

Clase Número: 7

Estructura de datos tipo lista (segunda parte)

Listas: ordenamiento de sus elementos

Otro algoritmo muy común que debe conocer y entender un programador es el ordenamiento de una lista de datos.

El ordenamiento de una lista se logra intercambiando las componentes de manera que:
`lista[0] <= lista[1] <= lista[2]` etc.

El contenido de la componente `lista[0]` sea menor o igual al contenido de la componente `lista[1]` y así sucesivamente.

Si se cumple lo dicho anteriormente decimos que la lista está ordenado de menor a mayor. Igualmente podemos ordenar una lista de mayor a menor.

Tengamos en cuenta que la estructura de datos lista en Python es mutable, eso significa que podemos modificar sus elementos por otros.

Se puede ordenar tanto listas con componentes de tipo `int`, `float` como cadena de caracteres. En este último caso el ordenamiento es alfabético.

Problema 1:

Se debe crear y cargar una lista donde almacenar 5 sueldos. Desplazar el valor mayor de la lista a la última posición.

La primera aproximación para llegar en el próximo problema al ordenamiento completo de una lista tiene por objetivo analizar los intercambios de elementos dentro de la lista y dejar el mayor en la última posición.

El algoritmo consiste en comparar si la primera componente es mayor a la segunda, en caso que la condición sea verdadera, intercambiamos los contenidos de las componentes.

Vamos a suponer que se ingresan los siguientes valores por teclado:

1200
750
820
550
490

En este ejemplo: ¿es 1200 mayor a 750? La respuesta es verdadera, por lo tanto intercambiamos el contenido de la componente 0 con el de la componente 1.

Luego comparamos el contenido de la componente 1 con el de la componente 2: ¿Es 1200 mayor a 820?

La respuesta es verdadera entonces intercambiamos.

Si hay 5 componentes hay que hacer 4 comparaciones, por eso el for se repite 4 veces.

Generalizando: si la lista tiene N componentes hay que hacer N-1 comparaciones.

Cuando	x = 0	x = 1	x = 2	x = 3
	750	750	750	750
	1200	820	820	820
	820	1200	550	550
	550	550	1200	490
	490	490	490	1200

Podemos ver cómo el valor más grande de la lista desciende a la última componente.

Empleamos una variable auxiliar (aux) para el proceso de intercambio:

Programa:

```

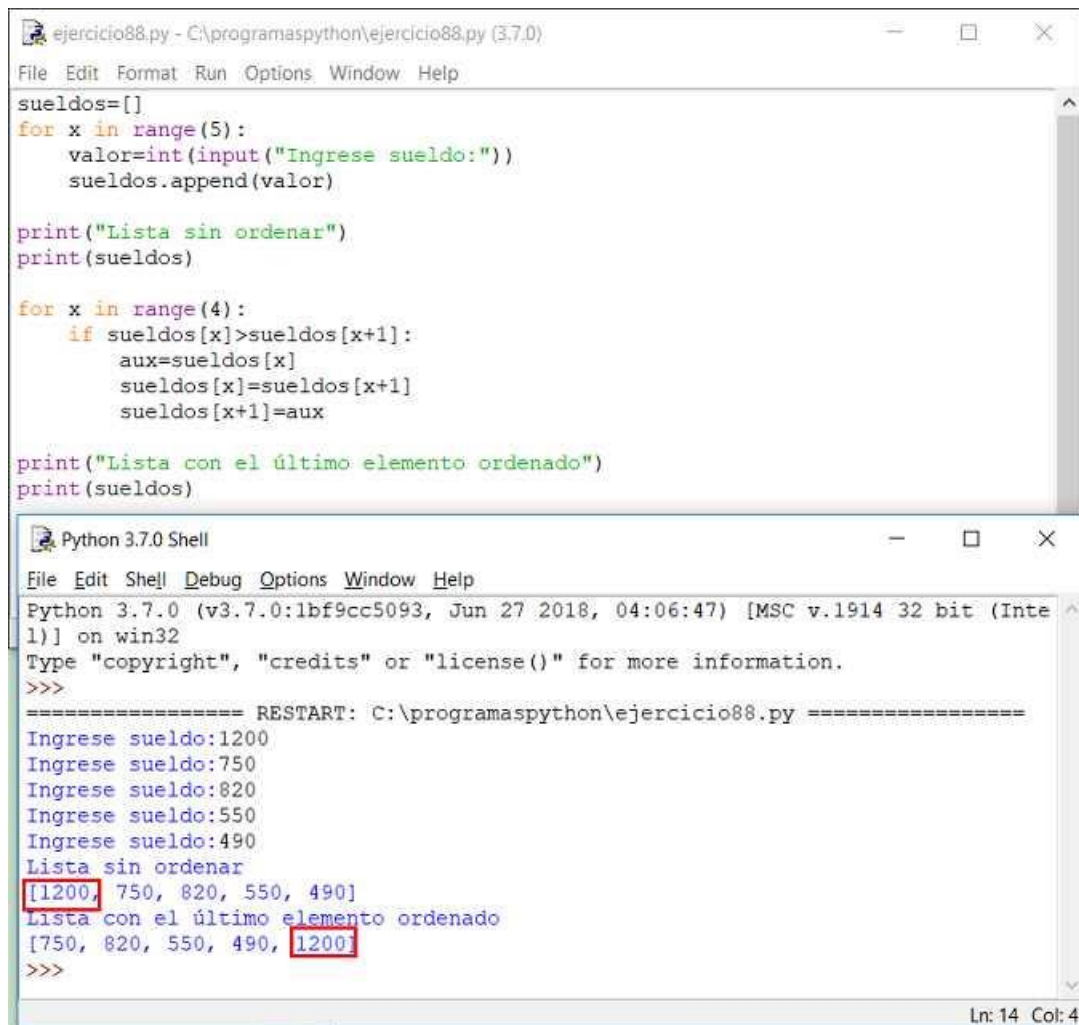
suealdos=[]
for x in range(5):
    valor=int(input("Ingrese sueldo:"))
    suealdos.append(valor)

print("Lista sin ordenar")
print(suealdos)

for x in range(4):
    if suealdos[x]>suealdos[x+1]:
        aux=suealdos[x]
        suealdos[x]=suealdos[x+1]
        suealdos[x+1]=aux

print("Lista con el último elemento ordenado")
print(suealdos)

```



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'ejercicio88.py - C:\programaspython\ejercicio88.py (3.7.0)', contains the following Python code:

```
sueldos=[]
for x in range(5):
    valor=int(input("Ingrese sueldo:"))
    sueldos.append(valor)

print("Lista sin ordenar")
print(sueldos)

for x in range(4):
    if sueldos[x]>sueldos[x+1]:
        aux=sueldos[x]
        sueldos[x]=sueldos[x+1]
        sueldos[x+1]=aux

print("Lista con el último elemento ordenado")
print(sueldos)
```

The bottom window, titled 'Python 3.7.0 Shell', shows the execution output:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio88.py =====
Ingrese sueldo:1200
Ingrese sueldo:750
Ingrese sueldo:820
Ingrese sueldo:550
Ingrese sueldo:490
Lista sin ordenar
[1200, 750, 820, 550, 490]
Lista con el último elemento ordenado
[750, 820, 550, 490, 1200]
>>>
```

The output shows that the list is [1200, 750, 820, 550, 490] after input and [750, 820, 550, 490, 1200] after the sorting loop. The element 1200 is now at the end of the list.

Al salir del for el contenido de la lista es la siguiente:

750
820
550
490
1200

Analizando el algoritmo podemos comprobar que el elemento mayor de la lista se ubica ahora en el último lugar.

Podemos volver a ejecutar el programa y veremos que siempre el elemento mayor queda al final.

Pero con un único for no se ordena una lista. Solamente está ordenado el último elemento de la lista.

Problema 2:

Se debe crear y cargar una lista donde almacenar 5 sueldos. Ordenar de menor a mayor la lista.

Ahora bien como vimos en el problema anterior con los 4 elementos que nos quedan podemos hacer el mismo proceso visto anteriormente, con lo cual quedará ordenado otro elemento de la lista. Este proceso lo repetiremos hasta que quede ordenado por completo la lista.

Como debemos repetir el mismo algoritmo podemos englobar todo el bloque en otra estructura repetitiva.

Programa:

