

Clase Número: 9

Funciones (segunda parte)

Funciones (parámetros de tipo lista)

Hemos visto que si trabajamos con programación estructurada debemos dividir nuestro problema en trocitos y plantear algoritmos que resuelvan cada uno de los trozos.

La división en Python se hace implementando funciones. La comunicación entre estas funciones se produce mediante el envío de datos a través de los parámetros y el valor que retorna la función.

Hasta ahora hemos resuelto problemas enviando datos simples como enteros, float y cadenas de caracteres. En este concepto veremos que una función puede recibir tanto datos simples como estructuras de datos.

La estructura de datos vista hasta este momento y que podemos enviarle a una función es la lista.

Problema 1:

Definir por asignación una lista de enteros en el bloque principal del programa. Elaborar tres funciones, la primera recibe la lista y retorna la suma de todos sus elementos, la segunda recibe la lista y retorna el mayor valor y la última recibe la lista y retorna el menor.

Programa:

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

def mayor(lista):
    may=lista[0]
    for x in range(1,len(lista)):
        if lista[x]>may:
            may=lista[x]
    return may

def menor(lista):
    men=lista[0]
    for x in range(1,len(lista)):
        if lista[x]<men:
            men=lista[x]
    return men

# bloque principal del programa
```

```
listavalores=[10, 56, 23, 120, 94]
print("La lista completa es")
print(listavalores)
print("La suma de todos su elementos es", sumarizar(listavalores))
print("El mayor valor de la lista es", mayor(listavalores))
print("El menor valor de la lista es", menor(listavalores))
```

La función `sumarizar` recibe un parámetro de tipo lista, dentro de la misma lo recorremos mediante un `for` y accedemos a sus elementos por medio de un subíndice:

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

Desde el bloque principal de nuestro programa llamamos a `sumarizar` enviando el dato retornado a la función `print` para que lo muestre:

```
print("La suma de todos su elementos es", sumarizar(listavalores))
```

De forma similar definimos las otras dos funciones que también reciben la lista y luego implementan los algoritmos para identificar el mayor y el menor de la lista:

```
def mayor(lista):
    may=lista[0]
    for x in range(1,len(lista)):
        if lista[x]>may:
            may=lista[x]
    return may

def menor(lista):
    men=lista[0]
    for x in range(1,len(lista)):
        if lista[x]<men:
            men=lista[x]
    return men
```

Las llamadas a estas dos funciones desde el bloque principal es similar a cuando llamamos a `sumarizar` (como cada una retorna un entero procedemos a llamarlas dentro de la función `print`):

```
print("El mayor valor de la lista es", mayor(listavalores))
print("El menor valor de la lista es", menor(listavalores))
```

El nombre de la lista que definimos en el bloque principal del programa es `listavalores`:

```
listavalores=[10, 56, 23, 120, 94]
```

No hay ningún problema de definir dicha variable con el mismo nombre que el parámetro de la función, es decir sería correcto nuestro programa si definiéramos en el bloque principal el siguiente código:

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

```

def mayor(lista):
    may=lista[0]
    for x in range(1,len(lista)):
        if lista[x]>may:
            may=lista[x]
    return may

def menor(lista):
    men=lista[0]
    for x in range(1,len(lista)):
        if lista[x]<men:
            men=lista[x]
    return men

# bloque principal del programa

lista=[10, 56, 23, 120, 94]
print("La lista completa es")
print(lista)
print("La suma de todos su elementos es", sumarizar(lista))
print("El mayor valor de la lista es", mayor(lista))
print("El menor valor de la lista es", menor(lista))

```

Debe quedar claro que la variable global definida en el bloque principal se llama "lista" y los parámetros de las tres funciones también se llaman "lista".

Problema 2:

Crear y cargar por teclado en el bloque principal del programa una lista de 5 enteros. Implementar una función que imprima el mayor y el menor valor de la lista.

Programa:

```

def mayormenor(lista):
    may=lista[0]
    men=lista[0]
    for x in range(1,len(lista)):
        if lista[x]>may:
            may=lista[x]
        else:
            if lista[x]<men:
                men=lista[x]
    print("El valor mayor de la lista es", may)
    print("El valor menor de la lista es", men)

# bloque principal

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

```

En el bloque principal de nuestro programa cargamos dentro de un for 5 valores enteros y los añadimos a la estructura de datos "lista".

Fuera del for procedemos a llamar a la función `mayormenor` y procedemos a enviarle la lista para que pueda consultar sus datos.

Es importante notar que la función `mayormenor` no retorna nada sino ella es la que tiene el objetivo de mostrar el mayor y menor valor de la lista.

Problemas propuestos

1 - Crear una lista de enteros por asignación. Definir una función que reciba una lista de enteros y un segundo parámetro de tipo entero. Dentro de la función mostrar cada elemento de la lista multiplicado por el valor entero enviado.

