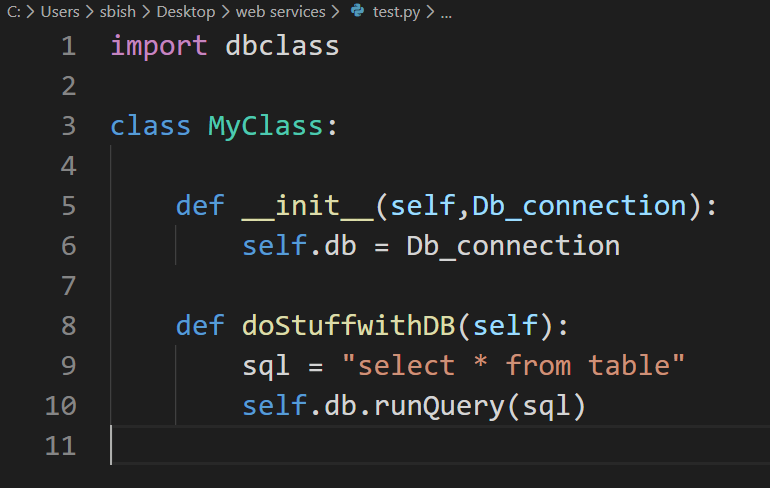
Dependency injection design pattern.

Dependency injection involves passing an object to a class that depends on an external object to function rather than instantiating this external object within the class. This frees a class from having to instantiate its own dependencies. The benefit of this approach supports the modularity of the classes and maintains a loosely coupled relationship between them. A great example of this approach can be implemented when a database connection is required by a given class. Usually a database connection requires a username, password, database name, and a host name (Thinking of PHP/MySQL here). Should any of these values change, each class that uses the database connection would have to be updated to keep the connection from breaking. Dependency injection can solve this issue by having the database connection instantiated perhaps in an initialization file and then the connection object can be passed to the class that needs it via its constructor.

Example without DI



Example with DI



The second example shows that the implantation of the database connection object can be passed the example class rather than having to have it instantiated inside the example class. The connection can be defined once in another class and have its object then passed to where it is needed. The example class should not have to worry about the details of usernames or passwords, hosts etc. The example class just needs to know that there is a connection to a database that is usable. