SUGGESTION

Considering the project submission and learning process, you don't actually need to implement the whole dynamic variable strategy, but at least enable the hooks to do so while setting defaults. You would get the same result but in a scalable manner.

You can achieve that by leveraging [Python's native os package](https://docs.python.org/3/library/os.html) as demonstrated below:

DB\_HOST = os.getenv('DB\_HOST', '127.0.0.1:5432')

DB\_USER = os.getenv('DB\_USER', 'postgres')

DB\_PASSWORD = os.getenv('DB\_PASSWORD', 'postgres')

DB\_NAME = os.getenv('DB\_NAME', 'trivia')

DB\_PATH = 'postgresql+psycopg2://{}:{}@{}/{}'.format(DB\_USER, DB\_PASSWORD, DB\_HOST, DB\_NAME)

This strategy allows your code to be compliant with two of the twelve essential rules provided by the [twelve-factor app strategy](https://12factor.net/), as explained below:

1. [Codebase control](https://12factor.net/codebase) - what controls the target connection is the environment variables so that the same code can handle different environments
2. [Dev/prod parity](https://12factor.net/dev-prod-parity) - again, by leveraging environment variables to control your connection, the same code can connect on different targets accordingly.