Task**: Create simple web app in python that works with sql and store drink items in SQL table.**

Currently App implements **Web Server** side while the **Web Client** side is simulated by **Postman**

Web Server that will get REST API REQUESTS (**GET** / **GET** by **id** / **POST** / **DELETE**) from Web Client (**Postman**) and update the table drinks in my DB (**drinks**.**db**)

Task works with packages:

1. Flask
2. Flask\_sqlalchemy

Flask, **by default** uses local host ip address: [**http://127.0.0.1**](http://127.0.0.1)means that only my pc can access this URL (IP address)

SQLalchemy, by default gives a table name = name of a class that defines its db.Model

If I defined that my class that will define a drink item that I will store in table drinks in my DB drinks.db will be called: Drink

A screen shot of a computer program

AI-generated content may be incorrect.

The table that will be created is drink (same name as a class but lover case !!!)

But we can define table name explicitly by adding attribute to the class:

**\_\_tablename\_\_= ‘my\_drinks’** # see how I did in my class

There is important thing we must remember – the context of the app   
all that you wish to happen must be written in the context of the app

main is not a context of the app !!! I mean

if \_\_name\_\_ == ‘\_\_main\_\_’:

xxxxxx

xxxxx

xxx

all the xxxx are not in the context of the app therefore will not run

What we must define at the very beginning of the run of the app:

1. # 1 Create db obj - of the type SQLAlchemy (no word about the app)  
   db = SQLAlchemy()

Our DB will be based on the package SQLAlchemy

1. # 2 Create app (use Flask module)  
   app = Flask(\_\_name\_\_)

Our app is created by package Flask

1. # 3 Here we connect between DB and the app  
   # Configure the app to work with SQLALCHEMY (app will use db called: drinks.db)  
   app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///drinks.db'  
   app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = *False*
2. # 4 Initialize db with the app - this is actual binding between app and db  
   db.init\_app(app)
3. # db\_path is set with a location where will be created a db in the project  
   db\_path = "instance/drinks.db"
4. # 5 create data base model - in simple words it creates a table of the type Drink, which name = class name but in lowercase  
   # class defines the class model of the table.  
   # DB is drinks.db but it will contain 1 table inside it, called drink  
   # because we use SQLAlchemy, SQLAlchemy automatically creates a table named drink in the database.  
   # if we use this attribute in the class Drink: \_\_tablename\_\_ = 'my\_drinks' # Set the table name explicitly to be 'my\_drinks' instead of drink (table name = DB name)  
   *class* Drink(db.Model):  
    # I set here the explicit name for the table that will be created in my DB  
    # If we do not set the name explicitly, its default name will be as a class name, but in lover case (e.g: drink)  
    \_\_tablename\_\_ = 'my\_drinks'  
     
    # here I define that in table 'drinks' will be 3 columns: id, name, description  
    id = db.Column(db.Integer, primary\_key=*True*)  
    name = db.Column(db.String(80), unique=*True*,nullable=*False*)  
    description = db.Column(db.String(80))  
     
    # magic func of the class for object (self) representation  
    *def \_\_repr\_\_*(self):  
    *return* f"{self.name}-{self.description}"

Here we created the type of the each raw we will put in the table

We gona put items that are drinks

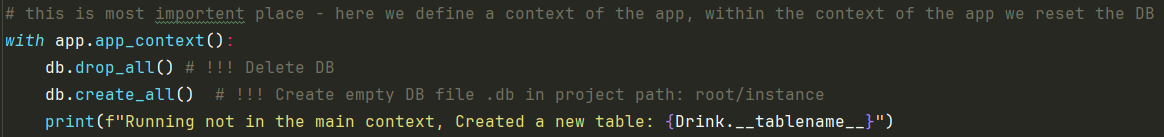
Each drink item has:

1. Id
2. Name
3. Description
4. Each time we run the app it should start from the empty table so we should delete old DB and the table
5. # remove db from the project if it created previously  
   *if* os.path.exists(db\_path):  
    os.remove(db\_path)  
    print("Old database deleted.")

Deletion must be done after we define the DB and connected between app and the db and the class model

1. The most important thing is to create a new DB – new context (as idea)

This must be done after all is ready and known



Now is the time to define hitters for our app:

These are functions that will be invoked upon requests from the WEB Client side (that currently is implemented by Postman)