9. Tipos Estructurados en Python (Listas) – Actividades

Los siguientes ejercicios están tomados del capítulo 5 y 6 del libro digital **"Introducción a la Programación con Python 3.**

Ejercicio 1:
Hacer el ejercicio 209

▶ 209 ¿Sabrías decir que resultados se mostrarán al ejecutar estas sentencias?

```
>>> [1, 2] < [1, 2]
>>> [1, 2, 3] < [1, 2]
>>> [1, 1] < [1, 2]
>>> [1, 3] < [1, 2]
>>> [10, 20, 30] > [1, 2, 3]
>>> [10, 20, 3] > [1, 2, 3]
>>> [10, 2, 3] > [1, 2, 3]
>>> [1, 20, 30] > [1, 2, 3]
>>> [0, 2, 3] <= [1, 2, 3]
>>> [1] < [2, 3]
>>> [1] < [1, 2]
>>> [1, 2] < [0]
```

Ejercicio 2: (ejercicio 210)

▶ 210 Diseña un programa que tras asignar dos listas a sendas variables nos diga si la primera es menor que la segunda. No puedes utilizar operadores de comparación entre listas para implementar el programa.

Ejercicio 3:
Diseña un programa que construya una lista con los **n primeros** números primos**.**

Ejercicio 4: (ejercicio 221)

▶ 221 Diseña un programa que lea una lista de 10 enteros, pero asegurándose de que todos los números introducidos por el usuario son positivos. Cuando un número sea negativo, lo indicaremos con un mensaje y permitiremos al usuario repetir el intento cuantas veces sea preciso.

Ejercicio 5: (ejercicio 224)

▶ 224 Diseña un programa que elimine de una lista todos los elementos de *índice* par y muestre por pantalla el resultado.
(Ejemplo: si trabaja con la lista [1, 2, 1, 5, 0, 3], esta pasará a ser [2, 5, 3]).

No usar "del".

Ejercicio 6: (ejercicio 225)

▶ 225 Diseña un programa que elimine de una lista todos los elementos de *valor* par y muestre por pantalla el resultado.
(Ejemplo: si trabaja con la lista [1, -2, 1, -5, 0, 3], esta pasará a ser [1, 1, -5, 3]).

No usar "del".

Ejercicio 7:
Escribir un programa que, dada una lista de 20 números enteros cargados aleatoriamente, los ordene de mayor a menor.

Ejercicio 8:
Escribir un programa que, dada una lista de 20 números enteros cargados aleatoriamente, ordene de menor a mayor los números pares.

Ejercicio 9:
Escribir un programa que lea datos de 20 alumnos.
Los datos son:
- Apellido
- Nota (valor entre 1 y 10)

El programa deberá guardar en una lista los apellidos, y en otra lista las notas.
Luego, deberá mostrar:
a. Los datos (Apellido y Nota) ordenados por Apellido.
b. Los datos (Apellido y Nota) ordenados por Nota.

Ejercicio 10:
Escribir un programa que lea temperaturas máximas y mínimas para los doce meses del año.
El programa deberá guardar en una lista las temperaturas máximas, y en otra lista las temperaturas mínimas.
Luego, deberá:
a. Mostrar promedio de temperaturas máximas.
b. Mostrar promedio de temperaturas mínimas.
c. Generar y mostrar una lista con las diferencias de temperaturas por mes.
d. Ordenar la lista anterior de mayor a menor, mostrando a qué mes corresponde cada dato.

Ejercicio 11:

▶ 296 Tenemos los tiempos de cada ciclista y etapa para los participantes en la última vuelta ciclista local. La lista *ciclistas* contiene una serie de nombres. La matriz *tiempos* tiene una fila por cada ciclista, en el mismo orden con que aparecen en *ciclistas*. Cada fila tiene el tiempo en segundos (un valor flotante) invertido en cada una de las 5 etapas de la carrera. ¿Complicado? Quizás te ayude este ejemplo de lista *ciclistas* y de matriz *tiempos* para 3 corredores.

```
ciclistas = ['Pere Porcar', 'Joan Beltran', 'Lledó Fabra']
tiempo = [[10092.0, 12473.1, 13732.3, 10232.1, 10332.3],
          [11726.2, 11161.2, 12272.1, 11292.0, 12534.0],
          [10193.4, 10292.1, 11712.9, 10133.4, 11632.0]]
```

En el ejemplo, el ciclista Joan Beltran invirtió 11161.2 segundos en la segunda etapa.
Se pide:

- Una función que reciba la lista y la matriz y devuelva el ganador de la vuelta (aquel cuya suma de tiempos en las 5 etapas es mínima).
- Una función que reciba la lista, la matriz y un número de etapa y devuelva el nombre del ganador de la etapa.
- Un procedimiento que reciba la lista, la matriz y muestre por pantalla el ganador de cada una de las etapas.

Ejercicio 12:

▶ 299 Diseña una función que reciba los tres coeficientes de una ecuación de segundo grado de la forma $ax^2 + bx + c = 0$ y devuelva una lista con sus soluciones reales. Si la ecuación solo tiene una solución real, devuelve una lista con dos copias de la misma. Si no tiene solución real alguna o si tiene infinitas soluciones, devuelve una lista con dos copias del valor None.

Ejercicio 13:

▶ 300 Diseña una función que reciba una lista de palabras (cadenas) y devuelva, simultáneamente, la primera y la última palabras según el orden alfabético.

