Argentina programa 4.0

.UBA fiuba

FACULTAD DE INGENIERÍA

PYTHON

Listas
Características y Manejo

Lic. Gustavo Bianchi

Junio 2023

Contenido

- Estructura Interna, composición y organización.
- Cómo acceder a los elementos que la componen.
- Cómo agregar, modificar y eliminar elementos.
- Cómo hacemos para recorrerlas.
- Cómo podemos generarlas.
- Cómo las podemos ordenar.

¿Qué son las listas?

➔ Son una **Colección o agrupación de elementos** en secuencia, encerrados entre corchetes

[2, 23, 8, 48, 5, 0]

➔ Pueden ser de distinto tipo

[235, "azul", 28.3, "rojo"]

➔ Son **MUTABLES**, su contenido puede ser modificado: **podemos reemplazar, agregar y eliminar elementos**.

¿Cómo se accede a los elementos?

➔ Cada elemento ocupa una posición comenzando en cero

```
[2, 23,  8, 48, 5, 0]
 0   1   2   3  4  5
```

➔ Se accede a los elementos a través de la posición que ocupan

```
>>> lista = [2, 23, 8, 48, 5, 0]
>>> lista[3]
48
```

➔ También puedo acceder a una posición, desde el final de la lista

```
>>> lista[ - 1]
0
```

Accedo al último elemento de la lista

```
>>> lista[ -4 ]
8
```

Accedo al 4to. elemento desde el final

¿Qué operaciones podemos hacer?

➔ Se pueden concatenar listas y obtener una nueva lista formada por todos los elementos

Ejemplos

```
>>> lista = lista + [-1, -10, 23]
>>> lista = lista_1 + lista_2
>>> lista = [51, -100, 3] + [-15, 23] + otra_lista
```

➔ Se puede declarar una lista sin valores, asignando una lista vacía

```
>>> lista = [ ]
```

¿Al ser Mutables qué podemos hacer?

		lista
Creamos la lista	>>> lista = [2, 11, 7, 4]	[2, 11, 7, 4]
Reemplazamos el valor de la posición 2	>>> lista[2] = 53	[2, 11, 53, 4]
Agregamos un valor al final de la lista	>>> lista.append(-15)	[2, 11, 53, 4, -15]
Otra forma de agregar un valor al final	>>> lista += [6]	[2, 11, 53, 4, -15, 6]
Agregamos los elementos de un iterable	>>> lista.extend(range(4))	[2, 11, 53, 4, -15, 6, 0, 1, 2, 3]

Nota: Reproducí en la consola lo que se muestra en esta y las siguientes diapositivas

¿Qué otras cosas más podemos hacer por ser Mutables?

		lista = [2, 11, 53, 4, -15, 6, 1]
Insertamos el valor 20 en la posición 1	>>> lista.insert(1, -20)	[2, -20, 11, 53, 4, -15, 6, 1]
Eliminamos el valor de la posición 0	>>> del(lista[0])	[-20, 11, 53, 4, -15, 6, 0, 1, 2, 3]
Retira el anteúltimo elemento y retorna su valor	>>> lista.pop(-2)	[-20, 11, 53, 4, -15, 6, 0, 1, 3]
Elimina la primer ocurrencia del valor -15	>>> lista.remove(-15)	[-20, 11, 53, 4, 6, 0, 1, 3]
Elimina todos los elementos de la lista	>>> lista.clear( )	[ ]

Y además también podemos ……

```
>>> lista = [2, 23, 8, 48, 5, 8, 0]
```

Conocer la cantidad de elementos que hay en la lista

```
>>> len( lista )
7
```

Conocer la cantidad de ocurrencias de un elemento

```
>>> lista.count(8)
2
```

Conocer la posición de un elemento

```
>>> lista.index(23)
1
```

Podemos revertir los elementos en la lista

```
>>> lista.reverse()
[0, 8, 5, 48, 8, 23, 2]
```

Ejercicio: Jugando con una lista en la consola

Aplicando lo visto hasta acá, abrí la consola de Python y realizá las siguientes operaciones:

1. Crea una lista con los siguientes valores, respetando el orden dado:
   1 23 -100 55 48 23 -50 2 0
2. Inserta el valor 200 en la posición 3
3. Agrega a la lista el valor 35
4. Elimina la primer ocurrencia del valor 23
5. Elimina las posiciones 6,7 y 8
6. Agrega a la lista los valores enteros pares entre 500 y 520 inclusive
7. Reemplaza el valor en la posición 12, por el valor -50
8. Elimina el último de los elementos
9. Consulta cuántos elementos hay en la lista en este momento
10. Invertí la lista
11. Calcula el total de la suma de valores que forman parte de la lista
12. Suma todos los elementos que se encuentran en posiciones pares

¿Cómo hacemos para recorrerlas?

```
>>> lista = [2, 8, 23, 48, 5, 8]
```

Si quisiéramos mostrar cada elemento de la lista en una línea diferente

```
for elemento in lista:
    print(elemento)
```

```
2
8
23
48
5
8
```

Si quisiéramos mostrar un elemento por línea hasta encontrar el primer elemento impar

```
posicion = 0
while posicion < len(lista) and \
      lista[posicion] % 2 == 0:
   print(lista[posicion])
   posicion += 1
```

```
2
8
```

¿Cómo podemos generarlas?

➔ Según sea el caso hay distintas alternativas para generar una lista, o almacenar elementos en ella. Veamos algunos ejemplos.

Si quisiéramos generar una lista que contenga los cuadrados de los números del 1 al 10

```
lista = [ ]
for valor in range(1, 11):
    lista.append( valor**2 )
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

¿Cómo podemos generarlas?

➔ Supongamos ahora otro caso diferente al anterior

➔ Si quisiéramos cargar palabras en una lista y detenernos cuando la palabra ingresada sea “FIN”

```
l_palabras = [ ]
palabra = input("Palabra: ")
while palabra != "FIN":
      lista.append( palabra )
      palabra = input("Palabra: ")
```

```
Palabra: Sol
Palabra: Luna
Palabra: Tierra
Palabra: FIN
```

```
['Sol', 'Luna', 'Tierra']
```

¿Cómo hacemos para ordenarlas?

```
>>> lista = [2, 23, 8, 48, 5, 8, 0]
```

Alternativas

Podemos ordenarla sobre sí misma usando el **método sort()**

```
>>> lista.sort()
[0, 2, 5, 8, 8, 23, 48]
```

o Podemos obtener una lista ordenada sin afectar la original usando la **función sorted()**

```
>>> lista_ordenada = sorted(lista)
>>> lista_ordenada
[0, 2, 5, 8, 8, 23, 48]
>>> lista
[2, 23, 8, 48, 5, 8, 0]
```

Nota: Tanto el método sort(), como la función sorted(), disponen de un parámetro nombrado "reverse" cuyo valor por omisión es False. En caso de querer que la lista quede **ordenada descendentemente** se debe usar el parámetro del siguiente modo: **reverse = True**.

Ejercicio: Operaciones varias sobre una lista de valores

Escribir un programa modular (compuesto por funciones)**, que haciendo uso de listas**:

1. **Solicite el ingreso de una secuencia de valores**, que termina con el valor 0.
   A medida que se solicita y se ingresa un valor se debe almacenar en una lista.
   El valor 0 no debe almacenarse.
2. **Muestre los valores ingresados**.
3. **Muestre los valores hasta encontrar el 3er. valor par** ingresado inclusive.
   En caso de haber ninguno o menos de 3 valores pares en la lista, se mostrarán todos los valores.
4. **Muestre los elementos** que se encuentren en **posiciones pares**.
5. **Muestre los elementos ordenados de menor a mayor**, sin repetirlos.
   Si hay 2 o más valores iguales, sólo mostrar el valor una vez, y a su lado colocar “valor repetido”.

En todos los casos, **las salidas deben ser presentadas con un título** que indique lo que se está mostrando y mostrar un valor por línea.

www.**ingenieria**.uba.ar

/ingenieriauba

/FIUBAoficial