

NIVEL 3

PATRONES DE RECORRIDO





REPASEMOS

(PATRONES DE RECORRIDO DE LISTAS Y
EN GENERAL DE CUALQUIER SECUENCIA)

RECORRIDO TOTAL



Se usa cuando se necesita recorrer **TODA** la lista (o la cadena de caracteres).

Ejemplos:

- ✓ Calcular la suma de TODOS los valores numéricos almacenados en una lista
- ✓ Obtener el elemento de menor valor de una lista
- ✓ ...

Con while

Con for-in
range

Con for-in

OPCIONES

CON WHILE

Índice para movernos
en la lista empieza en
CERO

Condición para continuar: índice
menor que la longitud de la lista

```
EjemploRecorridoTotal.py ✕  
1 def suma_recorrido_total_con_while(lista: list) -> int:  
2     i = 0  
3     suma = 0  
4     while i < len(lista) :  
5         suma += lista[i]  
6         i += 1  
7     return suma
```

Avance: **incremento**
en 1 del índice

IMPORTANTE: la variable **i** sirve para recorrer
los índices de la lista, por esa razón debemos
extraer el elemento con el operador **[]**

CON FOR-IN RANGE

Índice para movernos en la lista:
• va desde la posición 0 hasta la longitud de la lista menos 1

```
def suma_recorrido_total_con_for_in_range(lista: list) -> int:  
    suma = 0  
    for i in range(0, len(lista)):  
        suma += lista[i]  
    return suma
```

IMPORTANTE: la variable *i* sirve para recorrer los índices de la lista, por esa razón debemos extraer el elemento con el operador []

CON FOR-IN

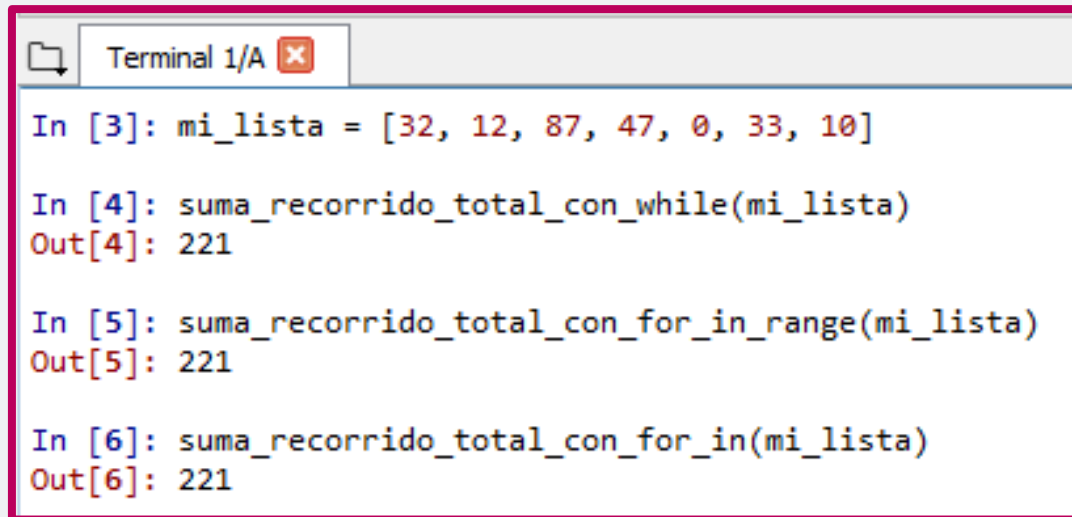
Variable para recorrer cada elemento de la lista

```
def suma_recorrido_total_con_for_in(lista: list) -> int:  
    suma = 0  
    for cada_numero in lista:  
        suma += cada_numero  
    return suma
```

IMPORTANTE: la variable `cada_elemento` sirve para recorrer los elementos de la lista, por esa razón **NO** necesitamos extraer el elemento con el operador `[]`

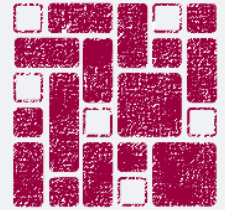
VEAMOS...