Ejercicio 14:

▶ 310 Diseña una función que reciba dos listas y devuelva los elementos comunes a ambas, sin repetir ninguno (intersección de conjuntos).

Ejemplo: si recibe las listas `[1, 2, 1]` y `[2, 3, 2, 4]`, devolverá la lista `[2]`.

Ejercicio 15:

▶ 311 Diseña una función que reciba dos listas y devuelva los elementos que pertenecen a una o a otra, pero sin repetir ninguno (unión de conjuntos).

Ejemplo: si recibe las listas `[1, 2, 1]` y `[2, 3, 2, 4]`, devolverá la lista `[1, 2, 3, 4]`.

Ejercicio 16:

▶ 312 Diseña una función que reciba dos listas y devuelva los elementos que pertenecen a la primera pero no a la segunda, sin repetir ninguno (diferencia de conjuntos).

Ejemplo: si recibe las listas `[1, 2, 1]` y `[2, 3, 2, 4]`, devolverá la lista `[1]`.

Ejercicio 17:

▶ 313 Diseña una función que, dada una lista de números, devuelva otra lista que solo incluya sus números impares.

Ejercicio 18:

▶ 314 Diseña una función que, dada una lista de nombres y una letra, devuelva una lista con todos los nombres que empiezan por dicha letra.

Ejercicio 19:

▶ 315 Diseña una función que, dada una lista de números, devuelva otra lista con solo aquellos números de la primera que son primos.

Ejercicio 20:

▶ 316 Diseña una función que, dada una lista de números, devuelva una lista con todos los pares de números que podemos formar con uno de la primera lista y otro de la segunda. Por ejemplo, si se suministran las listas `[1, 3, 5]` y `[2, 5]`, la lista resultante es

`[[1, 2], [1, 5], [3, 2], [3, 5], [5, 2], [5, 5]]`.

Ejercicio 21:

▶ 330 Diseña una función que reciba una lista y devuelva otra lista cuyo contenido sea el resultado de concatenar la lista original consigo misma. La lista original no debe modificarse.

Ejercicio 22:

▶ 331 Diseña una función que reciba una lista y devuelva otra lista cuyo contenido sea la lista original, pero con sus componentes en orden inverso. La lista original no debe modificarse.

Ejercicio 23:

▶ 335 Diseña una función que reciba una matriz y, si es cuadrada (es decir, tiene igual número de filas que de columnas), devuelva la suma de todos los componentes dispuestos en la diagonal principal (es decir, todos los elementos de la forma $A_{i,i}$). Si la matriz no es cuadrada, la función devolverá `None`.

Ejercicio 24:

▶ 336 Guardamos en una matriz de $m \times n$ elementos la calificación obtenida por m estudiantes (a los que conocemos por su número de lista) en la evaluación de n ejercicios entregados semanalmente (cuando un ejercicio no se ha entregado, la calificación es -1).

Diseña funciones y procedimientos que efectúen los siguiente cálculos:

- Dado el número de un alumno, devolver el número de ejercicios entregados.
- Dado el número de un alumno, devolver la media sobre los ejercicios entregados.
- Dado el número de un alumno, devolver la media sobre los ejercicios entregados si los entregó todos; en caso contrario, la media es 0.
- Devolver el número de todos los alumnos que han entregado todos los ejercicios y tienen una media superior a 3,5 puntos.
- Dado el número de un ejercicio, devolver el número de estudiantes que lo han presentado.
- Dado el número de un ejercicio, devolver la nota media obtenida por los estudiantes que lo presentaron.
- Dado el número de un ejercicio, devolver la nota más alta obtenida.
- Dado el número de un ejercicio, devolver la nota más baja obtenida.
- Devolver el número de abandonos en función de la semana. Consideramos que un alumno abandonó en la semana s si no ha entregado ningún ejercicio desde entonces. Este procedimiento mostrará en pantalla el número de abandonos para cada semana (si un alumno no ha entregado nunca ningún ejercicio, abandonó en la «semana cero»).

Ejercicio 25:

▶ 241 Diseña un programa que lea dos matrices y calcule la diferencia entre la primera y la segunda.

Ejercicio 26:

▶ 242 Diseña un programa que lea una matriz y un número y devuelva una nueva matriz: la que resulta de multiplicar la matriz por el número. (El producto de un número por una matriz es la matriz que resulta de multiplicar cada elemento por dicho número).

Ejercicio 27:

▶ 243 La traspuesta de una matriz A de dimensión $m \times n$ es una matriz A^T de dimensión $n \times m$ tal que $A^T_{i,j} = A_{j,i}$. Por ejemplo, si

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 12 & 6 \\ 1 & 0 & -3 \\ 10 & -1 & 0 \end{pmatrix}$$

entonces:

$$A^T = \begin{pmatrix} 1 & 2 & 1 & 10 \\ 2 & 12 & 0 & -1 \\ 3 & 6 & -3 & 0 \end{pmatrix}$$

Diseña un programa que lea una matriz y muestre su traspuesta.

Ejercicio 28:

▶ 246 Diseña un programa que, dada una matriz, determine si la suma de los elementos de cualquiera de sus filas es igual a la suma de los elementos de cualquiera de sus columnas.

Ejercicio 29:

▶ 247 Una matriz cuadrada es triangular superior si todos los elementos por debajo de la diagonal principal son nulos. Por ejemplo, esta matriz es triangular superior:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 12 & 6 \\ 0 & 0 & -3 \end{pmatrix}$$

Diseña un programa que diga si una matriz es o no es triangular superior.