«Talento Tech»

Iniciación a la Programación con Python

CLASE 13

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 13 | Bases de datos

Temario:

- Concepto y utilidad de los módulos en Python
- Introducción a bases de datos.
- Idea de tabla, campo, índice, clave, etc.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Introducción a los módulos en Python.

En esta clase, vamos a ver cómo trabajar con bases de datos en Python para que la información que guardemos pueda persistir en el tiempo. Hasta ahora, veníamos usando un diccionario para almacenar los datos, pero cuando cerrábamos el programa, esos datos se perdían. Las bases de datos nos permiten guardar la información de manera permanente.

Pero antes de eso, hay algo clave que necesitamos conocer: **los módulos en Python.** Un módulo es simplemente un conjunto de funciones y variables que otros pueden reutilizar. Python viene con muchos módulos incorporados, lo que significa que ya están listos para ser usados sin necesidad de instalarlos.

Los módulos son muy útiles porque nos permiten no tener que reinventar la rueda cada vez que necesitemos hacer algo. Si Python ya tiene funciones creadas para hacer algo, ¡aprovechemos eso! Vamos a ver cómo podemos importar y usar módulos en nuestro programa.

¿Cómo importar módulos?

Para poder usar un módulo primero tenemos que importarlo. La sintaxis es simple:

```
import nombre_del_modulo
```

Una vez que lo importamos, podemos acceder a las funciones que contiene ese módulo escribiendo **nombre_del_modulo.funcion()**. Veamos algunos ejemplos prácticos usando dos módulos muy comunes que ya vienen con Python: *math y random.*

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejemplo 1: El módulo *math.*

El módulo math contiene muchas funciones matemáticas disponibles. Vamos a importarlo y usar algunas de sus funciones.

```
import math

# Redondear hacia abajo
numero = 4.7
redondeado = math.floor(numero)
print("El número redondeado hacia abajo es:", redondeado)

# Calcular la raíz cuadrada de un número
raiz = math.sqrt(16)
print("La raíz cuadrada de 16 es:", raiz)
```

En este ejemplo, utilizamos **math.floor()** para redondear un número hacia abajo y **math.sqrt()** para calcular la raíz cuadrada de un número.

```
El número redondeado hacia abajo es: 4
La raíz cuadrada de 16 es: 4.0
```

El módulo math nos proporciona muchas funciones útiles para hacer cálculos matemáticos sin tener que escribir por nuestra cuenta las fórmulas.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejemplo 2: El módulo *random.*

El módulo random nos permite generar números aleatorios, lo cual es muy útil en muchos programas. Veamos cómo usarlo.

```
import random

# Generar un número aleatorio entre 1 y 10
numero_aleatorio = random.randint(1, 10)
print("Número aleatorio entre 1 y 10:", numero_aleatorio)

# Seleccionar un elemento aleatorio de una lista
colores = ["rojo", "verde", "azul", "amarillo"]
color_aleatorio = random.choice(colores)
print("Color aleatorio:", color_aleatorio)
```

Acá usamos **random.randint**(1, 10) para generar un número aleatorio entre 1 y 10. También utilizamos **random.choice()** para elegir un elemento aleatorio de una lista.

```
Número aleatorio entre 1 y 10: 8
Color aleatorio: azul
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Importar sólo una parte de un módulo.

A veces no necesitamos todo el módulo, sino sólo una función específica. En lugar de importar todo, podemos importar sólo lo que nos interesa. La sintaxis es la siguiente:

```
from nombre_del_modulo import nombre_de_la_funcion
```

Veamos un ejemplo con el módulo math:

```
from math import sqrt

# Ahora podemos usar sqrt directamente sin escribir math.
raiz = sqrt(25)
print("La raíz cuadrada de 25 es:", raiz)
```

En este caso, importamos únicamente la función sqrt del módulo math, lo que nos permite usarla directamente sin tener que escribir math.sqrt() cada vez.

Ahora que sabemos cómo importar módulos y usarlos, estamos listos para trabajar en la próxima clase con el módulo que nos permitirá interactuar con bases de datos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Bases de datos.

Una base de datos es un sistema que nos permite almacenar y organizar información de manera estructurada. A diferencia de lo que hacíamos en la clase 12, donde usábamos diccionarios en Python para guardar los productos del inventario, las bases de datos nos permiten conservar esa información de manera permanente. Esto significa que, aunque apaguemos el programa o la computadora, los datos siguen estando ahí, listos para ser consultados cuando los necesitemos nuevamente.

Los diccionarios son útiles para trabajar con datos temporales. Es decir, podemos almacenar y acceder a la información mientras el programa esté corriendo, pero una vez que cerramos el programa, todo lo que estaba guardado en el diccionario desaparece. Por eso, los diccionarios nos sirven cuando trabajamos con pequeñas cantidades de información o cuando no necesitamos que los datos se mantengan disponibles después de cerrar el programa.

En cambio, **las bases de datos nos permiten guardar grandes cantidades de información y mantenerlas accesibles en cualquier momento.** Es como si tuviéramos una libreta o un archivo donde podemos anotar todo lo que necesitamos, evitando correr el riesgo de perderlo o minimizándolo. Además, nos permiten llevar adelante varias funcionalidades que serían complicadas de gestionar por medio de los diccionarios, como buscar registros específicos rápidamente, actualizar datos de manera eficiente o eliminar información de manera controlada. También podemos trabajar con varios usuarios y usuarias al mismo tiempo, manteniendo todo organizado y seguro.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Tablas.

En una base de datos, la información se organiza en tablas. Podemos imaginar una tabla como una hoja de cálculo donde cada fila representa un registro (como un producto del inventario) y cada columna representa un campo (por ejemplo, el nombre del producto, el precio o la cantidad disponible). Esta estructura es muy práctica porque facilita la búsqueda, el filtrado y la modificación de datos.

ID Producto	Nombre	Descripción	Cantidad	Precio	Categoría
1	Manzana	Fruta fresca	50	0.5	Frutas
2	Pan	Pan casero	20	1.0	Panadería
3	Leche	Leche descremada	100	0.75	Lácteos
4	Jugo	Jugo de naranja natural	30	1.5	Bebidas

Ejemplo simple de una tabla

Campos y registros en las tablas.

En una base de datos, un **campo** es una columna dentro de una tabla que almacena un tipo específico de información sobre los **registros** que están en esa tabla. Por ejemplo, si pensamos en una tabla de productos, cada producto sería un registro y cada campo almacenaría una característica específica de ese producto, como *su nombre, cantidad o precio.*

Los tipos de datos que podemos almacenar en los campos de una tabla son importantes porque determinan qué clase de información puede guardarse en ese campo. Al definir una tabla, tenemos que indicar qué tipo de dato se acepta en cada campo para asegurarnos de que la información se almacene correctamente.

Estos son algunos de los tipos de datos más comunes y útiles para un proyecto de gestión de inventario:

Texto: Este tipo de dato se utiliza para almacenar cadenas de texto. Es útil cuando queremos guardar nombres de productos, descripciones o categorías. Por ejemplo, podríamos tener un campo llamado "nombre" para guardar el nombre del producto ("Manzana", "Leche", etc.) y otro llamado "descripción" para una breve descripción del producto.

Números enteros: Este tipo de dato se utiliza para almacenar números enteros. Es ideal para cantidades de productos o para campos que necesiten valores numéricos sin decimales, como el código de un producto (por ejemplo, un ID único que identifique cada producto). Un campo de tipo entero sería perfecto para registrar cuántas unidades de cada producto tenemos en el inventario.