```
lista=[3, 7, 8, 10, 2]
multiplicar(lista,3)
```

2 - Desarrollar una función que reciba una lista de string y nos retorne el que tiene más caracteres. Si hay más de uno con dicha cantidad de caracteres debe retornar el que tiene un valor de componente más baja.

En el bloque principal iniciamos por asignación la lista de string:

```
palabras=["enero", "febrero", "marzo", "abril", "mayo", "junio"]
print("Palabra con mas caracteres:",mascaracteres(palabras))
```

3 - Definir una lista de enteros por asignación en el bloque principal. Llamar a una función que reciba la lista y nos retorne el producto de todos sus elementos. Mostrar dicho producto en el bloque principal de nuestro programa.

Solución a los problemas

problema 1

```
def multiplicar(lista,va):
    for x in range(len(lista)):
        multi=lista[x]*va
        print(multi)

# bloque principal

lista=[3, 7, 8, 10, 2]
print("Lista original:",lista)
print("Lista multiplicando cada elemento por 3")
multiplicar(lista,3)
```

problema 2

```
def mascaracteres(palabras):
    pos=0
    for x in range(len(palabras)):
        if len(palabras[x])>len(palabras[pos]):
            pos=x
    return palabras[pos]

# bloque principal

palabras=["enero", "febrero", "marzo", "abril", "mayo", "junio"]
print("Palabra con mas caracteres:",mascaracteres(palabras))
```

problema 3

```
def producto(lista):
    prod=1
    for x in range(len(lista)):
        prod=prod*lista[x]
    return prod

# bloque principal

lista=[1, 2, 3, 4, 5]
print("Lista:", lista)
print("Multiplicacion de todos sus elementos:",producto(lista))
```

Funciones (retorno de una lista)

Hemos avanzado y visto que una función puede recibir como parámetros tipos de datos simples como enteros, flotantes etc. y estructuras de datos tipo lista.

También hemos visto que la función mediante la palabra clave return puede retornar un tipo de dato simple desde donde se la invocó.

Lo nuevo en este concepto es que una función también puede retornar una estructura de datos tipo lista. Con esto estamos logrando que una función retorne varios datos ya que una lista es una colección de datos.

Problema 1:

Confeccionar una función que cargue por teclado una lista de 5 enteros y la retorne. Una segunda función debe recibir una lista y mostrar todos los valores mayores a 10. Desde el bloque principal del programa llamar a ambas funciones.

Programa:

```
def carga_lista():
    li=[]
    for x in range(5):
        valor=int(input("Ingrese valor:"))
        li.append(valor)
    return li

def imprimir_mayor10(li):
    print("Elementos de la lista mayores a 10")
    for x in range(len(li)):
        if li[x]>10:
            print(li[x])

# bloque principal del programa

lista=carga_lista()
imprimir_mayor10(lista)
```

Recordemos que el programa empieza a ejecutarse desde el bloque principal y no se ejecutan cada función en forma lineal. Lo primero que hacemos en el bloque principal del programa es llamar a la función carga_lista y se la asignamos a una variable que recibirá la referencia de la función que se creará en carga_lista:

```
lista=carga_lista()
```

La función carga_lista como vemos no tiene parámetros:

```
def carga_lista():
```

Dentro de la función definimos una variable local de tipo lista llamada li (el nombre de la variable local puede ser cualquiera, inclusive se podría llamar lista ya que no hay conflictos en definir variables con el mismo nombre en una función y en el bloque principal):

