Actividad 3 –Tuplas, Listas y Algoritmos de búsqueda.

Programación en Python - IAI - ECyT - UNSAM

1er cuatrimestre 2020

El objetivo de esta actividad es ejercitar las estructuras de datos llamadas *tuplas* (*tuples*), *listas* (*lists*) (unidad 7) e incursionar en diferentes algoritmos de búsqueda, recorrido y combinación de listas y tuplas (unidad 8).

Como siempre, por favor envíen los ejercicios resueltos a: `python@unsam.edu.ar` y completen la "Evaluación Actividad 3". Los ejercicios marcados con un asterisco (*) son optativos, los (**) son desafíos.

Leer sección 7 (y su apéndice) del libro "Apunte Teórico – Algoritmos y Programación – FIUBA" antes de resolver Ejs. 1 a 12. Leer la sección 8 y su apéndice para el resto.

Ejercicio 1. Escribir una función que reciba una tupla de elementos e indique si se encuentran ordenados de menor a mayor o no. p.ej `('algo','nada','poco')` ó `(1,45,3.45,-1)` Nota: el orden de las cadenas (*strings*) es lexicográfico.

Ejercicio 2. *Dominó:* Escribir una función que indique si dos fichas de dominó son compatibles o no, para el caso en que … (Nota: utilizar la función `split()` para las cadenas.)

a) Las fichas son recibidas en dos tuplas, por ejemplo: `(3,4)` y `(5,4)`

b) Las fichas son recibidas en una cadena, por ejemplo: `'3-4 2-5'`.

Ejercicio 3. *Campaña electoral*

a) Escribir una función que reciba una tupla con nombres, y para cada nombre imprima el mensaje *Estimado <nombre>, vote por mí*.

b) Escribir una función que reciba una tupla con nombres p.ej. `('juan','mario','maría')`, una posición de origen *p* y una cantidad *n*, e imprima el mensaje *Estimado <nombre>, vote por mí*. para los *n* nombres que se encuentran a partir de la posición *p*.

c) Modificar las funciones anteriores para que tengan en cuenta el género del destinatario, para ello, deberán recibir una tupla de tuplas, conteniendo el nombre y el género. p.ej. `(('juan','m'),('mario','m'),('maría','f'))`

Ejercicio 4. (*) *Vectores matemáticos codificados como tuplas de python:*

a) Escribir una función que reciba dos vectores y devuelva su producto escalar.

b) Escribir una función que reciba dos vectores y diga si son o no ortogonales.

c) Escribir una función que reciba dos vectores y diga si son paralelos o no.

d) Escribir una función que reciba un vector y devuelva su norma.

Nota: definición en la Wikipedia (español) **o** más claro (inglés)

Ejercicio 5. Dada una lista de números enteros y un entero *k*, escribir una función que:

a) Devuelva tres listas, una con los menores, otra con los mayores y otra con los iguales a *k*

b) Devuelva una lista con aquellos que son múltiplos de *k*.

Ejercicio 6. Escribir una función que reciba una lista de tuplas `(Apellido, Nombre, Inicial_segundo_nombre)` y devuelva una lista de cadenas donde cada una contenga primero el nombre, luego la inicial con un punto, y luego el apellido, p.ej: `[('Bond','James','J.'),...] --> ['James J. Bond',...]`

Ejercicio 7. *Inversión de listas*

a) Escribir una función que, dada una lista, devuelva una nueva lista cuyo contenido sea igual a la original pero invertida. Así, dada la lista `['Di', 'buen', 'día', 'a', 'papa']`, deberá devolver `['papa', 'a', 'día', 'buen', 'Di']`.

b) Escribir otra función que reciba una lista e invierta la misma lista, sin usar listas auxiliares ni crear listas nuevas.

