IWI 131-PROGRAMACIÓN

Ayudantía 13

Ayudante: Anastasiia Fedorova

Paralelo: 212

Fecha: 29.07.2020

EJERCICIO 1 (CERTAMEN 2 2013-2)

La tienda Pylabella maneja la información de sus clientes en un diccionario cuya clave corresponde al RUT del cliente y su valor a una lista con su información. Esta información está compuesta por el nombre del cliente, su edad, la sucursal donde realizó su inscripción, y una tupla con los montos de las tres útlimas compras. Considere las siguiente estructura **como ejemplo**

```
clientes = {'16034124-5': ('Fernando Ruiz Diaz', 30, 'Vina', (100, 200, 300)),
            '5436576-2': ('Mike Portnoy', 20, 'Quilpue', (100, 100, 50)),
            '3333333-3': ('Slash', 50, 'Vina', (500, 550, 300)),
            '1234567-8': ('Cliff Burton', 35, 'Valparaiso’, (10, 10, 100))}

#RUT: [nombre,edad,sucursal,(monto1,monto2,monto3)]
```

EJERCICIO 1 (CERTAMEN 2 2013-2)

Escriba la función `misma_sucursal(RUT1, RUT2, clientes)` que reciba RUT de dos clientes, y además un diccionario `clientes`. La función debe retornar `True` si los clientes se inscribieron en la misma sucursal y `False` en el caso contrario.

```
>>> misma_sucursal('16034124-5','3333333-3',clientes)
True
>>> misma_sucursal('5436576-2','3333333-3',clientes)
False
```

SOLUCIÓN

```
def misma_sucursal(RUT1, RUT2, clientes):
    if clientes[RUT1][-2] == clientes[RUT2][-2]:
        return True
    return False
```

EJERCICIO 1 (CERTAMEN 2 2013-2)

Escriba la función `mayores_que(edad_minima, clientes)` que devuelva una lista con los nombres de clientes cuya edad es igual o mayor a la `edad_minima`.

```
>>> mayores_que(50,clientes)
['Slash']
>>> mayores_que(25,clientes)
['Fernando Ruiz Diaz','Cliff Burton','Slash']
```

SOLUCIÓN

```
def mayores_que(edad_minima, clientes):
    lista = []
    for rut in clientes:
        nombre, edad, _, _ = clientes[rut]
        if edad >= edad_minima:
            lista.append(nombre)
    return lista
```

EJERCICIO 1 (CERTAMEN 2 2013-2)

Escriba la función `es_cliente_vip(RUT, monto_compra, clientes)` la cual recibe como parámetro el `RUT` de cliente, el `monto_compra` que representa el monto mínimo para ser considerado cliente VIP y el diccionario `clientes`. Esta función debe retornar `True` si la suma de tres últimas compras de cliente es igual o superior a `monto_compra`, y `False` en caso contrario.

```
>>> es_cliente_vip('5436576-2',500,clientes)
False
>>> es_cliente_vip('3333333-3',500,clientes)
True
>>> es_cliente_vip('1234567-8',150,clientes)
False
```

SOLUCIÓN

```
def es_cliente_vip(RUT, monto_compra, clientes):
    nombre, _, _, compras = clientes[RUT]
    sum_compra = 0
    for compra in compras:
        sum_compra += compra
    if sum_compra >= monto_compra:
        return True
    return False
```

EJERCICIO 2 (CERTAMEN 3 2017-1)

Se mantiene la información relevante a la autopista en 3 archivos: un registro diario de patentes que circulan por ella, la información de patentes que contienen o no contienen TAG (indicado con 1 y 0), información de dueños y patentes asociadas. En `info_dueños` los dueños aparecen solo 1 vez.

`registro_diario.txt`

```
abmn32
crtj12
dflp11
hb5101
```

`patentes.txt`

```
crtj12,1
abmn32,0
hb5101,0
dflp11,1
tljg99,0
jfzo10,0
...
```

`info_duenos.txt`

```
Alex Perez;crtj12,hb5101,kcfl36,emda16
Aquiles Castro ;abmn32,tljg99,avrv33
Maria Gana;ab7677
Fede Santos;utfs90,dflp11
```

EJERCICIO 2 (CERTAMEN 3 2017-1)

Se necesita saber cuántos vehículos tiene cada usuario. Construya una función patentes_por_dueno(archivo_duenos) que recibe el nombre de archivo que contiene a los dueños y retorna un diccionario. Este debe tener como llave el nombre y apellido de la persona y como valor el número de vehículos que posee.

```
>>> patentes_por_dueno('info_duenos.txt')

{'Alex Perez': 4, 'Aquiles Castro ': 3,
 'Fede Santos':2,'Maria Gana': 1}
```

SOLUCIÓN

```
def patentes_por_dueno(archivo_duenos):
    d = {}
    a = open(archivo_duenos, "r")
    for linea in a:
        nombre, patentes = linea.strip().split(";")
        patentes = patentes.split(",")
        cant = 0
        for patente in patentes:
            cant += 1
        d[nombre] = cant
    a.close()
    return d
```

EJERCICIO 2 (CERTAMEN 3 2017-1)

