listas

Son estructuras de información que contienen un grupo de datos, no necesariamente homogéneos. Serían semejantes a las cadenas pero conteniendo elementos que pueden ser de distintos tipos.

Se accede a cada dato empleando un subíndice entra corchetes al igual que las cadenas pero además admiten ponerse en el lado izquierdo de una asignación.

Las constantes de tipo lista se escriben también entre corchetes, por ejemplo una lista de enteros sería [5, 23, -4, 67, 12, 9]

```
mi_lista=[5,23,-4,67,12,9] # una lista de enteros
```

listas

```
mi_lista=[5,23,-4,67,12,9] # una lista de enteros
print(mi_lista)
print(mi_lista[0])
print(mi_lista[-2])
print(mi_lista[2])
print(mi_lista[1:4])
print(mi_lista[-3:])
print(mi_lista[::2])
print(mi_lista[3::-1])
print(mi_lista[::-1])
```

listas

```
a=5
otra_lista=["Milán","dos",a>0,a/2,[9,-12],a*3]
```

En este caso otra_lista es heterogénea.

```
otra_lista[0] es una cadena
otra_lista[1] es también una cadena
otra_lista[2] es un booleano
otra_lista[3] es un real
otra_lista[4] es también una lista
otra_lista[5] es un entero
```

listas

```
lista=[5,23,-4,67,12,9]

# recorrido de una lista sin modificar sus elementos
for elemento in lista:
    print(elemento)

# recorrido también sin modificar (pero se podría)
for i in range(len(lista)):
    print(lista[i])

# recorrido modificando (sumando 1 en este caso)
for i in range(len(lista)):
    lista[i]=lista[i]+1
print(lista)
```

peculiaridades

1) las rodajas de listas se especifican entre corchetes exactamente igual que las rodajas de cadenas pero además se pueden poner al lado izquierdo de una asignación

```
p=[10,4,"hola",-2.5,70,True,"un test",[2,3]]
p[2:5]=[62,"otra",5.8,False,"no"]
print(p)
```

p es ahora [10,4,62,"otra",5.8,False,"no",True,"un test",[2, 3]], es decir se ha sustituido la rodaja ["hola",-2.5,70] por [62,"otra",5.8,False,"no"]

peculiaridades

2) cuidado con las asignaciones cuando en el lado derecho hay una variable de tipo lista. En este caso no se crea una lista nueva sino que se crea un alias a la otra. Ejemplo:

```
p=[10,-5,61,0,999,-17]
q=p
p[2]=28
print(p) # muestra [10,-5,28,0,999,-17]
print(q) # muestra [10,-5,28,0,999,-17]
```

p y q son la misma lista, para que fueran diferentes debería haberse puesto en el lado derecho una expresión como `q=p+[]` o `q=list(p)`

funciones con cadenas

`len(p)`	devuelve la longitud de la cadena
`str(p)`	convierte el parámetro a cadena
`chr(p)`	dado un entero devuelve el carácter asociado
`ord(p)`	dado un carácter devuelve el entero de su código
`max(p)`	dada una cadena devuelve el carácter de código más alto
`min(p)`	dada una cadena devuelve el carácter de código más bajo
`list(p)`	devuelve una lista con los elementos de la cadena
`sorted(p)`	devuelve una lista ordenada con los elementos de la cadena

El parámetro p debe ser una cadena excepto en chr() que debe ser un entero y ord() que deba ser una cadena de un solo carácter

operadores para cadenas

`+`	concatenación de cadenas
`*`	repetición de cadenas
`in`	cadena contenida en otra
`[i]`	selección de partes de una cadena, un solo carácter
`[i:j]`	selección de partes de una cadena, una subcadena
`[i:j:k]`	selección de partes de una cadena, una subcadena de k en k
`[i:]`	selección de partes de una cadena, una subcadena hasta el final
`[:j]`	selección de partes de una cadena, subcadena desde el principio
`[:]`	selección de partes de una cadena, toda la cadena

`> >= < <= == !=` mayor, mayor o igual, menor, menor o igual, igual y diferente, devuelven un resultado booleano al igual que el operador in

instrucciones para cadenas

bucle for iterable, permite recorrer una cadena carácter a carácter para procesar cada uno de ellos

```
for caracter in cadena:
    # cuerpo del bucle
    ...
    ...
    ...
```

métodos para cadenas

`.split(p)`	corta una cadena y devuelve una lista de subcadenas
`.join(p)`	une una lista de subcadenas en una cadena
`.find(p)`	posición de una cadena dentro de otra
`.count()`	cuántas veces aparece una cadena dentro de otra
`.lstrip(p)`	elimina subcadenas por la izquierda
`.rstrip(p)`	elimina subcadenas por la derecha
`.strip(p)`	elimina subcadenas por la izquierda y por la derecha
`.replace(p,q)`	reemplaza la cadena p por la cadena q
`.upper()`	cadena con todo mayúsculas
`.lower()`	cadena con todo minúsculas

funciones con listas

`len(p)`	devuelve la longitud de la lista
`max(p)`	devuelve el mayor elemento de la lista
`min(p)`	devuelve el menor elemento de la lista
`sum(p)`	devuelve la suma de los elementos de una lista numérica
`list(p)`	devuelve otra lista idéntica
`sorted(p)`	devuelve la lista con los mismos elementos pero ordenada

El parámetro p debe ser una lista. Para max y min debe ser una lista homogénea y para sum debe ser una lista numérica (solo enteros enteros y reales)

operadores para listas

`+`	concatenación de listas
`*`	repetición de listas
`in`	elemento contenido en lista
`[i]`	selección de partes de una lista, un solo elemento
`[i:j]`	selección de partes de una lista, una sublista
`[i:j:k]`	selección de partes de una lista, una sublista de k en k
`[i:]`	selección de partes de una lista, una sublista hasta el final
`[:j]`	selección de partes de una lista, una sublista desde el principio
`[:]`	selección de partes de una lista, toda la lista

`> >= < <= == !=` mayor, mayor o igual, menor, menor o igual, igual y diferente, devuelven un resultado booleano al igual que el operador in

instrucciones para listas

bucle for iterable, permite recorrer una lista elemento a elemento para procesar cada uno de ellos

```
for elemento in lista:
    # cuerpo del bucle
    ...
    ...
    ...
```

métodos para listas

`.append(p)`	añade un elemento a una lista
`.pop(p)`	devuelve y elimina el elemento en posición p
`.remove(p)`	elimina de la lista el elemento de valor p
`.insert(p,q)`	inserta en la posición p el elemento q
`.index(p)`	devuelve la posición del elemento p en la lista
`.count(p)`	devuelve el número de veces que está p en la lista
`.clear(p)`	elimina todos los elementos de la lista dejándola vacía
`.extend(p)`	añade la lista p a la lista

para hacer:

- programa que pida enteros y los guarde en una lista. Al final debe mostrar el contenido de la lista y generar dos listas, una con los pares y otra con los impares mostrándolas también. La cadena vacía debe ser el fin de la entrada de datos
- modifíquese el programa anterior para que además muestre el número de elementos pares, el de impares, la suma de los pares y la suma de los impares
- modifíquese el programa anterior para que los cálculos los haga sin emplear la función len(p) ni la función sum(p)
- haga un programa que pida cadenas de caracteres creando dos listas, una con las cadenas que contienen alguna vocal y otra con las que son de longitud par mostrando ambas al final. Fin con la cadena vacía.