# Listas en Python

## **🧠 ¿Qué es una lista en Python?**

Una **lista** es una colección de elementos almacenados en una sola variable. Es uno de los cuatro tipos de datos integrados en Python para almacenar colecciones:

* **Lista**: ordenada, modificable, permite duplicados
* **Tupla**: ordenada, inmodificable, permite duplicados
* **Conjunto (Set)**: no ordenado, no indexado, sin duplicados
* **Diccionario**: no ordenado (aunque desde Python 3.6+ conserva el orden de inserción), pares clave-valor

## **📚 Propiedades clave de las listas**

| **Propiedad** | **Descripción** |
| --- | --- |
| **Ordenada** | Los elementos mantienen el orden en que fueron agregados |
| **Indexada** | Cada elemento tiene una posición: Mi\_lista[0] es "manzana" |
| **Modificable** | Puedes modificar, agregar o eliminar elementos |
| **Permite duplicados** | Se pueden repetir valores dentro de la lista |

## Caso de implementación:

En nuestro programa, utilizamos listas para separar cada columna del csv, y poder manejar los datos por separado donde fuese necesario y también con una lista de diccionarios conseguimos tener acceso a todo el contenido dentro del csv

## **🧠 Diccionarios en Python**

Un **diccionario** se utiliza para almacenar valores de datos en pares **clave:valor**.

Es una colección que es **ordenada** (a partir de Python 3.7), **modificable** y **no permite duplicados**.

Los diccionarios se escriben con **llaves** {}, y cada elemento tiene una clave y un valor.

## **📚 Características de los diccionarios**

| **Propiedad** | **Descripción** |
| --- | --- |
| **Ordenados** | Desde Python 3.7, los elementos mantienen el orden de inserción |
| **Modificables** | Se pueden cambiar, agregar o eliminar elementos |
| **Sin duplicados** | No se pueden tener dos claves iguales; si se repite una clave, se sobrescribe |

## **🔄 ¿Ordenado o no ordenado?**

* **Python 3.7 en adelante**: los diccionarios son **ordenados**.
* **Python 3.6 y versiones anteriores**: los diccionarios son **no ordenados**.

Ordenado significa que los elementos tienen un orden definido que no cambia.  
 No ordenado significa que no se puede acceder a los elementos por índice.

## **✏️ Modificable**

Puedes cambiar, agregar o eliminar elementos después de crear el diccionario.

## **No se permiten claves duplicadas**

Si se repite una clave, el valor anterior se sobrescribe:

Diccionario = {

"marca": "Ford",

"modelo": "Mustang",

"año": 1964,

"año": 2020

}

print(Diccionario) # El valor de "año" será 2020

## Caso de implementación:

El uso del diccionario en nuestro programa es fundamental, ya que con él conseguimos tener acceso a cada una de las líneas del csv y sus datos correspondientes, para luego poder procesarlos durante el funcionamiento del programa.

## **🧠 Funciones en Python**

Una **función** es un bloque de código que solo se ejecuta cuando se llama.

Puedes pasarle datos a una función, conocidos como **parámetros**.

Una función puede **devolver datos** como resultado.

## **❓¿Parámetros o argumentos?**

Los términos **parámetro** y **argumento** pueden usarse para referirse a lo mismo: la información que se pasa a una función.

Desde la perspectiva de una función:

* Un **parámetro** es la variable que aparece entre paréntesis en la definición de la función.
* Un **argumento** es el valor que se envía a la función cuando se llama.

## Caso de implementación:

Las funciones cumplen un rol clave a la hora de poder modularizar el código , dividiendo cada una de sus funcionalidades para su reuso dentro del programa y para facilitar la lectura del código

## **🧠 Condiciones y sentencias if en Python**

Python admite las condiciones lógicas habituales de las matemáticas:

* **Igual a**: a == b
* **Distinto de**: a != b
* **Menor que**: a < b
* **Menor o igual que**: a <= b
* **Mayor que**: a > b
* **Mayor o igual que**: a >= b

Estas condiciones se pueden usar de varias formas, pero lo más común es en **sentencias if** y **bucles**.

