

Pontificia Universidad Católica de Chile
Escuela de ingeniería
Departamento de Ciencia de la Computación
Primer Semestre del 2014

IIC1103 Introducción a la Programación

Control Diccionarios-Listas-Tuplas-Búsqueda-Ordenamiento

La perfumería de nombre "Buenos Olores" decidió almacena la información de sus productos usando un diccionario en Python, con la siguiente estructura:

```
catalogo = {marca_1:[(producto11,sexo11,fomato11,precio11),
                     (producto12,sexo12,fomato12,precio12), …],
            marca_2:[(producto21,sexo21,fomato21,precio21),
                     (producto22,sexo22,fomato22,precio22), …],
            marca_3:[(producto31,sexo31fomato31,precio31),
                     (producto32,sexo31,fomato32,precio32), …], …
            marca_n:[(producto_n1,sexo_n1,fomato_n1,precio_n1),
                     (producto_n2,sexon2,fomaton2,precion3), …],
   }
```

Es decir el ***catalogo*** de sus productos esta compuesto por un diccionario, donde las claves son los nombres (marca) de los fabricantes del perfume, y los valores asociados a estas claves son una lista de tuplas, donde cada tupla almacena información de un producto de esa marca.

Ejemplo, un diccionario de *catalogo* sería:

```
catalogo = {'Carolina Herrera':[('CHIC EDP', 'Mujer',30,16900),
                                ('CHIC EDT', 'Varon', 30,16900),
                                ('Carolina Herrera EDP', 'Mujer', 100,52000),
                               …],
              …
              }
```

Así una tupla de la marca Carolina Herrera, como la tupla **('CHIC EDP', 'Mujer',30,16900)**, indica que el perfume **CHIC EDP,** es un perfume de **mujer**, que viene en un formato de envase de **30** ml y que cuesta $**16.900**.

Pregunta 1:

Para recomendar ofertas a sus clientes la perfumería desea poder generar una lista ordenada (por precio) con los perfumes más barato de cada marca (independiente del formato del perfume), para ello se te pide que crees una función que permita generar esta lista a partir del catálogo de productos. En tu función debes buscar el perfume más barato de cada marca, ponerlo en la lista, y una vez que tienes todos los perfumes más baratos de cada marca debes ordenar de acuerdo al valor del producto y retornar esta lista como resultado.

Observación: La lista resultado, tiene el producto más barato de cada una de las marcas, por lo que tu lista podría tener elementos del tipo (marca, producto), donde marca es una marca que esta en catalogo y producto es la tupla con la información del producto mas barato de esa marca.

Solución:

```
# Se implementa sólo si se usó la function sorted para ordenar
def clave1(item):
      return item[4]


def baratos_precio(catalogo):
      resultado = []

      # paso 1
      # buscar los mas baratos por marca
      for marca in catalogo:
            producto_m = catalogo[marca][0]
            precio_m   = producto_m[3]
            for i in range(1,len(catalogo[marca])):
                  producto = catalogo[marca][i]
                  if producto[3] < precio_m:
                        producto_m = producto
                        precio_m   = producto[3]
            resultado.append((marca,producto_m[0],producto_m[1],
                              producto_m[2], producto_m[3]))
      # paso 2
      # Si se ordena la lista usando sorted, entonces es obligatorio
      # tener implementada la función que retorna la posición a buscar
      # en este caso es clave1 definida el comienzo
      # Si no se usa sorted, entonces cualquier método de ordenamiendo
      # sirve (inserción o selección)
      aux = sorted(resultado, key = clave1)
      resultado_final = []

      # paso 3
      # dejar la lista de respuesta final en el format
      #(marca,producto)
      # entendiendo que un producto es una tupla
      for elemento in aux:

      resultado_final.append((elemento[0],(elemento[1],elemento[2],
                              elemento[3],elemento[4])))
      # hay otras alternativas de ordenar la lista
      return resultado_final
```

Pregunta 2:

Los compradores más expertos, que hacen rendir más su dinero, han observado que el mayor beneficio por compra se calcula de la relación valor versus formato (tamaño), es decir para comparar productos, en términos de decir cual es más barato, importa el precio, pero también el formato (tamaño), y la mejor relación entre formato del producto y precio, se da por lo general con aquellos perfumes de mayor formato, pese a que el valor puede ser más alto. Es decir, el valor que ellos asocian al producto para comprar es $valor/formato. En base a este criterio, se te pide que escribas una función en Python, que

reciba como parámetro el catalogo de productos, y retorne una lista ordenada con los perfumes de cada marca que tengan la mejor relación precio formato.

Solución:

```
# Se implementa sólo si se usó la function sorted para ordenar
def clave2(item):
      return item[5]

def baratos_ml(catalogo):
      resultado = []

      # paso 1
      # buscar por marca que perfume tiene la mejor relacion precio
      # cantidad
      for marca in catalogo:
            producto_m = catalogo[marca][0]
            relacion   = producto_m[3]/producto_m[2]
            for i in range(1,len(catalogo[marca])):
                  producto = catalogo[marca][i]
                  if producto[3]/producto[2] < relacion:
                        producto_m = producto
                        indice   = producto[3]/producto[2]
            resultado.append((marca,producto_m[0],producto_m[1],
                              producto_m[2], producto_m[3],relacion))

      # paso 2
      # Si se ordena la lista usando sorted, entonces es obligatorio
      # tener implementada la función que retorna la posición a buscar
      # en este caso es clave2 definida el comienzo
      # Si no se usa sorted, entonces cualquier método de ordenamiendo
      # sirve (inserción o selección)
      aux = sorted(resultado, key = clave2)
      resultado_final = []

      # paso 3
      # dejar la lista de respuesta final en el format
      # (marca,producto) entendiendo que un producto es una tupla
      for elemento in aux:
            resultado_final.append((elemento[0],(elemento[1],
                              elemento[2], elemento[3],elemento[4])))
      return resultado_final
```