```
suellos=[]
for x in range(5):
    valor=int(input("Ingrese sueldo:"))
    sueldos.append(valor)

print("Lista sin ordenar")
print(sueldos)

for k in range(4):
    for x in range(4):
        if sueldos[x]>sueldos[x+1]:
            aux=sueldos[x]
            sueldos[x]=sueldos[x+1]
            sueldos[x+1]=aux

print("Lista ordenada")
print(sueldos)
```

Realicemos una prueba del siguiente algoritmo:

Cuando k = 0

x = 0	x = 1	x = 2	x = 3
750	750	750	750
1200	820	820	820
820	1200	550	550
550	550	1200	490
490	490	490	1200

Cuando k = 1

x = 0	x = 1	x = 2	x = 3
750	750	750	750
820	550	550	550
550	820	490	490
490	490	820	820
1200	1200	1200	1200

Cuando k = 2

x = 0	x = 1	x = 2	x = 3
550	550	550	550
750	490	490	490
490	750	750	750
820	820	820	820
1200	1200	1200	1200

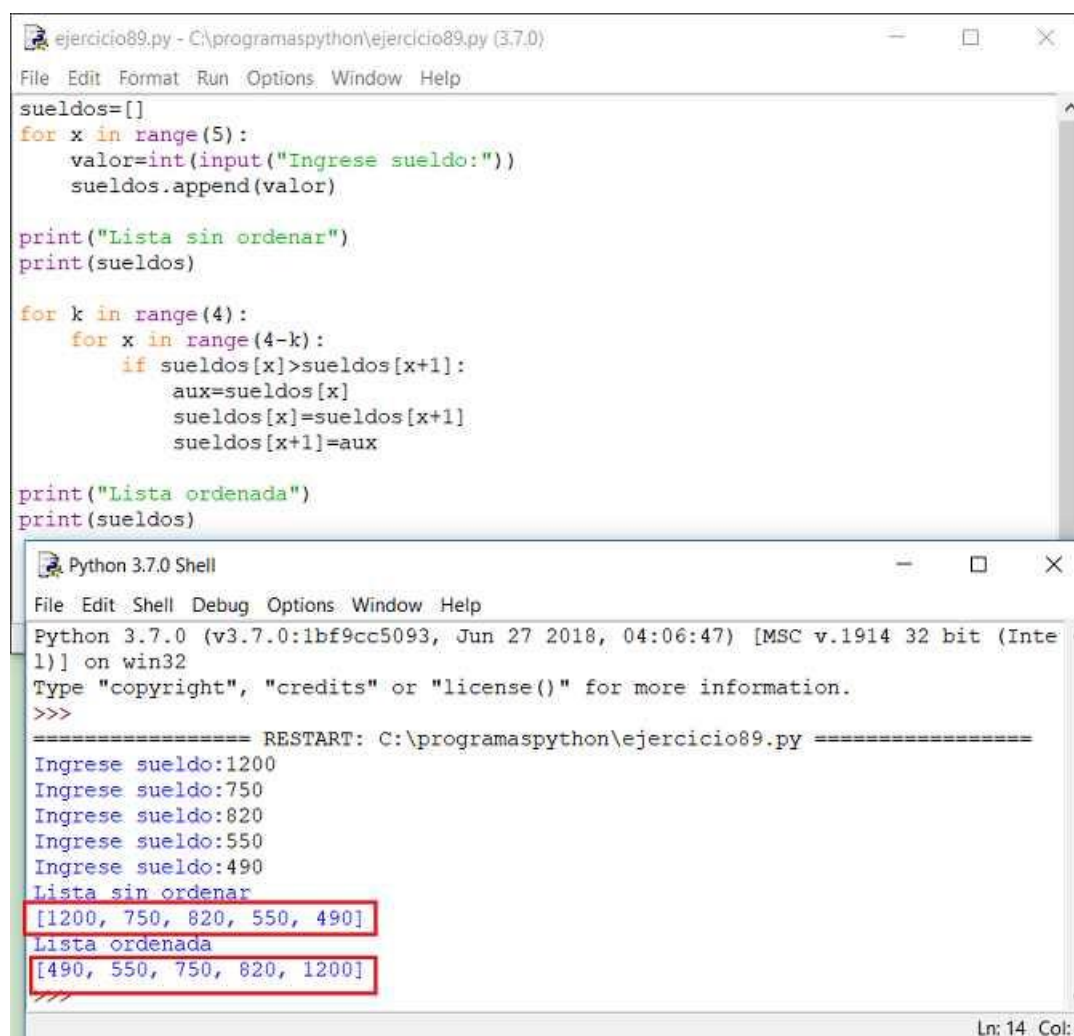
Cuando k = 3

x = 0	x = 1	x = 2	x = 3
490	490	490	490
550	550	550	550
750	750	750	750
820	820	820	820
1200	1200	1200	1200

¿Porque repetimos 4 veces el for externo?

Como sabemos cada vez que se repite en forma completa el for interno queda ordenada una componente de la lista. A primera vista diríamos que deberíamos repetir el for externo la cantidad de componentes de la lista, en este ejemplo la lista sueldos tiene 5 componentes.

Si observamos, cuando quedan dos elementos por ordenar, al ordenar uno de ellos queda el otro automáticamente ordenado (podemos imaginar que si tenemos una lista con 2 elementos no se requiere el for externo, porque este debería repetirse una única vez)



The image shows a screenshot of a Python IDE window titled 'ejercicio89.py - C:\programaspython\ejercicio89.py (3.7.0)'. The code defines a list 'sueldos' and uses nested loops to implement a bubble sort. The first loop iterates 5 times, and the second loop iterates 4 times, with the range decreasing by 1 in each iteration. The output shows the list before and after sorting.

```
ejercicio89.py - C:\programaspython\ejercicio89.py (3.7.0)
File Edit Format Run Options Window Help
sueldos=[]
for x in range(5):
    valor=int(input("Ingrese sueldo:"))
    sueldos.append(valor)

print("Lista sin ordenar")
print(sueldos)

for k in range(4):
    for x in range(4-k):
        if sueldos[x]>sueldos[x+1]:
            aux=sueldos[x]
            sueldos[x]=sueldos[x+1]
            sueldos[x+1]=aux

print("Lista ordenada")
print(sueldos)
```

Python 3.7.0 Shell

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio89.py =====
Ingrese sueldo:1200
Ingrese sueldo:750
Ingrese sueldo:820
Ingrese sueldo:550
Ingrese sueldo:490
Lista sin ordenar
[1200, 750, 820, 550, 490]
Lista ordenada
[490, 550, 750, 820, 1200]
```

Ln: 14 Col: 4

Una última consideración a este ALGORITMO de ordenamiento es que los elementos que se van ordenando continuamos comparándolos.

Ejemplo: En la primera ejecución del for interno el valor 1200 queda ubicado en la posición 4 de la lista. En la segunda ejecución comparamos si el 820 es mayor a 1200, lo cual seguramente será falso.

Podemos concluir que la primera vez debemos hacer para este ejemplo 4 comparaciones, en la segunda ejecución del for interno debemos hacer 3 comparaciones y en general debemos ir reduciendo en uno la cantidad de comparaciones.

Si bien el algoritmo planteado funciona, un algoritmo más eficiente, que se deriva del anterior es el plantear un for interno con la siguiente estructura:

```
for k in range(4):
    for x in range(4-k):
        if sueldos[x]>sueldos[x+1]:
            aux=sueldos[x]
            sueldos[x]=sueldos[x+1]
            sueldos[x+1]=aux
```

Es decir restarle el valor del contador del for externo.

Problema propuesto

- 1 - Crear una lista y almacenar los nombres de 5 países. Ordenar alfabéticamente la lista e imprimirla.
- 2 - Solicitar por teclado la cantidad de empleados que tiene la empresa. Crear y cargar una lista con todos los sueldos de dichos empleados. Imprimir la lista de sueldos ordenados de menor a mayor.
- 3 - Cargar una lista con 5 elementos enteros. Ordenarla de menor a mayor y mostrarla por pantalla, luego ordenar de mayor a menor e imprimir nuevamente.

Solución a los problemas

```
países=[]
for x in range(5):
    nom=input("Ingrese el nombre de país:")
    países.append(nom)

for k in range(4):
    for x in range(4-k):
        if países[x]>países[x+1]:
            aux=países[x]
            países[x]=países[x+1]
            países[x+1]=aux

print("Listado de países")
print(países)
```

```
cantidad=int(input("Cuántos empleados tiene la empresa?"))
sueldos=[]
for x in range(cantidad):
    su=int(input("Ingrese sueldo:"))
    sueldos.append(su)

# ordenamos la lista
for k in range(cantidad-1):
    for x in range(cantidad-1-k):
        if sueldos[x]>sueldos[x+1]:
            aux=sueldos[x]
            sueldos[x]=sueldos[x+1]
            sueldos[x+1]=aux

print("Lista de sueldos ordenados")
print(sueldos)
```

```
lista=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)

# ordenamos de menor a mayor
for k in range(4):
    for x in range(4-k):
        if lista[x]>lista[x+1]:
            aux=lista[x]
            lista[x]=lista[x+1]
            lista[x+1]=aux

print("Lista ordenada de menor a mayor")
print(lista)

# ordenamos de mayor a menor
for k in range(4):
    for x in range(4-k):
        if lista[x]<lista[x+1]:
            aux=lista[x]
            lista[x]=lista[x+1]
            lista[x+1]=aux
```

```
print("Lista ordenada de mayor a menor")  
print(lista)
```


Listas: ordenamiento con listas paralelas

Cuando se tienen listas paralelas y se ordenan los elementos de una de ellas hay que tener la precaución de intercambiar los elementos de las listas paralelas.

Problema 1:

Confeccionar un programa que permita cargar los nombres de 5 alumnos y sus notas respectivas. Luego ordenar las notas de mayor a menor. Imprimir las notas y los nombres de los alumnos.

Debe quedar claro que cuando intercambiamos las notas para dejarlas ordenadas de mayor a menor debemos intercambiar los nombres para que las listas continúen paralelas (es decir que en los mismos subíndices de cada lista continúe la información relacionada)

Programa:

```
alumnos=[]
notas=[]
for x in range(5):
    nom=input("Ingrese el nombre del alumno:")
    alumnos.append(nom)
    no=int(input("Ingrese la nota de dicho alumno:"))
    notas.append(no)

for k in range(4):
    for x in range(4-k):
        if notas[x]<notas[x+1]:
            aux1=notas[x]
            notas[x]=notas[x+1]
            notas[x+1]=aux1
            aux2=alumnos[x]
            alumnos[x]=alumnos[x+1]
            alumnos[x+1]=aux2

print("Lista de alumnos y sus notas ordenadas de mayor a menor")
for x in range(5):
    print(alumnos[x],notas[x])
```

Definimos y cargamos dos listas con cinco elementos cada una:

```
alumnos=[]
notas=[]
for x in range(5):
    nom=input("Ingrese el nombre del alumno:")
    alumnos.append(nom)
    no=int(input("Ingrese la nota de dicho alumno:"))
    notas.append(no)
```

Lo nuevo se presenta cuando ordenamos la lista de notas de mayor a menor. La condición dentro de los dos ciclos repetitivos depende de la lista notas, pero en el caso que se verifique verdadera intercambiamos tanto los elementos de la lista notas como el de la lista alumnos con el fin que continúen paralelas:

```
for k in range(4):
```

```
for x in range(4-k):
    if notas[x]<notas[x+1]:
        aux1=notas[x]
        notas[x]=notas[x+1]
        notas[x+1]=aux1
        aux2=alumnos[x]
        alumnos[x]=alumnos[x+1]
        alumnos[x+1]=aux2
```

Imprimimos las dos listas:

```
for x in range(5):
    print(alumnos[x],notas[x])
```

Algo que no habíamos utilizado en Python hasta ahora es imprimir varios datos en la misma línea, esto se logra pasando más de un parámetro a la función print separándolos por una coma:

```
print(alumnos[x],notas[x])
```

El resultado de ejecutar este programa:

```
ejercicio93.py - C:\programaspython\ejercicio93.py (3.7.0)
File Edit Format Run Options Window Help

alumnos=[]
notas=[]
for x in range(5):
    nom=input("Ingrese el nombre del alumno:")
    alumnos.append(nom)
    no=int(input("Ingrese la nota de dicho alumno:"))
    notas.append(no)

for k in range(4):
    for x in range(4-k):
        if notas[x]<notas[x+1]:
            aux1=notas[x]
            notas[x]=notas[x+1]
            notas[x+1]=aux1
            aux2=alumnos[x]
            alumnos[x]=alumnos[x+1]
            alumnos[x+1]=aux2

print("Lista de alumnos y sus notas ordenadas de mayor a menor")
for x in range(5):
    print(alumnos[x],notas[x])

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio93.py =====
Ingrese el nombre del alumno:juan
Ingrese la nota de dicho alumno:7
Ingrese el nombre del alumno:ana
Ingrese la nota de dicho alumno:10
Ingrese el nombre del alumno:carlos
Ingrese la nota de dicho alumno:4
Ingrese el nombre del alumno:maria
Ingrese la nota de dicho alumno:8
Ingrese el nombre del alumno:pablo
Ingrese la nota de dicho alumno:2
Lista de alumnos y sus notas ordenadas de mayor a menor
ana 10
maria 8
juan 7
carlos 4
pablo 2
>>> |
```

Problema propuesto

1 - Crear y cargar en un lista los nombres de 5 países y en otra lista paralela la cantidad de habitantes del mismo. Ordenar alfabéticamente e imprimir los resultados. Por último ordenar con respecto a la cantidad de habitantes (de mayor a menor) e imprimir nuevamente.

Solución a los problemas

```
países=[]
habitantes=[]
for x in range(5):
    nom=input("Ingrese el nombre del país:")
    países.append(nom)
    cant=int(input("Cantidad de habitantes"))
    habitantes.append(cant)

# ordenamiento alfabético
for k in range(4):
    for x in range(4-k):
        if países[x]>países[x+1]:
            aux1=países[x]
            países[x]=países[x+1]
            países[x+1]=aux1
            aux2=habitantes[x]
            habitantes[x]=habitantes[x+1]
            habitantes[x+1]=aux2

print("Listado de países en orden alfabético")
for x in range(5):
    print(países[x],habitantes[x])

# ordenamiento por cantidad de habitantes
for k in range(4):
    for x in range(4-k):
        if habitantes[x]<habitantes[x+1]:
            aux1=países[x]
            países[x]=países[x+1]
            países[x+1]=aux1
            aux2=habitantes[x]
            habitantes[x]=habitantes[x+1]
            habitantes[x+1]=aux2

print("Listado de países por cantidad de habitantes")
for x in range(5):
    print(países[x],habitantes[x])
```

Listas: componentes de tipo lista

Hasta ahora hemos trabajado con listas cuyos componentes son de tipo:

enteros

flotantes

cadenas de caracteres

Ejemplo

```
notas=[8, 6, 8]
alturas=[1.73, 1.55, 1.92]
dias=["lunes", "martes", "miércoles"]
```

Pero lo que la hace tan flexible a esta estructura de datos es que podemos almacenar componentes de tipo LISTA.

```
notas=[[4,5], [6,9], [7,3]]
```

En la línea anterior hemos definido una lista de tres elementos de tipo lista, el primer elemento de la lista es otra lista de dos elementos de tipo entero. De forma similar los otros dos elementos de la lista notas son listas de dos elementos de tipo entero.

Problema 1:

Crear una lista por asignación. La lista tiene que tener cuatro elementos. Cada elemento debe ser una lista de 3 enteros.


Imprimir sus elementos accediendo de diferentes modos.

Programa:

```
lista=[[1,2,3], [4,5,6], [7,8,9], [10,11,12]]

# imprimimos la lista completa
print(lista)
print("-----")
# imprimimos la primer componente
print(lista[0])
print("-----")
# imprimimos la primer componente de la lista contenida
# en la primer componente de la lista principal
print(lista[0][0])
print("-----")
# imprimimos con un for la lista contenida en la primer componente
for x in range(len(lista[0])):
    print(lista[0][x])
print("-----")
# imprimimos cada elemento entero de cada lista contenida en la lista
for k in range(len(lista)):
    for x in range(len(lista[k])):
        print(lista[k][x])
```

El resultado de ejecutar este programa es:



The screenshot shows a Python IDE with two windows. The top window, titled 'ejercicio95.py - C:\programaspython\ejercicio95.py (3.7.0)', contains the following code:

```
lista=[[1,2,3], [4,5,6], [7,8,9], [10,11,12]]

# imprimimos la lista completa
print(lista)
print("-----")
# imprimimos la primer componente
print(lista[0])
print("-----")
# imprimimos la primer componente de la lista contenida
# en la primer componente de la lista principal
print(lista[0][0])
print("-----")
# imorimimos con un for la lista contenida en la primer componente
```

The bottom window, titled 'Python 3.7.0 Shell', shows the execution output:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio95.py =====
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
-----
[1, 2, 3]
-----
1
-----
1
2
3
-----
1
2
3
4
5
6
7
8
9
10
11
12
>>> |
```

The status bar at the bottom right indicates 'Ln: 27 Col:'.

Al principio puede complicarse trabajar con listas de listas pero a medida que practiquemos esta estructura de datos veremos que podemos desarrollar algoritmos más complejos.

Para definir y crear por asignación una lista de listas tenemos:

```
lista=[[1,2,3], [4,5,6], [7,8,9], [10,11,12]]
```

Queda claro que el primer elemento de lista es:

```
[1,2,3]
```

El segundo elemento de la variable lista es (y así sucesivamente):

```
[4,5,6]
```

La función `print` si le pasamos como parámetro el nombre de la lista nos muestra la lista completa por pantalla:

```
print(lista)
```

Aparece:

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

Cuando pasamos a la función `print` el primer elemento de la lista:

```
print(lista[0])
```

Nos muestra la lista contenida en la primer componente de la lista principal:

```
[1, 2, 3]
```

Si queremos acceder al primer entero almacenado en la lista contenida en la primer componente de la lista principal:

```
print(lista[0][0])
```

Nos muestra:

```
1
```

Para acceder mediante un `for` a todos los elementos de la lista contenida en la primer componente de la lista principal debemos codificar:

```
for x in range(len(lista[0])):
    print(lista[0][x])
```

Recordemos que la función `len` retorna la cantidad de elementos que contiene una lista. En este caso le pasamos como parámetro `lista[0]` que hace referencia a la primer componente de la lista principal.

El resultado de `len(lista[0])` es un 3 que es la cantidad de elementos que tiene la lista contenida en la primer componente de la lista principal.

Cada ciclo del `for` accedemos a: `lista[0][0]` cuando `x` vale 0, `lista[0][1]` cuando `x` vale 1 y `lista[0][2]` cuando `x` vale 2.

Mediante este ciclo podemos acceder a cada elemento y procesarlo.

Por último con el ciclo anidado `k` podemos acceder a cada elemento de la lista principal y mediante el `for` interno acceder a cada elemento entero de las listas contenidas en la lista principal:

```
for k in range(len(lista)):
    for x in range(len(lista[k])):
        print(lista[k][x])
```

Problema 2:

Crear una lista por asignación. La lista tiene que tener 2 elementos. Cada elemento debe ser una lista de 5 enteros.

Calcular y mostrar la suma de cada lista contenida en la lista principal.

Programa:

```
lista=[[1,1,1,1,1], [2,2,2,2,2]]

suma1=lista[0][0]+lista[0][1]+lista[0][2]+lista[0][3]+lista[0][4]
print(suma1)
suma2=lista[1][0]+lista[1][1]+lista[1][2]+lista[1][3]+lista[1][4]
print(suma2)
print("-----")

suma1=0
for x in range(len(lista[0])):
    suma1=suma1+lista[0][x]
suma2=0
for x in range(len(lista[1])):
    suma2=suma2+lista[1][x]
print(suma1)
print(suma2)
print("-----")

for k in range(len(lista)):
    suma=0
    for x in range(len(lista[k])):
        suma=suma+lista[k][x]
    print(suma)
```

Hemos resuelto el problema de tres formas.

La primer forma es acceder a cada elemento en forma secuencial, esto se resuelve de esta forma si tenemos que acceder a un pequeño número de elementos, es decir si la lista es pequeña:

```
suma1=lista[0][0]+lista[0][1]+lista[0][2]+lista[0][3]+lista[0][4]
print(suma1)
suma2=lista[1][0]+lista[1][1]+lista[1][2]+lista[1][3]+lista[1][4]
print(suma2)
```

La segunda forma es utilizar una estructura repetitiva para sumar todos los elementos de una lista (el primer subíndice siempre es 0 y el segundo varía con la variable x):

```
suma1=0
for x in range(len(lista[0])):
    suma1=suma1+lista[0][x]
suma2=0
for x in range(len(lista[1])):
    suma2=suma2+lista[1][x]
print(suma1)
print(suma2)
```

La última forma planteada es utilizar una estructura repetitiva anidada que suma cada fila, el for externo (k) se repite 2 veces que es el tamaño de la variable "lista":

```
for k in range(len(lista)):
    suma=0
    for x in range(len(lista[k])):
```



```
        suma=suma+lista[k][x]
    print(suma)
```

Es importante notar que el acumulador suma lo iniciamos en 0 cada vez que comenzamos a sumar una nueva lista.

Problema 3:

Crear una lista por asignación. La lista tiene que tener 5 elementos. Cada elemento debe ser una lista, la primera lista tiene que tener un elemento, la segunda dos elementos, la tercera tres elementos y así sucesivamente.

Sumar todos los valores de las listas.

Programa:

```
lista=[[1], [1,2], [1,2,3], [1,2,3,4], [1,2,3,4,5]]

suma=0
for k in range(len(lista)):
    for x in range(len(lista[k])):
        suma=suma+lista[k][x]
print(suma)
```

Lo primero que es importante notar que las listas contenidas en la lista principal no tienen porque ser del mismo tamaño.

La forma más sencilla es utilizar dos ciclos repetitivos. El primero se repite tantas veces como elementos tenga la lista principal:

```
for k in range(len(lista)):
```

El segundo ciclo nos sirve para recorrer y acceder a cada elemento entero de cada lista:

```
    for x in range(len(lista[k])):
        suma=suma+lista[k][x]
```

La cantidad de veces que se repite el for interno depende de la cantidad de elementos que tiene la lista que estamos sumando en ese momento.

Problemas propuestos

1 - Se tiene la siguiente lista:

```
lista=[[100,7,85,8], [4,8,56,25], [67,89,23,1], [78,56]]
```

Imprimir la lista. Luego fijar con el valor cero todos los elementos mayores a 50 del primer elemento de "lista".

Volver a imprimir la lista.

2 - Se tiene la siguiente lista:

```
lista=[[4,12,5,66], [14,6,25], [3,4,5,67,89,23,1], [78,56]]
```

Imprimir la lista. Luego fijar con el valor cero todos los elementos mayores a 10 contenidos en el primer elemento de la variable "lista".

Volver a imprimir la lista.

3 - Crear una lista por asignación con la cantidad de elementos de tipo lista que usted desee. Luego imprimir el último elemento de la lista principal.

Solución a los problemas

```
lista=[[100,7,85,8], [4,8,56,25], [67,89,23,1], [78,56]]

print(lista)

for x in range(len(lista[0])):
    if lista[0][x]>50:
        lista[0][x]=0

print(lista)
```

```
lista=[[4,12,5,66], [14,6,25], [3,4,5,67,89,23,1], [78,56]]

print(lista)

for k in range(len(lista)):
    for x in range(len(lista[k])):
        if lista[k][x]>10:
            lista[k][x]=0

print(lista)
```

```
lista=[["juan","ana"], ["luis"], ["pedro","carlos","maria"]]
print(lista[len(lista)-1])
```

Listas: carga por teclado de componentes de tipo lista

En el concepto anterior vimos que fácilmente podemos definir por asignación una lista cuyas componentes sean también listas:

```
lista=[[1,2,3], [7,4], [9,2]]
```

En muchas situaciones debemos crear una nueva lista ingresando los datos por teclado o por operaciones del mismo programa.

Problema 1:

Crear y cargar una lista con los nombres de tres alumnos. Cada alumno tiene dos notas, almacenar las notas en una lista paralela. Cada componente de la lista paralela debe ser también una lista con las dos notas. Imprimir luego cada nombre y sus dos notas.

Si cargáramos los datos por asignación sería algo parecido a esto:

```
alumnos=["juan", "ana", "luis"]
notas=[[10,8], [6,5], [2,8]]
```

En la componente 0 de la lista alumnos tenemos almacenado el nombre "juan" y como son listas paralelas en la componente 0 de la lista notas tenemos almacenado una lista con las dos notas 10 y 8.

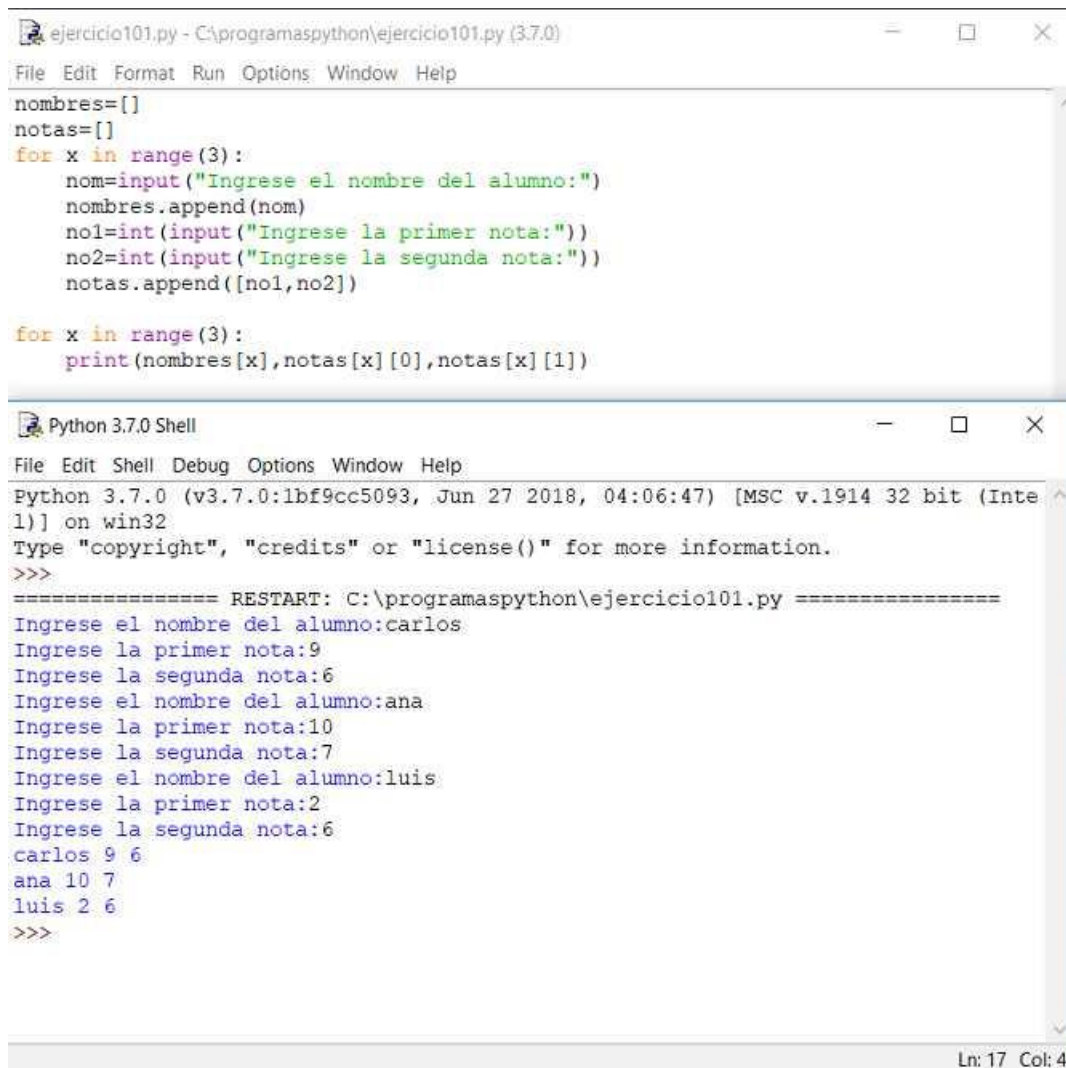
Nuestro objetivo como lo pide el problema es cargar los datos por teclado.

Programa:

```
nombres=[]
notas=[]
for x in range(3):
    nom=input("Ingrese el nombre del alumno:")
    nombres.append(nom)
    no1=int(input("Ingrese la primer nota:"))
    no2=int(input("Ingrese la segunda nota:"))
    notas.append([no1,no2])

for x in range(3):
    print(nombres[x],notas[x][0],notas[x][1])
```

La ejecución del programa tiene una salida por pantalla similar a:

The image shows a screenshot of a Python IDE with two windows. The top window, titled 'ejercicio101.py - C:\programaspython\ejercicio101.py (3.7.0)', contains the following Python code:

```
nombres=[]
notas=[]
for x in range(3):
    nom=input("Ingrese el nombre del alumno:")
    nombres.append(nom)
    no1=int(input("Ingrese la primer nota:"))
    no2=int(input("Ingrese la segunda nota:"))
    notas.append([no1,no2])

for x in range(3):
    print(nombres[x],notas[x][0],notas[x][1])
```

The bottom window, titled 'Python 3.7.0 Shell', shows the execution of the script. It displays the Python version and architecture, followed by a restart message and the program's output:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio101.py =====
Ingrese el nombre del alumno:carlos
Ingrese la primer nota:9
Ingrese la segunda nota:6
Ingrese el nombre del alumno:ana
Ingrese la primer nota:10
Ingrese la segunda nota:7
Ingrese el nombre del alumno:luis
Ingrese la primer nota:2
Ingrese la segunda nota:6
carlos 9 6
ana 10 7
luis 2 6
>>>
```

La creación de las dos listas no difiere una de otra:

```
nombres=[]
notas=[]
```

En la estructura repetitiva for procedemos a cargar un nombre y agregarlo a la lista en forma similar como lo hemos hecho en conceptos anteriores:

```
for x in range(3):
    nom=input("Ingrese el nombre del alumno:")
    nombres.append(nom)
```

Lo nuevo se presenta cuando queremos añadir elementos a la lista notas, lo hacemos con el mismo método append pero le pasamos como parámetro una lista con dos elementos:

```
no1=int(input("Ingrese la primer nota:"))
no2=int(input("Ingrese la segunda nota:"))
notas.append([no1,no2])
```

Cuando imprimimos el nombre lo accedemos por un solo subíndice, pero para acceder a cada una de las notas debemos indicar dos subíndices, el primer subíndice es con respecto a que

elemento accedemos de la lista principal y el segundo subíndice hace referencia a la lista contenida en dicha componente:

```
for x in range(3):  
    print(nombres[x], notas[x][0], notas[x][1])
```

Problema 2:

Se tiene que cargar la siguiente información:

- Nombres de 3 empleados
- Ingresos en concepto de sueldo, cobrado por cada empleado, en los últimos 3 meses.

Confeccionar el programa para:

- a) Realizar la carga de los nombres de empleados y los tres sueldos por cada empleado.
- b) Generar una lista que contenga el ingreso acumulado en sueldos en los últimos 3 meses para cada empleado.
- c) Mostrar por pantalla el total pagado en sueldos a cada empleado en los últimos 3 meses
- d) Obtener el nombre del empleado que tuvo el mayor ingreso acumulado

En este problema definiremos tres listas:

Una lista para almacenar los nombres de los empleados, por ejemplo si lo cargamos por asignación:

```
nombres=["juan", "ana", "luis"]
```

Una lista paralela a la anterior para almacenar los sueldos en los últimos tres meses por cada empleado:

```
sueldos=[[5000,5100,5000], [7000,6500,6000], [2500,2500,2500]]
```

Otra lista paralela donde almacenamos el total de sueldos cobrados por cada empleado que resulta de sumar los tres sueldos de cada empleado contenidos en la lista sueldos:

```
totalsueldos=[15100, 19500, 7500]
```

Es importante notar que estos datos no los cargaremos por asignación sino los cargaremos por teclado a las dos primeras listas y la tercera la generaremos extrayendo los datos de la lista sueldos.

Programa:

```
nombres=[]  
sueldos=[]  
totalsueldos=[]  
  
for x in range(3):  
    nom=input("Ingrese el nombre del empleado:")  
    nombres.append(nom)  
    su1=int(input("Ingrese el primer sueldo:"))  
    su2=int(input("Ingrese el segundo sueldo:"))  
    su3=int(input("Ingrese el tercer sueldo:"))  
    sueldos.append([su1, su2, su3])  
  
for x in range(3):
```

```

        total=sueldos[x][0]+sueldos[x][1]+sueldos[x][2]
        totalsueldos.append(total)

for x in range(3):
    print(nombres[x], totalsueldos[x])

posmayor=0
mayor=totalsueldos[0]
for x in range(1,3):
    if totalsueldos[x]>mayor:
        mayor=totalsueldos[x]
        posmayor=x
print("Empleado con mayores ingresos en los ultimos tres meses")
print(nombres[posmayor])
print("con un ingreso de",mayor)

```

Definimos 3 listas:

```

nombres=[]
sueldos=[]
totalsueldos=[]

```

En el primer for realizamos la carga de cada nombre de empleado y sus tres últimos sueldos, en la lista nombres añadimos strings y en la lista sueldos añadimos otra lista con tres enteros que representan los sueldos:

```

for x in range(3):
    nom=input("Ingrese el nombre del empleado:")
    nombres.append(nom)
    su1=int(input("Ingrese el primer sueldo:"))
    su2=int(input("Ingrese el segundo sueldo:"))
    su3=int(input("Ingrese el tercer sueldo:"))
    sueldos.append([su1, su2, su3])

```

En el segundo ciclo repetitivo generamos automáticamente la lista totalsueldos (es decir no cargamos por teclado), este valor resulta de sumar los tres sueldos de cada empleado que se encuentran en cada una de las listas contenidas en la lista sueldos:

```

for x in range(3):
    total=sueldos[x][0]+sueldos[x][1]+sueldos[x][2]
    totalsueldos.append(total)

```

Imprimimos ahora el nombre del empleado seguido por el ingreso total en sueldos de los últimos tres meses:

```

for x in range(3):
    print(nombres[x], totalsueldos[x])

```

Finalmente el problema requiere mostrar el nombre del empleado que recaudó más en sueldos en los últimos tres meses.

Iniciamos dos variables previas al for indicando en una la posición de importe en sueldos mayor y en otra dicho importe. Hacemos la suposición antes de ingresar al for que el mayor está en la posición 0:

```

posmayor=0
mayor=totalsueldos[0]

```

Disponemos un for cuya variable se inicia en 1 y dentro del for controlamos el importe del total de sueldos si supera al que hemos considerado mayor hasta ese momento "mayor" procedemos a actualizar las dos variables: mayor y posmayor:

```
for x in range(1,3):
    if totalsueldos[x]>mayor:
        mayor=totalsueldos[x]
        posmayor=x
```

Finalmente cuando salimos del ciclo procedemos a mostrar el nombre del empleado que recaudó más en los últimos tres meses:

```
print("Empleado con mayores ingresos en los ultimos tres meses")
print(nombres[posmayor])
print("con un ingreso de",mayor)
```

Problema 3:

Solicitar por teclado dos enteros. El primer valor indica la cantidad de elementos que crearemos en la lista. El segundo valor indica la cantidad de elementos que tendrá cada una de las listas internas a la lista principal.

Mostrar la lista y la suma de todos sus elementos.

Por ejemplo si el operador carga un 2 y un 4 significa que debemos crear una lista similar a:

```
lista=[[1,1,1,1], [1,1,1,1]]
```

Programa:

```
lista=[]
elementos=int(input("Cuantos elementos tendra la lista:"))
sub=int(input("Cuantos elementos tendran las listas internas:"))
for k in range(elementos):
    lista.append([])
    for x in range(sub):
        valor=int(input("Ingrese valor:"))
        lista[k].append(valor)

print(lista)

suma=0
for k in range(len(lista)):
    for x in range(len(lista[k])):
        suma=suma+lista[k][x]

print("La suma de todos sus elementos:",suma)
```

Lo primero que hacemos en este problema además de definir la lista es cargar dos enteros por teclado:

```
lista=[]
elementos=int(input("Cuantos elementos tendra la lista:"))
sub=int(input("Cuantos elementos tendran las listas internas:"))
```


El primer for se repetirá tantas veces como indica el primer valor ingresado por teclado almacenado en la variable "elementos", cada vuelta de este for se crea un elemento en la "lista" y se carga una lista vacía []:

```
for k in range(elementos):  
    lista.append([])
```

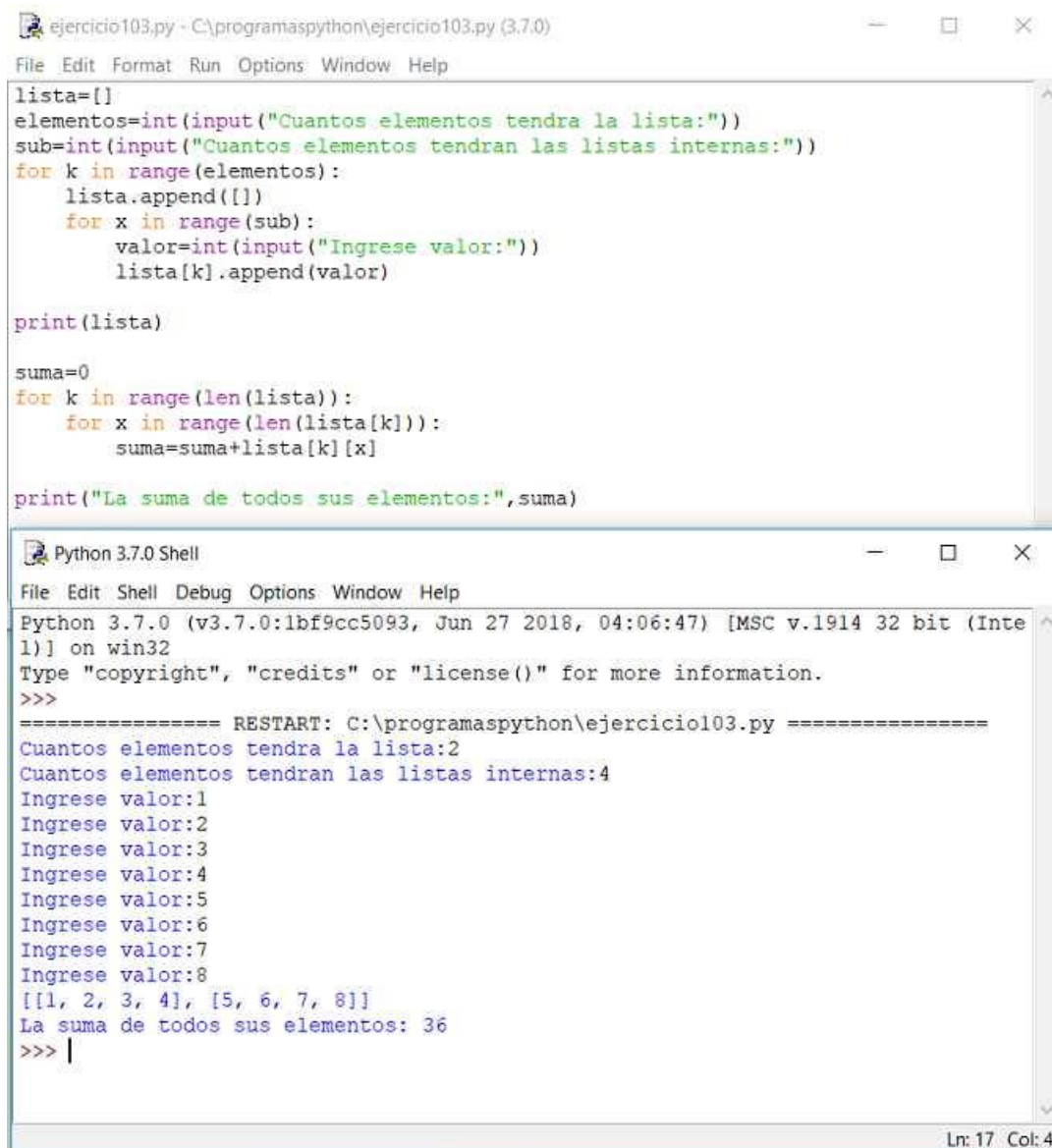
En el for interior procedemos a cargar tantos valores como lo indicamos en la variable "sub" y los vamos añadiendo en la lista vacía que creamos antes de este for:

```
for x in range(sub):  
    valor=int(input("Ingrese valor:"))  
    lista[k].append(valor)
```

Finalmente para sumar todos los elementos enteros almacenados en "lista" debemos disponer estructuras repetitivas anidadas:

```
suma=0  
for k in range(len(lista)):  
    for x in range(len(lista[k])):  
        suma=suma+lista[k][x]
```

El for de las "k" se repite tantas veces como elementos tenga "lista" y el for de las x se repite tantas veces como elementos tenga la lista en la posición k.



The image shows a screenshot of a Python IDE with two windows. The top window, titled 'ejercicio103.py - C:\programaspython\ejercicio103.py (3.7.0)', contains the following Python code:

```
lista=[]
elementos=int(input("Cuantos elementos tendra la lista:"))
sub=int(input("Cuantos elementos tendran las listas internas:"))
for k in range(elementos):
    lista.append([])
    for x in range(sub):
        valor=int(input("Ingrese valor:"))
        lista[k].append(valor)

print(lista)

suma=0
for k in range(len(lista)):
    for x in range(len(lista[k])):
        suma=suma+lista[k][x]

print("La suma de todos sus elementos:",suma)
```

The bottom window, titled 'Python 3.7.0 Shell', shows the execution of the script. It displays the prompts and user inputs, followed by the output of the program:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio103.py =====
Cuantos elementos tendra la lista:2
Cuantos elementos tendran las listas internas:4
Ingrese valor:1
Ingrese valor:2
Ingrese valor:3
Ingrese valor:4
Ingrese valor:5
Ingrese valor:6
Ingrese valor:7
Ingrese valor:8
[[1, 2, 3, 4], [5, 6, 7, 8]]
La suma de todos sus elementos: 36
>>> |
```

The status bar at the bottom right of the shell window indicates 'Ln: 17 Col: 4'.

Problema 4:

Definir dos listas de 3 elementos.

La primer lista cada elemento es una sublista con el nombre del padre y la madre de una familia. La segunda lista está constituida por listas con los nombres de los hijos de cada familia. Puede haber familias sin hijos.

Imprimir los nombres del padre, la madre y sus hijos.

También imprimir solo el nombre del padre y la cantidad de hijos que tiene dicho padre.

Un ejemplo si se define por asignación:

```
padres=[["juan","ana"], ["carlos","maria"], ["pedro","laura"]]
hijos=[["marcos","alberto","silvia"], [], ["oscar"]]
```

Como son listas paralelas podemos decir que la familia cuyos padres son "juan" y "ana" tiene tres hijos llamados "marcos", "alberto", "silvia". La segunda familia no tiene hijos y la tercera tiene solo uno.

Programa:

```
padres=[]
hijos=[]
for k in range(3):
    pa=input("Ingrese el nombre del padre:")
    ma=input("Ingrese el nombre de la madre:")
    padres.append([pa, ma])
    cant=int(input("Cuantos hijos tienen esta familia:"))
    hijos.append([])
    for x in range(cant):
        nom=input("Ingrese el nombre del hijo:")
        hijos[k].append(nom)

print("Listado del padre, madre y sus hijos")
for k in range(3):
    print("Padre:",padres[k][0])
    print("Madre:",padres[k][1])
    for x in range(len(hijos[k])):
        print("Hijo:",hijos[k][x])

print("Listado del padre y cantidad de hijos que tiene")
for x in range(3):
    print("padre:",padres[x][0])
    print("Cantidad de hijos:",len(hijos[x]))
```

Comenzamos definiendo las dos listas:

```
padres=[]
hijos=[]
```

El primer for es para cargar y crear cada elemento de la lista "padres", crear una elemento de la lista hijos con una lista vacía y solicitar que se cargue cuantos hijos tiene la familia:

```
for k in range(3):
    pa=input("Ingrese el nombre del padre:")
    ma=input("Ingrese el nombre de la madre:")
    padres.append([pa, ma])
    cant=int(input("Cuantos hijos tienen esta familia:"))
    hijos.append([])
```

El for interno se ingresan los nombres de los hijos y se agregan a la lista hijos en la posición respectiva. El for interno puede llegar a no ejecutarse si se ingresa un 0 en cantidad de hijos:

```
    for x in range(cant):
        nom=input("Ingrese el nombre del hijo:")
        hijos[k].append(nom)
```

Para imprimir los nombres de ambos padres y sus hijos también implementamos un for anidado:

```
print("Listado del padre, madre y sus hijos")
for k in range(3):
    print("Padre:",padres[k][0])
    print("Madre:",padres[k][1])
    for x in range(len(hijos[k])):
        print("Hijo:",hijos[k][x])
```

Para mostrar la cantidad de hijos que tiene un determinado padre llamamos a la función len de cada una de las sublistas contenidas en la lista hijos:

```
print("Listado del padre y cantidad de hijos que tiene")
for x in range(3):
    print("padre:",padres[x][0])
    print("Cantidad de hijos:",len(hijos[x]))
```

Problemas propuestos

1 - Se desea saber la temperatura media trimestral de cuatro países. Para ello se tiene como dato las temperaturas medias mensuales de dichos países. Se debe ingresar el nombre del país y seguidamente las tres temperaturas medias mensuales.

Seleccionar las estructuras de datos adecuadas para el almacenamiento de los datos en memoria.

a - Cargar por teclado los nombres de los países y las temperaturas medias mensuales.

b - Imprimir los nombres de los países y las temperaturas medias mensuales de las mismas.

c - Calcular la temperatura media trimestral de cada país.

c - Imprimir los nombres de los países y las temperaturas medias trimestrales.

b - Imprimir el nombre del país con la temperatura media trimestral mayor.

2 - Definir una lista y almacenar los nombres de 3 empleados.

Por otro lado definir otra lista y almacenar en cada elemento una sublista con los números de días del mes que el empleado faltó.

Imprimir los nombres de empleados y los días que faltó.

Mostrar los empleados con la cantidad de inasistencias.

Finalmente mostrar el nombre o los nombres de empleados que faltaron menos días.

3 - Desarrollar un programa que cree una lista de 50 elementos. El primer elemento es una lista con un elemento entero, el segundo elemento es una lista de dos elementos etc.

La lista debería tener esta estructura y asignarle esos valores a medida que se crean los elementos:

```
[ [1], [1,2], [1,2,3], [1,2,3,4], [1,2,3,4,5], etc.... ]
```

Solución a los problemas

```
países=[]
temperaturas=[]
promediotemp=[]

for x in range(4):
    nom=input("Ingrese el nombre del país:")
    países.append(nom)
    temp1=int(input("Ingrese primer temperatura:"))
    temp2=int(input("Ingrese segunda temperatura:"))
    temp3=int(input("Ingrese tercer temperatura:"))
    temperaturas.append([temp1, temp2, temp3])

print("Países y temperaturas medias de los últimos tres meses mensuales")
for x in range(4):
    print(países[x], temperaturas[x][0], temperaturas[x][1], temperaturas[x][2])

for x in range(4):
    pro=(temperaturas[x][0]+temperaturas[x][1]+temperaturas[x][2])/3
    promediotemp.append(pro)

print("Países y temperaturas medias trimestrales")
for x in range(4):
    print(países[x], promediotemp[x])

posmayor=0
for x in range(1,4):
    if promediotemp[x]>promediotemp[posmayor]:
        posmayor=x
print("País con temperatura media trimestral mayor:", países[posmayor])

empleados=[]
faltas=[]

for k in range(3):
    nom=input("Ingrese el nombre de empleado:")
    empleados.append(nom)
    cant=int(input("Cuantos días faltó:"))
    faltas.append([])
    for x in range(cant):
        dia=int(input("Ingrese el número de día que faltó:"))
        faltas[k].append(dia)

print("Nombres de empleados y días que faltó")
for k in range(3):
    print(empleados[k])
    for x in range(len(faltas[k])):
        print(faltas[k][x])

print("Nombres de empleados y cantidad de inasistencias")
for x in range(3):
    print(empleados[x], len(faltas[x]))

men=len(faltas[0])
for x in range(1,3):
    if len(faltas[x])<men:
        men=len(faltas[x])

print("Empleado o empleados que faltaron menos")
for x in range(3):
    if len(faltas[x])==men:
```

```
print(empleados[x])
```

```
lista=[]  
cant=1  
for k in range(50):  
    lista.append([])  
    valor=1  
    for x in range(cant):  
        lista[k].append(valor)  
        valor=valor+1  
    cant=cant+1  
  
print(lista)
```

Listas: eliminación de elementos

Hemos visto que una lista la podemos iniciar por asignación indicando sus elementos.

```
lista=[10, 20, 30, 40]
```

También podemos agregarle elementos al final mediante el método `append`:

```
lista.append(120)
```

Si ahora imprimimos la lista tenemos como resultado:

```
[10, 20, 30, 40, 120]
```

Otra característica fundamental de las listas en Python es que podemos eliminar cualquiera de sus componentes llamando al método `pop` e indicando la posición del elemento a borrar:

```
lista.pop(0)
```

Ahora si imprimimos la lista luego de eliminar el primer elemento el resultado es:

```
[20, 30, 40, 120]
```

Otra cosa que hay que hacer notar que cuando un elemento de la lista se elimina no queda una posición vacía, sino se desplazan todos los elementos de la derecha una posición.

El método `pop` retorna el valor almacenado en la lista en la posición indicada, aparte de borrarlo.

```
lista=[10, 20, 30, 40]
print(lista.pop(0)) # imprime un 10
```

Problema 1:

Crear una lista por asignación con 5 enteros. Eliminar el primero, el tercero y el último de la lista.

Programa:

```
lista=[10, 20, 30, 40, 50]

print(lista)

lista.pop(0)
lista.pop(1)
lista.pop(2)

print(lista)
```

Parecería que con esas tres llamadas al método `pop` se eliminan los tres primeros elementos pero no es así, si imprimimos cada vez que borramos uno veremos que estamos borrando el primero, tercero y quinto.

```
lista=[10, 20, 30, 40, 50]
```

```

print(lista)
# se imprime [10, 20, 30, 40, 50]
lista.pop(0)
print(lista)
# se imprime [20, 30, 40, 50]
lista.pop(1)
print(lista)
# se imprime [20, 40, 50]
lista.pop(2)
# se imprime [20, 40]
print(lista)

```

Problema 2:

Crear una lista y almacenar 10 enteros pedidos por teclado. Eliminar todos los elementos que sean iguales al número entero 5.

Programa:

```

lista=[]
for x in range(10):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)

print(lista)

posicion=0
while posicion<len(lista):
    if lista[posicion]==5:
        lista.pop(posicion)
    else:
        posicion=posicion+1

print(lista)

```

Mediante un for cargamos 10 elementos en la lista:

```

lista=[]
for x in range(10):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)

```

Como es posible que se eliminen 0, 1, 2 etc. elementos de la lista utilizamos un ciclo while (no podemos usar un for, ya que el contador del mismo no indicará correctamente el subíndice a analizar)

Llevamos un contador llamado "posicion" que nos indica que elemento de la lista estamos verificando en el if, en el caso que se debe borrar llamamos al método pop pasando el contador y no incrementamos en uno el contador "posicion" ya que los elementos de la derecha se desplazan una posición a izquierda.

En el caso que no se debe borrar se incrementa en uno el contador "posicion" para analizar el siguiente elemento de la lista en la próxima vuelta del ciclo:

```

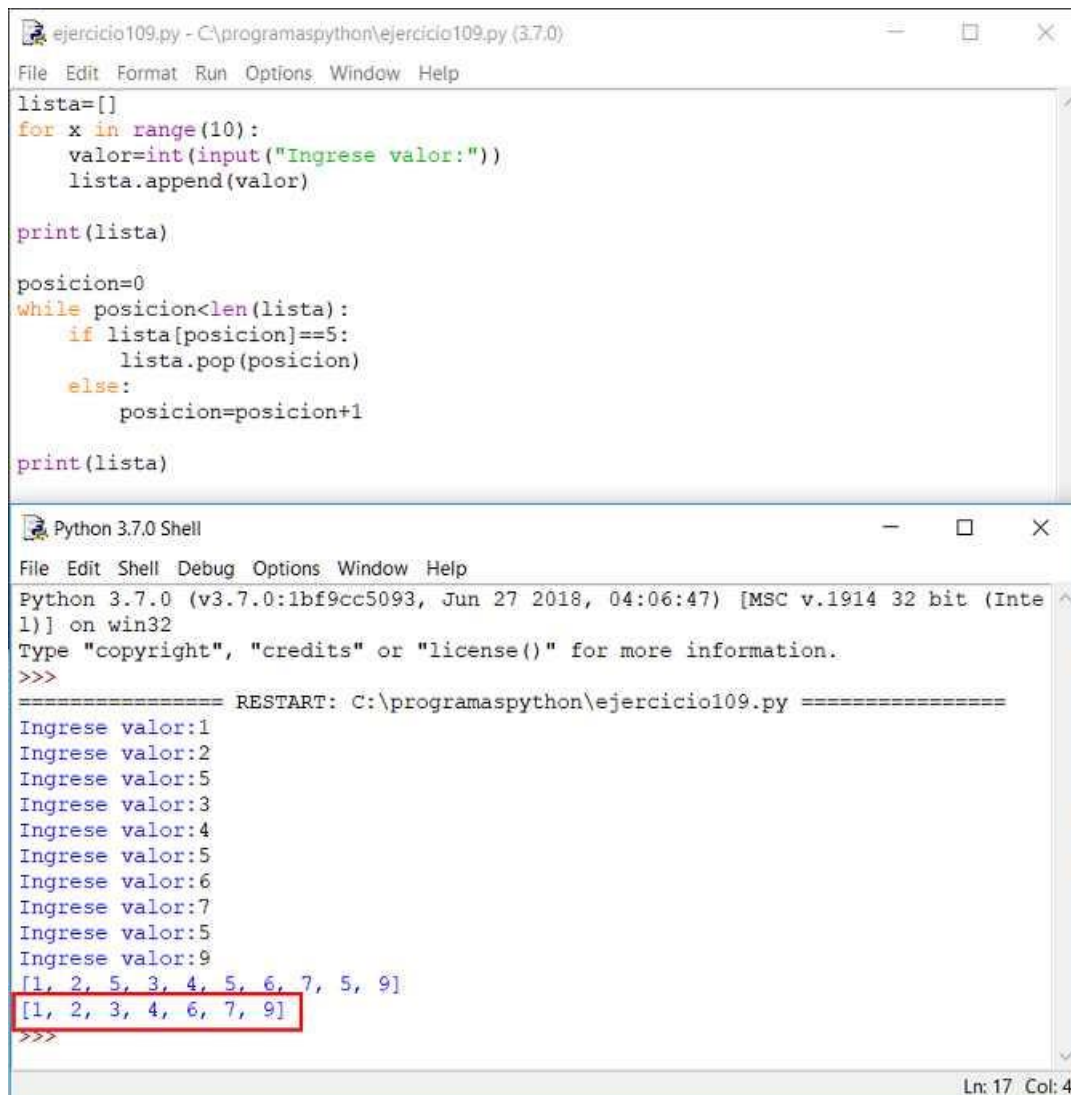
posicion=0
while posicion<len(lista):
    if lista[posicion]==5:
        lista.pop(posicion)
    else:

```



```
posicion=posicion+1
```

Si ejecutamos y cargamos tres cincos en distintas posiciones de la lista tenemos como resultado:



```
ejercicio109.py - C:\programaspython\ejercicio109.py (3.7.0)
File Edit Format Run Options Window Help

lista=[]
for x in range(10):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)

print(lista)

posicion=0
while posicion<len(lista):
    if lista[posicion]==5:
        lista.pop(posicion)
    else:
        posicion=posicion+1

print(lista)

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\programaspython\ejercicio109.py =====
Ingrese valor:1
Ingrese valor:2
Ingrese valor:5
Ingrese valor:3
Ingrese valor:4
Ingrese valor:5
Ingrese valor:6
Ingrese valor:7
Ingrese valor:5
Ingrese valor:9
[1, 2, 5, 3, 4, 5, 6, 7, 5, 9]
[1, 2, 3, 4, 6, 7, 9]
>>>
```

Acotación: otra forma de eliminar elementos de una lista

Para eliminar elementos de una lista también es empleada la función "del" pasando como parámetro la referencia de la componente a eliminar:

```
lista=[10, 20, 30, 40, 50]

print(lista)

del(lista[0])
del(lista[1])
del(lista[2])

print(lista) # 20 40
```

Problemas propuestos

1 - Crear dos listas paralelas. En la primera ingresar los nombres de empleados y en la segunda los sueldos de cada empleado.

Ingresa por teclado cuando inicia el programa la cantidad de empleados de la empresa.

Borrar luego todos los empleados que tienen un sueldo mayor a 10000 (tanto el sueldo como su nombre)

2 - Crear una lista de 5 enteros y cargarlos por teclado. Borrar los elementos mayores o iguales a 10 y generar una nueva lista con dichos valores.

Solución a los problemas

```
empleados=[]
sueldos=[]
cant=int(input("Cuantos empleados tiene la empresa:"))
for x in range(cant):
    nom=input("Ingrese el nombre:")
    empleados.append(nom)
    su=int(input("Ingrese el importe del sueldo:"))
    sueldos.append(su)

print("Listado completo de empleados")
for x in range(len(sueldos)):
    print(empleados[x],sueldos[x])

posicion=0
while posicion<len(sueldos):
    if sueldos[posicion]>10000:
        sueldos.pop(posicion)
        empleados.pop(posicion)
    else:
        posicion=posicion+1

print("Listado de empleados que cobran 10000 o menos")
for x in range(len(sueldos)):
    print(empleados[x],sueldos[x])

listal=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    listal.append(valor)

print("Lista original")
print(listal)

lista2=[]
posicion=0
while posicion<len(listal):
    if listal[posicion]>=10:
        lista2.append(listal.pop(posicion))
    else:
        posicion=posicion+1

print("Lista despues de borrar los elementos mayores o iguales a 10")
print(listal)
print("Lista generada con los elementos mayores o iguales a 10")
print(lista2)
```