Puntos Clave

1. La lista **es un tipo de dato** en Python que se utiliza para **almacenar múltiples objetos**. Es una **colección ordenada y mutable** de elementos separados por comas entre corchetes, por ejemplo:

```
my_list = [1, None, True, "Soy una cadena", 256, 0]
```

2. Las listas se pueden **indexar y actualizar** , por ejemplo:

```
my_list = [1, None, True, 'Soy una cadena', 256, 0]
print(my_list[3])  # salida: Soy una cadena
print(my_list[-1])  # salida: 0

my_list[1] = '?'
print(my_list)  # salida: [1, '?', True, 'Soy una cadena', 256, 0]

my_list.insert(0, "primero")
my_list.append("último")
print(my_list)  # outputs: ['primero', 1, '?', True, 'Soy una
cadena', 256, 0, 'último']
```

3. Las listas pueden estar **anidadas**, por ejemplo:

```
my_list = [1, 'a', ["lista", 64, [0, 1], False]]
```

Aprenderás más sobre el anidamiento en el módulo 3.7; por el momento, solo queremos que sepas que algo como esto también es posible.

4. Los elementos de la lista y las listas se pueden **eliminar**, por ejemplo:

```
my_list = [1, 2, 3, 4]
del my_list[2]
print(my_list)  # salida: [1, 2, 4]
```

```
del my_list  # borra la lista entera
```

Nuevamente, aprenderás más sobre esto en el módulo 3.6, no te preocupes. Por el momento, intenta experimentar con el código anterior y verifica cómo cambiarlo afecta la salida.

5.Las listas pueden ser **iteradas** mediante el uso del bucle `for`, por ejemplo:

```
my_list = ["blanco", "purpura", "azul", "amarillo", "verde"]

for color in my_list:
    print(color)
```

6. La función `len()` se puede usar para **verificar la longitud de la lista**, por ejemplo:

```
my_list = ["blanco", "purpura", "azul", "amarillo", "verde"]
print(len(my_list))  # salida 5

del my_list[2]
print(len(my_list))  # salida 4
```

7. Una invocación típica de **función** tiene el siguiente aspecto: `result = function(arg)`, mientras que una invocación típica de un **método** se ve así: `result = data.method(arg)`.