Resultados de las ejecuciones



```
Terminal 1/A ✕  
In [3]: mi_lista = [32, 12, 87, 47, 0, 33, 10]  
  
In [4]: suma_recorrido_total_con_while(mi_lista)  
Out[4]: 221  
  
In [5]: suma_recorrido_total_con_for_in_range(mi_lista)  
Out[5]: 221  
  
In [6]: suma_recorrido_total_con_for_in(mi_lista)  
Out[6]: 221
```

RECORRIDO PARCIAL



- ✓ Se usa cuando **NO** se necesita recorrer **TODA** la lista
- ✓ Existe una condición que debemos verificar en cada iteración para saber si debemos detener el ciclo o volver a repetirlo

Ejemplos:

- ✓ Decidir si al menos un valor es positivo en una lista de números
- ✓ Buscar la posición de un valor dado
- ✓ ...

While con
centinela

While sin
centinela

OPCIONES

Un recorrido parcial **NO** se puede implementar con **for-in** porque es necesario interrumpir el ciclo cuando se encuentra lo que se está buscando



CON CENTINELA

Índice para movernos
en la lista empieza en
CERO

Centinela para parar el ciclo cuando
haya encontrado la posición del
elemento buscado, empieza en **False**

```
EjemploRecorridoParcial.py x
1 def posicion_recorrido_parcial_con_centinela(lista: list, buscado: int) -> int:
2     i = 0
3     posicion = -1
4     ya_encontre = False
5     while i < len(lista) and ya_encontre == False:
6         if lista[i] == buscado:
7             ya_encontre = True
8             posicion = i
9         i += 1
10    return posicion
```

Condición para continuar:

- índice **menor** que la longitud de la lista **Y (and)**
- el centinela es **False**

En cualquier caso se **incrementa el índice** para continuar el recorriendo

Al final, se retorna **la posición**
del elemento buscado **o -1** si
no lo encontró

Si se encuentra el elemento
buscado, se cambia el valor del
centinela a **True** para romper el
ciclo

CON CENTINELA

IMPORTANTE: controlar **SIEMPRE** que el índice que se usa para recorrer la lista esté dentro del tamaño de la lista para evitar que se muera el programa por “**Index out of range**”



```
EjemploRecorridoParcial.py x
1 def posicion_recorrido_parcial_con_centinela(lista: list, buscado: int) -> int:
2     i = 0
3     posicion = -1
4     ya_encontre = False
5     while i < len(lista) and ya_encontre == False:
6         if lista[i] == buscado:
7             ya_encontre = True
8             posicion = i
9         i += 1
10    return posicion
```

SIN CENTINELA

Índice para movernos
en la lista empieza en
CERO

Condición para continuar:

- índice **menor** que la longitud de la lista **Y (and)**
- **no hayamos encontrado** lo que estamos buscando

```
def posicion_recorrido_parcial_sin_centinela(lista: list, buscado: int) -> int:
    i = 0
    while i < len(lista) and lista[i] != buscado:
        i += 1

    if (i == len(lista)):
        posicion = -1
    else:
        posicion = i

    return posicion
```

Se **incrementa el**
índice para continuar
recorriendo la lista

Si al finalizar el ciclo, **el índice tiene el valor de la longitud de la lista**, quiere decir que **no se encontró** lo que se estaba buscando

Al final, se retorna **la posición**
del elemento buscado **o -1** si
no lo encontró

SIN CENTINELA

IMPORTANTE: controlar **SIEMPRE** que el índice que se usa para recorrer la lista esté dentro del tamaño de la lista para evitar que se muera el programa por “Index out of range”



