Funciones para Colecciones de Datos

Ejercicios para la solución de problemas que involucren la modificación de colecciones de datos almacenados en contenedores tipo lista o tupla en Pyhton. Se utilizarán los conceptos vistos en la unidad correspondientes a las funciones **map, reduce, filter, zip.**

Problema #1:
Utilizar la función incorporada **map()** para crear una función que retorne una lista con la longitud de cada palabra (separadas por espacios) de una frase. La función recibe una cadena de texto y retornará una lista.

Problema #2:
Crear una función que tome una lista de dígitos y devuelva al número al que corresponden. Por ejemplo [1,2,3] corresponde a el número ciento veintitrés (123). Utilizar la función **reduce**.

Problema #3:
Crear una función que retorne las palabras de una lista de palabras que comience con una letra en específico. Utilizar la función **filter**.

Problema #4:
Realizar una función que tome una lista de comprensión para devolver una lista de la misma longitud donde cada valor son las dos cadenas de L1 y L2 concatenadas con un conector entre ellas. **Ejemplo:** Listas: ['A', 'a'] ['B','b'] Conector: '-' Salida: ['A-a'] ['B-b']. Utilizar la función **zip**.

La función enumarate es otra de las herramientas para manipulación de colecciones de datosl en Python. *Consultar cuál es su finalidad y una vez teniendo claro su comportamiento, resolver los dos siguientes problemas propuestos:*

Problema #5:
Realizar una función que tome una lista y retorne un diccionario que contenga los valores de la lista como clave y el índice como el valor. Utilizar la función **enumerate**.

Problema #6:
Realizar una función que retorne el recuento de la cantidad de elementos en la lista cuyo valor es igual a su índice. Utilizar la función **enumerate**.

Soluciones:

Problema #1:
Utilizar la función incorporada **map()** para crear una función que retorne una lista con la longitud de cada palabra (separadas por espacios) de una frase. La función recibe una cadena de texto y retornará una lista.

Solución:

```
def longitud_palabras(frase): # Función
    palabra_len = list(map(len, frase.split())) # Longitud de cada palabra
    return palabra_len # Retornar resultado

longitud_palabras('Hola Luis, como estas?') # Prueba de la función
```

Problema #2:
Crear una función que tome una lista de dígitos y devuelva al número al que corresponden. Por ejemplo [1,2,3] corresponde a el número ciento veintitrés (123). Utilizar la función **reduce**.

Solución:

```
from functools import reduce

def digitos_a_numero(digitos):
    return reduce(lambda x,y:x*10 + y,digitos)

digitos_a_numero([4,3,9,2])
```

4392

Problema #3:
Crear una función que retorne las palabras de una lista de palabras que comience con una letra en específico. Utilizar la función **filter**.

Solución:

```
def filtro_palabras(lista_palabras, letra):
    return list(filter(lambda word:word[0]==letra,lista_palabras))

filtro_palabras(['Perro', 'Gato', 'Pelota', 'Manzana', 'Libro', 'Python'], 'P')
```

['Perro', 'Pelota', 'Python']

Problema #4:
Realizar una función que tome una lista de comprensión para devolver una lista de la misma longitud donde cada valor son las dos cadenas de L1 y L2 concatenadas con un conector entre ellas. **Ejemplo:** Listas: ['A', 'a'] ['B','b'] Conector: '-' Salida: ['A-a'] ['B-b']. Utilizar la función **zip**.

Solución:

```
def concatenacion(L1, L2, connector):
    return [word1+connector+word2 for (word1,word2) in zip(L1,L2)]
```

```
concatenacion(['A', 'a'],['B','b'],'-')
```

['A-a', 'B-b']

Casos de Consulta

Problema #5:
Realizar una función que tome una lista y retorne un diccionario que contenga los valores de la lista como clave y el índice como el valor. Utilizar la función **enumerate**.

Solución:

```
def d_list(L):
    return {key:value for value,key in enumerate(L)}
```

```
d_list(['a','b','c','d','e'])
```

{'a': 0, 'b': 1, 'c': 2}

Problema #6:
Realizar una función que retorne el recuento de la cantidad de elementos en la lista cuyo valor es igual a su índice. Utilizar la función **enumerate**.

Solución:

```
def count_match_index(L):
    return len([num for count,num in enumerate(L) if num==count])
```

```
count_match_index([0,2,2,1,5,5,6,10])
```