        ft = float(input("Nhập chiều dài (feet): "))
        print(f"{ft} feet = {feet_to_meters(ft)} mét")

    elif choice == '7':
        print("Thoát chương trình.")
        break

    else:
        print("Lựa chọn không hợp lệ, vui lòng chọn lại.")
```

```
Chương trình chuyển đổi đơn vị:
1. Chuyển đổi độ C sang độ F
2. Chuyển đổi độ F sang độ C
3. Chuyển đổi từ kg sang pao (lb)
4. Chuyển đổi từ pao (lb) sang kg
5. Chuyển đổi từ mét sang feet
6. Chuyển đổi từ feet sang mét
7. Thoát chương trình
```

```
20.0°C = 68.0°F

Chương trình chuyển đổi đơn vị:
1. Chuyển đổi độ C sang độ F
2. Chuyển đổi độ F sang độ C
3. Chuyển đổi từ kg sang pao (lb)
4. Chuyển đổi từ pao (lb) sang kg
5. Chuyển đổi từ mét sang feet
6. Chuyển đổi từ feet sang mét
7. Thoát chương trình
Thoát chương trình.
```

Bài 19.

Làm trò chơi búa, đá, giấy chơi với máy tính

Ý tưởng:

https://youtu.be/eXQNi8hA_yM

Gợi ý:

Cách 1: Nhập số từ bàn phím

import random

choice = int(input("User turn: "))

random_choice = random.randint(1, 3)

Cách 2: Nhập chữ từ bàn phím

import random

options = ["Rock","Paper","Scissors"]

AI = random.choice(options)

Người chơi có 5 lượt chơi với máy tính, sau 5 lượt chơi thì thống kê người chơi đã thắng, hòa, thua bao nhiêu lượt với máy tính.

In [ ]:
```python
import random

options = ["rock", "paper", "scissors"]

# Biến đếm số lượt thắng, thua và hòa
wins = 0
losses = 0
draws = 0

# Vòng lặp cho 5 lượt chơi
for i in range(5):
    print(f"\nLượt chơi {i+1}:")
    user_choice = input("Nhập lựa chọn của bạn (rock/paper/scissors): "

    if user_choice not in options:
        print("Lựa chọn không hợp lệ! Vui lòng nhập lại.")
        continue

    # Máy tính chọn ngẫu nhiên
    AI_choice = random.choice(options)
    print(f"Máy tính chọn: {AI_choice}")
```

```
    # So sánh kết quả
    if user_choice == AI_choice:
        print("Hòa!")
        draws += 1
    elif (user_choice == "rock" and AI_choice == "scissors") or \
         (user_choice == "scissors" and AI_choice == "paper") or \
         (user_choice == "paper" and AI_choice == "rock"):
        print("Bạn thắng!")
        wins += 1
    else:
        print("Bạn thua!")
        losses += 1

# Thống kê kết quả sau 5 lượt chơi
print("\nKết quả sau 5 lượt chơi:")
print(f"Thắng: {wins} lượt")
print(f"Hòa: {draws} lượt")
print(f"Thua: {losses} lượt")
```

In [ ]:

Bài 21.

Làm game chọn nhóm. Có một danh sách gồm 8 người chơi, hãy lựa chọn ngẫu nhiên 4 người chơi không trùng nhau để cho vào nhóm A, còn lại cho vào nhóm B.

In [19]:
```
import random
group = ["per1", "per2", "per3", "per4", "per5", "per6", "per7","per8"]

gr_A = random.sample(group, 4)

gr_B = list(filter ( lambda x: x not in (gr_A) ,group))

print("gr_A:",gr_A)
print("gr_B:",gr_B)
```

```
gr_A: ['per8', 'per1', 'per3', 'per7']
gr_B: ['per2', 'per4', 'per5', 'per6']
```

In [ ]:
```
Bài 22.
Tìm vị trí của giá trị chẵn đầu tiên trong mảng 1 chiều các số nguyên.
mảng không có giá trị chẵn thì sẽ trả về -1
```

In [22]:
```
def find_first_even(arr_num):
    for i in arr_num:
        if i % 2 == 0:
            return i
    return -1

arr_num = list(map(int, input("Nhập mảng: ").split()))
print("arr_num:", arr_num)
```

