Actividad 4 –Diccionarios, Conjuntos y Documentación.

Programación en Python - IAI - ECyT - UNSAM

1er cuatrimestre 2020

En esta clase vamos a estudiar dos estructuras de datos importantes en Pyhton: diccionarios (*dictionaries*) y conjuntos (*sets*). También vamos a discutir sobre la importancia de una adecuada documentación del código que escribimos y como hacerla.

Diccionarios.

Por favor leer la Unidad 9 de “Algoritmos y Programación” (p.127)

Ejercicio 1. Escribir una función que reciba una lista de duplas (tuplas de longitud dos), y que devuelva un diccionario en el cual las claves sean los primeros elementos de las tuplas, y los valores una lista con los segundos.

```
Ej.:    lista = [('Hola','don Pepito'),('Hola', 'don Jose')]
print(tuplas_a_diccionario(lista))
{ 'Hola': ['don Pepito', 'don Jose'] }
```

Ejercicio 2. Diccionarios usados para contar.

a) Escribir una función que reciba una frase y devuelva un diccionario con la cantidad de apariciones de cada palabra en la cadena. Ej.: si recibe `”Hay grandes libros en el mundo y grandes mundos en los libros”` debe devolver: `{'hay':1,'grandes':2, 'libros':2,'en':1,'el':1,'mundo':1,'y':1,'mundos':1}`. **Nota:** cuidado con las mayúsculas.

b) Escribir una función que cuente la cantidad de apariciones de cada caracter en una cadena de texto y los devuelva en un diccionario. “Graficar” ese diccionario anotando una letra por línea, seguida de un `#` por cada repetición. Ej: “Abracadabra”

```
a #####
b ##
r ##
c #
d #
```

Ejercicio 3. Escribir un programa que :

a) Simule un dado usando `import random` y `random.randint(1,6)`.
b) Tire un par de dados y guarde los resultados en una tupla.
c) Repita (b) cien veces y guarde las cien tuplas resultantes en una lista.
d) Reciba esa lista en una función que calcule la cantidad de veces que los dados suman cada uno de los valores posibles, y guarde esos resultados en un diccionario.
e) “Grafique” como antes (con barras para repeticiones de cada valor posible (del 2 al 12)).

Ejercicio 4. Continuación de la agenda. iniciada en el Ej. 15 Guía 3.

Modifiquemos la agenda que habíamos hecho. Cambiemos la lista por un *diccionario* donde las *claves* serán nombres de personas y sus *valores* asociados serán sus números de teléfono u otros datos asociados (puede haber varios *valores* para una misma *clave*), y permitamos que:

(a) El programa pida un nombre al usuario.
(b) Busque ese nombre en su diccionario con coincidencias parciales (p.ej. `'ric' in 'ricardo'`).
(c) Si las hay, devuelva las coincidencias (*claves*) y sus números de teléfonos (*valores*).
(d) Si el nombre no existe, ofrezca ingresar un texto asociado y guarde ambos en el diccionario.
(e) Sólo si el nombre es `'*'` el programa termina, perdiendo todos los datos (volveremos a ésto).
(f) Si el nombre comienza con …

`'-'`	`(menos)`	elimine del diccionario todas las claves coincidentes.
`'+'`	`(más)`	si existe, ofrezca como en (d). Sino no haga nada, solo avise.
`'?'`	`(pregunta)`	elimine del diccionario el valor asociado, pida uno nuevo y lo guarde.

Conjuntos

Leer las páginas 133 y 134, y compruebe en el intérprete la ejecución de las operaciones mostradas en los ejemplos del libro : `add()` , `union()` , `intersection()` , `difference()`.

Puede consultar la lista completa de métodos en la documentación de Python [aquí] o en castellano aquí.

Documentación

Por favor lean el capítulo 10 del libro (p. 135). Formaliza conceptos que hasta ahora les hemos transmitido en comentarios o sugerencias, y es bueno retomarlos y formalizarlos con más experiencia. No menciona que `help(función)` devuelve su docstring (probarlo), algo muy útil al usar herramientas escritas por otro. Después discutamos brevemente los cuatro casos presentados.

Ejercicio 5. Analizar cada una de las siguientes funciones. ¿Cuál es el contrato de la función? ¿Cómo sería su documentación? ¿Es necesario agregar comentarios? ¿Se puede simplificar el código y/o mejorar su legibilidad?

a)
```
def Abs(i):
    if i >= 0:
        return i
    else:
  return -i
```

b)
```
def emails(diccionario):
    for k, v in diccionario.items():
        print(f"El e-mail de {k} es {v}")
```

c)

```
def emails2(diccionario):
    buenos = {}
    for k, v in diccionario.items():
        if '@' in v:
      buenos[k] = v
    return buenos
```

d)

```
def emails3(nombres, direcciones):
    for nom in range(len(nombres)):
              if direcciones[nom] == None:
                  nombre, apellido = ' '.split(nombres[nom].lower())
                  direcciones[nom] = nombre[0] + apellido + '@ejemplo.com'
```

Ejercicio 6. Mastermind

Ver archivo adjunto.