Question1

Create a function that takes three integer arguments (a, b, c) and returns the amount of integers which are of equal value.

**Examples**

equal(3, 4, 3) ➞ 2

equal(1, 1, 1) ➞ 3

equal(3, 4, 1) ➞ 0

**Notes**

Your function must return 0, 2 or 3.

# Create a function that takes three integer arguments (a, b, c) and returns the amount of integers

# which are of equal value.

# Examples equal(3, 4, 3) ➞ 2

# equal(1, 1, 1) ➞ 3

# equal(3, 4, 1) ➞ 0

# Notes Your function must return 0, 2 or 3.

# Define a function, function1

def function1(a,b,c):

if a==b and a==c:

return 3

elif a == b and b!=c:

return 2

elif a!=b and b==c:

return 2

elif a!=b and a==c:

return 2

elif a!=b and a!=c and b!=c:

return 0

# call the function with different values and check it

print(function1(3, 4, 3)) # it should print 2

print(function1(1,1,1)) # it should print 3

print(function1(3, 4, 1)) # it should print 0

Question2

Write a function that converts a **dictionary** into a **list** of keys-values **tuples**.

### Examples

dict\_to\_list({

"D": 1,

"B": 2,

"C": 3

}) ➞ [("B", 2), ("C", 3), ("D", 1)]

dict\_to\_list({

"likes": 2,

"dislikes": 3,

"followers": 10

}) ➞ [("dislikes", 3), ("followers", 10), ("likes", 2)]

### Notes

Return the elements in the list in alphabetical order.

# Write a function that converts a dictionary into a list of keys-values tuples.

# Create a sample dictionary object and initialize it

dict1 = {

"D": 1,

"B": 2,

"C": 3

}

# Create a function, function1 which converts a dictionary into a list of keys-values tuples.

def function1(dictobj1):

list1 = [(i,j) for i,j in dictobj1.items()]

return list1

# call the function, function1 by passing dict object as parameter

print(function1(dict1))

Question3

Write a function that creates a dictionary with each **(key, value)** pair being the **(lower case, upper case)** versions of a letter, respectively.

### Examples

mapping(["p", "s"]) ➞ { "p": "P", "s": "S" }

mapping(["a", "b", "c"]) ➞ { "a": "A", "b": "B", "c": "C" }

mapping(["a", "v", "y", "z"]) ➞ { "a": "A", "v": "V", "y": "Y", "z": "Z" }

### Notes

All of the letters in the input list will always be lowercase.

# Write a function that creates a dictionary with each (key, value) pair being the (lower case, upper case)

# Examples mapping(["p", "s"]) ➞ { "p": "P", "s": "S" }

# mapping(["a", "b", "c"]) ➞ { "a": "A", "b": "B", "c": "C" }

# mapping(["a", "v", "y", "z"]) ➞ { "a": "A", "v": "V", "y": "Y", "z": "Z" }

# Notes All of the letters in the input list will always be lowercase.

# Define a function, function1 which takes list as input parameter

# and returns dictionary with a pair being the (lower case, upper case)

def function1(list1):

dict1 = {}

for i in list1:

dict1[i] = i.upper()

return dict1

# Call the function, function1 with different input parameters

print(function1(["p", "s"])) # the output should be { "p": "P", "s": "S" }

print(function1(["a", "b", "c"])) # the output should be { "a": "A", "b": "B", "c": "C" }

print(function1(["a", "v", "y", "z"])) # the output should be { "a": "A", "v": "V", "y": "Y", "z": "Z" }

Question4

Write a function, that replaces all vowels in a string with a specified vowel.

### Examples

vow\_replace("apples and bananas", "u") ➞ "upplus und bununus"

vow\_replace("cheese casserole", "o") ➞ "chooso cossorolo"

vow\_replace("stuffed jalapeno poppers", "e") ➞ "steffed jelepene peppers"

### Notes

All words will be lowercase. Y is not considered a vowel.

# Write a function, that replaces all vowels in a string with a specified vowel.

# Examples vow\_replace("apples and bananas", "u") ➞ "upplus und bununus"

# vow\_replace("cheese casserole", "o") ➞ "chooso cossorolo"

# vow\_replace("stuffed jalapeno poppers", "e") ➞ "steffed jelepene peppers"

# Notes All words will be lowercase. Y is not considered a vowel.

# Define a function, function1

def function1(string1, char1):

string1 = string1.replace('a', char1)

string1 = string1.replace('e', char1)

string1 = string1.replace('i', char1)

string1 = string1.replace('o', char1)

string1 = string1.replace('u', char1)

return string1

# Call the function, function1 with different input parameters given and check the output

print(function1("apples and bananas", "u")) # the output should be "upplus und bununus"

print(function1("cheese casserole", "o")) # the output should be "chooso cossorolo"

print(function1("stuffed jalapeno poppers", "e")) # the output should be "steffed jelepene peppers"

Question5

Create a function that takes a string as input and capitalizes a letter if its ASCII code is even and returns its lower case version if its ASCII code is odd.

### Examples

ascii\_capitalize("to be or not to be!") ➞ "To Be oR NoT To Be!"

ascii\_capitalize("THE LITTLE MERMAID") ➞ "THe LiTTLe meRmaiD"

ascii\_capitalize("Oh what a beautiful morning.") ➞ "oH wHaT a BeauTiFuL moRNiNg."

# Create a function that takes a string as input and capitalizes a letter if its ASCII code is even

# and returns its lower case version if its ASCII code is odd.

# Examples ascii\_capitalize("to be or not to be!") ➞ "To Be oR NoT To Be!"

# ascii\_capitalize("THE LITTLE MERMAID") ➞ "THe LiTTLe meRmaiD"

# ascii\_capitalize("Oh what a beautiful morning.") ➞ "oH wHaT a BeauTiFuL moRNiNg."

# create function, ascii\_capitalize()

# Use ord() function to get ASCII code of a character

def ascii\_capitalize(string1):

string2 = ""

for i in range(0,len(string1)):

if ord(string1[i])%2 == 0:

string2 = string2 + (string1[i].upper())

else:

string2 = string2 + (string1[i].lower())

return string2

# call the function, ascii\_capitalize() with given inputs and check the output

print(ascii\_capitalize("to be or not to be!")) # "To Be oR NoT To Be!"

print(ascii\_capitalize("THE LITTLE MERMAID")) # "THe LiTTLe meRmaiD"

print(ascii\_capitalize("Oh what a beautiful morning.")) # "oH wHaT a BeauTiFuL moRNiNg."