Números de punto flotante: Cuando necesitamos almacenar precios o valores que incluyan decimales, usamos un tipo de dato que permita números con fracciones. En un sistema de inventario, el precio de un producto es un ejemplo de un valor que suele requerir decimales.

Fechas y horas: A veces es necesario registrar cuándo ocurrió algo, como la fecha en que un producto fue agregado al inventario o la última vez que se actualizó su cantidad. Para esto, usamos un tipo de dato que pueda manejar fechas y horas. Estos campos son útiles para llevar un control de las acciones que realizamos en la base de datos.

Al crear una tabla, decidimos qué tipo de datos vamos a almacenar en cada campo según el tipo de información que necesitamos guardar. De esta forma nos aseguramos de que la base de datos esté organizada y opere con normalidad.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Campos clave.

Ahora vamos a hablar sobre un concepto clave en las bases de datos: el **campo clave**, o también conocido como **clave primaria**. Este campo es uno de los elementos más importantes cuando diseñamos una tabla en nuestra base de datos.

Un campo clave es un campo especial dentro de una tabla que se utiliza para identificar de manera **única** cada registro. Imaginemos que tenemos una tabla donde guardamos todos los productos del inventario. Cada producto tiene su propio nombre, precio, cantidad, etc., pero necesitamos un dato que permita diferenciar cada producto de manera única. Para esto, usamos una clave primaria.

La clave primaria tiene dos características principales:

Unicidad: No puede haber dos registros en la tabla que tengan el mismo valor en el campo clave. Esto es lo que permite identificar de manera única cada producto en el inventario. Por ejemplo, en nuestra tabla de productos, podríamos usar un campo llamado **"ID Producto"**, que asigna un número único a cada producto. Así, aunque tengamos dos productos llamados "Pan", cada uno tendría un número de ID diferente, lo que nos permite diferenciarlos.

No puede ser nulo: El campo clave debe tener siempre un valor. No puede quedar vacío. Si no existiera una clave para un registro, no tendríamos manera de identificarlo claramente dentro de la tabla, lo que generaría problemas al momento de buscar o actualizar información.

Por ejemplo, en nuestra tabla de productos, podemos tener un campo ID Producto que actúe como clave primaria. Supongamos que tenemos estos productos.

ID Producto	Nombre	Cantidad	Precio
1	Manzana	50	0.5
2	Pan	20	1.0
3	Jugo	30	1.5

En este ejemplo, el campo ID Producto es la clave primaria. Cada producto tiene un número único que lo diferencia de los demás, incluso si otros datos, como el nombre, llegaran a repetirse. El hecho de que sea único y obligatorio nos asegura que podemos identificar cualquier producto en la tabla sin confusión.

Una de las mayores ventajas de utilizar una clave primaria es que nos permite realizar búsquedas de manera rápida. Al tener un identificador único para cada registro, podemos encontrar lo que buscamos sin tener que comparar otros campos. Además, nos brinda seguridad al momento de actualizar datos. Si necesitamos cambiar la cantidad de un producto o modificar su precio, podemos hacerlo con la tranquilidad de que estamos editando el registro correcto, ya que está identificado de forma única. Por otro lado, la clave primaria también garantiza la **integridad de los datos.** Al no permitir valores repetidos o vacíos, ayuda a mantener la tabla organizada y libre de errores. Esto hace que sea una herramienta casi indispensable para la correcta gestión de los datos, sobre todo cuando trabajamos con grandes volúmenes de información, como en el caso de un sistema de inventario.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Del diccionario a la base de datos.

En la clase anterior decidimos que cada producto se almacenará dentro de un diccionario llamado inventario. En este, usamos un código numérico único (la clave) y los detalles del producto (nombre, descripción, cantidad, etc.) que estaban asociados a esa clave en otro diccionario. Esta era su estructura:

```
inventario = {
    1: {
        "nombre": "Manzana",
        "descripcion": "Fruta fresca y deliciosa",
        "cantidad": 50,
        "precio": 0.5,
        "categoria": "Frutas"
    },
    2: {
        "nombre": "Pan",
        "descripcion": "Pan casero recién horneado",
        "cantidad": 20,
        "precio": 1.0,
        "categoria": "Panadería"
    }
}
```

Hasta esta clase la estructura “diccionario” era nuestra única herramienta para implementar el guardado de nuestros datos. Veamos cómo reemplazarla por la base de datos.

Podemos transformar nuestro diccionario en una tabla de base de datos. Para eso, vamos a definir la estructura de la tabla teniendo en cuenta los datos que estamos guardando en el diccionario actual. La tabla se llamará **Productos**, un nombre descriptivo que representa

claramente el tipo de información que va a almacenar: cada fila será un producto del inventario y cada columna, un dato específico sobre ese producto.

Cada elemento del diccionario se convertirá en un campo dentro de la tabla. Primero, necesitamos un campo llamado **ID Producto**, que usaremos como clave primaria para identificar de manera única a cada producto. En el diccionario, este valor está representado por codigo_actual, que se incrementa cada vez que se registra un producto nuevo. En la tabla, el campo ID Producto será de tipo entero y tendrá la propiedad de incremento automático, lo que asegura que cada nuevo registro tenga un ID único sin necesidad de asignarlo manualmente. Este campo también se establece como clave primaria, de manera que ningún producto podrá tener el mismo ID que otro, garantizando que cada registro sea único e inconfundible.

El siguiente campo, **Nombre**, almacena el nombre del producto, como "Manzana" o "Jugo". Para este campo, usaremos el tipo de dato texto, ya que necesitamos almacenar cadenas de caracteres. Este dato es fundamental para el registro del producto, así que no puede quedar vacío.

La **Descripción** del producto será otro campo de tipo texto. Aquí se guarda una breve explicación sobre el producto, como “Fruta fresca” o “Bebida de naranja”. Este campo puede ser opcional, ya que sirve solo para dar información adicional y no es un dato esencial para el funcionamiento de la tabla.

El campo **Cantidad** indica la cantidad disponible de cada producto. Usaremos un tipo de dato entero, ya que solo necesitamos números sin decimales. Es importante que este campo tenga siempre un valor válido, así que lo establecemos como obligatorio.

El **Precio** del producto será otro campo, y al igual que la cantidad, es un dato que debe ser registrado obligatoriamente. En este caso, usaremos un tipo de dato que permita decimales, ya que los precios suelen incluir fracciones. Para esto, utilizamos un tipo de dato real que nos permite almacenar valores como 1.5 o 0.75.

Finalmente, el campo **Categoría** servirá para clasificar los productos, y será de tipo texto. Esto permite que agrupemos productos similares, como en las categorías "Frutas", "Bebidas" o "Lácteos". La categoría no es un campo esencial para el inventario, por lo que no requerimos que tenga siempre un valor.

A partir de todas estas decisiones, esta podría ser la estructura final de nuestra tabla:

Campo	Tipo de dato	Propiedades
ID Producto	INTEGER	PRIMARY KEY, AUTOINCREMENT
Nombre	TEXT	NOT NULL
Descripción	TEXT	
Cantidad	INTEGER	NOT NULL
Precio	REAL	NOT NULL
Categoría	TEXT	

NOT NULL indica que el campo no puede quedar vacío. En este caso, *Nombre, Cantidad y Precio* son datos esenciales, por lo que siempre deben tener un valor asignado.
AUTOINCREMENT hace que el ID se asigne automáticamente cada vez que se registra un nuevo producto, asegurando que sea único y evitando duplicados.
Con estas decisiones, nuestra tabla Productos queda estructurada para manejar cada producto del inventario con la flexibilidad y organización necesaria, permitiéndonos en el futuro realizar búsquedas, actualizaciones y eliminaciones fácil y eficientemente. Este diseño asegura que cada registro esté claramente identificado, que se almacenen sólo los datos válidos y que la información esté siempre organizada y fácil de manejar, características que siempre debemos buscar en nuestros proyectos.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios prácticos:

Conceptualización de una tabla

Imaginá que estás a cargo de organizar una pequeña biblioteca de la escuela en una base de datos. Los datos que se quieren registrar incluyen el título del libro, el autor, la fecha de publicación y el género. Definí:

- Un nombre adecuado para la tabla.
- Los campos que incluirías en la tabla, sus tipos de datos y por qué.
- Qué campo sería la clave primaria y por qué.

Generación de valores únicos con random

Escribe un programa en Python que genere cinco códigos únicos de cinco dígitos para usarlos como identificadores de productos en un inventario. Para esto, utiliza el módulo random. Cada código generado debe ser diferente de los otros.

Tip: Puedes usar **random.randint()** para generar números dentro de un rango determinado.

Simulación de precios con math y random

Supongamos que deseás simular los precios de 10 productos en un inventario. Escribí un programa que:

- Utilice el módulo random para generar 10 precios aleatorios entre $10.00 y $100.00.
- Redondee los precios generados a dos decimales usando una función del módulo math.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad