

Ejercicios para trabajar con listas

Ejercicio 01: Realizar un programa que dado dos listas, verifique que las listas estén solapadas o no. Una lista está solapada con otra, cuando hay un elemento común entre las dos listas. El programa escribirá como salida si las listas están solapadas o no. Las listas de prueba se deben escribir en el código del problema.

Ejemplo, sean A, B y C las siguientes listas:

A = [1, 2, 3, 4]
B = [5, 6, 7, 8, 9]
C = [2, 0, 9]

Las listas A y B no están solapadas, en cambio, las listas A y C si están solapadas, el elemento 2 es común a las 2.

Ejercicio 02: Diseñar un programa que calcule e imprima la suma de los números de cada una filas de una matriz. Luego debe imprimir las sumas de los elementos por columnas.

La matriz se crea en el código del programa.

```
#  
matriz = [[7,6,5], [8,3,4], [3,2,1]]  
#
```

la salida debe ser la siguiente:

```
/DAM/00-FICHEROS-PYTHON/UT03-LISTAS/SumaFilasC  
La fila 1 la suma de sus valores es: 18  
La fila 2 la suma de sus valores es: 15  
La fila 3 la suma de sus valores es: 6  
La columna 1 la suma de sus valores es: 18  
La columna 2 la suma de sus valores es: 11  
La columna 3 la suma de sus valores es: 10
```

Ejercicio 03: Modificar el programa anterior para los datos de las listas se ingresen por teclado.

Ejercicio 04: Queremos guardar en una lista los nombres y las edades de los alumnos de un curso. Realiza un programa que permita leer el nombre y la edad de cada alumno y forme la lista solicitada

El proceso de lectura de datos terminará cuando se introduzca como nombre un asterisco (*).

La lista debe ser de tuplas, y cada tupla contiene (nombre_del_alumno, edad)

Al finalizar la lectura de los datos el programa debe mostrar la siguiente información:

- Los nombres de los alumnos mayores de edad.
- Los nombres y edades de los 3 alumnos mayores.

Ejercicio 05: Modificar el ejercicio anterior, para tener en cuenta en el apartado dos, que pueda existir un empate en la edad del tercer alumno mayor con la del cuarto, la del quinto, etc. En este caso, se debe mostrar los datos también los datos de esos alumnos.

Ejercicio 06: Crea una lista e inicialízala con 5 cadenas de caracteres leídas por teclado. Copia los elementos de la lista en otra lista, pero en orden inverso, y muestra sus elementos por la pantalla.

Ejercicio 07: Se quiere realizar un programa que lea por teclado y almacene en una lista las 9 notas obtenidas por un alumno en las asignaturas que está cursando (comprendidas entre 0 y 10).

Se debe verificar que la nota esté entre 0 y 10, en caso de que no se cumpla, se debe indicar un mensaje al usuario indicando el error. Al tercer error, al indicar la misma nota, se debe abortar la ejecución del programa.

Una vez leídas las notas, el programa debe mostrarlas todas, calcular e imprimir la nota media, la nota mayor y menor que haya obtenido.

La forma de la salida debe ser la siguiente:

Notas obtenidas: nota1, nota2, nota3, nota4 nota5, nota6 nota7, nota8, nota9

Promedio de notas: *****

Nota más alta: *****

Nota más baja: *****

Ejercicio 08: Programa que cree una lista y la llene de números leídos por pantalla hasta que introduzcamos un número negativo. Entonces se debe imprimir la lista (como lista) (sólo los elementos introducidos).

Ejercicio 09: Crea un programa que pida un número al usuario que representa un número de mes (por ejemplo, el 4) y diga cuántos días tiene (por ejemplo, 30) y el nombre del mes. Debes usar listas. Para simplificarlo vamos a suponer que febrero tiene 28 días.

Ejercicio 10: Diseña un programa que cree tres listas 'lista1', 'lista2' y 'lista3'.