```
li=[]
```

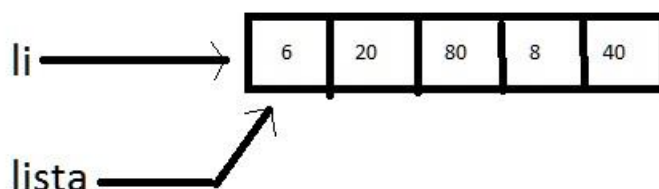
Seguidamente definimos un for que se repita 5 veces y solicitamos al operador que ingrese enteros y los añadimos a la lista:

```
for x in range(5):
    valor=int(input("Ingrese valor:"))
    li.append(valor)
return li
```

Cuando finaliza el for ya tenemos almacenado en la lista los 5 enteros y mediante la palabra clave de Python return procedemos a devolver a quien invocó la función la variable "li".

Esto quiere decir que en el bloque principal del programa lista recibe la referencia de "li":

```
lista=carga_lista()
```



Si no hacemos esa asignación la lista que devuelve la función carga_lista se perdería:

```
carga_lista() # incorrecto al no asignarla a una variable el dato devuelto
```

La segunda línea de nuestro bloque principal es llamar a la función imprimir_mayor10 y pasar la lista que cargamos previamente:

```
imprimir_mayor10(lista)
```

La segunda función de nuestro programa es la imprimir_mayor10 que recibe como parámetro una lista y procede a analizar cada elemento para ver si debe imprimirse:

```
def imprimir_mayor10(li):
    print("Elementos de la lista mayores a 10")
    for x in range(len(li)):
        if li[x]>10:
            print(li[x])
```

Problema 2:

Confeccionar una función que cargue por teclado una lista de 5 enteros y la retorne. Una segunda función debe recibir una lista y retornar el mayor y el menor valor de la lista. Desde el bloque principal del programa llamar a ambas funciones e imprimir el mayor y el menor de la lista.

Programa:

```
def carga_lista():
```

```

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

def retornar_mayormenor(li):
    ma=li[0]
    me=li[0]
    for x in range(1,len(li)):
        if li[x]>ma:
            ma=li[x]
        else:
            if li[x]<me:
                me=li[x]
    return [ma,me]

# bloque principal del programa

lista=carga_lista()
rango=retornar_mayormenor(lista)
print("Mayor elemento de la lista:",rango[0])
print("Menor elemento de la lista:",rango[1])

```

Lo interesante que tenemos que sacar de este ejemplo es que cuando tenemos que retornar varios valores podemos utilizar una lista. La función `retornar_mayormenor` debe devolver dos enteros, y la forma de hacerlo en Python es mediante una lista.

En la función `retornar_mayormenor` definimos dos variables locales donde almacenamos el primer elemento de la lista que recibimos como parámetro:

```

def retornar_mayormenor(li):
    ma=li[0]
    me=li[0]

```

Luego mediante un `for` procedemos a analizar el resto de los elementos de la lista para verificar si hay alguno que sea mayor al que hemos considerado mayor hasta ese momento, lo mismo con el menor:

```

    for x in range(1,len(li)):
        if li[x]>ma:
            ma=li[x]
        else:
            if li[x]<me:
                me=li[x]

```

Cuando sale del `for` ya tenemos almacenadas en las variables `ma` y `me` el valor mayor y el valor menor de la lista.

La sintaxis para devolver ambos datos es crear una lista e indicando en cada elementos la variable respectiva:

```

    return [ma,me]

```


Ahora cuando llamo a esta función desde el bloque principal del programa sabemos que nos retorna una lista y que el primer elemento representa el mayor de la lista y el segundo elemento representa el menor de la lista:

```
rango=retornar_mayormenor(lista)
print("Mayor elemento de la lista:",rango[0])
print("Menor elemento de la lista:",rango[1])
```

Importante

Hemos visto que podemos retornar en una función una lista. Python también nos permite descomponer los valores devueltos por la función en varias variables que reciban cada elemento de esa lista:

```
# bloque principal del programa

lista=carga_lista()
mayor, menor=retornar_mayormenor(lista)
print("Mayor elemento de la lista:",mayor)
print("Menor elemento de la lista:",menor)
```

Como vemos definimos las variables mayor y menor y le asignamos el valor que retorna la función mayormenor. Cada elemento de la lista se guarda en el mismo orden, es decir la componente 0 de la lista se guarda en la variable mayor y la componente 1 de la lista se guarda en la variable menor. Esto se hace para que el programa sea mas legible en cuanto a nombre de variables y el dato que almacenan.

Problema 3:

Desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de realizar la carga por teclado de todos los datos imprimir los nombres de las personas mayores de edad (mayores o iguales a 18 años)
Imprimir la edad promedio de las personas.

Programa:

```
def cargar_datos():
    nom=[]
    ed=[]
    for x in range(5):
        v1=input("Ingrese el nombre de la persona:")
        nom.append(v1)
        v2=int(input("Ingrese la edad:"))
        ed.append(v2)
    return [nom,ed]

def mayores_edad(nom,ed):
    print("Nombres de personas mayores de edad")
    for x in range(len(nom)):
        if ed[x]>=18:
            print(nom[x])

def promedio_edades(ed):
    suma=0
    for x in range(len(ed)):
        suma=suma+ed[x]
```

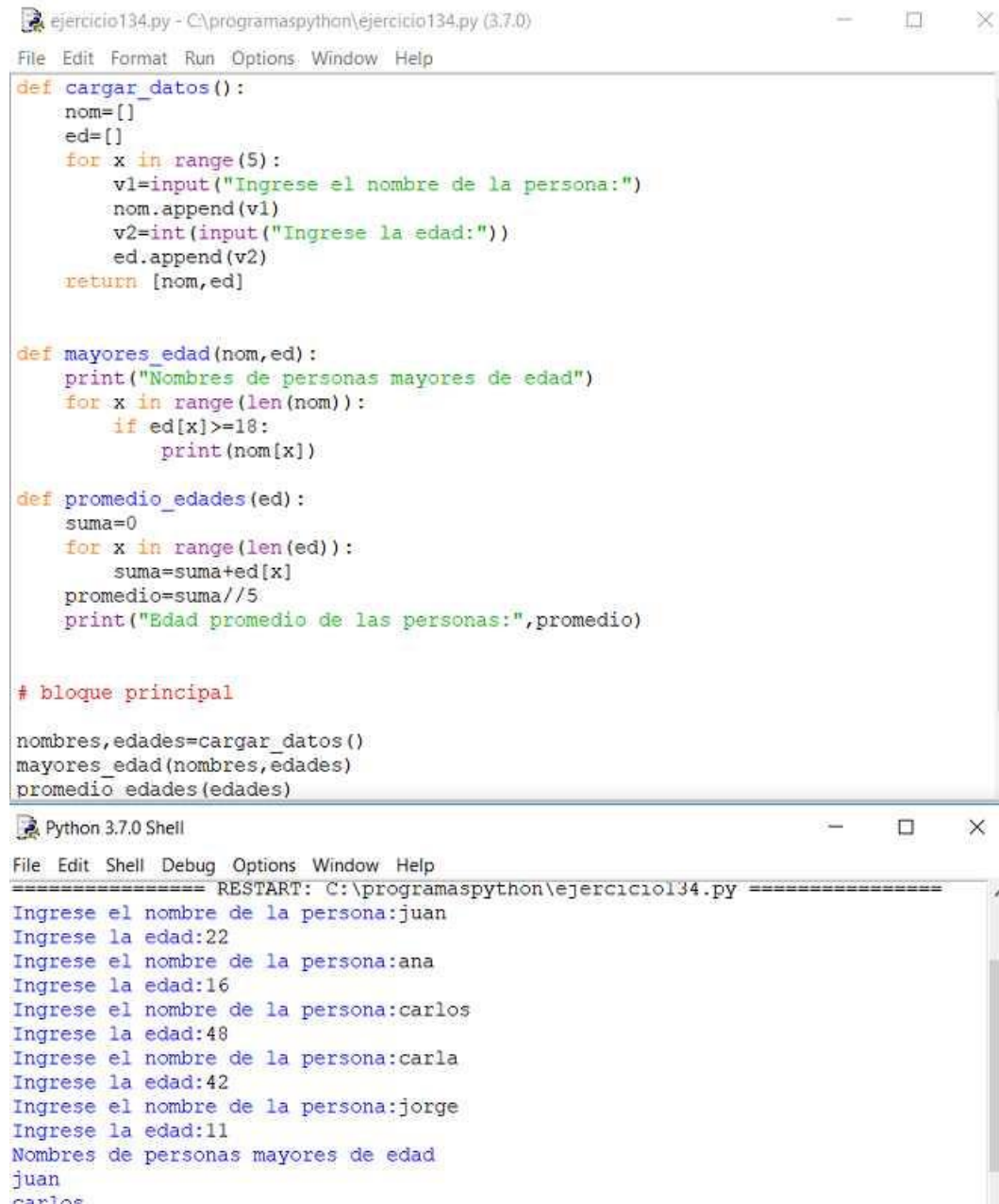
```

    promedio=suma//5
    print("Edad promedio de las personas:",promedio)