## **🛠️ Sentencia if**

Una sentencia if se escribe usando la palabra clave if:

a = 33

b = 200

if b > a:

print("b es mayor que a")

En este ejemplo usamos dos variables, a y b, para comprobar si b es mayor que a. Como b es 200 y a es 33, se imprime en pantalla que “b es mayor que a”.

## **🔠 Identación**

Python **depende de la indentación** (espacios en blanco al comienzo de una línea) para definir el alcance del código. Otros lenguajes usan llaves {} para esto.

### **Ejemplo incorrecto (sin indentación):**

a = 33

b = 200

if b > a:

print("b es mayor que a") # Esto generará un error

## **🔄 elif**

La palabra clave elif es la forma de Python de decir: “si las condiciones anteriores no fueron verdaderas, prueba esta”.

a = 33

b = 33

if b > a:

print("b es mayor que a")

elif a == b:

print("a y b son iguales")

Como a es igual a b, la primera condición no se cumple, pero la condición elif sí, por lo que se imprime “a y b son iguales”.

## **🧲 else**

La palabra clave else captura cualquier caso que no haya sido cubierto por las condiciones anteriores.

a = 200

b = 33

if b > a:

print("b es mayor que a")

elif a == b:

print("a y b son iguales")

else:

print("a es mayor que b")

Aquí, como a es mayor que b, ni if ni elif se cumplen, así que se ejecuta el bloque else.

También puedes usar else sin elif:

a = 200

b = 33

if b > a:

print("b es mayor que a")

else:

print("b no es mayor que a")

## Caso de implementación:

La estructura “if” es fundamental para indicarle al programa como comportarse en determinadas situación y guiarlo a que ejecute lo que nosotros queremos, es algo que podemos encontrar en todas partes dentro del código ya que es una manera muy clara de evaluar distintas condiciones dentro del código.

# **Funciones de python: sorted()**

La función sorted() devuelve una lista ordenada del objeto iterable especificado. Puedes especificar un orden ascendente o descendente. Las cadenas de texto se ordenan alfabéticamente y los números se ordenan numéricamente.

## Caso de implementación:

En nuestro programa, junto con la implementación de una sencilla función lambda() ,esta función nativa de python es utilizada para ordenar los datos de nuestros campos dentro del csv según el usuario así lo solicite.

# **Estadísticas básicas**

# **Archivo CSV**

Un archivo **CSV** (del inglés *Comma-Separated Values*, o **Valores Separados por Comas**) es un formato de archivo de texto plano muy simple que se utiliza para almacenar datos en una estructura de tabla, similar a una hoja de cálculo o una tabla de base de datos.

Es uno de los formatos más comunes para intercambiar datos entre diferentes aplicaciones.

### **📋 Características Principales**

* **Texto Plano:** Puedes abrir, leer e incluso editar un archivo CSV con cualquier editor de texto básico (como el Bloc de Notas en Windows, TextEdit en Mac o Gedit en Linux).
* **Estructura:**
  + Cada **línea** del archivo representa una **fila** de la tabla.
  + Los **valores** (columnas) dentro de cada fila están separados por un **delimitador**.
* **Delimitador:** Aunque el nombre "CSV" implica que el separador es una **coma (,)**, en la práctica se usan varios delimitadores:
  + **Coma (,)**: El estándar más común (ej. Juan,Pérez,30).
  + **Punto y coma (;)**: Muy común en regiones donde la coma se usa como separador decimal (ej. Juan;Pérez;30,5). Excel en español a menudo usa este por defecto.
  + **Tabulación (\t)**: A estos se les suele llamar archivos **TSV** (Tab-Separated Values).

## Caso de implementación:

Pieza fundamental del trabajo, ya que en un archivo csv se van a encontrar los datos que vamos a utilizar para que el programa funcione y despliegue toda su funcionalidad, y con el uso de la librería csv de python conseguimos extraerlo y procesarlos.