Ejercicio 8. Escribir una función `empaquetar()` para una lista, donde epaquetar significa indicar la repetición de valores consecutivos mediante una *tupla* `(valor, cantidad de repeticiones)`. Por ejemplo, empaquetar `([1, 1, 1, 3, 5, 1, 1, 3, 3])` debe devolver `[(1, 3), (3, 1), (5, 1), (1, 2), (3, 2)]`. (nota: esta es la base de muchos algoritmos de compresión)

Ejercicio 9. (*) *Algebra de Matrices*. (Nota1: sólo si sabe operar con matrices. Nota2: aunque existen herramientas para manipular matrices, hoy representemos una matriz como una tupla de tuplas.)

a) Escribir una función que reciba dos matrices *A* y *B* y devuelva su suma *A* + *B*.

b) Escribir una función que reciba dos matrices *A* y *B* y devuelva su producto *A* x *B*.

c) (**) Escribir una función que opere sobre una matriz y mediante eliminación gaussiana devuelva una matriz triangular superior.

d) (**) Escribir una función que indique si un grupo de vectores, recibidos mediante una lista, son linealmente independientes o no.

Ejercicio 10. (*) *Plegado de un texto*. Escribir una función que reciba un texto y una longitud y devuelva el texto completo fraccionado en una lista de cadenas de como máximo esa longitud. Las líneas deben ser cortadas correctamente en los espacios (sin cortar las palabras al medio) y ser lo más largas posible.

Ejercicio 11. *Funciones que reciben funciones*.

a) Escribir una funcion llamada `mapear()`, que reciba una función y una lista y devuelva la lista resultante de aplicar la función recibida a cada uno de los elementos de la lista original.

b) Escribir una función llamada `filtrar()`, que reciba una función y una lista y devuelva una lista con los elementos de la lista original para los cuales la función recibida devuelve un valor verdadero.

c) ¿En qué ejercicios de esta guía podría haber utilizado estas funciones?

Ejercicio 12. *Funciones que reciben funciones (II)*.

Volviendo al cifrado César de la guía anterior, rescate su función `cifrar (cadena)` y una funcion inversa `descifrar (cadena)`. Escriba entonces una función que, evaluando la primera letra de una cadena pueda distinguir si el mensaje debe ser descrifrado o cifrado, y le aplique la función correspondiente. (Convención : todos los mensajes comienzan con `'Hola'`).

Ejercicios de la Sección 8: Algoritmos de búsqueda.

Ejercicio 13. En cada caso, escribir una función que, dada una lista desordenada y un elemento:

a) Busque todos los elementos coincidan con el pasado por parámetro y devuelva la cantidad de coincidencias encontradas. `[6,3,6,9,3,1,0,5], 3` → `2`

b) Busque la primera coincidencia del elemento en la lista y devuelva su posición.

c) Utilizando la función anterior, devuelva una lista con las posiciones de todas las coincidencias.

Ejercicio 14. En cada caso, escribir una función que, dada una lista desordenada y un elemento:

a) Devuelva el valor máximo.

b) Devuelva una tupla que incluya el valor máximo y su posición.

c) ¿Qué sucede si los elementos son cadenas de caracteres?

Nota: no utilizar lista.sort()

Ejercicio 15. Agenda simplificada

Escribir una función que reciba una lista de tuplas `[(nombre_completo1, telefono1), (nombre_completo2, telefono2),...)]`, y una `cadena` a buscar, y devuelva una lista con todas las tuplas en las que el `nombre_completo` contenga la `cadena` (puede ser el nombre, el apellido o sólo una parte de cualquiera de ellos).

Ejercicio 16. (*) Sistema de facturación simplificado

Se desea generar una factura que incluya la cantidad, la descripción, el precio unitario y el precio total de cada producto comprado, y al final imprima el total general.

Para ello se cuenta con una lista ordenada de productos donde cada producto está representado por una tupla `(identificador, descripción, precio_unitario)`, y una lista de los productos comprados representados como tuplas `(identificador, cantidad)`.

Usando listas por comprehensión (Apéndice 8.A) aplique un descuento del 10% a todo producto del que se hayan comprado 3 ó mas unidades.

Escribir una función `facturar()` que reciba estas dos listas de tuplas y devuelva una factura.

Ejercicio 17. Escribir una función que reciba una lista ordenada y un elemento. Si el elemento se encuentra en la lista, encontrar su posición mediante búsqueda binaria. Si no se encuentra, agregarlo en la posición correcta. En ambos casos devolver su posición. No utilizar `lista.sort()`