# RogueLists Local Installation Instructions

*This tutorial was created using a fresh install of VS Code on a new user account using Windows 10 64-bit Home Edition. Some other technical questions may be answered within the README.txt in the root directory of the GitHub repository.*

*This tutorial assumes you have already downloaded the ‘settings.py’ file from the Moodle submission.*

## Step-By-Step

1. Download the ZIP file from the repository located at the following URL:

<https://github.com/k-bru/RogueLists>

1. Extract the contents
2. Place the settings.py file in the following directory:

‘capvenv\roguelists\roguelists\settings.py’

1. Open the folder with VS Code or another editor. (Tutorial was created using VS Code)
2. Open a cmd terminal. Shortcut is CTRL + `

*\*Default is sometimes PowerShell. The following lines MUST be executed in a standard command prompt.*

1. Execute the following commands:
2. cd capvenv

#This command activates the virtual environment

1. Scripts\activate.bat

cd roguelists

#This command activates the RogueLists application

python manage.py runserver --insecure

*\*Using the --insecure flag when running the Django development server allows static files such as CSS and JS to be served from unknown directories, making it easier to run the project locally without additional configuration. When DEBUG is turned on in Django, the --insecure flag is not necessary for serving static files because Django automatically serves them using a view designed for development purposes.*

If you have done the above steps correctly, your terminal should look as follows:

Django version 4.1.6, using settings 'roguelists.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

"GET /rogueapp/ HTTP/1.1" 200 43965

"GET /static/rogueapp/css/styles.css HTTP/1.1" 200 7292

"GET /static/rogueapp/js/app.js HTTP/1.1" 200 3687

"GET /static/rogueapp/images/rogue-logo.png HTTP/1.1" 200 21592

"GET /static/rogueapp/images/favicon.ico HTTP/1.1" 200 15406

1. Navigate to the url indicated by your terminal.

*\*Typically '*[*http://127.0.0.1:8000/*](http://127.0.0.1:8000/)*'*

1. Congratulations! You have successfully opened RogueLists in your local environment! To end the connection, press ‘CTRL + C’

## Other Notes

Standard user login:

Username:   the-quick-brown-fox

Password:    sK99Ktrp8FwAE33

Admin login:

Username:    kb

Password:    qwer1234

The web scraping script ‘*update-game-table-pythonanywhere.py*’ will not function correctly if executed locally, as the file is set up for the folder structure within the web host, PythonAnywhere.

To make the script work locally, simply change lines 23 and 24 to the following code:

23 db\_path = 'db.sqlite3'

24 csv\_path = 'roguelike.csv'

If you would like to change the search term, that string is located just below on line 28:

28 game = 'roguelike'

To execute this code, make sure your terminal is still in the virtual environment directory:

(capvenv) C:\Users\your-username\your-path-to\RogueLists-master\capvenv\roguelists>

If the site is running currently, you will either need to close the connection (CTRL+C) or open up a new instance of cmd and navigate to the directory the same way you did before.

Once you are in the directory, simply type the filename you wish to execute, in this case, its:

update-game-table-pythonanywhere.py

## Explanation of the missing typical SQL file for a Capstone submission:

From README.txt

\*Note about SQL in relation to this project:

Django uses an Object-Relational Mapping (ORM) framework, which is a technique of mapping between the relational database and the object-oriented programming language. Django provides a high-level abstraction layer that allows developers to work with databases using Python classes and methods instead of writing SQL scripts.

The models.py file in a Django app contains Python classes that represent database tables. These classes inherit from the django.db.models.Model class and define attributes that map to fields in the database table. The ORM framework takes care of translating these Python classes and their attributes into SQL queries and executing them against the database.

When Django is set up, it creates tables in the database for each of the defined models. This is done by running a series of migrations, which are scripts that Django generates based on changes to the models. These migrations are stored in a special folder within the app called migrations.

Django does not require a typical SQL script file to initiate a database because the models and the ORM handle the creation and management of database tables. The models.py file is similar to a SQL script file in that it defines the structure of the database, but it is written in Python and uses Django's ORM to interact with the database. You can find the models.py file for RogueLists in the app folder at: capvenv\roguelists\rogueapp\models.py

To see where SQL is used within the daily update web scraping script, look at lines 311-325 located at:

capvenv\roguelists\update-game-table-pythonanywhere.py

*# check if the game already exists in the database*

c.execute("SELECT COUNT(\*) FROM rogueapp\_game WHERE steam\_id=?", (steam\_id,))

result = c.fetchone()[0]

if result == 0:

*# game does not exist, insert a new record*

print(f"Inserting game {game\_title} with steam\_id {steam\_id}")

c.execute("INSERT INTO rogueapp\_game (steam\_id, game\_title, base\_price, current\_price, release\_date, genres) VALUES (?, ?, ?, ?, ?, ?)",

(steam\_id, game\_title, base\_price, current\_price, release\_date, genres))

else:

*# game already exists, update the record*

print(f"Updating game {game\_title} with steam\_id {steam\_id}")

c.execute("UPDATE rogueapp\_game SET game\_title=?, base\_price=?, current\_price=?, release\_date=?, genres=? WHERE steam\_id=?",

(game\_title, base\_price, current\_price, release\_date, genres, steam\_id))