📌 LECCIÓN 6: LISTAS, TUPLAS, DICCIONARIOS Y ARCHIVOS

🎯 Objetivo:

Aprender a manejar las estructuras de datos básicas de Python y realizar operaciones elementales de lectura y escritura de archivos de texto.

🔷 1. ¿Qué son las listas?

Una lista es una **secuencia ordenada** y **mutable** (puedes cambiarla) que admite elementos repetidos y de cualquier tipo. Internamente mantiene un índice: el primer elemento es 0, el segundo 1, etc.

Analogía cotidiana

Piénsala como tu **lista de compras** en el celular:

- Está ordenada (decides qué va primero).
- Puedes tachar, añadir o mover productos cuando quieras.
- Puedes tener el mismo producto dos veces si te hace falta.

⚙️ Propiedades de las listas (list)

Definición	colección ordenada y mutable.
Creación	numeros = [4, 7, 3]
Índice	numeros[0] # 4
Métodos básicos	append(x) – añade al final.
	remove(x) – elimina la primera aparición.
	len(lista) – longitud.
Funciones útiles	sum(lista), sorted(lista).

Recuerda que las listas en Python son una forma de **guardar varios datos en una sola variable**, y **puedes cambiarlas** fácilmente. Una lista en Python es como una **lista de cosas en tu cuaderno o celular**, por ejemplo, una lista de compras:

```
compras = ["pan", "leche", "huevos"]
```

Cada elemento tiene una **posición** (índice):

Índice	Elemento
0	"pan"
1	"leche"
2	"huevos"

🔷 ¿Cómo se usa una lista?

1. Ver un elemento de la lista

Para ver el primer elemento:

```
print(compras[0])  # Imprime "pan"
```

2. Agregar un elemento

Para añadir algo al final:

```
compras.append("frutas")
print(compras)
```

Resultado: ["pan", "leche", "huevos", "frutas"]

3. Eliminar un elemento

Para quitar algo:

```
compras.remove("leche")
print(compras)
```

Resultado: ["pan", "huevos", "frutas"]

4. Ver cuántos elementos hay

```
print(len(compras))  # Devuelve 3
```

5. Ordenar la lista alfabéticamente (sin cambiar la original)

```
ordenada = sorted(compras)
print(ordenada)
```

🧠 Ejercicio simple para practicar

```
mi_lista = []

# Paso 1: Agrega 3 frutas
mi_lista.append("manzana")
mi_lista.append("banana")
mi_lista.append("pera")

# Paso 2: Muestra la lista completa
print("Mi lista:", mi_lista)

# Paso 3: Elimina la banana
mi_lista.remove("banana")
print("Lista actualizada:", mi_lista)

# Paso 4: Imprime cuántos elementos quedan
print("Cantidad de elementos:", len(mi_lista))
```

📌 **Recuerda:**

- Las listas **se escriben entre corchetes []**.
- Puedes **guardar números, textos o mezclarlos**.
- Los elementos se pueden **agregar, quitar, modificar o contar**.

🖍️ **Mini reto:**

Crea una lista llamada “colores” con tus 4 colores favoritos.
Luego:

- Muestra el primer y último color.
- Elimina uno de ellos.
- Agrega un nuevo color al final.
- Imprime la lista completa y su longitud.

🔷 2. ¿Qué son las tuplas?

Una tupla es una **secuencia ordenada e inmutable**. Una vez creada, no puedes agregar ni quitar elementos. Se usa para **agrupar** valores que forman una unidad lógica y no deberían cambiar.

Analogía cotidiana

Imagina las **coordenadas (latitud, longitud)** de tu casa o la **fecha de nacimiento** (día, mes, año). Son datos que se guardan juntos y, en circunstancias normales, no cambian.

⚙️ Propiedades de las tuplas (tuple)

Definición	colección ordenada e **inmutable**.
Creación	punto = (3, 5)
Uso típico	devolver varios valores de una función o proteger datos que no deben cambiar.

Recuerda que una **tupla** en Python es **como una lista**, pero con una gran diferencia: **NO puedes cambiarla una vez creada**. Es decir: no puedes agregar, borrar ni modificar sus elementos.

🔧 ¿Cómo se usa una tupla?

