CS101. A21 - Flatten a list

Flatten a list. Eliminate empty lists or None values.

Eg. Input: [[ ], [1], [2, None, 3], [4, 5, 6]] . Return value: [1, 2, 3, 4, 5, 6]

Tasks to implement:

Implement the function flatten. Input is a nested list (max 2 levels). Return value: a list with all non-None elements in a non-nested list.

Code Template (flatten.py)

```
'''
Flatten a list
'''

def flatten(list1):
     pass
```

Test Cases

Function	Inputs	Output	
flatten	[[ ]]	[ ]	
flatten	[[ ], [None]]	[ ]	
flatten	[[ ], [None], [None, [ ]]]	[ ]	
flatten	[[ ], [None], [None, [ ]], [1, 2]]	[1, 2]	
flatten	[[ ], [None], [None, [ ]], [1], [2]]	[1, 2]	
flatten	[[ ], [None], [None, [ ]], [1], [[2]]]	[1, 2]	
flatten	[[ ], [None], [None, [ ]], [1], [[2, None]]]	[1, 2]	
flatten	[[ ], [None], [None, [ ]], [1, [None]], [[2, None]]]	[1, 2]	
flatten	[[3], [None], [None, [ ]], [1, [None]], [[2, None]]]	[3, 1, 2]	

****

Write a function to compare two lists. a_list and b_list are input to the function. Function returns True if an element in b_list is a square of an element of a_list. Every element in a_list should have at least one square in b_list. Eg. a_list = [10, 20, 30], b_list = [400, 100, 900]. compare_lists(a_list, b_list) returns True. Otherwise, the function returns False.

Tasks to implement:

Implement the function compare_lists. Input: 2 lists - a_list has Integers and b_list has only positive integers. Return value: True if every element of b_list is a square of at least one element from a_list; False otherwise.

Code Template (compare_lists.py)

```
'''
Compare lists
'''

def compare_lists(a_list, b_list):
     pass
```

Test Cases

Function	**Inputs**	**Output**	
`compare_lists`	[10], [100]	True	
`compare_lists`	[10, -10], [100]	True	
`compare_lists`	[10, -10, 1, -1], [100, 1]	True	
`compare_lists`	[10, -10, 1, -1], [100]	False	
`compare_lists`	[10, -10, 1], [100]	False	
`compare_lists`	[10, -10, 1, -1, 5], [1, 25, 100]	True	
`compare_lists`	[5], [25, 25, 25]	True	
`compare_lists`	[5], [-25]	ValueError	

****

CS101. A24 - Pangram

Ref: https://en.wikipedia.org/wiki/Pangram

Determine if a sentence is a pangram. A pangram (Greek: παν γράμμα, pan gramma, "every letter") is a sentence using every letter of the alphabet at least once. The best known English pangram is:

> The quick brown fox jumps over the lazy dog.

The alphabet used consists of ASCII letters a to z, inclusive, and is case insensitive. Input will not contain non-ASCII symbols.

Tasks to implement:

Implement the function pangram(sentence). Input is a sentence (a String). Return value: True if the sentence is a pangram; false if not.

Code Template (pangram.py)

```
'''
Is the sentence a pangram
'''

def is_pangram(sentence):
     pass
```

Test Cases

Function	Inputs	Output	
`pangram`	“”	False	
`pangram`	“abcdefghijklmnopqrstuvwxyz”	True	
`pangram`	“the quick brown fox jumps over the lazy dog"	True	
`pangram`	“a quick movement of the enemy will jeopardize five gunboats”	False	
`pangram`	"the_quick_brown_fox_jumps_over_the_lazy_dog"	True	
`pangram`	"the 1 quick brown fox jumps over the 2 lazy dogs"	True	
`pangram`	“7h3 qu1ck brown fox jumps ov3r 7h3 lazy dog”	False	
`pangram`	"Five quacking Zephyrs jolt my wax bed."	True	
`pangram`	"the quick brown fox jumps over with lazy FX"	False	
`pangram`	"bcdefghijklmnopqrstuvwxyz"	False	
`pangram`	"abcdefghijklmnopqrstuvwxy"	False	

****