```
print("Value:", find_first_even(arr_num))
```

```
arr_num: [1, 3, 5, 7]
Value: -1
```

Bài 23.

Given the year number. You need to check if this year is a leap year. If it is, print LEAP, otherwise print COMMON.
The rules in Gregorian calendar are as follows:
a year is a leap year if its number is exactly divisible by 4 and is not exactly divisible by 100
a year is always a leap year if its number is exactly divisible by 400

In [28]:
```
year = int( input("Enter year: "))
print(year)

if(year%400 ==0 or (year%4 == 0 and year%100 != 0) ):
    print("LEAP")
else:
    print("COMMON")
```

```
2017
COMMON
```

Bài 24.

Viết chương trình tính tổng S = 1 + 1/2 + 1/3 + ...+ 1/n với n là số nguyên dương nhập từ bàn phím.

In [34]:
```
n = int(input("Enter n: "))
S = 0
for i in  range (1, n + 1):
    S = S + 1/i

print (S)
```

```
1.8333333333333333
```

Bài 25.

Liệt kê tất cả các ước số của số nguyên dương n.

In [36]:
```
def find_divisor(n):
    arr=[]
    for i in range ( 1, n+1):
        if n % i == 0:
            arr.append(i);
    return arr
```

```
# Input
n = int(input("Enter n:"))

print (find_divisor(n))
```

```
[1, 2, 5, 10]
```

Bài 26.

Phân tích một số thành tích các thừa số nguyên tố Input: n = 120 Output: 2 2 2 3 5

In [45]:
```
def prime_factorization(n):
    arr = []
    i = 2
    # Chia n cho các số nguyên tố từ 2 trở đi
    while n > 1:
        if n % i == 0:  # Nếu n chia hết cho i
            arr.append(i)  # Thêm i vào danh sách thừa số
            n = n // i  # Chia n cho i
        else:
            i += 1  # Tăng i để kiểm tra số tiếp theo
    return arr

# Input
n = int(input("Enter n: "))

print(prime_factorization(n))
```

```
[2, 2, 2, 3, 5]
```

Bài 27.

Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..len(str)-1 inclusive).
missing_char('kitten', 1) → 'ktten'
missing_char('kitten', 0) → 'itten'
missing_char('kitten', 4) → 'kittn'

In [47]:
```
def missing_char(s, n):
    print(s[:n])
    print(s[n+1:])
    return s[:n] + s[n+1:]


print(missing_char('kitten', 4))  # Output: kittn
```

```
kitt
n
kittn
```

Bài 28.

Given a string, return a new string where the first and last chars have been exchanged.
front_back('code') → 'eodc'
front_back('a') → 'a'
front_back('ab') → 'ba'

In [25]:
```
text = input("Enter text: ")
n =len(text)
if(n ==1):
    print(text[0])
else:
    print(text[n-1] + text[1:n-1] + text[0])
```

```
ba
```

Bài 29.

You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases.
caught_speeding(60, False) → 0
caught_speeding(65, False) → 1
caught_speeding(65, True) → 0

In [30]:
```
speed = int(input ("Enter speed: "))

def fine_driver (speed):
    if speed <= 60:
        return 0
    elif speed >= 61 and speed <= 80 :
        return 1
    elif speed >= 81:
        return 2


print ("ticket = ", fine_driver (speed))
```

```
ticket =  1
```

Bài 30

Given 3 int values, a b c, return their sum. However, if one of the values is 13 then it does not count towards the sum and values to its right do not count. So for example, if b is 13, then both b and c do not count.

lucky_sum(1, 2, 3) → 6
lucky_sum(1, 2, 13) → 3
lucky_sum(1, 13, 3) → 1

In [36]:
```
def lucky_sum(a,b,c):
    if a == 13:
        return 0
    elif b == 13:
        return  a
    elif c == 13:
        return  a + b
    else:
        return a+b+c

a = int(input ("Enter a: "))
b = int(input ("Enter b: "))
c = int(input ("Enter c: "))
print("sum =" , lucky_sum(a,b,c))
```

```
sum = 10
```

Bài 31.

Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in min(v1, v2) and max(v1, v2) functions return the smaller or larger of two values.
big_diff([10, 3, 5, 6]) → 7
big_diff([7, 2, 10, 9]) → 8
big_diff([2, 10, 7, 2]) → 8

In [39]:
```
def big_diff (arr):
    return max(arr)- min(arr)


arr = list(map(int, input("Enter arr: " ).split()))

print ("Difference: " , big_diff (arr))
```

```
Difference:  0
```

Bài 32.

Tìm số chẵn cuối cùng trong mảng 1 chiều các số nguyên. Nếu mảng không có giá trị chẵn thì trả về -1

In [43]:
```
def last_even(arr):
    for i in range(len(arr) -1 , 0, -1):
        if arr[i] % 2 == 0:
            return arr[i]
    return -1
```

```
arr = list(map(int, input("Enter arr: " ).split()))

print ("last even number: " , last_even(arr))
```

```
last even number:  4
```

Bài 33.

Given a dictionary containing counts of both upvotes and downvotes, return what vote count should be displayed. This is calculated by subtracting the number of downvotes from upvotes.

Examples

get_vote_count({ "upvotes": 13, "downvotes": 0 }) ➞ 13

get_vote_count({ "upvotes": 2, "downvotes": 33 }) ➞ -31

get_vote_count({ "upvotes": 132, "downvotes": 132 }) ➞ 0

In [45]:
```
def get_vote_count(votes):
    return votes['upvotes'] - votes['downvotes']

print(get_vote_count({ "upvotes": 13, "downvotes": 0 }))  # Output: 13
print(get_vote_count({ "upvotes": 2, "downvotes": 33 }))  # Output: -31
print(get_vote_count({ "upvotes": 132, "downvotes": 132 }))  # Output:
```

```
13
-31
0
```

Bài 34.

Sắp xếp mảng 1 chiều tăng dần

Input: 3 5 2 8 10 7 12

Output: 2 3 5 7 8 10 12

Lưu ý: Không sử dụng hàm sort()

In [47]:
```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]  # swap

arr = list(map(int, input("Enter arr:").split()))

bubble_sort(arr)

print("Mảng sau khi sắp xếp:", arr)
```

```
Mảng sau khi sắp xếp: [2, 3, 5, 6, 6, 75]
```

Bài 35.

Đây là một chương trình tính tổng và trừ 2 số, viết lại chương trình này bằng cách sử dụng hàm.

```
if command == "add":
        print("lets add some numbers")
        input1 = input("Number 1>")
        input2 = input("Number 2>")
        number1 = int(input1)
        number2 = int(input2)
        result = number1 + number2
        output = str(result)
        print(input1 + " + " + input2 + " = " + output)

elif command == "subtract":
        print("lets subtract some numbers")
        input1 = input("Number 1>")
        input2 = input("Number 2>")
        number1 = int(input1)
        number2 = int(input2)
        result = number1 - number2
        output = str(result)
        print(input1 + " - " + input2 + " = " + output)
```

In [54]:
```
def sum_num(input1, input2):
    return input1 + input2


def subtract_num(input1, input2):
    return input1 - input2


def calculate(command, input1, input2):
    if command == "add":
        print(f"Add number: {input1} + {input2}  = {sum_num(input1, inp
        return sum_num(input1, input2)
    if command == "subtract":
        print(f"Subtract {input1} - {input2}  = {subtract_num(input1, i
        return subtract_num(input1, input2)


command = str(input("Enter add or subtract: "))
input1 = int(input("Number 1>"))
input2 = int(input("Number 2>"))
print( calculate(command, input1, input2))
```