1. Creación

```
coordenada = (4.657, -74.093)
```

2. Acceder a un valor

Igual que en una lista, por su posición (empezando desde 0):

```
print(coordenada[0])  # Muestra 4.657
```

3. No puedes modificarla

Esto da error:

```
coordenada[1] = -74.000  # ❌ ERROR
```

🧠 Ejercicio simple para practicar

Las tuplas son muy útiles para devolver varios resultados de una función:

```
def resumen_notas():
    return (3.5, 4.2, 5.0)  # devuelvo tres valores a la vez

notas = resumen_notas()
print("Tus notas son:", notas)
```

✅ ¿Cuándo usar tuplas?

Caso	¿Lista o tupla?
Lista de tareas del día	Lista 🟢
Fecha de nacimiento o coordenadas GPS	Tupla 🔵

Caso	¿Lista o tupla?
Resultados fijos que no deben modificarse	Tupla 🔵
Alumnos inscritos (que se pueden actualizar)	Lista 🟢

🖍️ Mini reto:

Crea una tupla “horario” con tres elementos: la hora, el minuto y si es AM o PM.

Ejemplo: (8, 30, "AM").

Imprime el mensaje:
"Tu clase comienza a las 8:30 AM" usando los elementos de la tupla.

🔷 3. ¿Qué es un diccionario?

Un diccionario es una **colección de pares clave-valor**; cada clave es única y se usa para encontrar rápidamente su valor. Desde Python 3.7 mantiene el **orden de inserción**. Es **mutable**: puedes añadir, modificar o eliminar entradas.

Analogía cotidiana

Piensa en la **agenda de contactos** del teléfono: el nombre (clave) apunta al número (valor). Buscas “Ana” y obtienes su teléfono sin leer toda la lista.

⚙️ Propiedades de los diccionarios (dict)

Definición	pares clave-valor, sin orden garantizado (en versiones ≥3.7 sí mantienen orden de inserción).
Creación	persona = {"nombre": "Ana", "edad": 22}
Acceso	persona["nombre"] → "Ana"
Métodos básicos	keys(), values(), items()
	get(clave, valor_defecto)

🔧 Operaciones básicas con diccionarios

```
# Crear un diccionario
persona = {"nombre": "Carlos", "edad": 16}
```


```
# Acceder a un valor
print(persona["nombre"])  # Carlos

# Agregar o modificar
persona["curso"] = "10A"

# Eliminar
del persona["edad"]

# Recorrer claves y valores
for clave, valor in persona.items():
    print(clave, "→", valor)
```

🧠 **Ejercicio simple para practicar**

```
# Diccionario de contactos: claves = nombres, valores = teléfonos

contactos = {"Ana":  "321-555-1234", "Luis": "300-111-2222"}

print(contactos["Ana"]) # Accede solo al número de Ana

contactos["Pedro"] = "311-000-9999" # Añadir nuevo contacto

for nombre, numero in contactos.items():
    print(nombre, "→", numero)

# 1) contactos.items() produce una lista de tuplas (clave, valor), por ejemplo, [("Ana", "321-555-1234"), ("Luis", "300-111-2222"), ...]
# 2) En el for, cada tupla se "desempaqueta" en las variables nombre y numero
```

Puntos clave:

1. items() devuelve cada par **(clave, valor)** del diccionario.
2. El for puede asignar esos dos elementos directamente a dos variables (nombre, numero); así se **crean y actualizan** en cada vuelta, sin definirlas antes.

🎯 **¿Cuándo usar un diccionario?**

Situación	¿Usar diccionario?
Buscas valores por una clave única	✅ Sí
Necesitas cambiar valores fácilmente	✅ Sí

Situación	¿Usar diccionario?
El orden importa mucho	⚠️ Desde Python 3.7 sí guarda el orden

---

🔷 4. Conjunto (set) (opcional, pero útil)

Colección **no ordenada** de elementos **únicos**. Sirve para eliminar duplicados y para operaciones de teoría de conjuntos (unión, intersección...).

Analogía cotidiana

Tu **colección de cromos**: nunca guardas dos iguales y el orden te da igual.

📝 Ejemplo:

```
cromos = {"Messi", "Mbappé", "Rodrygo", "Messi"}
print(cromos)        # {'Rodrygo', 'Messi', 'Mbappé'}
```

🧠 Ejercicio simple para practicar

1. Crea una lista con nombres repetidos.

   Ejemplo: ["Ana", "Luis", "Ana", "Pedro", "Luis"]

2. Convierte la lista en un conjunto usando “set()” para eliminar duplicados.

3. Muestra el resultado.

```
nombres = ["Ana", "Luis", "Ana", "Pedro", "Luis"]
sin_repetidos = set(nombres)
print("Nombres únicos:", sin_repetidos)
```

💬 *Reflexiona:* ¿Por qué el orden no se conserva? ¿Qué ventaja tiene usar un set?