# bloque principal

nombres,edades=cargar_datos()
mayores_edad(nombres,edades)
promedio_edades(edades)

```



The screenshot shows a Python IDE window titled 'ejercicio134.py - C:\programaspython\ejercicio134.py (3.7.0)'. The code defines three functions: `cargar_datos()` which prompts for 5 names and ages, `mayores_edad(nom, ed)` which prints names of people aged 18 or older, and `promedio_edades(ed)` which calculates and prints the average age. The main block calls these functions. Below the code window is the 'Python 3.7.0 Shell' window showing the execution output, including the restart message and the interactive prompts for names and ages, followed by the printed results for older people and the average age.

```

def cargar_datos():
    nom=[]
    ed=[]
    for x in range(5):
        v1=input("Ingrese el nombre de la persona:")
        nom.append(v1)
        v2=int(input("Ingrese la edad:"))
        ed.append(v2)
    return [nom,ed]

def mayores_edad(nom,ed):
    print("Nombres de personas mayores de edad")
    for x in range(len(nom)):
        if ed[x]>=18:
            print(nom[x])

def promedio_edades(ed):
    suma=0
    for x in range(len(ed)):
        suma=suma+ed[x]
    promedio=suma//5
    print("Edad promedio de las personas:",promedio)

# bloque principal

nombres,edades=cargar_datos()
mayores_edad(nombres,edades)
promedio_edades(edades)

```

```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\programaspython\ejercicio134.py =====
Ingrese el nombre de la persona:juan
Ingrese la edad:22
Ingrese el nombre de la persona:ana
Ingrese la edad:16
Ingrese el nombre de la persona:carlos
Ingrese la edad:48
Ingrese el nombre de la persona:carla
Ingrese la edad:42
Ingrese el nombre de la persona:jorge
Ingrese la edad:11
Nombres de personas mayores de edad
juan
carlos

```

Para resolver este problema debemos crear y cargar dos listas paralelas. En una guardamos los nombres de las personas y en la otra almacenamos las edades.

Definimos una función `cargar_datos` que crea y carga las dos listas paralelos. Las variables locales que almacenan dichas listas son `nom` y `ed`:

```

def cargar_datos():
    nom=[]

```

```

ed=[]
for x in range(5):
    v1=input("Ingrese el nombre de la persona:")
    nom.append(v1)
    v2=int(input("Ingrese la edad:"))
    ed.append(v2)
return [nom,ed]

```

Esta función retorna una lista cuyos elementos son listas (es decir estamos en presencia de listas con elementos tipo lista):

```

return [nom,ed]

```

Cuando llamamos a esta función desde el bloque principal de nuestro programa le asignamos a dos variables:

```

nombres,edades=cargar_datos()

```

Ahora en las variables globales nombres y edades tenemos almacenados los dos listas que cargamos en la función cargar_datos.

Como la función cargar_datos() retorna una lista de listas luego las dos variables receptoras son listas.

En el bloque principal luego de llamar a cargar_datos que nos retorna las dos listas procedemos a llamar a las otras funciones mayores_edad y promedio_edades (nombres y edades son dos variables globales que normalmente se inicializan con datos que devuelve una función y luego se las enviamos a otras funciones para que las procesen):

```

# bloque principal

nombres,edades=cargar_datos()
mayores_edad(nombres,edades)
promedio_edades(edades)

```

El algoritmo de la función que imprime los nombres de las personas mayores de edad es (recibe las dos listas):

```

def mayores_edad(nom,ed):
    print("Nombres de personas mayores de edad")
    for x in range(len(nom)):
        if ed[x]>=18:
            print(nom[x])

```

El algoritmo que calcula el promedio de edades es:

```

def promedio_edades(ed):
    suma=0
    for x in range(len(ed)):
        suma=suma+ed[x]
    promedio=suma//5
    print("Edad promedio de las personas:",promedio)

```

Problemas propuestos

- 1 - En una empresa se almacenaron los sueldos de 10 personas.
Desarrollar las siguientes funciones y llamarlas desde el bloque principal:
 - a) Carga de los sueldos en una lista.
 - b) Impresión de todos los sueldos.
 - c) Cuántos tienen un sueldo superior a \$4000.
 - d) Retornar el promedio de los sueldos.
 - e) Mostrar todos los sueldos que están por debajo del promedio.
- 2 - Desarrollar una aplicación que permita ingresar por teclado los nombres de 5 artículos y sus precios.
Definir las siguientes funciones:
 - a) Cargar los nombres de artículos y sus precios.
 - b) Imprimir los nombres y precios.
 - c) Imprimir el nombre de artículo con un precio mayor
 - d) Ingresar por teclado un importe y luego mostrar todos los artículos con un precio menor igual al valor ingresado.
- 3 - Confeccionar un programa que permita:
 - a) Cargar una lista de 10 elementos enteros.
 - b) Generar dos listas a partir de la primera. En una guardar los valores positivos y en otra los negativos.
 - c) Imprimir las dos listas generadas.

Solución a los problemas

problema 1

```
def cargar_sueldos():
    sueldos=[]
    for x in range(10):
        su=int(input("Ingrese sueldo:"))
        sueldos.append(su)
    return sueldos

def imprimir_sueldos(sueldos):
    print("Listado de sueldos")
    for x in range(len(sueldos)):
        print(sueldos[x])

def sueldos_mayor4000(sueldos):
    cant=0
    for x in range(len(sueldos)):
        if sueldos[x]>4000:
            cant=cant+1
    print("Cantidad de empleados con un sueldo superior a 4000:",cant)

def promedio(sueldos):
    suma=0
    for x in range(len(sueldos)):
        suma=suma+sueldos[x]
    promedio=suma//10
    return promedio

def sueldos_bajos(sueldos):
    pro=promedio(sueldos)
    print("Sueldo promedio de la empresa:",pro)
    print("Sueldos inferiores al promedio")
    for x in range(len(sueldos)):
        if sueldos[x]<pro:
            print(sueldos[x])

# bloque principal

sueldos=cargar_sueldos()
imprimir_sueldos(sueldos)
sueldos_mayor4000(sueldos)
sueldos_bajos(sueldos)
```

problema 2

```
def cargar_datos():
    articulos=[]
    precios=[]
    for x in range(5):
        ar=input("Ingrese el nombre del articulo:")
        articulos.append(ar)
        pre=int(input("Ingrese el precio de dicho articulo:"))
        precios.append(pre)
    return [articulos,precios]
```

```

def imprimir(articulos,precios):
    print("Listado completo de articulos y sus precios")
    for x in range(len(articulos)):
        print(articulos[x],precios[x])

def precio_mayor(articulos,precios):
    may=precios[0]
    pos=0
    for x in range(1,len(precios)):
        if precios[x]>may:
            may=precios[x]
            pos=x
    print("Articulo con un precio mayor es :",articulos[pos],"su precio
es:",may)

def consulta_precio(articulos,precios):
    valor=int(input("Ingrese un importe para mostrar los articulos con un
precio menor:"))
    for x in range(len(precios)):
        if precios[x]<valor:
            print(articulos[x],precios[x])

# bloque principal

articulos,precios=cargar_datos()
imprimir(articulos,precios)
precio_mayor(articulos,precios)
consulta_precio(articulos,precios)

```

problema 3

```

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

def generar_listas(lista):
    listanega=[]
    listaposi=[]
    for x in range(len(lista)):
        if lista[x]<0:
            listanega.append(lista[x])
        else:
            if lista[x]>0:
                listaposi.append(lista[x])
    return [listanega,listaposi]

def imprimir(lista):
    for x in range(len(lista)):
        print(lista[x])

# programa principal

lista=cargar()
listanega,listaposi=generar_listas(lista)

```

```
print("Lista con los valores negativos")
imprimir(listanega)
print("Lista con los valores positivos")
imprimir(listaposi)
```