Se requiere listar a aquellas patentes que circularon durante el día sin tener TAG. Construya la función multar_patentes(archivo_registro, archivo_patentes) que recibe nombres de archivo con registros diarios y con patentes y TAGs. Esta función debe retornar una lista con patentes que cumplen con el criteo mencionado.

```
>>> patentes_multadas('registro_diario.txt', 'patentes.txt')
['abmn32', 'hb5101']
```

SOLUCIÓN

```
def patentes_multadas(archivo_registro, archivo_patentes):
    a2 = open(archivo_patentes, "r")
    lista = []
    for linea in a2:
        patente, TAG = linea.strip().split(",")
        if TAG == "0":
            a1 = open(archivo_registro, "r")
            for registro in a1:
                r = registro.strip()
                if r == patente:
                    lista.append(r)
            a1.close()
    a2.close()
    return lista
```

EJERCICIO 2 (CERTAMEN 3 2017-1)

Se necesita listar los dueños con patentes multadas. Para esto, se les pide construir la función personas_multadas(archivo_registros,archivo_patentes,archivo_duenos) que debe retornar un listado con los nombres de los dueños de patentes multadas. Este listado no debe contener nombres repetidos.

```
>>> personas_multadas('registro_diario.txt', 'patentes.txt',
'info_duenos.txt')
['Aquiles Castro ', 'Alex Perez']
```

SOLUCIÓN

```
def personas_multadas(archivo_registro, archivo_patentes, archivo_duenos):
    lista = []
    multas = patentes_multadas(archivo_registro, archivo_patentes)
    a = open(archivo_duenos, "r")
    for linea in a:
        nombre, patentes = linea.strip().split(";")
        patentes = patentes.split(",")
        for patente in patentes:
            if patente in multas:
                if nombre not in lista:
                    lista.append(nombre)
    a.close()
    return lista
```

EJERCICIO 3 (CERTAMEN 3 2017-1)

Una pequeña empresa chocolatera tiene una base de datos nutricional de materia prima que usa para crear su mercancía. Los archivos tienen el formato nombre_producto, marca, cantidad (en gramos). Desde la segunda línea se indica la información nutricional del producto. Excepto por las calorías, todos los valores están en gramos, sin importar si son líquidos o sólidos.

CHOCOLATEAMARGO txt
```
CHOCOLATE AMARGO,LA FETE,100
AZUCARES 10
CALORIAS 400
CARBOHIDRATOS 40
GRASAS 20
PROTEINAS 28
```

LICORMENTA TXT
```
LICOR DE MENTA,LIDER,4
AZUCARES 1.6
CALORIAS 20
CARBOHIDRATOS 1.6
GRASAS 0
PROTEINAS 0.2
```

EJERCICIO 3 (CERTAMEN 3 2017-1)

La receta de un producto es un archivo con nombre RECETA-PRODUCTO.txt. En la primera línea se indica su peso en gramos, y en la segunda en adelante se describen las cantidades en gramos de los ingredientes a utilizar.

```
RECETA-CHOCOLATE_CON_MENTA TXT
160
CHOCOLATEAMARGO 150
LICORMENTA 10
```

EJERCICIO 3 (CERTAMEN 3 2017-1)

Se les pide a usted que escriba la función calcular_total(archivo_receta) que genera un archivo con la información nutricional de nuevo producto. El nombre de archivo debe ser NUTRICION-NOMBRE_RECETA.txt. Tenga en cuenta que el cálcula de la información nutricional debe considerar la información nutricional de materia prima, la cual debe ser proporcional a la cantidad de gramos indicada en la receta.

```
NUTRICION-CHOCOLATE_CON_MENTA.txt
CHOCOLATE CON MENTA,160.0
**********
AZUCARES 19.0
CALORIAS 650.0
CARBOHIDRATOS 64.0
GRASAS 30.0
PROTEINAS 42.5
```

```
def calcular_total(archivo_receta):
    a_receta = open(archivo_receta, "r")
    count = 1
    dicc = {}
    receta = (archivo_receta.split("-")[1]).split(".")[0]
    a_nutricion = open("NUTRICION-" + receta + ".txt", "w")
    for linea in a_receta:
        if count == 1:
            peso = float(linea.strip())
            receta = receta.split("_")
            receta = " ".join(receta)
            a_nutricion.write(",".join([receta, str(peso) + "\n"]))
            a_nutricion.write("*" * 10 + "\n")
            count += 1
        else:
            producto, cantidad = linea.strip().split(" ")
            arch_aux = open(producto + ".txt", "r")
            count += 1
            count_aux = 1
```

```
        for l in arch_aux:
                    if count_aux == 1:
                        _, _, porcion = l.strip().split(",")
                        count_aux += 1
                    else:
                        count_aux += 1
                        ingrediente, cant = l.strip().split(" ")
                        # cantidad en receta es proporcional
                        cantidad_proporcional = (
                            float(cant) / float(porcion)) * float(cantidad)
                        if ingrediente not in dicc:
                            dicc[ingrediente] = cantidad_proporcional
                        else:
                            dicc[ingrediente] += cantidad_proporcional
                arch_aux.close()
    a_receta.close()
    for ing in dicc:
        a_nutricion.write(" ".join([ing, str(dicc[ing]) + "\n"]))
    a_nutricion.close()
```