

PRÁCTICA Nº 6

Cátedra Programación II

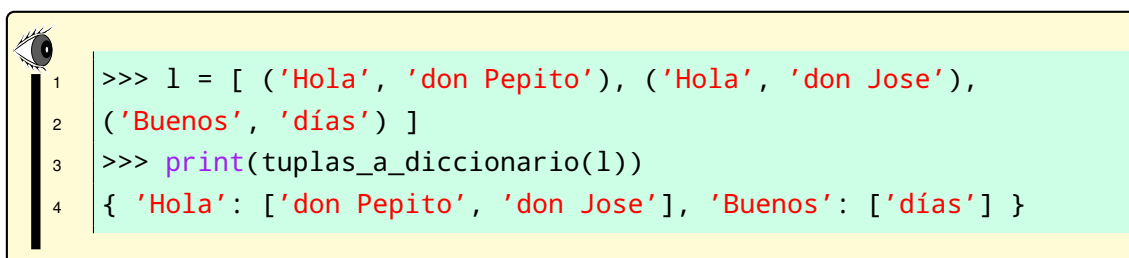
Octubre 2018

1. Listas y Secuencias

- 1) Escribir una función que reciba una lista desordenada y un elemento, que:
 - a) devuelva la cantidad de apariciones del elemento recibido como parámetro, en la lista;
 - b) busque la primera coincidencia del elemento en la lista y devuelva su posición;
 - c) utilizando la función anterior, busque todos los elementos que coincidan con el recibido como parámetro y devuelva una lista con las posiciones.
- 2) Escribir una función que tome una lista de números desordenada, que:
 - a) devuelva el valor máximo;
 - b) devuelva una tupla que incluya el valor máximo y su posición;
 - c) ¿qué sucede si los elementos son listas de caracteres?
- 3) Escribir una función que tome una lista ordenada y un elemento. Si el elemento se encuentra en la lista, debe encontrar su posición mediante búsqueda binaria y devolverlo. Si no se encuentra, debe agregarlo a la lista en la posición correcta y devolver esa nueva posición.

2. Diccionesarios

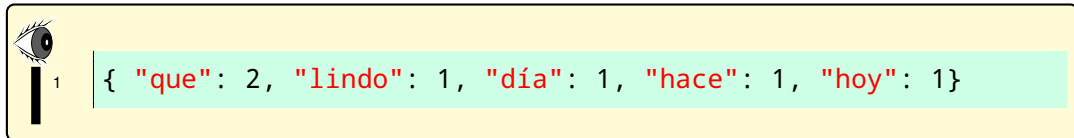
- 1) Escribir una función que reciba una lista de tuplas, y que devuelva un diccionario en donde las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundo. Por ejemplo:



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

2) Diccionarios usados para contar:

- a) Escribir una función que reciba una cadena y devuelva un diccionario con la cantidad de apariciones de cada palabra en la cadena. Por ejemplo, si recibe *"Qué lindo día que hace hoy"* debe devolver:



- b) Escribir la cantidad de apariciones de cada caracter en una cadena de texto, y los devuelva en un diccionario.
- c) Escribir una función que reciba una cantidad de iteraciones de una tirada de 2 dados a realizar y devuelva la cantidad de veces que se observa cada valor de la suma de los dos dados.
- 3) Escribir una función que reciba un texto y para cada caracter presente en el texto devuelva la cadena más larga en la que se encuentra ese caracter.

3. Archivos

1. Escribir un programa, llamado **head** que reciba un archivo y un número **N** e imprima las primeras **N** líneas del archivo.
2. Escribir un programa, llamado **cp.py**, que copie todo el contenido de un archivo a otro, de modo que quede exáctamente igual. Observación: usar `archivo.read(bytes)` para leer como máximo una cantidad de bytes.
3. 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.
4. 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.
5. 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 caracter comprendido entre la *a* y la *z*, se le suma 13 y luego se aplica el módulo 26, para obtener un nuevo caracter.
6. Persistencia de un diccionario
 - a) Escribir una función *cargarDatos* 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 diccionario.

- b) Escribir una función *guardarDatos* que reciba un diccionario y un nombre de archivo, y guarde el contenido del diccionario en el archivo, con el formato *clave, valor*.