

Python Básico

Prof. Ariel Palazzesi

Comenzamos a grabar la clase

Módulo 2:

Control de flujo

Clase 06

Bucles for

1. Bucle for.
2. Recorriendo cadenas con for.
3. Uso de range()

Clase 07

Listas y Tuplas I

1. Creación y manipulación de listas y Tuplas.
2. Diferencias entre listas y tuplas.
3. Métodos básicos de listas: len(), append(), insert(), extend(), remove(), pop(), clear(), in
4. Slices de listas, tuplas y cadenas.

Clase 08

Listas y Tuplas II

1. Listas y tuplas con for y while.
2. Uso de list()
3. Métodos avanzados de listas: sort(), sorted(), index(), in, count().

Listas

Listas (Repaso)



✓ Una **lista** es una secuencia ordenada de elementos. Cada elemento es un dato. Pueden ser del mismo o distinto tipo, aunque esto último es poco frecuente.

Las listas en Python son un tipo **contenedor compuesto**, se usan para almacenar conjuntos de elementos relacionados, y las utilizamos en el video anterior.

Listas | Cantidad de elementos

✓ La función **len()** devuelve la cantidad de elementos que hay en una lista.

Código Python

```
# Definimos una lista de frutas
frutas = ["manzana", "banana", "cereza", "kiwi"]

# Obtener la cantidad de elementos en la lista
cantidad = len(frutas)

# Mostrar el resultado
print("La lista contiene", cantidad, "elementos.")
# Salida: La lista contiene 4 elementos.
```



Listas | Agregar un elemento

✓ El método **append()** permite agregar elementos **al final** de una lista. Se usa con la notación punto:

Código Python

```
frutas = ["manzana", "banana", "cereza"]      # Lista inicial con 3 elementos
print("Lista inicial de frutas:", frutas)      # Mostrar la lista inicial

# Pedimos una fruta al usuario...
nueva_fruta = input("Ingresa una fruta para agregar a la lista: ")
frutas.append(nueva_fruta)  # Aquí se agrega la nueva fruta a la lista

# Mostrar la lista actualizada
print("Lista actualizada de frutas:", frutas)
```

Listas | Agregar un elemento



✓ Es posible agregar un elemento en una posición específica de una lista utilizando el método **insert()**:

Código Python

```
# Lista inicial con 3 elementos
frutas = ["manzana", "banana", "cereza"]

frutas.insert(2, "pera") #(entre "banana" y "cereza")

# Mostrar la lista actualizada
print("Lista actualizada:", frutas)

# ['manzana', 'banana', 'pera', 'cereza']"
```


Listas | Agregar múltiples elementos

✓ El método **extend()** permite agregar múltiples elementos al final de una lista. A diferencia de **append()**, que añade un solo elemento, **extend()** toma otra lista y extiende la original con los elementos de esa secuencia:

Código Python

```
frutas = ["manzana", "banana", "cereza"]      # Lista inicial
nuevas_frutas = ["kiwi", "pera", "naranja"]    # Frutas a agregar

# Extender la lista original con las nuevas frutas
frutas.extend(nuevas_frutas)

# Mostrar la lista actualizada
print("Lista actualizada de frutas:", frutas)
```

Terminal

```
Lista actualizada de frutas:
['manzana', 'banana', 'cereza',
'kiwi', 'pera', 'naranja']
```

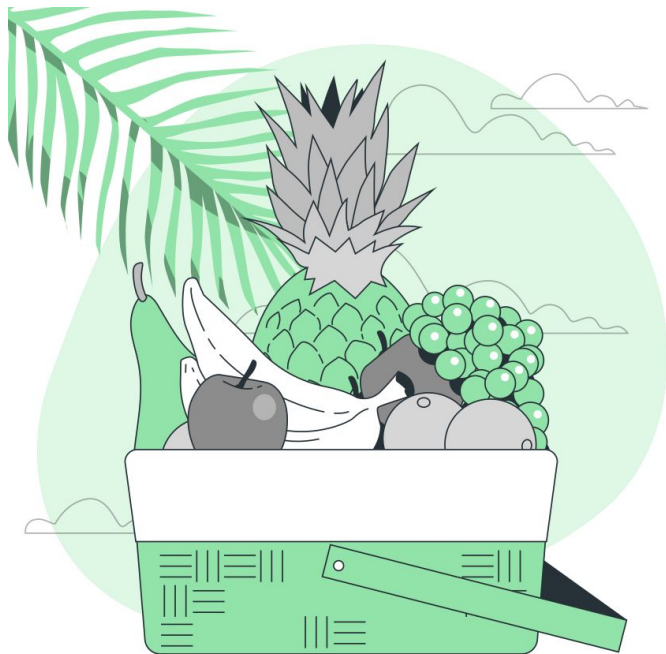
Listas | Eliminar elementos



Existen varios métodos para eliminar elementos de una lista:

| Método | Descripción |
|--------------------------------|---|
| <code>remove(valor)</code> | Elimina la primera ocurrencia del valor especificado. Si no existe, da un <code>ValueError</code> . |
| <code>pop(indice)</code> | Elimina el elemento en el índice especificado y lo devuelve. <code>pop()</code> elimina el último elemento de la lista. |
| <code>del lista[indice]</code> | Elimina el elemento o rango de elementos por índice. |
| <code>clear()</code> | Elimina todos los elementos de la lista. |

Listas | Eliminar elementos



✓ Veamos algunos ejemplos de estos métodos:

Código Python

```
frutas = ["uva", "banana", "higo", "piña", "mango"]

# Eliminar el elemento "higo"
frutas.remove("higo")
print(frutas)      # ['uva', 'banana', 'piña', 'mango']

# Eliminar el elemento con índice 1:
x = frutas.pop(1)
print(x)           # banana
print(frutas)      # ['uva', 'piña', 'mango']
```

Listas | Eliminar elementos

✓ Más ejemplos:

Código Python

```
frutas = ["uva", "banana", "higo", "piña", "mango"]

# Eliminar el elemento en la posición 0
del frutas[0]
print(frutas)    # ['banana', 'higo', 'piña', 'mango']

# Eliminar desde el índice 1 hasta el 2
del frutas[1:3]
print(frutas)    # ['banana', 'mango']
```



Listas | Eliminar todos los elementos



✓ El método **clear()** se usa para eliminar todos los elementos de una lista. Después de llamar a **clear()**, la lista quedará vacía, pero seguirá existiendo como una lista vacía.

Código Python

```
# Lista inicial de frutas
frutas = ["manzana", "banana", "cereza"]

frutas.clear() # Vaciar usando clear()

print(frutas) # [] (Lista vacia)
```

Listas | ¿Está en la lista, o no?

✓ El operador **in** se utiliza para verificar si un elemento está presente en una lista. Devuelve un valor booleano: **True** si el elemento está en la lista, y **False** si no lo está.

Código Python

```
frutas = ["manzana", "banana", "cereza"]  
  
print("banana" in frutas)  # Salida: True  
print("kiwi" in frutas)   # Salida: False
```



Tuplas

Tuplas

✓ Una **tupla** es una estructura de datos en Python similar a una **lista**, pero con una diferencia clave: **es inmutable**, es decir, una vez creada, no se puede modificar (no se pueden agregar, eliminar ni cambiar elementos).

| Característica | Lista | Tupla |
|---------------------|---|--|
| Definición | <code>mi_lista = [1, 2, 3]</code> | <code>mi_tupla = (1, 2, 3)</code> |
| Mutabilidad | Mutable (se puede modificar) | Inmutable (no se puede modificar) |
| Métodos disponibles | Muchos (<code>append()</code> , <code>remove()</code> , etc.) | Pocos (<code>count()</code> , <code>index()</code>) |
| Uso | Datos que pueden cambiar | Datos que no deben cambiar |

Desafíos

Desafíos de la clase:

✓ Desafío 1: Filtrar números pares de una lista

Crea un programa en Python que permita al usuario ingresar una lista de números enteros.

Luego, el programa debe crear una nueva lista que contenga solo los números pares de la lista original.

Finalmente, muestra la lista filtrada.



Desafíos de la clase:

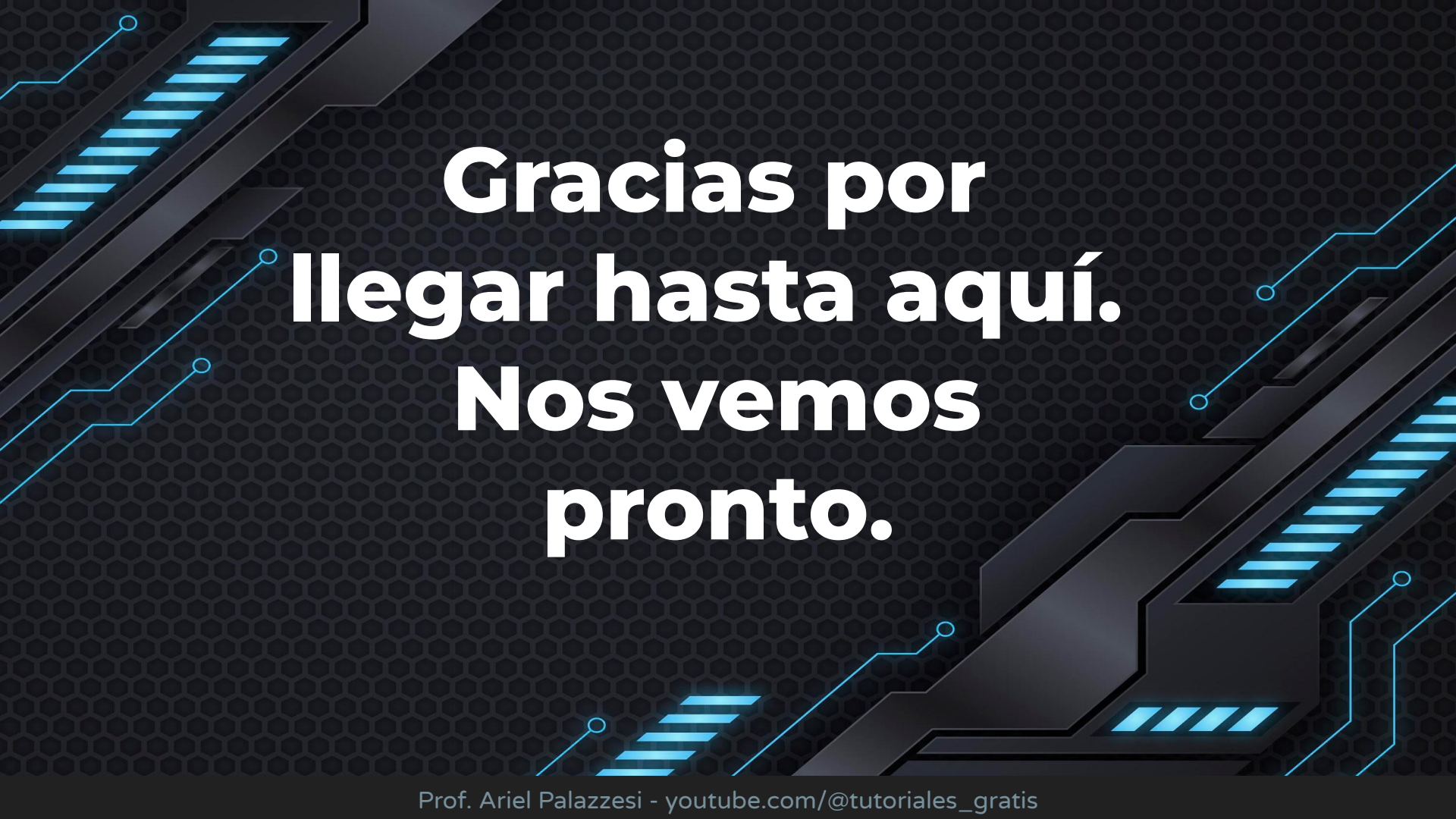
✓ Desafío 2: Administrador de tareas

Crea un programa que funcione como una lista de tareas pendientes. El programa debe permitir al usuario:

- Agregar una tarea a la lista.
- Eliminar una tarea por su nombre si ya está en la lista.
- Mostrar todas las tareas pendientes.
- Limpiar todas las tareas si el usuario lo decide.

El programa debe ejecutarse en un bucle hasta que el usuario elija salir.





**Gracias por
llegar hasta aquí.
Nos vemos
pronto.**