Buenos Aires aprende
Agencia de Habilidades para el Futuro

Talento Tech

Iniciación con Python

Clase 13 - “Bases de datos”

Buenos Aires Ciudad

¡Les damos la bienvenida!

 Vamos a comenzar a grabar la clase

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Talento Tech

Clase 12.

Ruta de avance

1. Creamos las funciones necesarias para el Proyecto Integrador

Clase 13.

Bases de Datos

1. Concepto y utilidad de los módulos en Python
2. Introducción a Bases de datos.
3. Idea de tabla, campo, índice, clave, etc.

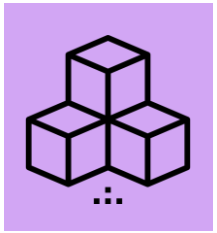
Clase 14.

Fundamentos SQL

1. Instalación y uso del módulo SQLite
2. Consultas SQL básicas desde Python con SQLite
3. Consultas SQL básicas: SELECT, INSERT, UPDATE, DELETE

Pero antes...

¡Resolvamos los “**Ejercicios prácticos**” de la clase anterior!



Módulos

Módulos en Python

- Un módulo es simplemente un conjunto de funciones y variables que otros pueden reutilizar.
- Python viene con muchos módulos incorporados, que están listos para ser usados sin necesidad de instalarlos.
- Los módulos son muy útiles: no necesitamos reinventar la rueda cada vez que necesitemos hacer algo.

¿Cómo importar módulos?

Para poder usar un módulo, primero tenemos que importarlo:

Ejemplo:

```
# Asi importamos el módulo llamado "nombre_del_modulo"
import nombre_del_modulo
```

Una vez que lo importamos, podemos acceder a las funciones que contiene ese módulo utilizando la sintaxis ***nombre_del_modulo.funcion()***

El módulo math

El módulo math contiene muchas funciones matemáticas útiles. Podemos importarlo y usar sus funciones:

Ejemplo:

```python
import math                          # Importar el módulo
numero = 4.7
redondeado = math.floor(numero)  # Redondear hacia abajo
print("El número redondeado hacia abajo es:", redondeado)

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

El módulo random

El módulo random nos permite generar números aleatorios, lo cual es muy útil en muchos programas para realizar simulaciones o crear datos de prueba.

Ejemplo:

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

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

Importación parcial

A veces no necesitamos todo el módulo, sino sólo una función específica. En esos casos podemos importar sólo lo que nos interesa:

Ejemplo:

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

La próxima clase aprenderemos a usar un módulo que nos permitirá interactuar con bases de datos.

Base de datos

Una base de datos es un sistema que nos permite almacenar y organizar información de manera estructurada. Entre muchas otras, posee estas ventajas:

- Permite almacenar información de forma permanente.
- Proporciona mecanismos para realizar búsquedas, actualizaciones y borrado de datos.
- Organiza la información en tablas, con campos y registros.
- Puede utilizar campos clave para facilitar las búsquedas.

Tablas

En una base de datos, los datos se organizan en **tablas**. Podemos pensarlas 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 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

Campos y registros

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.

CAMPO

REGISTRO

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

Tipos de datos para los campos

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. Por ejemplo, para guardar el nombre del producto ("Manzana", "Leche", etc.) o su descripción.
- **Números enteros:** Es ideal para cantidades de productos o para campos que necesiten valores numéricos sin decimales, como el código de un producto.
- **Números de punto flotante:** Ideal para almacenar precios o valores que incluyan decimales.
- **Fechas y horas:** Ideales para registrar cuándo ocurrió algo, como la fecha en que un producto fue agregado al inventario.

Campo clave

El campo clave, también conocido como clave primaria, se utiliza para identificar de manera única cada registro. Permite diferenciar cada producto de manera única. Posee dos características principales:

- **Unicidad:** No puede haber dos registros en la tabla que tengan el mismo valor en el campo clave. 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.

Diccionario vs. base de datos

En la clase anterior decidimos que cada producto se almacenará dentro de un diccionario llamado inventario:

```
inventario = {
    1: {
        "nombre": "Manzana",
        "descripcion": "Fruta fresca y deliciosa",
        "cantidad": 50,
        "precio": 0.5,
        "categoria": "Frutas"
    }
}
```

Diccionario vs. base de datos

A partir de lo que hemos aprendido en esta clase, podríamos reemplazar ese diccionario por una tabla como esta:

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	

PRIMARY KEY indica que ese campo no admite duplicados.
NOT NULL indica que el campo no puede quedar vacío.
AUTOINCREMENT hace que el ID se asigne automáticamente cada vez que se registra un nuevo producto.

¡Vamos a la práctica! 🚀

Ejercicios prácticos

Optativos | No entregables

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.

Ejercicios prácticos

Optativos | No entregables

Generación de valores únicos con random

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

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

Ejercicios prácticos

Optativos | No entregables

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.