Seminario de Lenguajes
Opción Python
Práctica 3

Importante: todos los ejercicios debe probarlos tanto en Python 2.x como Python 3.x, salvo que se indique lo contrario.
La utilización de algunos caracteres de nuestro lenguaje puede generar error unicode, por lo cual es necesario agregar el comienzo del script:
 # -- coding: utf-8 -*-*

Funciones y Módulos

1. Implemente las funciones vistas de operaciones sobre conjuntos (unión, intersección, diferencia simétrica y resta) que reciban los dos conjuntos y devuelvan el resultante.

2. Desarrolle mediante funciones y listas el funcionamiento de las colas FIFO (el primero en entrar es el primero en salir). Implemente un módulo que contenga las operaciones **pop**():, **push**(): y **length**(): recibiendo la lista como parámetro.
   Escriba un programa que permita probar las funciones definidas.
   Aclaración: En la función push si no se envía como parámetro ningún elemento por defecto el valor será 0 (cero).

3. Implemente la función **está()**: que reciba una lista **variable** de números y un diccionario. Por cada uno de ellos tiene que imprimir si está o no ese número como clave en el diccionario.

4. Defina las siguientes funciones para el ejercicio 4 de la práctica 2:
   a. **agregar_evento(eventos, datos):** Agregar los “datos” de un evento a la estructura de datos “eventos”
   b. **agregar_invitado(eventos, nom, desc_event):** Agrega en “eventos” un invitado de nombre “nom” al evento con descripción “desc_event”
   c. **confirmar_presencia(eventos, nom, desc_event):** Confirma la presencia del invitado de nombre “nom” al evento con descripción “desc_event” en “eventos”

5. A partir del ejercicio 8 de la práctica 2, el cálculo del máximo común divisor, resuélvalo a través de la función **mcd():** que sea llamada una sola vez y dentro de la misma no utilice las estructuras de control **for** ni **while**.

6. Dado el archivo “datos” disponible en el sitio de la cátedra, implemente un módulo que contenga que contenga las siguientes funciones:
    a. Función **max_min():** que devuelve el valor máximo y mínimo de la temperatura.
    b. Función **promedio()**: que devuelve el promedio de la humedad.
    c. Función **imprimo_info()**: que imprime la temperatura del último dato informando nombre de la ciudad que reciba como parámetro, se debe definir el valor por defecto ‘La Plata’ como ciudad en caso que no reciba.

    Escriba un programa que permita probar las funciones definidas.
    Aclaración: Todas las funciones deben contener los docstring correspondientes

7. Retomando el ejercicio3 de la práctica 2, agregue las notas:
    {‘Gabriela’ : 6, ‘José’: 10}
    a. Realice la función **sorteo()** que reciba el diccionario y devuelva el nombre de un alumno de manera aleatoria.
    b. Implemente la función **eligiendo()** que llame 20 veces a la función anterior y guarde en una estructura aparte la cantidad que salió cada alumno.
    c. Realice la función **salio_mas()** que calcule qué estudiante salió más veces en el ítem anterior.
    d. Realice la función **promedio()** que devuelva el promedio de las notas.

    Escriba un programa que permita probar las funciones definidas.
    Aclaración: Para el desarrollo de este ejercicio utilice los módulos Random y Collections

8. Modifique el ejercicio de 12 de la práctica 2, que en lugar ingresar la ubicación del objeto, lo ubique en una posición aleatoria.

9. Realice un programa que imprima un listado de los archivos y carpetas que contiene el directorio actual con información del tamaño de los mismos y la fecha de su último acceso. ¿Cambia algo si se realiza en Windows o Linux? Hágalo de tal forma que funcione en cualquier plataforma.

10. Escriba un programa que imprima qué versión de Python está utilizando, la codificación de caracteres por defecto, la plataforma en la que se está ejecutando, la cantidad de argumentos que recibe y el tamaño en bytes de cada uno de ellos.

11. Implemente un módulo con las siguientes funciones:

a. **doble(lista):** Multiplica por 2 cada elemento de “lista”. Es decir si tengo [2,3,4,5,6] el resultado sería [4,6,8,10,12]
b. **elevar(lista):** Eleva al cuadrado los números impares de “lista” y al cubo los pares.
c. **simplificar(lista):** Convierte cada elemento de “lista” a un carácter simple. Es decir, si lista = ['+zA[Z)E9G', 'h9nXDp89f', 'M8V-mZq+7', 'A7R6bX-*c',')]) (nms43','Z7ETzqT9@'], el resultado sería [‘+’, ‘z’, ’A’, ‘[‘, ‘Z’, ‘)’, ...]
d. **contabilizar(lista):** Informa la cantidad de apariciones de cada carácter de “lista” luego de haber aplicado la función simplificar().

Escriba un programa que permita probar las funciones definidas. Para a) y b) los números de la lista serán cargados por teclado (hasta recibir el -100).
Aclaración: Para el inciso d) utilice la colección Counter del módulo Collections

12. Realice los ejercicios del inciso anterior utilizando las funciones **lambda**, **map** y **filter**