```
Add number: 4 + 8  = 12
12
```

Bài 36.

Viết chương trình nhập vào 2 số nguyên dương a và b. Tìm ước số chung lớn nhất của a và b.

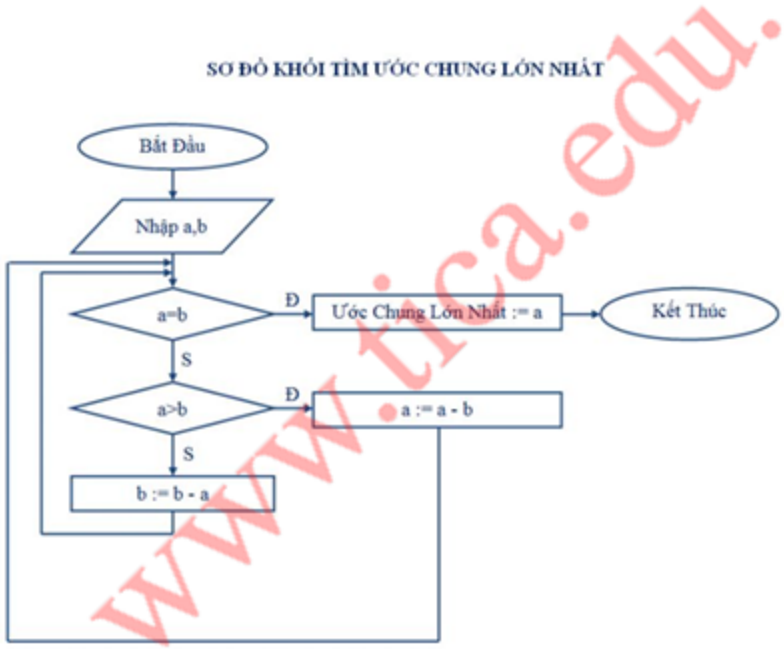
Ví dụ:

- Input:
  - a = 30
  - b = 40
- Output:
  - UCLN = 10
  - BCNN = 120

Ước chung lớn nhất của hai số nguyên a và b là số nguyên dương lớn nhất mà a và b chia hết.

Bội số chung nhỏ nhất của hai số nguyên a và b là số nguyên dương nhỏ nhất chia hết cho cả a và b.

Nếu có số tự nhiên a chia hết cho số tự nhiên b thì ta gọi a là bội của b và b là ước của a.



Ví dụ:

Tìm ước chung lớn nhất của 27 và 45?

UCLN(27,45)=9

In [61]:
```
def ucln(a, b):
    if a==b:
        return a
    while b != 0:
        a, b = b, a % b #swap
    return a

def bcnn(a, b):
    return a * b // ucln(a, b)  # BCNN = a * b / UCLN

a = int(input("Enter a: "))
b = int(input("Enter b: "))
```

```
ucln_val = ucln(a, b)
bcnn_val = bcnn(a, b)

# In kết quả
print(f"UCLN = {ucln_val}")
print(f"BCNN = {bcnn_val}")
```

```
UCLN = 3
BCNN = 3
```

Bai 37

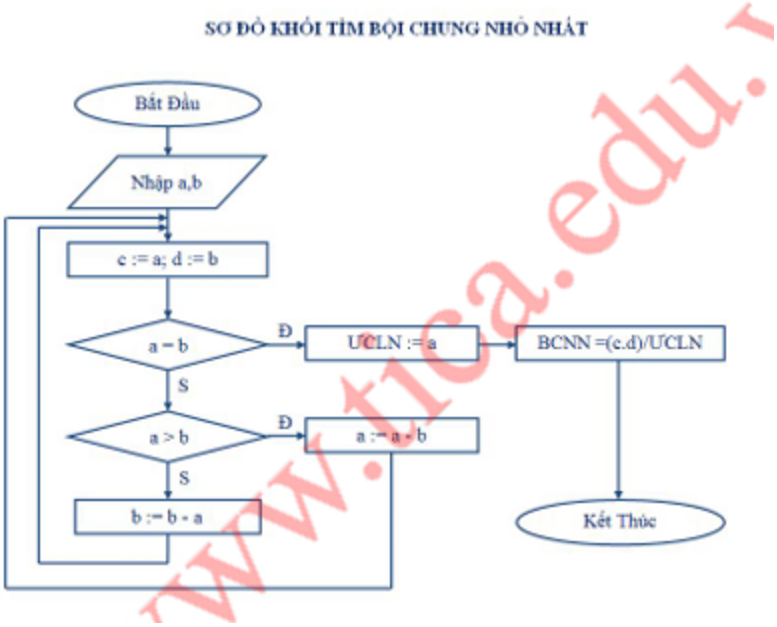
Viết chương trình nhập vào 2 số nguyên dương a và b. Tìm bội số chung nhỏ nhất của a và b.

Ví dụ:

BCNN(6,10) = 30

Bội số của 6: 6 12 18 24 30 36 .....

Bội số của 10: 10 20 30 40 ....



In [62]:
```
def ucln(a, b):
    if a==b:
        return a
    while b != 0:
        a, b = b, a % b #swap
    return a

