Diccionarios

Temas incluidos en la guía

- Diccionarios.
- Ciclos.

Ejercicios

> Nota: (★☆☆, ★★☆, ★★★) Esta notación indica la dificultad (ascendente).

1. ★☆☆ ¿Cual de los siguientes no es un método de diccionarios?

    1. get
    2. keys
    3. sort
    4. clear

2. ★☆☆ Imprima por pantalla las claves de un diccionario

3. ★☆☆ Imprima por pantalla los valores de un diccionario.

4. ★☆☆ Escriba una función que recibe un número y devuelve un diccionario cuyas claves son desde el número 1 hasta el número indicado, y los valores sean los cuadrados de las claves.

5. ★☆☆ Escriba una función que recibe una cadena y devuelve un diccionario con la cantidad de apariciones de cada carácter en la cadena.

6. ★★☆ Escribir un programa que guarde el diccionario `{'Euro':'€', 'Dollar':'$', 'Yen':'¥'}` en una variable, pregunte al usuario por una divisa y muestre su símbolo por pantalla, o un mensaje de aviso si la divisa no está en el diccionario.

7. ★★☆ Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sea el primer elemento de la tupla y el valor el segundo, asumir que nunca estarán repetidas las claves. Por ejemplo:

```
>>> l = [ ('papa', 3), ('cebolla', 479),('radicheta', 1) ]
>>> print(lista_de_tuplas_a_diccionario(l))
{ 'papa': 3, 'cebolla': 479 , 'radicheta': 1 }
```

8. ★★★ Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundos. Por ejemplo:

```
>>> l = [ ('Hola', 'don Pepito'), ('Hola', 'don Jose'),('Buenos', 'días') ]
>>> print(tuplas_a_diccionario(l))
{ 'Hola': ['don Pepito', 'don Jose'], 'Buenos': ['días'] }
```

9. ★★☆ Escribir una función que reciba la cantidad de iteraciones a realizar de una tirada de 2 dados y devuelva la cantidad de veces que se observa cada valor de la suma de los dos dados.

   > Nota: utilizar el módulo random para obtener tiradas aleatorias.

   ```
   >>> print(frecuencia_de_tiradas(9))
   { '2': 1, '5': 3 , '6': 2, '9': 2, '13': 1}
   ```

10. ★★☆ Implementamos una agenda con un diccionario, donde almacenamos nombres y números telefónicos. Escribir un programa que vaya solicitando al usuario que ingrese nombres, luego:

    1. Si el nombre se encuentra en la agenda, debe mostrar el teléfono y, opcionalmente, permitir modificarlo si no es correcto.
    2. Si el nombre no se encuentra, debe permitir ingresar el teléfono correspondiente.

    El usuario puede utilizar la cadena "*", para salir del programa.

11. ★★★ Escribir un archivo llamado `words.py` que contenga una variable que sea una lista con palabras (strings). Escribir, luego, una función que lea las palabras en `words.py` y las almacene como valores en un diccionario. La clave a la que pertenece cada palabra es su propia longitud. Luego se pueden recuperar rápidamente todas las palabras de una determinada longitud.

12. ★★★ Lea que es un bit de paridad [https://en.wikipedia.org/wiki/Parity_bit] y cree una función que reciba una lista de strings con 7 bits y devuelva un diccionario donde estos strings sean las claves y el *parity bit* el valor.

    ```
    >>> print(parity_bit(["0110100","1011010"]))
    { '0110100': 1, '1011010': 0 }
    ```

13. ★★★ Lea que es un Cyclic redundancy check [https://en.wikipedia.org/wiki/Cyclic_redundancy_check] y cree una función que reciba una lista de strings con 7 bits y devuelva un diccionario donde estos strings sean las claves y el CRC (de 4 bits de largo) el valor asociado.