INFORMATORIO

TALLER DE PROGRAMACIÓN WEB

LISTAS

Subsecretaría de Empleo
Chaco Gobierno de todos

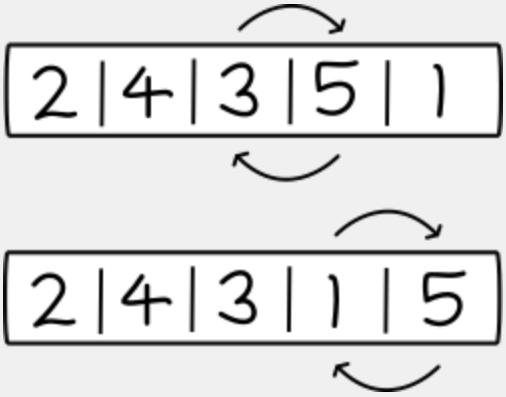
Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

Listas

Las **listas** en Python son conjuntos de elementos ordenados y pueden estar conformadas por elementos de distintos tipo (enteros, float, cadenas, funciones, otras listas, etc).

Las listas en Python son **mutables**, lo que significa que sus elementos pueden *cambiar o reordenarse.*



Las listas se delimitan por **corchetes [ ]** y sus elementos se separan por comas.

Los elementos de una lista pueden accederse mediante su índice.

Ejemplos:

```
>>> lista_colores = ["rojo", "azul", "verde", "amarillo", "naranja"]
>>> lista_numeros = [3, 45, 23, 67]
>>> lista_variada = [45, 23.4, "Lunes", True, "rojo"]
>>> frutas = ["manzana","naranja","banana","frutilla"]
>>> lista_compra = ["pan", "huevo", "fideo", "arroz"]
```

También puede contener otras listas dentro

```
>>> lista = ["abc", ["a","e","i"], 45, True]
```

Una lista puede estar vacía:

```
>>> lista_sin_elementos = [ ]
```

Para crear una lista podremos hacerlo encerrando los elementos entre corchetes y separados por comas:

```
>>> mi_lista = [1, 2, 3, 4, 5]
>>> otra_lista = ["uno", "dos", "tres", "cuatro", "cinco"]
>>> lista = ["uno", 55, True]
```

Podemos crear listas vacías de dos formas:

```
>>> lista = [ ]
>>> lista = list()
```

Si queremos conocer la cantidad de elementos en una lista podemos utilizar la función `len()`

```
>>> lista = ["hola", "chau"]
>>> cantidad_elementos = len(lista)
>>> print(cantidad_elementos)
2
```

Podemos acceder a los distintos elementos de la lista por medio de su índice.

```
>>> estaciones = [“Verano”, “Otoño”,“Invierno”, “Primavera”]
>>> print(estaciones[0])
Verano
>>> print(estaciones[1])
Otoño
>>> print(estaciones[2])
Invierno
```

los índices de las listas comienzan en 0 (cero). Por eso usamos el índice 0 para acceder al primer elemento, el índice 1 para el segundo y así sucesivamente.
También podemos usar de índices enteros negativos. correspondiendo el -1 al último elemento, el -2 al anteúltimo y así sucesivamente.

```
>>> print(estaciones[-1])
Primavera
>>> print(estaciones[-2])
Invierno
```

Una lista anidada se considera como un solo elemento, independientemente de cuántos elementos contenga.

```
>>> lista = [1, 2, 3, ["a","b","c"], 44, 77]
>>> print(len(lista))
6
>>> elemento = lista[3]
>>> print(elemento)
["a","b","c"]
>>> print(lista[4])
44
```

Modificar Elementos

Se puede modificar cualquier elemento de la lista haciendo referencia a su posición y asignándole el nuevo elemento.

```
>>> animales = ["gato", "perro", "sapo", "rata", "mono"]
Podemos cambiar el cuarto elemento de la siguiente manera
>>> animales[3] = "tortuga"
ahora veremos al imprimir la lista que el cuarto elemento cambio de "rata" a "tortuga"
>>> print(animales)
["gato", "perro", "sapo", "tortuga", "mono"]
```

Recorrer una lista

Puedes leer los elementos de la lista utilizando un ciclo for utilizando range():

```
>>> lista = ["a","b","c"]
>>> for index in range(0, len(lista)):
>>>     print(lista[index])
a
b
c
```

pero python también nos permite recorrer listas de la siguiente manera

```
>>> lista = ["a","b","c"]
>>> for elemento in lista:
>>>     print(elemento)
a
b
c
```

Agregar elementos en una lista

Podemos agregar un elemento a la lista utilizando `append()`

```
>>> lista_de_compras = ["arroz", "harina", "huevo"]
>>> print(lista_de_compras)
["arroz", "harina", "huevo"]
>>> lista_de_compras.append("tomate")
>>> print(lista_de_compras)
["arroz", "harina", "huevo","tomate"]
```

Como vemos el elemento que pasamos entre paréntesis en append() se agregó al final de la lista.

Otra forma de insertar elementos es con `insert()`, que aparte de indicarle el elemento que queremos insertar le indicamos el índice.

```
>>> series = ["The IT Crowd", "Halt and Catch Fire", "Silicon Valley"]
>>> series.insert(2, "Black Mirror")
>>> print(series)
["The IT Crowd", "Halt and Catch Fire", "Black Mirror", "Silicon Valley"]
```

Podemos agregar todos los elementos de otra lista al final con el método `extend()`

```
>>> una_lista = ["Hola", 56, True, 3.44]
>>> otra_lista = ["A", 23, "B", "C"]
>>> una_lista.extend(otra_lista)
>>> print(una_lista)
['Hola', 56, True, 3.44, 'A', 23, 'B', 'C']
>>> print(otra_lista)
['A', 23, 'B', 'C']
```

vemos como **una_lista** se extendió con los elementos de **otra_lista** y **otra_lista** no se ve afectada.

Prueba el siguiente código

```
>>> lista_uno = ["a","b","c"]
>>> lista_dos = [1, 2, 3]
>>> lista_uno.append(lista_dos)
>>> print(lista_uno)
['a', 'b', 'c', [1, 2, 3]]
```

Ahora prueba el siguiente.

```
>>> lista_uno = ["a","b","c"]
>>> lista_dos = [1, 2, 3]
>>> lista_uno.extend(lista_dos)
>>> print(lista_uno)
['a', 'b', 'c', 1, 2, 3]
```

¿Cúal es la diferencia entre append y extend?

Obtener sublistas (slicing)

Se puede obtener sublistas a partir de una lista usando [:] de esta manera:

```
>>> mi_lista = ["a","b","c","d","e","f"]
>>> lista_cortada = mi_lista[1:3]
>>> print(lista_cortada)
['b', 'c']
```

El primer número indica el índice del primer elemento, y el segundo en donde detenerse y dejar de tomar elementos.

```
>>> mi_lista = ["a","b","c","d","e","f"]
>>> mi_lista[3:5]
['d', 'e']
>>> mi_lista[0:6]
["a","b","c","d","e","f"]
```

Si se decide omitir el primer número, se comenzará desde el inicio.

```
>>> mi_lista[:4]
["a","b","c","d"]
```

`mi_lista[:4] es igual a hacer mi_lista[0:4]`

Si se elimina el segundo número se tomarán elementos hasta el final.

```
>>> mi_lista[0:]
["a","b","c","d","e","f"]
>>> mi_lista[4:]
['d', 'e', 'f']
```

`mi_lista[3:] es igual a hacer mi_lista[3:len(lista)]`

Ahora si no pones ningún número, y solo deja los dos puntos, se copiará la lista completa.

```
>>> mi_lista[:]
["a","b","c","d","e","f"]
```

`mi_lista[:] es igual a mi_lista[0:len(lista)]`

Copiar un lista.
Si necesitamos una copia de la lista en otra variable podremos hacer

```
>>> copia = mi_lista[:]
```

o también utilizar el método copy()

```
>>> copia = mi_lista.copy()
```

¿Por qué no podríamos simplemente hacer ***copia*** = ***mi_lista*** directamente?

prueba el siguiente código.

```
>>> frutas = ["naranja", "banana", "frutilla"]
>>> copia = frutas
>>> print(frutas)
['naranja', 'banana', 'frutilla']
>>> print(copia)
['naranja', 'banana', 'frutilla']
>>> copia.append("uva")
>>> print(copia)
['naranja', 'banana', 'frutilla', 'uvas']
>>> print(frutas)
['naranja', 'banana', 'frutilla', 'uvas']
```

¿qué sucedió?

Ahora prueba:

```
>>> frutas = ["naranja", "banana", "frutilla"]
>>> copia = frutas[:]
#también podrías usar frutas.copy()
#copia = frutas.copy()
>>> copia.append("uva")
>>> print(copia)
['naranja', 'banana', 'frutilla', 'uvas']
>>> print(frutas)
['naranja', 'banana', 'frutilla']
```

¿y ahora?

También puedes modificar varios elementos a la vez operando con sublistas.

```
>>> colores = ["rojo","verde","azul","rosado"]
>>> colores[1:3] = ["amarillo", "violeta"]
>>> print(colores)
["rojo","amarillo","violeta","rosado"]
```

Ordenar una lista

Para ordenar una lista podemos usar el método sort()

```
>>> numeros = [56, 34, 67, 89, 23, 12, 62, 73, 27]
>>> numeros.sort()
>>> print(numeros)
[12, 23, 27, 34, 56, 62, 67, 73, 89]
```

para ordenarlos de manera inversa, puedes pasar el parámetro reverse=True al método sort()

```
>>> numeros.sort(reverse=True)
>>> print(numeros)
[89, 73, 67, 62, 56, 34, 27, 23, 12]
```

otros ejemplos

```
>>> paises = ["ECU", "BRA","DEU", "CHN","ARG", "PRY", "CHL"]
>>> paises.sort()
>>> print(paises)
['ARG', 'BRA', 'CHL', 'CHN', 'DEU', 'ECU', 'PRY']
```

Invertir una lista

Para invertir el orden de una lista, se puede usar el método reverse()

```
>>> animales = ["gato", "perro", "tortuga", "loro"]
>>> animales.reverse()
>>> print(animales)
['loro', 'tortuga', 'perro', 'gato']
```

Saber si un elemento se encuentra en la lista

para saber si un elemento se encuentra en la lista, podemos usar el operador in.

```
>>> alumnos = ["Juan", "Lara", "Micaela", "Jorge"]
>>> print("Lara" in alumnos)
True
>>> print("Maria" in alumnos)
False

>>> if "Micaela" in alumnos:
>>>     print("se encuentra en la lista")
```

para saber cuantas veces se encuentra un elemento dentro de una lista podremos usar el método count()

```
>>> numeros = [1, 4, 6, 3, 1, 1, 5, 6]
>>> numeros.count(1)
3
>>> numeros.count(6)
6
```

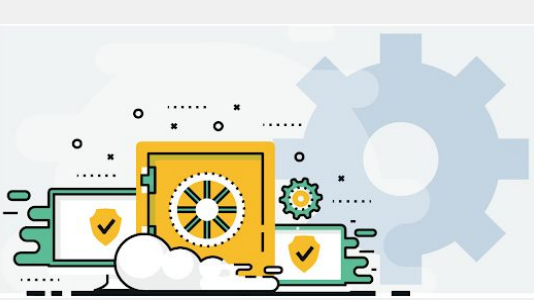
Conocer el índice de un elemento

Podemos conocer el índice de un elemento utilizando el método index()

```
>>> nombres = ["Mirian", "Sandra", "Walter", "Kevin", "Emilce"]
>>> indice = nombres.index("Sandra")
>>> print(indice)
1
>>> print(nombres[indice])
Sandra
```

En caso de tener más de un elemento igual, index() solo va devolver el índice del primero.

```
>>> lista = ["b", "a", "a","c","a"]
>>> print( lista.index("a") )
1
```



Eliminar elementos de una lista.

puedes eliminar elementos de una lista utilizando la palabra reservada ***del***

```
>>> letras = ["a", "b", "c", "d", "e"]
>>> print(letras)
["a", "b", "c", "d", "e"]
>>> del letras[3]
>>> print(letras)
["a", "b", "c", "e"]
```

en el ejemplo se borro "d" que se encontraba en el índice 3.

Podemos borrar de a varios elementos operando sublistas [:]

```
>>> letras = ["a", "b", "c", "d", "e"]
>>> del letras[2:5]
>>> print(letras)
["a","b"]
```

Otra manera de eliminar elementos es con el método remove() y especificando el elemento a borrar.

```
>>> palabras = ["hola", "casa", "zapatillas", "puerta", "tenedor"]
>>> palabras.remove('casa')
>>> print(palabras)
["hola", "zapatillas", "puerta", "tenedor"]
```

Otra forma de quitar elementos es utilizando el método pop(), que no sólo quitará un elemento de la lista si no que nos retornará el valor quitado.

```
>>> lista = ["mouse", "teclado", "monitor", "impresora"]
>>> elemento_quitado = lista.pop()
>>> print(elemento_quitado)
"impresora"
>>> print("lista")
["mouse", "teclado", "monitor"]
```

También podemos especificar el índice del elemento que queremos quitar.


```
>>> colores = ["amarillo", "verde", "azul", "rojo"]
>>> color_quitado = lista.pop(2)
>>> print(color_quitado)
"azul"
>>> print(lista)
["amarillo", "verde", "rojo"]
```

Algunas funciones de python que puedes utilizar en listas son:

len() para saber la longitud(cantidad de elementos) que contiene una lista
min() devuelve el mínimo
max() devuelve el máximo
sum() devuelve la suma entre sus elementos

```
>>> mi_lista = [1,2,3,4,5,6]
>>> longitud = len(mi_lista)
>>> print(longitud)
6
>>> minimo = min(mi_lista)
>>> print(minimo)
>>> 1
>>> maximo = max(mi_lista)
>>> print(maximo)
6
>>> suma = sum(mi_lista)
>>> print(suma)
21
```

También puedes utilizar operadores matemáticos en la listas

```
>>> una_lista = ["a","b","c"]
>>> otra_lista = [1, 2, 3]
>>> resultado = una_lista + otra_lista
>>> print(resultado)
["a","b","c",1,2,3]
>>> animales = ["elefante", "leon", "tigre"]
>>> print(animales)
>>> ["elefante", "leon", "tigre"]
>>> animales += ["mono", "perro"]
>>> print(animales)
["elefante", "leon", "tigre", "mono", "perro"]
>>> resultado = una_lista * 3
>>> print(resultado)
["a","b","c","a","b","c","a","b","c"]
```

¿Observaste algunas similitudes entre listas y cadenas de caracteres?