🔷 5. Funciones integradas clave

Función	Qué hace	Ejemplo en código	Ejemplo cotidiano
len()	Devuelve la longitud	len([1,2,3]) → 3	¿Cuántos ítems hay en la lista de compras?
sum()	Suma numérica	sum([4,5]) → 9	Sumar precios para ver el total que gastarás
sorted()	Devuelve una copia ordenada	sorted([3,1,2]) → [1,2,3]	Ordenar alfabéticamente los apellidos de un grupo

Función	Qué hace	Ejemplo en código	Ejemplo cotidiano
max() / min()	Mayor / menor elemento	max([7,2]) → 7	Nota más alta o más baja de un examen

---

🔷 6. Manejo de archivos de texto

Un archivo de texto es como una **libreta** guardada en disco. Leer un archivo es abrir la libreta y hojearla; escribir es anotar algo nuevo. Para el manejo de archivos de texto, Python trae la función open() en su biblioteca estándar; basta con especificar **modo** (***"r", "w", "a"...***).

Analogía cotidiana

- **Lectura ("r")**: sacas la libreta, la lees y la guardas intacta.
- **Escritura ("w")**: abres la libreta en la primera página y la reescribes desde cero.
- **Añadir ("a")**: escribes una nota al final, sin borrar lo anterior.

📘 Diagrama de flujo simple para manejo de archivos:

1. Usuario escribe algo →
2. Python abre archivo en modo `"w"` o `"a"` →
3. Python escribe el contenido →
4. Se cierra el archivo automáticamente con “with” →
5. Python lo abre de nuevo en modo "r" para leerlo →
6. El contenido se muestra en pantalla.

⚙️ Apertura y cierre manual

```
archivo = open("datos.txt", "r")  # "r" = read
contenido = archivo.read()
archivo.close()
```

⚙️ Context manager (with) – recomendado

```
with open("datos.txt", "r") as f:
    contenido = f.read()      # f ya está cerrado al salir del bloque
```

⚙️ Modos comunes

Modo	Significado
"r"	Solo lectura (archivo debe existir)
"w"	Escritura, **sobrescribe** si existe

Modo	Significado
"a"	Añadir al final (append)
"r+"	Lectura y escritura

📝 **Ejemplo:**

```
# Escribir
with open("diario.txt", "w", encoding="utf-8") as f:
    f.write("Hoy aprendí listas y diccionarios.\n")

# Leer
with open("diario.txt", "r", encoding="utf-8") as f:
    contenido = f.read()
    print(contenido)
```

🧠 **Ejercicio simple para practicar**

📝 **Diario acumulativo:**

1. Pide al usuario que escriba un mensaje para su diario.
2. Abre el archivo `diario.txt` en modo `"a"` para **agregar** la entrada sin borrar las anteriores.
3. Muestra todo el contenido actualizado del archivo.

```
entrada = input("¿Qué quieres agregar al diario hoy? ")

with open("diario.txt", "a", encoding="utf-8") as f:
    f.write(entrada + "\n")

with open("diario.txt", "r", encoding="utf-8") as f:
    print("Contenido completo del diario:")
    print(f.read())
```

💬 *Reflexiona:* ¿Qué diferencias hay entre usar "a" y "w"?

🎯 ACTIVIDADES DE APRENDIZAJE

A continuación, se presentan una serie de **actividades guiadas para estudiantes principiantes**, organizadas por tipo de estructura de datos. Para cada actividad deberá realizar **pseudocódigo o diagrama de flujo previo.** Cada ejercicio posee una orientación paso a paso y utiliza ejemplos contextualizados.

📌 Ejercicio 1: Mi lista de compras 🛒

Objetivo: Crear y modificar una lista con operaciones básicas.

Antes de codificar:

✏️ Dibuja un diagrama que muestre la lista y cómo se modificará tras cada paso (agregar, eliminar, ordenar).

Instrucciones:

1. Crea una lista llamada compras con tres productos de tu elección.
2. Agrega otro producto con append().
3. Elimina uno con remove().
4. Muestra la lista original y su versión ordenada alfabéticamente.

📌 *Reflexiona:* ¿Qué pasa si tratas de eliminar un producto que no existe?