La lista1 y lista2 se llenan con números que se leen por teclado.
La lista3 contiene la suma elemento por elemento de lista1 y lista2.
Al final el programa debe mostrar el resultado de la siguiente forma:

```
nro1lista1 + nro1lista2 = nro1lista3  
nro2lista1 + nro2lista2 = nro2lista3  
nro3lista1 + nro3lista2 = nro3lista3  
nro4lista1 + nro4lista2 = nro4lista3  
nro5lista1 + nro5lista2 = nro5lista3
```

La lista 1 es = [nro1lista1, nro2lista1, nro3lista1, nro4lista1, nro5lista1]
La lista 2 es = [nro1lista2, nro2lista2, nro3lista2, nro4lista2, nro5lista2]
La lista 3 es = [nro1lista3, nro2lista3, nro3lista3, nro4lista3, nro5lista3]

Ejercicio 11: Se quiere realizar un programa que sume dos matrices. La matriz resultante de la suma de dos matrices se realiza sumando elemento por elemento y asignándolo a la misma posición.

Veamos un ejemplo:

$A = \begin{bmatrix} 2 & 3 \\ 1 & 0 \end{bmatrix}$ y $B = \begin{bmatrix} 8 & 7 \\ -7 & 4 \end{bmatrix}$ el resultado de la suma de $A + B = \begin{bmatrix} 10 & 10 \\ -6 & 4 \end{bmatrix}$

Se tiene que tener en cuenta que las matrices A y B deben coincidir en numero de filas y deben coincidir en número de columnas, por ejemplo:

$A = \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ -2 & 5 \end{bmatrix}$ y $B = \begin{bmatrix} 8 & 7 \\ -7 & 4 \\ 6 & 9 \end{bmatrix}$ la suma de $A + B = \begin{bmatrix} 10 & 10 \\ -6 & 4 \\ 4 & 14 \end{bmatrix}$

Las matrices A y B se leen por teclado, para ello, se debe pedir primero el número de filas y de columnas de cada matriz, se debe verificar que se puede realizar la suma.

En caso de que no se pueda realizar la suma se indica mensaje y se aborta el programa.

Si se puede realizar se debe mostrar de la siguiente forma:

```
1 2  3 4  4  6  
-1 8  6 3  7 11
```

Recuerden como hacer la escritura en columnas, explicado en un ejercicio en clase:

Ejercicio 12: Se quiere modificar el programa anterior de tal forma que además de sumar matrices el problema pueda restarlas ($A - B$). La resta de matrices tiene las mismas restricciones que la suma y los elementos del resultado se obtienen restando a los elemento de la matriz A los de la matriz B.

El programa debe ampliarse para que al comenzar a ejecutarse contemple el siguiente menú:

1. Leer A
2. Leer B
3. Verificar que A y B son compatibles
4. $A + B$
5. $A - B$
9. Salir

Ejercicio 13: Un cuadrado mágico es una matriz cuadrada (el número de filas es igual número de columnas) en el que la suma de los valores de cualquier fila y de cualquier columna y sus diagonales suman 15.

Diseñe un programa que dada una matriz cuadrada leída por teclado verifique si representa un cuadrado mágico o no.

Este es un cuadrado mágico de orden 3 (3 filas y columnas)

4	9	2
3	5	7
8	1	6

Y este el de orden 4, la suma ahora debe dar 34.

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Ejercicio 14: La transpuesta de una matriz es la matriz a la que obtiene al cambiar las filas por columnas y viceversa de la matriz original.

Veamos un ejemplo:

Si la matriz A es: 1 2 3

4 5 6

La transpuesta de la matriz A es:

1 4
2 5
3 6

Realice un programa que lea una matriz, calcule su transpuesta y muestre ambas por pantalla.

Ejercicio 15: Una matriz cuadrada se dice en “espejo” cuando la matriz y su transpuesta son iguales.

La matriz de abajo es “en espejo”

1 4 5
4 2 6
5 6 3

Y esta no lo es:

1 4 5
4 2 7
8 6 3

Ejercicio 16: Hacer un programa que inicialice una lista de números con valores aleatorios (10 valores), y posterior ordene los elementos de menor a mayor y los muestre por pantalla.

Ejercicio 17: Realizar un programa que inicialice una lista con 10 valores aleatorios (del 1 al 10) y posteriormente muestre en pantalla cada elemento de la lista junto con su cuadrado y su cubo.

Ejercicio 18: Crea una lista de números de un tamaño pasado por teclado, la lista contendrá números aleatorios entre 1 y 300 y mostrar aquellos números que acaben en un dígito que nosotros le indiquemos por teclado (debes controlar que se introduce un numero correcto), estos números deben guardarse en una nueva lista

Ejercicio 19: Se quiere simular un juego en el que participan N jugadores y otra persona que hace de árbitro. Cada jugador elige 4 números en el rango [1, 10], pudiendo estar repetidos. A continuación, el árbitro, sin conocer los números que ha elegido cada jugador, selecciona 2 números A y B.

El programa debe ser capaz de calcular cuántos números de los seleccionados por cada jugador están comprendidos entre los valores A y B. Ganará el jugador que más números tenga en dicho intervalo.

Se pide implementar un programa modular que simule el juego para 3 jugadores, teniendo en cuenta que:

- Tanto los 4 datos de cada jugador, como los valores para A y B se introducirán por teclado. En todos los casos, el programa detectará la entrada de números erróneos, solicitando nuevamente el dato hasta que sea válido.
- Se deben mostrar por pantalla no solo los aciertos de cada jugador sino los datos que ha introducido cada jugador y los que ha seleccionado el árbitro. Por último, hay que imprimir la media aritmética de los aciertos de todos los jugadores

Ejercicio 20: Crea una lista de caracteres que contenga de la 'A' a la 'Z' (sólo las mayúsculas). Después, ve pidiendo posiciones de la lista por teclado y si la posición es correcta, se añadira el carácter a una cadena que se mostrara al final, se dejará de insertar cuando se introduzca un -1.

Por ejemplo, si escribo los siguientes números:

```
0 //Añadira la 'A'
5 //Añadira la 'F'
25 //Añadira la 'Z'
50 //Error, inserte otro numero
-1 //fin
```

Cadena resultante: AFZ

Ejercicio 22: Escribir un programa en Python que indique si dos fichas de dominó cuyos valores se leen por el teclado *encajan* o no. Las fichas se almacenan en tuplas, por ejemplo: (3,4) y (5,4).

Ejecuciones del programa pueden ser la siguientes:

```
Indique la primera ficha: 3,5
Indique la segunda ficha: 4,6
```

Las fichas no encajan

```
Indique la primera ficha: 2,3
Indique la segunda ficha: 0,3
```

Las fichas si encajan

Debe comprobarse que se leen fichas correctas de dominó y no se permite seguir la ejecución del programa hasta que se introduzca datos correctos.

Ejercicio 23: Escribir un programa en Python que permita empaquetar una lista, donde empaquetar significa indicar la repetición de valores consecutivos mediante una tupla (valor, cantidad de repeticiones). Por ejemplo, empaquetar la lista [1, 1, 1, 3, 5, 1, 1, 3, 3] debe generar la siguiente lista de tuplas: [(1, 3) , (3, 1) , (5, 1), (1, 2), (3, 2)].

Los elementos de la primera lista deben leerse por teclado.

Ejercicio 24: Realizar un programa en Python para llevar la gestión de los precios de los artículos de una tienda, así como la cantidad vendida en total de cada artículo para lo cual, habrá que tener 3 vectores.

1. Nombres: guardará los nombres de los artículos
2. Precios: guardará los precios de los artículos
3. Cantidades vendidas: guardará la cantidad vendida en total de cada artículo

Además, este programa debe implementar las siguientes opciones que van a mostrarse en un menú:

1. Introducir un artículo nuevo – Registrar un nuevo artículo con nombre y precio
2. Hacer una venta – Solicitar nombre de producto y cantidad a vender
3. Mostrar información – Mostrar nombre, precio, cantidad vendida e importe de los artículos, además del total
4. Borrar un artículo
5. Borrar todos los artículos
6. Salir