# Assignment 14

#### 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.

**Answer:**

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]

**def** show\_the\_love(in\_list):  
 out\_list = in\_list.copy()  
 sum\_num = 0  
 **for** ele **in** range(len(out\_list)):  
 **if** out\_list[ele] **is** **not** min(out\_list):  
 sum\_num += out\_list[ele]/4  
 out\_list[ele] = out\_list[ele]-(out\_list[ele]/4)  
 out\_list[out\_list.index(min(out\_list))] = sum\_num +min(out\_list)  
 print(f'show\_the\_love({in\_list}) ➞ {out\_list}')  
   
show\_the\_love([4, 1, 4])  
show\_the\_love([16, 10, 8])  
show\_the\_love([2, 100])

show\_the\_love([4, 1, 4]) ➞ [3.0, 3.0, 3.0]  
show\_the\_love([16, 10, 8]) ➞ [12.0, 7.5, 14.5]  
show\_the\_love([2, 100]) ➞ [27.0, 75.0]

#### 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.

**Answer:**

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".

**def** letters(s\_one,s\_two):  
 s\_one\_set = set(s\_one)  
 s\_two\_set = set(s\_two)  
 out\_list = []  
 out\_list.append(''.join(sorted(s\_one\_set.intersection(s\_two\_set))))  
 out\_list.append(''.join(sorted(s\_one\_set.difference(s\_two\_set))))  
 out\_list.append(''.join(sorted(s\_two\_set.difference(s\_one\_set))))   
 print(f'letters{s\_one,s\_two} ➞ {out\_list}')  
   
letters("sharp", "soap")  
letters("board", "bored")  
letters("happiness", "envelope")  
letters("kerfuffle", "fluffy")  
letters("match", "ham")

letters('sharp', 'soap') ➞ ['aps', 'hr', 'o']  
letters('board', 'bored') ➞ ['bdor', 'a', 'e']  
letters('happiness', 'envelope') ➞ ['enp', 'ahis', 'lov']  
letters('kerfuffle', 'fluffy') ➞ ['flu', 'ekr', 'y']  
letters('match', 'ham') ➞ ['ahm', 'ct', '']

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

**Answer:**

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([]) ➞ []

**def** pairs(in\_list):  
 in\_list\_clone = in\_list.copy()  
 output = []  
 **while** True:  
 **if** len(in\_list) > 0:  
 **if** len(in\_list) == 1:  
 output.append([in\_list[0],in\_list.pop(0)])  
 **else**:  
 output.append([in\_list.pop(0),in\_list.pop(-1)])  
 **else**:  
 **break**  
 print(f'pairs({in\_list\_clone}) ➞ {output}')  
   
pairs([1, 2, 3, 4, 5, 6, 7])  
pairs([1, 2, 3, 4, 5, 6])  
pairs([5, 9, 8, 1, 2])  
pairs([])

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([]) ➞ []

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

**Answer:**

Examples:  
add\_str\_nums("4", "5") ➞ "9"  
add\_str\_nums("abcdefg", "3") ➞ "-1"  
add\_str\_nums("1", "") ➞ "1"  
add\_str\_nums("1874682736267235927359283579235789257", "32652983572985729") ➞ "1874682736267235927391936562808774986"

**def** add\_str\_nums(in\_one,in\_two):  
 in\_one = in\_one **if** len(in\_one) > 0 **else** "0"  
 in\_two = in\_two **if** len(in\_two) > 0 **else** "0"  
 **if** in\_one.isdigit() == False **or** in\_two.isdigit() == False:  
 output = -1  
 **else**:   
 output = int(in\_one)+int(in\_two)  
 print(f'add\_str\_nums{in\_one,in\_two} ➞ {str(output)}')  
   
add\_str\_nums("4", "5")  
add\_str\_nums("abcdefg", "3")  
add\_str\_nums("1", "")  
add\_str\_nums("1874682736267235927359283579235789257", "32652983572985729")

add\_str\_nums('4', '5') ➞ 9  
add\_str\_nums('abcdefg', '3') ➞ -1  
add\_str\_nums('1', '0') ➞ 1  
add\_str\_nums('1874682736267235927359283579235789257', '32652983572985729') ➞ 1874682736267235927391936562808774986

#### 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.

**Answer:**

Examples:  
unmix("123456") ➞ "214365"  
unmix("hTsii s aimex dpus rtni.g") ➞ "This is a mixed up string."  
unmix("badce") ➞ "abcde"

**def** unmix(in\_string):  
 output = ''  
 **for** ele **in** range(0,len(in\_string)-1,2):  
 output += in\_string[ele+1]+in\_string[ele]  
 **if** (len(in\_string)%2 != 0 **and** ele == len(in\_string)//2 ):  
 output += in\_string[-1]   
 print(f'unmix({in\_string}) ➞ {output}')  
  
unmix("123456")  
unmix("hTsii s aimex dpus rtni.g")  
unmix("badce")

unmix(123456) ➞ 214365  
unmix(hTsii s aimex dpus rtni.g) ➞ This is a mixed up string.  
unmix(badce) ➞ abcde