1. Given a list of numbers, create a function that removes 25% from every number in the list except the smallest number, and adds the total amount removed to the smallest number.

**Examples**

show\_the\_love([4, 1, 4]) ➞ [3, 3, 3]

show\_the\_love([16, 10, 8]) ➞ [12, 7.5, 14.5]

show\_the\_love([2, 100]) ➞ [27, 75]

**Ans:**

def show\_the\_love(lst):

total=0

m=min(lst)

for i in range(len(lst)):

if lst[i]==m:

continue

val=(int(lst[i])\*25)/100

total+=val

lst[i]-=val

index=lst.index(m)

lst[index]+=total

print(lst)

2. Create a function that takes in two words as input and returns a list of three elements, in the following order:

1.Shared letters between two words.

2.Letters unique to word 1.

3.Letters unique to word 2.

Each element should have unique letters, and have each letter be alphabetically sorted.

**Examples**

letters("sharp", "soap") ➞ ["aps", "hr", "o"]

letters("board", "bored") ➞ ["bdor", "a", "e"]

letters("happiness", "envelope") ➞ ["enp", "ahis", "lov"]

letters("kerfuffle", "fluffy") ➞ ["flu", "ekr", "y"]

# Even with multiple matching letters (e.g. 3 f's), there should

# only exist a single "f" in your first element.

letters("match", "ham") ➞ ["ahm", "ct", ""]

# "ham" does not contain any letters that are not found already

# in "match".

**Ans:**

def letters(str1, str2):

l=[]

shared=[]

diff1=[]

diff2=[]

set1=set(str1)

set2=set(str2)

shared=list(set1.intersection(set2))

diff1=list(set1.difference(set2))

diff2=list(set2.difference(set1))

shared.sort()

diff1.sort()

diff2.sort()

l.append("".join(shared))

l.append("".join(diff1))

l.append("".join(diff2))

print(l)

3. Write a function that pairs the first number in an array with the last, the second number with the second to last, etc.

**Examples**

pairs([1, 2, 3, 4, 5, 6, 7]) ➞ [[1, 7], [2, 6], [3, 5], [4, 4]]

pairs([1, 2, 3, 4, 5, 6]) ➞ [[1, 6], [2, 5], [3, 4]]

pairs([5, 9, 8, 1, 2]) ➞ [[5, 2], [9, 1], [8, 8]]

pairs([]) ➞ []

**Ans:**

def pairs(lst):

first=0

last=len(lst)-1

if len(lst)%2==0:

length=len(lst)//2

else:

length=(len(lst)//2)+1

l=[]

for i in range(length):

sub\_l=[]

sub\_l.append(lst[first])

sub\_l.append(lst[last])

l.append(sub\_l)

first+=1

last-=1

print(l)

4. Write a function that adds two numbers. The catch, however, is that the numbers will be strings.

Examples

add\_str\_nums("4", "5") ➞ "9"

add\_str\_nums("abcdefg", "3") ➞ "-1"

add\_str\_nums("1", "") ➞ "1"

add\_str\_nums("1874682736267235927359283579235789257", "32652983572985729") ➞ "1874682736267235927391936562808774986"

**Ans:**

def add\_str\_nums(str1, str2):

if str1=="":

str1="0"

if str2=="":

str2="0"

if str1.isdigit() and str2.isdigit():

print(str(int(str1)+int(str2)))

else:

print("-1")

5. lPaeesh le pemu mnxit ehess rtnisg! Oh, sorry, that was supposed to say: Please help me unmix these strings!

Somehow my strings have all become mixed up; every pair of characters has been swapped. Help me undo this so I can understand my strings again.

Examples

unmix("123456") ➞ "214365"

unmix("hTsii s aimex dpus rtni.g") ➞ "This is a mixed up string."

unmix("badce") ➞ "abcde"

**Ans:**

def unmix(string):

new\_str=""

for i in range(0, len(string), 2):

sub\_str=string[i:i+2]

new\_str+=sub\_str[::-1]

print(new\_str)