Creating a Context Manager class

Rather than rewrite Python’s open method here, we’ll create a context manager that can create a SQLite database connection and close it when it’s done. Here’s a simple example:

```
import sqlite3


class DataConn:
    """"""

    def __init__(self, db_name):
        """Constructor"""
        self.db_name = db_name

    def __enter__(self):
        """
        Open the database connection
        """
        self.conn = sqlite3.connect(self.db_name)
        return self.conn

    def __exit__(self, exc_type, exc_val, exc_tb):
        """
        Close the connection
        """
        self.conn.close()
        if exc_val:
            raise

if __name__ == '__main__':
    db = 'test.db'
    with DataConn(db) as conn:
        cursor = conn.cursor()
```

In the code above, we created a class that takes a path to a SQLite database file. The **__enter__** method executes automatically where it creates and returns the database connection object. Now that we have that, we can create a cursor and write to the database or query it. When we exit the with statement, it causes the **__exit__** method to execute and that closes the connection.

Let’s try creating a context manager using another method.