

Introducción a Python

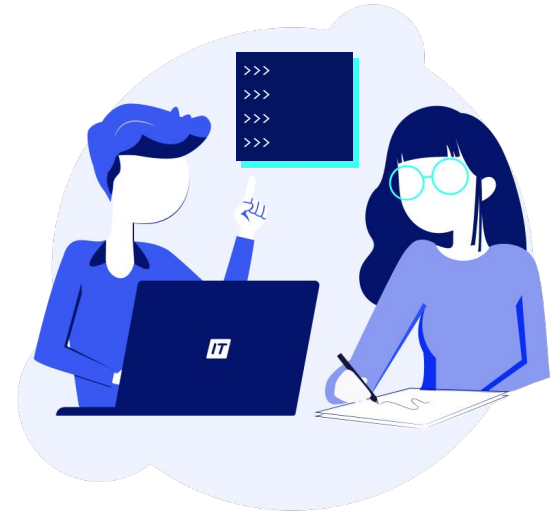
Módulo 2

Listas

Listas

Las listas nos permiten **agrupar varios objetos** (de cualquiera de los tipos de dato básicos) en una misma variable. Cada uno de estos objetos se conoce como **elemento**.

Para crear una lista vamos a escoger un nombre e indicar, entre corchetes y separados por comas, los elementos.



Veamos los ejemplos de la derecha.

En el primero creamos una lista de cinco números primos. El segundo es una lista con nombres de alumnos. En el último vemos una lista con valores de distintos tipos de dato.

La disposición de los elementos al momento de crear una lista es importante. **Las listas son colecciones ordenadas de objetos.**

A cada uno de los elementos se le asigna un **índice**, que no es otra cosa que la **posición** en la que se encuentra en la lista, empezando del 0. El índice nos permite luego traer un elemento en particular de una lista cualquiera.

```
>>> numeros_primos = [2, 3, 5, 7, 11]
```

```
>>> alumnos = ["Jorge", "Matias", "Juan", "Martin"]
```

```
>>> datos = [3.14, True, -1, False, "Hola mundo"]
```



Para acceder a un elemento de una lista a partir de su índice, lo vamos a indicar entre corchetes luego del nombre de la lista.

```
>>> numeros_primos[0]
2
>>> numeros_primos[1]
3
>>> numeros_primos[2]
5
>>> numeros_primos[3]
7
>>> numeros_primos[4]
11
```

Ahora bien, podemos agregar un elemento al final de la lista (luego de haberla creado) usando la sintaxis:

`nombre_lista.append(elemento)`

Por ejemplo, agreguemos un nuevo elemento en la quinta posición:

```
>>> numeros_primos.append(13)
```

```
>>> numeros_primos[5]
13
```

La consola interactiva nos mostrará la lista completa si indicamos su nombre sin corchetes ni índice.

Vía la sintaxis `nombre_lista.insert(índice, elemento)` insertamos un nuevo elemento en la posición especificada. En este caso, el elemento 3.14 se inserta en la posición 2, es decir, entre los números 3 y 5.

Nótese que todos los elementos subsiguientes al que acabamos de insertar sufren un desplazamiento hacia la derecha. De modo que el número 13, que antes estaba en la quinta posición, ahora está en la sexta.

La operación `insert` es menos frecuente que `append`.

```
>>> numeros_primos
[2, 3, 5, 7, 11, 13]
```

```
>>> numeros_primos.insert(2, 3.14)
```

```
>>> numeros_primos[2]
3.14
>>> numeros_primos
[2, 3, 3.14, 5, 7, 11, 13]
```

```
>>> numeros_primos[6]
13
```

La sintaxis para **quitar un elemento de una lista** es `del nombre_lista[indice]`. Vamos a remover el elemento que acabamos de insertar:

Del mismo modo los elementos de una lista se reorganizan luego de haber eliminado uno de ellos, a menos que haya sido el último. En particular, todos aquellos que estaban a la derecha del elemento removido se desplazan una posición hacia la izquierda.

Así, ahora `numeros_primos[2]` es **5** de nuevo.

```
>>> del numeros_primos[2]
```

```
>>> numeros_primos  
[2, 3, 5, 7, 11, 13]
```

Otra operación frecuente sobre listas es obtener el número de elementos que contiene, vía la instrucción `len(nombre_lista)`.

Existe una relación entre la cantidad de elementos de una lista y la posición del último elemento. Puesto que los índices comienzan de cero, el último será `len(nombre_lista) - 1`.

A partir de esto es claro que entre los corchetes podemos colocar un número o cualquier expresión que retorne un número.

```
>>> len( numeros_primos )  
6
```

```
>>> # Accede al último elemento.  
... numeros_primos[ len( numeros_primos ) - 1 ]  
13
```

Ten en cuenta que los “...” son agregados por la consola interactiva, al igual que “>>>”, y no forman parte del código de Python.




Por ejemplo, los siguientes métodos son todos similares para acceder al cuarto elemento (es decir, aquel cuyo índice es 3).

```
>>> numeros_primos[3]
7
>>> numeros_primos[2 + 1]
7
>>> indice = 3
>>> numeros_primos[indice]
7
```

Por último, para crear una lista vacía usamos dos corchetes sin elementos:

```
>>> lista_vacia = []
```

Esto puede resultar útil cuando no conocemos los elementos de la lista al momento de crearla, pero los iremos añadiendo vía **append** o **insert** en el transcurso del programa.



**¡Sigamos
trabajando!**