```
# 1.
#Write a function that check if a string is in another string
print("Q # 1")

def str_in_str(str1,str2):
    if ((str1 in str2) or (str2 in str1)):
        return True

    return False


# 2.Bisection Algorithm
# From the course book with minor adjustments

# a.
print("Q # 2A")

def findRoot_a(x, n, epsilon):
    """Assumes x and epsilon int or float, n an int, x>1,
            epsilon > 0 & n >= 1
       Returns float y such that y**n is within epsilon of x.
            If such a float does not exist, it returns None"""

    low = 0
    high = x
    ans = (high + low)/2.0
    while abs(ans**n - x) >= epsilon:
        if ans**n < x:
            low = ans
        else:
            high = ans
        ans = (high + low)/2.0
    return ans

# b.
print("Q # 2B")

def findRoot_b(x, n, epsilon):
    """Assumes x and epsilon int or float, n an int, x>0,
            epsilon > 0 & n >= 1
       Returns float y such that y**n is within epsilon of x.
            If such a float does not exist, it returns None"""

    low = 0
    high = max(1.0, x) # if you're not familiar with "max" you can do it with "if"
    ans = (high + low)/2.0
    while abs(ans**n - x) >= epsilon:
        if ans**n < x:
            low = ans
        else:
            high = ans
        ans = (high + low)/2.0
    return ans

# c.
print("Q # 2C")

def findRoot_c(x, n, epsilon):
    """Assumes x and epsilon int or float, n an int,
            epsilon > 0 & n >= 1
       Returns float y such that y**power is within epsilon of x.
            If such a float does not exist, it returns None"""
    if x < 0 and n%2 == 0: #Negative number has no even-powered
                           #roots
        return None
    low = min(-1.0, x) # if you're not familiar with "min" you can do it with "if"
    high = max(1.0, x) # if you're not familiar with "max" you can do it with "if"
    ans = (high + low)/2.0
    while abs(ans**n - x) >= epsilon:
        if ans**n < x:
            low = ans
        else:
            high = ans
        ans = (high + low)/2.0
    return ans


 # 3.
 #Check function from Q2 with other param
print("Q # 3")
def test_findRoot (x):
    for n in range (1,6):
        print ("The",n,"root of",x,",with epsilon=0.001,is",findRoot_c(x,n,0.001))
        if x!=0: # if x=0 then x=-x and 1/x is not defined
            print ("The",n,"root of",-x,",with epsilon=0.001, is",findRoot_c(-x,n,0.001))
            print ("The",n,"root of",1/x,",with epsilon=0.001, is",findRoot_c(1/x,n,0.001))

#5
#Get 2 numbers and return dev
print("Q # 4")
def division (n1, n2):
  #  if n2 == 0:
   #     return ("The second number must be different than 0")
  #  else:
        return print(f"The div is: {n1/n2}")
def call_num():
    n = 0
    while n == 0:
        n1 = float(input("Enter first number: "))
        n2 = float(input("Enter second number: "))
        if n2 == 0:
            print ("The second number must be different than 0")
        else:
            n = 1
            division (n1,n2)
call_num()




#6
#write fun mul_2nums that get 2 numbers and return * and if neg return 0

print("Q # 5")
def Mult (n1, n2):
    if (n2*n1) < 0:
        return print(0)
    else:
        return print(f"The mul is: {n1*n2}")
def call_num():
    n1 = float(input("Enter first number: "))
    n2 = float(input("Enter second number: "))
    Mult(n1, n2)

call_num()

#7
#Make list of 5 items and take out all items jump of 2

print("Q # 7")
Listcheck = 0
while Listcheck==0:
    lst1 = [int(item) for item in input("Enter list with 5 items seperated with apace: ").split()]

    if len(lst1)==5:
        Listcheck=1

print("the original list:" ,lst1)
lst1=lst1[::2]
print("after we removed the items:" ,lst1)

#8
#Make a list summer(), that get a list and sum
print("Q # 8")
def summer(lst1):
    if len(lst1)==0: #Check extreme case
        return 0
    lst2 = lst1[0] #enter first element, for the type
    for element in lst1[1:]:# addimg all other
        lst2 += element
    return lst2

print(summer([10, 11, 12, 0.75]))
print(summer([True, False, True, True]))
print(summer(['aa', 'bb', 'cc']))
print(summer([[1, 2, 3, 'a'], [4, 'b', 'c', 'd']]))
```