Ejercitación № 11: Mas sobre Listas, Archivos , y Diccionarios

Cátedra Programación II Mayo 2019

Ejercitación

Desarrollar cada uno de los siguientes ejercicios de forma completa, detallando cada una de las etapas involucradas en la construcción de un programa, tal cual las presentamos en la receta. Para la etapa de testing, se deberá emplear el módulo `pytest`. Tener en cuenta que, deberá crear funciones que permitan el pasaje de argumentos para posteriormente testear el programa.

1. Mas sobre Listas y Secuencias

Ejercicio 1. Escribir una función que reciba una lista y un elemento, que:

1. devuelva la cantidad de apariciones del elemento recibido como parámetro, en la lista;
2. busque la primera coincidencia del elemento en la lista y devuelva su posición;
3. utilizando la función anterior, busque todos los elementos que coincidan con el recibido como parámetro y devuelva una lista con las posiciones.

Ejercicio 2. Escribir una función que tome una lista de números que:

1. devuelva el valor máximo;
2. devuelva el valor mínimo;
3. devuelva una tupla con el valor máximo y mínimo.
4. devuelva una tupla que incluya el valor máximo y su posición;
5. devuelva una tupla que incluya el valor mínimo y su posición;
6. ¿qué sucede si hay mas de un máximo y un mínimo?
7. ¿qué sucede si los elementos son listas de caracteres?

Ejercicio 3. Resolver el ejercicio “Administración de una biblio-ludoteca” dado en un archivo aparte.

Ejercicio 4. Resolver el ejercicio “Lista de supermercado!” dado en un archivo aparte.

2. Archivos

Ejercicio 5. Escribir un programa, llamado **head.py** que reciba un archivo y un número **N** e imprima las primeras **N** líneas del archivo.

Ejercicio 6. Escribir un programa, llamado **cp.py**, que copie todo el contenido de un archivo a otro, de modo que quede exáctamente igual.

Ejercicio 7. Escribir un programa, llamado **wc.py**, que reciba un archivo, lo procese e imprima por pantalla cuántas líneas, cuántas palabras y cuántos caracteres tiene el archivo.

Ejercicio 8. Escribir un programa, llamado **greep.py**, que reciba una expresión y un archivo e imprima, por pantalla, las líneas del archivo que contienen la expresión recibida.

Ejercicio 9. Escribir un programa, llamado **rot13.py**, que reciba un archivo de texto de origen y uno de destino, de modo que para cada línea del archivo origen, se guarde una línea *cifrada* en el archivo destino. El algoritmo de cifrado será muy sencillo: "a cada carácter comprendido entre la *a* y la *z*, se le suma 13 y luego se aplica el módulo 26, para obtener un nuevo caracter".

3. Diccionarios

Ejercicio 10. Veterinaria Crear un diccionario para guardar información sobre las mascotas. Las *claves* serán los nombres de los animales, y el *valor* será qué tipo de animal es. Por ejemplo: pedrito: loro, bobby: perro, kitty: gato siames, ico: caballo, etc.

- Agregar a los ya dados, por lo menos 5 elementos más al diccionario.
- Utilizar un ciclo para imprimir mensajes como el siguiente: "Bobby es un perro", "Kitty es un gato siames"

Ejercicio 11. Población en las Ciudades Crear un diccionario para guardar información sobre la población en las ciudades del mundo. La información de la población estará discriminada de la siguiente forma: *Población Femenina, Población Masculina, Población de Mascotas*. Las *claves* serán los nombres de las ciudades, y el *valor* será una tupla con los tres valores antes mencionados.

1. Generar un diccionario con por lo menos cinco valores. **{"Casilda": (20582,17452,15222) . . . }**
2. Utilizar un ciclo para imprimir mensajes como el siguiente:

```
1  ------------------------------------
2  Casilda:
3  ------------------------------------
4  Población Total:38034,
5  Población Femenina: 20582  (54,1\%)
6  Población Masculina: 17452 (55,9\%)
7  Población de Mascotas: 15222
8  ------------------------------------
```

3. Diseñar funciones independientes para las cuáles, dado un *diccionario de población* retorne:
   - La ciudad que tiene la máxima población.
   - El menor valor de la población de mascotas.
   - La ciudad que tiene mayor población femenina.

Ejercicio 12. Persistencia de datos sobre Diccionarios

Para los diccionarios creados en los ejercicios "Veterinaria" y "Población de Ciudades" definir las siguientes funciones para cargar y guardar los datos en un archivo.

1. Escribir una función *cargarDatosDicc* que reciba un nombre de archivo, cuyo contenido tiene el formato: *clave, valor* y devuelva un diccionario con el primer campo como clave y el segundo como valor.
2. Escribir una función *guardarDatosDicc* que reciba un diccionario y un nombre de archivo, y guarde el contenido del diccionario en el archivo, con el formato *clave, valor*.

Agregar también un *menú* en los mismos para que estas opciones aparezcan junto con otras que permitan agregar un dato, mostrar el diccionario, y eliminar un elemento del mismo.

Ejercicio 13. Cifrados

Utilizando la estructura de los diccionarios, implementar funciones que permitan para un mismo archivo, generar otro equivalente modulo mediante el cifrado. Los cifrados que emplearemos son los siguientes:

1. Cifrado de la Clave Morse.

   Utilizar el archivo *codigomorse.py* el cual almacena el diccionario asociado al mencionado código para realizar la traducción. Para ello se puede incluir en el archivo de codificación la siguiente línea: **from codigomorse import morse**

2. Cifrado de Cesar.

CIFADO CESAR
ABCDEFGHIJKLMNÑOPQRSTUVWXYZ
GHIJKLMNÑOPQRSTUVWXYZABCDEF

3. Cifrado de Rot13 (Cesar con rotación 13).

ROT13
ABCDEFGHIJKLMNOPQRSTUVWXYZ
NOPQRSTUVWXYZABCDEFGHIJKLM

Así el programa resolverá la traducción codificada que el usuario indique. Emplee un menú el cual permite traducir el texto ingresado por el usuario, junto con la opción de almacenarlo codificado en un archivo independiente.

Ejercicio 14. Revisamos la Lista de Contactos

Reimplementar el programa visto en clase para manipular una lista de contactos, ahora utilizando una estructura diccionario. Emplear como *clave* el nombre del contacto, y como *valor* una estructura con campos que modelen los restantes datos del contacto. Permitir que tenga mas de un teléfono asociado a un contacto, modelando dicha situación usando una lista.

Referencias

[1] Think Python: How to Think Like a Computer Scientist, Allen B. Downey, 2nd Edition, Version 2.2.18.

[2] Algoritmos y Programación I, Aprendiendo a programar usando Phyton como herramienta, Rosita Wachenchauzer et.al., 2016, (sin publicar).