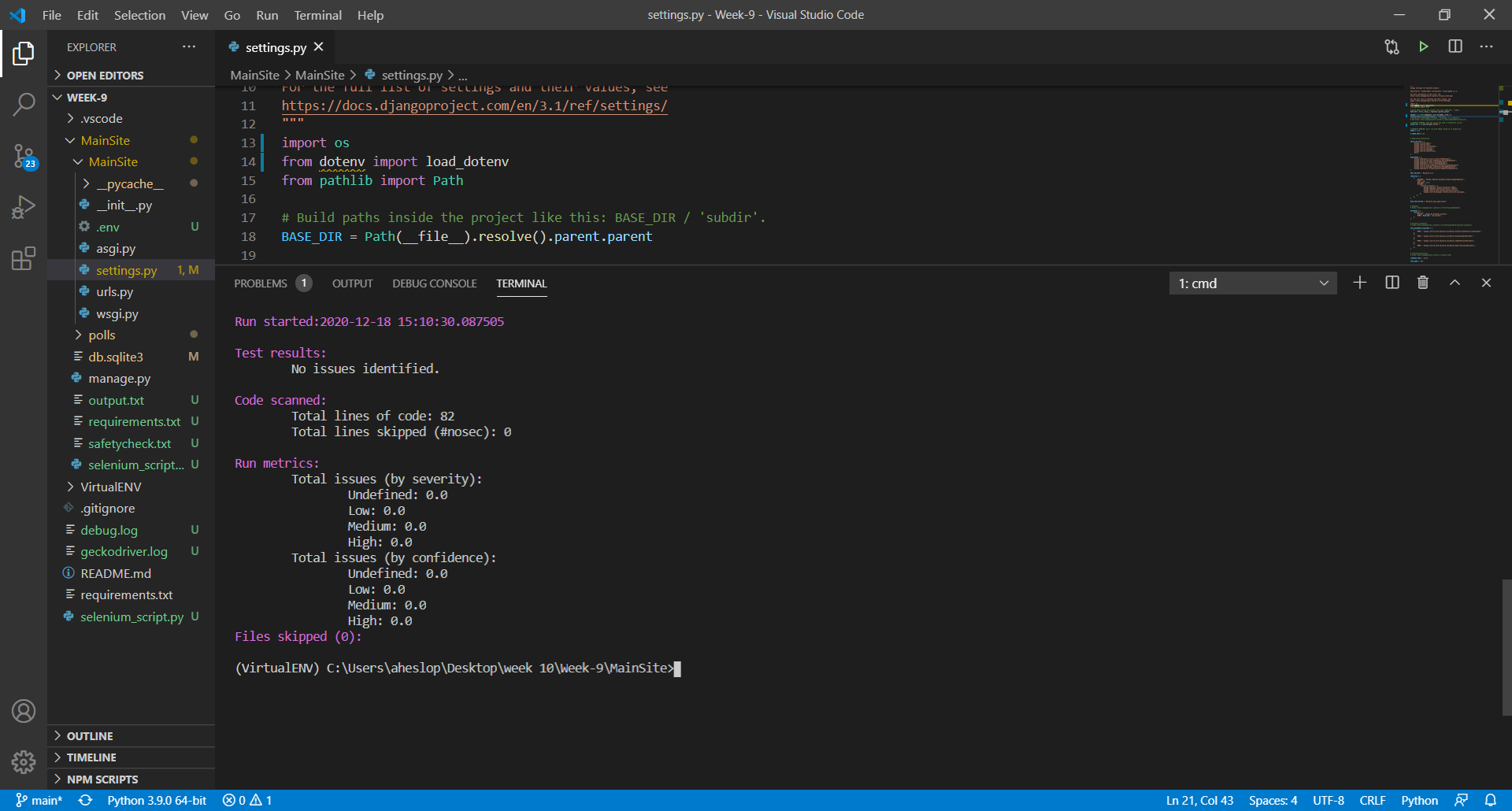
Secure Coding Evidence

* Bandit
* Safety
* Secret key/env variables

# Bandit

When testing the security of an application using Python we can use many tools to help us. Bandit is a tool which processes each file, and builds an AST, and uses plugins with the AST nodes.

Bandit produces an easy-to-read out onscreen for the user to review and make changes. Bandit can be ran using the command “bandit -r <foldername>”. Result below:



Bandit is a tool designed to find common security issues in Python code. To do this Bandit processes each file, builds an AST from it, and runs appropriate plugins against the AST nodes. Once Bandit has finished scanning all the files it generates a report.

# Safety

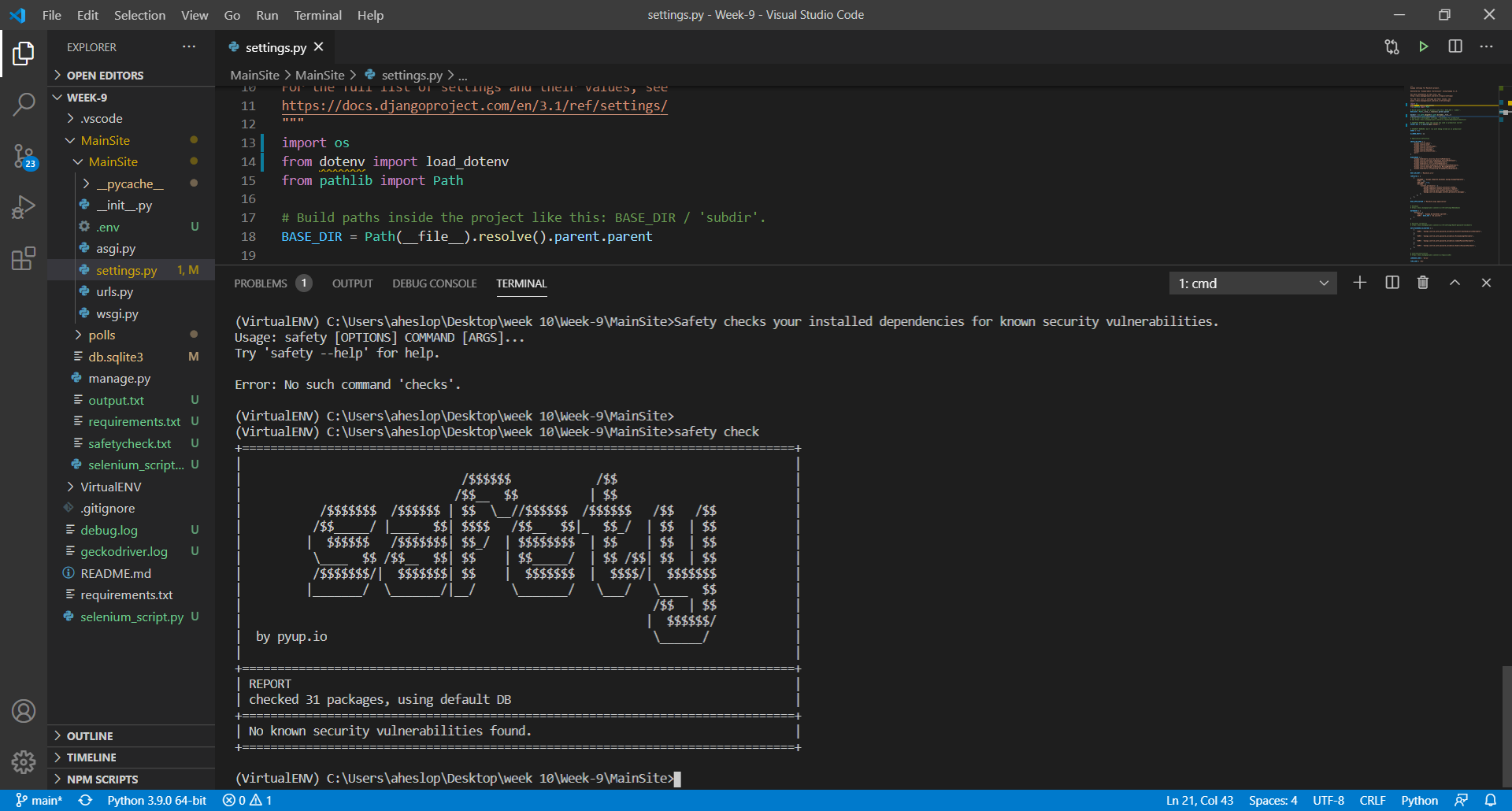
Safety checks your installed dependencies for known security vulnerabilities.

In order to install Safety please use:

pip install safety

In order to run safety please use:

safety check



As we can see from this print screen, we currently have ‘no known security vulnerabilities found’ in the 31 packages installed.

# Secret Key

In order for the application to run and be secure a Secret Key is in place. Secret Keys are used for authentication (username/password) and anything secret that you would prefer your visitors to not see. Secret Keys can be stored on the local computer for development (Environment Variables) and can be called using the os.environ.get command. Alternatively, Secret Keys can also be set in a separate file in which you can hide in the .gitignore file but is less secure.

The example below is using the Secret Key in a separate file, and is correctly assigned the path: