Herramientas de Programación

2.2.2 Tratamiento de datos estructurados: listas de listas

Ejercicios con listas de listas homogéneas (matrices)

1. **Máximo de cada columna**. Diseña una función `max_columnas(lst)` que, dada una lista de listas, que representa una matriz de nf filas y nc columnas, devuelva una lista con el cálculo del mayor valor de cada columna. Ejemplo:

```
>>> max_columnas([[2,3,4],[3,5,3],[1,1,2]])
[3, 5, 4]
```

2. **Media de cada fila**. Diseña una función `media_filas(lst)` que, dada una **lista de listas**, que representa una matriz (los elementos de la lista son las filas de la matriz), de `nf` filas y `nc` columnas, devuelva una **lista** con el cálculo del valor medio de cada fila. Ejemplo:

```
>>> M = [[1,2,3,4],[1,3,5,7]
>>> media_filas(M)
[2.5, 4.0]
```

3. **Media de cada columna**. Diseña una función `media_columnas(lst2)` que, dada una lista de listas, que representa una matriz de nf filas y nc columnas, devuelva una lista con el cálculo del valor medio de cada columna. Ejemplo:

```
>>> M2 = [[2,3,4],[3,3,3]]
>>> media_columnas(M2)
[2.5, 3.0, 3.5]
```

4. **Mayor fila**. Diseña una función `mayor_fila(m)` que, dada una lista de listas, que representa una matriz de nf filas y nc columnas, devuelva la fila cuya suma de elementos sea máxima. Se tiene que devolver el índice de la fila y el valor de su suma. Ejemplo:

```
>>> mayor_fila([[1,2,3],[3,3,3],[1,3,2]])
(1, 9)
```

5. **Es diagonal?**. Diseña una función `es_diagonal(m)` que, dada una matriz cuadrada (por ejemplo, 3x3), nos diga si es una matriz diagonal. Una matriz es diagonal si los elementos `m[i][j] = 0`, para `i ≠ j`. Ejemplo:

```
>>> es_diagonal([[1,0,0],[0,2,0],[0,0,3]])
True
```

6. **Es identidad?**. Diseña una función `es_identidad(m)` que, dada una matriz cuadrada (por ejemplo, 3x3), nos diga si es una matriz identidad. Una matriz identidad es una matriz diagonal con los valores de la diagonal iguales a 1. Ejemplo:

```
>>> es_identidad([[1,0,0],[0,1,0],[0,0,1]])
True
```

Listas de listas heterogéneas

1. ***Gestionar Pensionistas***. Los gastos de un grupo de pensionistas se guardan en una lista de listas donde cada sub-lista representa la información de un pensionista y está formada por el identificador del pensionista (string), un entero que indica la edad y una serie de gastos mensuales que se guardan (enteros). El número de gastos mensuales puede variar entre pensionistas. Por ejemplo, el pensionista con identificador '1111A' se llama 'Carlos' tiene 68 años y tiene 3 gastos mensuales de 640, 589 y 573.

```
lst = [['1111A','Carlos',68,640,589,573],
       ['2222D','Lucia',69,710,550,570,698,645,512],
       ['3333J','Paula',72,530,534],
       ['4444N','Luis',75,770,645,630,650,590,481,602]]
```

   a) Diseña una función mediaGastos(p) en que, dada una lista que codifica los datos de **un pensionista p**, devuelva el promedio de los gastos con dos cifras decimales. Ejemplo:

```
>>> mediaGastos(lst[0])
600.67
```

   b) Diseña una función mediaEdad(lst) en que, dada una lista de los pensionados, devuelva el promedio de las edades con un decimal. Del ejemplo lst:

```
>>> mediaEdad(lst)
71.0
```

   c) Diseña una función EdadesExtremas(lst) en que, dada una lista como la mostrada en el ejemplo, devuelva la edad menor y la mayor de los pensionistas. Del ejemplo lst:

```
>>> EdadesExtremas(lst)
(68, 75)
```

   d) Diseña una función sumaPromedio(lst) en que, dada una lista como la mostrada en el ejemplo, devuelva la suma del promedio de los gastos de todos los pensionistas de la lista. Redondea a 2 decimales. Del ejemplo lst:

```
>>> sumaPromedio(lst)
2370.84
```

   e) Diseña una función MediaMaxima(lst) en que, dada una lista como la mostrada en el ejemplo, retorne el mayor promedio de los gastos entre todos los pensionistas de la lista. Del ejemplo lst:

```
>>> MediaMaxima(lst)
624.0
```

   f) Diseña una función GastoPromedio(lst) en que, dada una lista de pensionistas devuelve otra lista con el gasto medio mensual de cada persona. La lista resultante debe estar ordenada de forma ascendente.

```
>>> GastoPromedio(lst)
[532.0, 600.67, 614.17, 624.0]
```

g) Diseña una función gastoPromedioSuperior(lst, g) en que, dada una lista como la mostrada en el ejemplo y un gasto g, devuelva otra lista con el identificador y la edad de la primera persona de la lista que tenga un promedio de gasto superior a g. En caso de que no haya ninguna, la función devolverá la lista vacía.

```
>>> gastoPromedioSuperior(lst, 600)
['1111A', 68]
```

2. **Lista de datos de pacientes**. Diseña una función que, dada una lista de datos de pacientes en listas que contienen como elementos "listas" con elementos heterogéneos: *apellido*, *código de la seguridad social* y *edad*, calcule la edad promedio de los pacientes (con una cifra decimal de precisión). Ejemplo de paciente: `paciente1 = ['Puig','c1234', 56]`.

```
>>> paciente1 = ['Puig','c1234', 56]  # apellido, código, edad
>>> lst = [paciente1,['Marti','c3456',39],['Smith','c543',45]]
>>> edad_m_pacientes(lst)
46.7
```

3. **Clientes vip**. Se dispone de la información anual de las compras de los clientes de un centro comercial en una lista lst. En lst, cada cliente aparece en una sublista, cuyo primer elemento es un string que identifica al cliente y luego aparecen los importes de cada una de las compras realizadas por el cliente. Diseñar una función `vip(lst)` que retorne la lista de los clientes que han hecho compras por un importe total superior a 1000 euros y han comprado al menos tres veces. En la lista a retornar sólo deben aparecer los identificadores de los clientes que cumplen las dos condiciones y debe estar ordenada alfabéticamente. Ejemplo:

```
>>> vip([['pepe', 350, 505, 200], ['juan', 590, 650], ['paula', 100, 105,  450], ['ana', 650, 351, 5]])
['ana', 'pepe']
>>>  vip([['juan', 590, 650], ['paula', 100, 105, 450, 95], ['sara', 350, 350, 300]])
[]
>>>  vip([['89GR', 348, 943, 304], ['09PY', 932, 894, 49, 49], ['64FF', 8932, 3290, 320]])
['09PY', '64FF', '89GR']
```

4. **Mejor herramienta**. Una fábrica de herramientas de precisión realiza un proceso de calidad, realizando múltiples mediciones con la misma herramienta.
   Diseña una función `mejor_herramienta(Lmediciones)` que, dada una lista de mediciones reales `Lmediciones`, devuelva el nombre de la herramienta que tenga mayor precisión, es decir, aquella cuya desviación estándar sea menor. Podemos suponer que para cada herramienta se han realizado al menos dos mediciones. La desviación estándar de la muestra, s, de una lista de N números x[0], [1], … , x[N-1] es:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(x[i] - \bar{x})^2}$$

donde $\bar{x}$ es la media de los números de la lista. Por ejemplo, dada la lista de 8 mediciones [1.2, 1.3, 1.0, 1.5, 1.2, 1.4, 1.3, 1.2], la media $\bar{x}$ es 1.2625 y la desviación estándar muestral es 0.15059406173077153. Ejemplo de sesión:

```
>>> m = [['herramienta1',1.29708186, 1.31266035, 1.33536083, 1.29651107,
...                      1.27489226, 1.31669962, 1.296645  , 1.31999788,
...                      1.30039536, 1.28287777, 1.28805435, 1.30090147,
...                      1.27151722, 1.26324628, 1.2893031 , 1.30099176,
...                      1.29308506, 1.30773716, 1.29497462, 1.29530172],
...         ['herramienta2',2.82486964, 2.79124532, 2.80591011, 2.80944679,
...                      2.81057422, 2.7981161 , 2.77104033, 2.79758669,
...                      2.78554721, 2.80345295, 2.79900319, 2.7846449 ,
...                      2.80583578, 2.81153274, 2.80816726, 2.81009209,
...                      2.7946134 , 2.7883018 , 2.7968774 , 2.78145684],
...         ['herramienta3',1.59912295, 1.60199575, 1.59920467, 1.60108844,
...                      1.60497468, 1.59742797, 1.6009022 , 1.59948543,
...                      1.60242975, 1.60092409, 1.59868339, 1.59909426,
...                      1.59670533, 1.59309514, 1.60266672, 1.59587891,
...                      1.59662104, 1.60692734, 1.60238514, 1.60106771],
...         ['herramienta4',0.91837844, 0.91031125, 0.87873479, 0.91919775,
...                      0.94548981, 0.92281759, 0.87520564, 0.90950628,
...                      0.91250966, 0.90622502, 0.90045178, 0.89656651,
...                      0.92379851, 0.90519854, 0.87822092, 0.83423747,
...                      0.92680545, 0.9184275 , 0.9108524 , 0.88113948],
...         ['herramienta5',1.89940911, 1.90002205, 1.89982581, 1.90027163,
...                      1.89981981, 1.89991447, 1.89988227, 1.90011783,
...                      1.90013265, 1.89962221, 1.90012783, 1.89985387,
...                      1.90009801, 1.89989183, 1.89987518, 1.8993762 ,
...                      1.89976188, 1.89942767, 1.89992038, 1.89967331],
...         ['herramienta6',2.02130853, 1.99060463, 2.01337742, 1.99429791,
...                      1.98339322, 1.99800895, 2.0008932 , 2.0228473 ,
...                      2.00771829, 2.00002554, 1.99437905, 2.01117355,
...                      1.99537268, 2.00808408, 1.99111077, 1.98352936,
...                      1.99376165, 1.98541568, 2.00602666, 2.00810295]]
>>> mejor_herramienta(m)
'herramienta5'
```