# Assignment 25

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

**Answer:**

**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.

def equal(a,b,c):  
 if a==b==c:  
 print(f'{a,b,c} ➞ {3}')  
 elif a==b or b==c:  
 print(f'{a,b,c} ➞ {2}')  
 else:  
 print(f'{a,b,c} ➞ {0}')  
  
equal(3, 4, 3)  
equal(1, 1, 1)  
equal(3, 4, 1)

(3, 4, 3) ➞ 0  
(1, 1, 1) ➞ 3  
(3, 4, 1) ➞ 0

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

**Answer:**

**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.

**def** dict\_to\_list(in\_dict):  
 out\_list = []  
 **for** keys,values **in** in\_dict.items():  
 out\_list.append((keys,values))  
 print(f'{in\_dict} ➞ {out\_list}')  
   
dict\_to\_list({"D": 1,"B": 2,"C": 3})  
dict\_to\_list({"likes": 2,"dislikes": 3,"followers": 10})

{'D': 1, 'B': 2, 'C': 3} ➞ [('D', 1), ('B', 2), ('C', 3)]  
{'likes': 2, 'dislikes': 3, 'followers': 10} ➞ [('likes', 2), ('dislikes', 3), ('followers', 10)]

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

**Answer:**

**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.

def mapping(in\_list):  
 out\_dict = {}  
 for ele in in\_list:  
 out\_dict[ele] = ele.upper()  
 print(f'{in\_list} ➞ {out\_dict}')  
   
mapping(["p", "s"])  
mapping(["a", "b", "c"])  
mapping(["a", "v", "y", "z"])

['p', 's'] ➞ {'p': 'P', 's': 'S'}  
['a', 'b', 'c'] ➞ {'a': 'A', 'b': 'B', 'c': 'C'}  
['a', 'v', 'y', 'z'] ➞ {'a': 'A', 'v': 'V', 'y': 'Y', 'z': 'Z'}

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

**Answer:**

**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.

def vow\_replace(in\_string,vow\_char):  
 vowels = ['a','e','i','o','u']  
 out\_string = ''  
 for ele in in\_string:  
 if ele in vowels:  
 out\_string += vow\_char  
 else:  
 out\_string += ele  
 print(f'{in\_string} ➞ {out\_string}')  
   
vow\_replace("apples and bananas", "u")  
vow\_replace("cheese casserole", "o")  
vow\_replace("stuffed jalapeno poppers", "e")

apples and bananas ➞ upplus und bununus  
cheese casserole ➞ chooso cossorolo  
stuffed jalapeno poppers ➞ steffed jelepene peppers

#### 5.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.

**Answer:**

**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."

def ascii\_capitalize(in\_string):  
 out\_string = ''  
 for ele in in\_string.lower():  
 if (ord(ele)%2 == 0):  
 out\_string += ele.upper()  
 else:  
 out\_string += ele  
 print(f'{in\_string} ➞ {out\_string}')  
   
ascii\_capitalize("to be or not to be!")  
ascii\_capitalize("THE LITTLE MERMAID")  
ascii\_capitalize("Oh what a beautiful morning.")

to be or not to be! ➞ To Be oR NoT To Be!  
THE LITTLE MERMAID ➞ THe LiTTLe meRmaiD  
Oh what a beautiful morning. ➞ oH wHaT a BeauTiFuL moRNiNg.