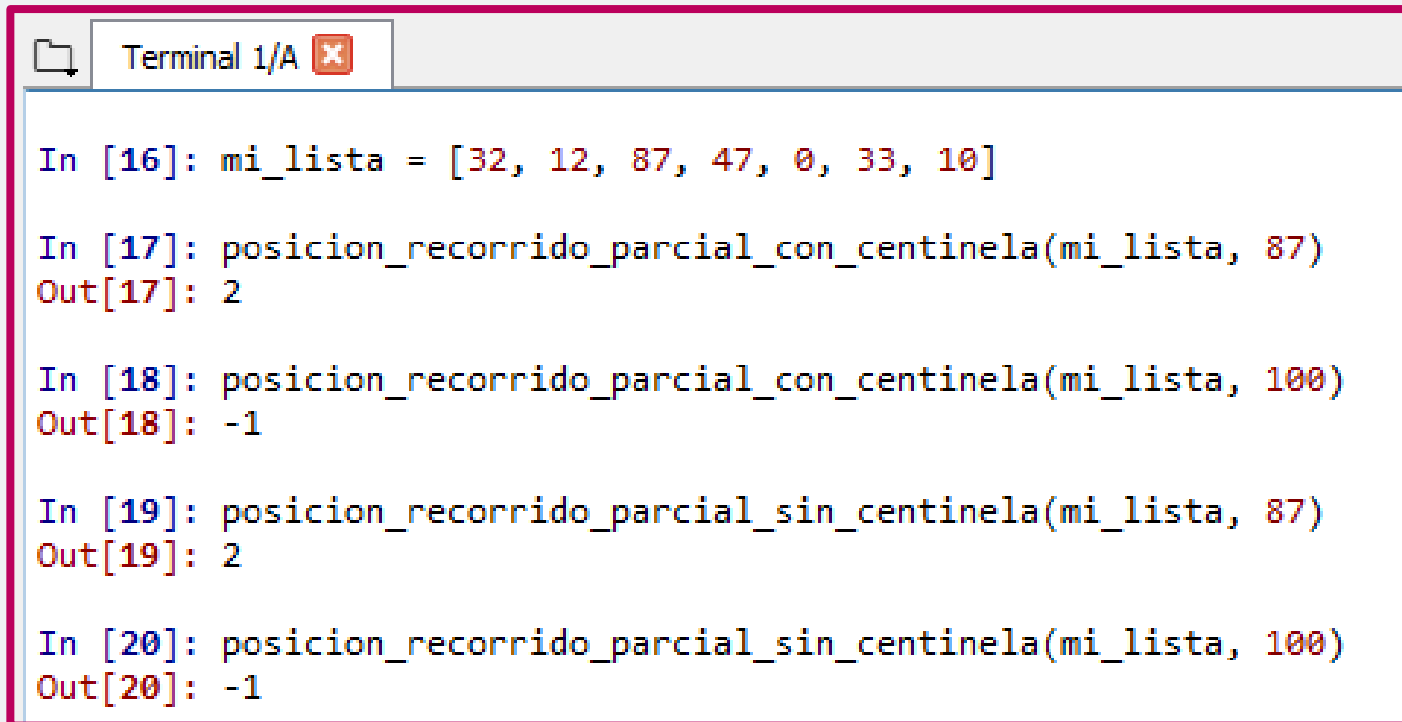
```
def posicion_recorrido_parcial_sin_centinela(lista: list, buscado: int) -> int:
    i = 0
    while i < len(lista) and lista[i] != buscado:
        i += 1

    if (i == len(lista)):
        posicion = -1
    else:
        posicion = i

    return posicion
```

VEAMOS...

Resultados de las ejecuciones



```
Terminal 1/A [X]

In [16]: mi_lista = [32, 12, 87, 47, 0, 33, 10]

In [17]: posicion_recorrido_parcial_con_centinela(mi_lista, 87)
Out[17]: 2

In [18]: posicion_recorrido_parcial_con_centinela(mi_lista, 100)
Out[18]: -1

In [19]: posicion_recorrido_parcial_sin_centinela(mi_lista, 87)
Out[19]: 2

In [20]: posicion_recorrido_parcial_sin_centinela(mi_lista, 100)
Out[20]: -1
```