def bcnn(a, b):
    return a * b // ucln(a, b)  # BCNN = a * b / UCLN

a = int(input("Enter a: "))
b = int(input("Enter b: "))

ucln_val = ucln(a, b)
bcnn_val = bcnn(a, b)
```

```
print(f"BCNN = {bcnn_val}")
```

```
BCNN = 36
```

Bài 38.

Viết một hàm với đầu vào là tọa độ hai điểm trên mặt phẳng hai chiều và trả về độ dài đoạn thẳng nối hai điểm đó.
Ví dụ:
line_length([15, 7], [22, 11]) ⟶ 8.06
line_length([0, 0], [0, 0]) ⟶ 0
line_length([0, 0], [1, 1]) ⟶ 1.41

In [ ]:
```
import math

def line_length(point1, point2):

  x1, y1 = point1
  x2, y2 = point2
  distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
  return distance

print(line_length([15, 7], [22, 11]))  # Output: 8.06
```

Bài 39.

Viết một hàm có tên capital_indexes. Hàm nhận một tham số duy nhất là một chuỗi. Hàm của bạn sẽ trả về một danh sách tất cả các chỉ số (index) trong chuỗi có chữ in hoa.
Ví dụ: gọi capital_indexes ("HeLlO") sẽ trả về danh sách [0, 2, 4].

In [64]:
```
def capital_indexes(string):
  result = []
  for i in range(len(string)):
    if string[i].isupper():
      result.append(i)
  return result
print(capital_indexes("HeLlO"))
```

```
[0, 2, 4, 8]
```

Bài 40.

Kiểm tra đối xứng
Một chuỗi gọi là đối xứng khi đọc từ trái qua phải hay phải qua trái thì kết quả giống nhau.
Ví dụ: Chuỗi "bob" và chuỗi " abba" là đối xứng
Chuỗi " abcd" không phải đối xứng vì "abcd" != "dcba".

Viết một hàm có tên palindrome kiểm tra tính đối xứng. Hàm trả True nếu đối xứng, False nếu không đối xứng.

In [ ]:
```
def palindrome(string):
  reversed_string = string[::-1]
  return string == reversed_string

print(palindrome("bob"))  # Output: True
print(palindrome("abba"))  # Output: True
print(palindrome("abcd"))  # Output: False
```

Bài 41.

Hãy viết chương trình nhập vào số nguyên dương n. Kiểm tra xem n có phải là số chính phương hay không? (số chính phương là số khi lấy căn bậc 2 có kết quả là nguyên). Hãy viết chương trình kiểm tra số chính phương.

In [ ]:
```
import math

def is_perfect_square(number):
  root = int(math.sqrt(number))
  return root * root == number

number = int(input("Nhập số nguyên dương: "))
if is_perfect_square(number):
  print(number, "là số chính phương")
else:
  print(number, "không phải là số chính phương")
```

Bài 42.

Viết chương trình nhập vào số n.. xuất số đảo ngược của n đó..
Vd: n = 123 => 321
n = 4320 → 0234

In [9]:
```
num = input("Enter num: ")  # Không cần phải chuyển sang int, chỉ cần x

# slicing
print(num[::-1])
```

```
54321
```

Bai 43:

Cho số nguyên dương X, khi đảo ngược trật tự các chữ số của X ta sẽ thu được một số nguyên dương Y, Y được gọi là số đảo ngược của X.
Ví dụ: X = 613 thì Y = 316 là số đảo ngược của X.
Số nguyên dương Y được gọi là số nguyên tố nếu nó chỉ có hai ước số là 1 và

chính nó, số 1 không phải là số nguyên tố.
Cho hai số nguyên dương P và Q (1 ≤ P ≤ Q ≤ 2109; Q - P ≤ 105).

Yêu cầu: Hãy tìm tất cả các số nguyên dương X nằm thỏa mãn P≤X≤Q và số đảo ngược của số X là số nguyên tố.
Dữ liệu vào: Cho trong file văn bản TimSo.txt có cấu trúc như sau:

- Dòng 1: Ghi hai số nguyên dương PQ,hai số được ghi cách nhau ít nhất một dấu cách.
  Dữ liệu ra: Ghi ra file văn bản KetQua.txt trên nhiều dòng,mỗi dòng ghi một số nguyên X tìm dược

Ví dụ:
TimSo.txt
10 19

KetQua.txt
11
13
14
16
17

In [18]:
```
import math


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


def reverse_number(n):
    return int(str(n)[::-1])

# read file TimSo.txt
with open('E:\\4.Python\\4.Github repo\\Timso.txt', 'r') as file:
    # Lấy hai số P và Q
    line = file.readline().strip()
    P, Q = map(int, line.split())


result = []


for X in range(P, Q + 1):
    Y = reverse_number(X)
    if is_prime(Y):
        result.append(X)
```

```
# Ghi kết quả vào file KetQua.txt
with open('E:\\4.Python\\4.Github repo\\KetQua.txt', 'w') as file:
    for num in result:
        file.write(f"{num}\n")
```

Bài 44.

Viết một chương trình chấp nhận chuỗi từ do người dùng nhập vào, phân tách nhau
bởi dấu phẩy và in những từ đó thành chuỗi theo thứ tự bảng chữ cái, phân tách nhau bằng dấu phẩy.
Giả sử đầu vào được nhập là: without,hello,bag,world, thì đầu ra sẽ là: bag,hello,without,world

In [39]:
```
str_val = list(map(str, input("Enter str: ").split(',')))
str_val.sort()
result = ','.join(str_val)
print(result)
```

```
bag,hello,without,world
```

Bài 45.

Write a function named add_dots that takes a string and adds "." in between each letter.

For example, calling add_dots("test") should return the string "t.e.s.t".

Then, below the add_dots function, write another function named remove_dots that removes all dots from a string. For example, calling remove_dots("t.e.s.t") should return "test".

If both functions are correct, calling remove_dots(add_dots(string)) should return back the original string for any string.

In [32]:
```
# add '.'
def add_dots(string):
    return '.'.join(string)

# remove all '.'
def remove_dots(string):
    return string.replace('.', '')

test_str = "test"

print("Adding dots:", add_dots(test_str))

print("Removing dots:", remove_dots(dots_added))
```

```
print("Original string after both operations:", remove_dots(add_dots(te
```

```
Adding dots: t.e.s.t
Removing dots: test
Original string after both operations: test
```

Bài 46.

Viết một chương trình chấp nhận đầu vào là một câu, đếm số chữ cái và chữ số trong câu đó. Giả sử đầu vào sau được cấp cho chương trình:

Input: hello world! 123

Thì đầu ra sẽ là:

Số chữ cái là: 10

Số chữ số là: 3

In [4]:
```
p = str(input("Enter str: "))

digit = 0
character = 0
for i in p:
    if (i.isalpha() == True):
        character+=1
    if (i.isdigit() == True):
        digit+=1

print("Count digits:",digit )
print("Count character:",character )
```

```
hjsjahs 276 gjdj
Count digits: 3
Count character: 11
```

Bài 47.

Thiết kế trò chơi đoán từ vựng như chiếc nón kỳ diệu. Máy tính sẽ hiện các ô chữ tương ứng với số chữ cái của từ bí mật.

Ví dụ:

Tên một loài động vật bơi ở biển có 4 chữ cái ?

Từ bí mật là Fish gồm 4 chữ cái

Welcome to Hangman!

_ _ _ _

Guess your letter: I

Người dùng nhập từ ký tự muốn đoán. Sau đó chương trình sẽ hiện tất cả các ký tự đó nếu có trong từ bí mật.

_ I _ _

Trò chơi chỉ cho phép bạn đoán sai tối đa 5 lần, nếu quá 5 lần thì chương trình in ra từ bí mật.

In [ ]:
```
import random

words = ["fish", "shark", "whale", "dolphin", "seal"]

secret_word = random.choice(words).lower()
print(secret_word)

guessed_word = ["_" for _ in secret_word]

max_err = 5
attempts = 0
wrong_guesses = []

print("Question: Tên một loài động vật sống dưới biển.")
print(" ".join(guessed_word))

while attempts < max_err and "_" in guessed_word:
    guess = input("Nhập dự đoán: ").lower()

    if len(guess) != 1 and guess.isalpha() == False:
        print("Chuỗi có ký tự không hợp lệ !")
        continue

    if guess in words and guess in wrong_guesses:
        print(f"Bạn đã đoán ký tự '{guess}' rồi!")
        max_err = max_err - 1
        continue

    if guess in secret_word:
        print("Ban da doan dung: ", guess)
        break
    else:
        attempts += 1
        wrong_guesses.append(guess)
        print(f"Không có '{guess}' . Bạn còn {max_err - attempts} lần đ
        print(" ".join(guessed_word))
```

Bài 48.

Đọc nội dung của một file INPUT.txt có cấu trúc như sau:

- Dòng đầu tiên ghi số lượng số nguyên có trong file
- Dòng tiếp theo là một dãy số nguyên

Ví dụ: Nội dung file INPUT.txt

8

1 2 3 4 5 6 7 8

Ghi vào file RESULT.txt các số nguyên tố có trong mảng

Ví dụ: Kết quả file KETQUA.txt

1 3 5 7

In [42]:
```
n = 0
arr = []


def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True


def filter_prime_numbers(numbers):
    return list(filter(is_prime, numbers))


with open('E:\\4.Python\\4.Github repo\\INPUT.txt', 'r') as file:
    n = file.readline()
    arr = list(map(int, file.readline().strip().split()))
    prime_numbers = filter_prime_numbers(arr)
    print ("\n n: ", n, "\n arr: ", arr,  "\n prime_numbers: ", prime_n


with open('E:\\4.Python\\4.Github repo\\RESULT.txt', 'w') as file:
    file.write(" ".join(map(str, prime_numbers)))
```

```
n:  8

arr:  [1, 2, 3, 4, 5, 6, 7, 8]
prime_numbers:  [2, 3, 5, 7]
```

Bài 49.

Cho một dữ liệu file input_49.txt

Hello! Welcome to Ha noi

Ha noi is the capital city of Vietnam

Good Luck!

Đọc nội dung của file tìm ra từ đầu tiên có độ dài lớn nhất trong file trên. Các từ trong file sẽ cách nhau một khoảng trắng.

Kết quả: Từ Welcome có độ dài 7 ký tự.

In [49]:
```
arr = []
with open ('E:\\4.Python\\4.Github repo\\input_49.txt','r') as file:
    arr = list(map(str,file.read().split()))
    print(arr)
```

```
print("Word has max length: ", max(arr, key=len))
```

```
['Hello!', 'Welcome', 'to', 'Ha', 'noi', 'Ha', 'noi', 'is', 'the', 'capi
tal', 'city', 'of', 'Vietnam', 'Good', 'Luck!']
Word has max length:  Welcome
```