# Listas en Python

Una **lista** es una colección ordenada de valores. Una lista puede contener cualquier cosa. En Python, el tipo de datos que representa a las listas se llama **list**.

## Como crear listas

Las dos maneras principales de crear una lista son:

1. Crear una lista con los valores entre corchetes.
   - L = [1, 2, 3, 4]
   - nom = [‘Juan’,’Ana’,’Pedro’]
   - datos = [‘Luis’,34,’ced: 3456789’]
2. Usar la función **list** .
   - texto = list(‘Hola’) → el contenido de texto será: ['H', 'o', 'l', 'a'], crea una lista con los caracteres de la palabra ‘Hola’
   - números = list(range(10)) → el contenido de números será: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], llena la lista con los valores que va generando range, en este caso entre 0 y 10.

## Como imprimir una lista

print (numeros)

En consola veremos: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

## Operaciones sobre listas

- **len(lista)** entrega cuantos elementos hay en la lista

  len(nom) → indica 3

  len(Texto) → indica 4

- **lista[i]** entrega el **i-esimo** valor de la lista. El valor i se llama **índice** de la lista. Los índices parten de cero.

  L[0] → entrega → 1

  Datos[2] → entrega ced: 3456789

- Modificar el valor del **i-esimo** elemento de la lista

  Nom[1] = “María” si se imprime la lista nom tenemos nom = [‘Juan’,’Maria’,’Pedro’]

  Si el índice i indica un índice que no está en la lista, ocurre un **error de índice**

  Datos[3], al querer referenciar esta posición de la lista el Pytho índica:

  **Traceback (most recent call last):**

  **File "<stdin>", line 1, in <module>**

  **IndexError: list index out of range ←Indice Fuera de rango**

  Si el índice es negativo, los elementos se cuentan desde el final hacia atrás

  L[-1] → entrega 4

L[-4] → entrega 1

- **lista.append(x)** agrega el elemento x al final de la lista

  Nom.append( “Jose”) → si se imprime la lista nom tenemos:

  nom = [‘Juan’, ’María’, ’Pedro’, ’José’]

  **Nota:** Las listas no controlan si se insertan elementos repetidos, si necesitamos exigir unicidad, debemos hacerlo mediante el código de nuestros programas.

- **sum**(x) entrega la suma de los valores de la lista

  sum(L) → entrega 10

- Operador **+** concatena listas.

  numero + nom → entrega [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ‘Juan’, ’María’, ’Pedro’, ’José’]

- Para saber si un elemento x está en la lista, se usa **x in** lista. La versión negativa de **in** es **not in**

  8 in número → entrega True

  12 not in numero → entrega True

  8 not in numero → entrega False

- **lista[i:j]** es el operador de **rebanado**, que entrega una nueva lista que tiene desde el **i-esimo** hasta justo antes del **j-esimo** elemento de la lista, el rebanado no modifica la lista, sino que crea una nueva:

  nom[1:3] →entrega ['María', 'Pedro']

  nom[1:4] → entrega ['María', 'Pedro', 'José']

- **lista.count(x)** cuenta cuantas veces esta el elemento **x** en la lista.

  letras = list('paralelepipedo') → al mostrar **letras** vemos:

  ['p', 'a', 'r', 'a', 'l', 'e', 'l', 'e', 'p', 'i', 'p', 'e', 'd', 'o']

  letras.count('p') entrega → 3

- **l.remove(x)** elimina el elemento **x** de la lista.

  nom.remove('Juan') al mostrar la lista entrega → ['María', 'Pedro', 'José']

  numeros.remove(3) al mostrar la lista entrega → [0, 1, 2, 4, 5, 6, 7, 8, 9]

  **Advertencia:** Si el valor a borrar no existe, se produce un error:
  `ValueError: list.remove(x): x not in list`

- **lista.reverse()** invierte la lista.

  nom.reverse() entrega → ['José', 'Pedro', 'María'].

- **lista.sort()** ordena la lista.

  nom.append('Ana')

  nom.append('Berta') al agregar estos nombres a la lista **nom** esta queda → ['José', 'Pedro', 'María', 'Ana', ’Berta’]de

  nom.sort() entrega → ['Ana', 'Berta', 'José', 'María', 'Pedro']

## Como recorrer una lista

Una lista es un objeto **iterable**. Esto significa que sus valores se pueden recorrer usando un ciclo **for**, sea

```
valores = [6, 1, 7, 8, 9]
for i in valores:
        print i ** 2
```

En cada iteración del **for**, la variable **i** toma uno de los valores de la lista, por lo que este programa imprime los valores 36, 1, 49, 64 y 81.

## Uso de Split

Puedes usar el método **split** para dividir el texto en palabras:

```
Frase=”La casita de la pradera”
Mylista=Frase.split()
print(Mylista)
[‘La’,’casita’,’de’,’la’,’pradera’]
```

¿Qué sucede si utilizamos otro divisor que no sea el espacio?

```
Frase=”La-casita-de-la-pradera”
Mylista=Frase.split(‘-‘) ← Indico el separador entre palabras
print(Mylista)
[‘La’,’casita’,’de’,’la’,’pradera’]
```

## Unir una lista

El proceso opuesto de dividir una cadena en una lista de cadenas es unirlas para formar una cadena. Puedes unir los elementos de la lista para hacer una cadena utilizando el método **join** de esta manera:

```
Mylista = [‘La’,’casita’,’de’,’la’,’pradera’]
Separador = ’ ‘ ← Indicamos el separador entre palabras
Frase = separador.join(Mylista)
print(Frase)
”La casita de la pradera”
```