📌 Ejercicio 2 – Datos inmutables 🗓️

Objetivo: Usar tuplas para representar información que no cambia.

Antes de codificar:

✏️ Representa con un diagrama cómo una fecha está compuesta por día, mes y año.

Instrucciones:

1. Crea una tupla nacimiento con tu fecha de nacimiento: (día, mes, año).
2. Muestra el mes accediendo al segundo valor de la tupla.
3. Intenta cambiar el año y observa lo que ocurre.

📌 *Reflexiona:* ¿Por qué las tuplas son útiles para guardar este tipo de datos?

📌 Ejercicio 3 – Agenda telefónica 📱

Objetivo: Crear un diccionario y recorrer sus elementos.

Antes de codificar:

✏️ Haz una tabla con nombres y teléfonos. Representa cómo se accede a un número usando el nombre.

Instrucciones:

1. Crea un diccionario agenda con al menos tres contactos.
2. Agrega un nuevo contacto.
3. Imprime solo el número de una persona.
4. Usa un for para mostrar todos los nombres y números.

📌 *Reflexiona:* ¿Qué ocurre si cambias un número? ¿Y si repites un nombre?

📌 Ejercicio 4 – Contador de notas 🖊️

Objetivo: Aplicar len(), sum() y max() sobre una lista de números.

Antes de codificar:

🖊️ Escribe en tu cuaderno una lista con cinco notas y calcula a mano el promedio y la nota más alta.

Instrucciones:

1. Crea una lista con cinco notas numéricas.
2. Calcula el promedio usando sum() y len().
3. Muestra la nota más alta y la más baja.

📌 *Reflexiona:* ¿Por qué es importante usar funciones como max() y min()?

📌 Ejercicio 5 – Diario de aprendizaje 📘

Objetivo: Escribir y leer un archivo de texto.

Antes de codificar:

🖊️ Dibuja un esquema donde un usuario escribe una línea que se guarda en el archivo y luego se muestra.

Instrucciones:

1. Pide al usuario una frase sobre lo que aprendió hoy.
2. Guárdala en un archivo llamado diario.txt.
3. Luego, lee e imprime el contenido del archivo.

📌 *Reflexiona:* ¿Qué pasaría si usas "a" en lugar de "w" como modo de apertura?

🧠 Recomendación final para estudiantes

- Antes de programar, **piensa en el problema como si fueras a resolverlo en papel**.
- Comienza por **representar gráficamente** lo que ocurre con listas, tuplas o archivos.
- Prueba tus programas varias veces con **diferentes datos** para verificar su comportamiento.
- Haz **pequeños cambios** a tu código para comprobar y comprender su funcionamiento sin dañar. En caso de ser necesario inhabilita parte del código seleccionando el fragmento y usando # en cada

línea (CTRL + K y luego CTRL + C). Si quieres añadir un comentario recuerda iniciar con # la palabra o frase que vayas a añadir.

¡Excelente trabajo! Con esta unidad finalizada, al fin podremos comenzar a crear algunas aplicaciones interesantes. ¡Felicitaciones! 🚀😊

🖊️ **Autor:** L. Nova
📅 **Fecha de creación:** 23 de julio de 2025

ORIENTACIONES PARA LOS DOCENTES

Ejercicio 1

```
compras = ["pan", "leche", "huevos"]
compras.append("frutas")
compras.remove("leche")
print("Lista actual:", compras)
print("Ordenada:", sorted(compras))
```

Ejercicio 2

```
nacimiento = (15, 6, 2006)
print("Mes de nacimiento:", nacimiento[1])
# nacimiento[2] = 2007  ← Esto dará error
```

Ejercicio 3

```
agenda = {"Sofía": "3101112233", "Carlos": "3202223344", "Julia": "3113334455"}
agenda["Marta"] = "3124445566"
print("Tel. de Carlos:", agenda["Carlos"])

for nombre, numero in agenda.items():
    print(nombre, "→", numero)
```

Ejercicio 4

```
notas = [3.5, 4.2, 3.8, 4.7, 4.0]
promedio = sum(notas) / len(notas)
print("Promedio:", promedio)
print("Máxima:", max(notas), "Mínima:", min(notas))
```

Ejercicio 5

```
entrada = input("¿Qué aprendiste hoy? ")

with open("diario.txt", "w", encoding="utf-8") as f:
    f.write(entrada + "\n")

with open("diario.txt", "r", encoding="utf-8") as f:
    contenido = f.read()
    print("Contenido del diario:")
    print(contenido)
```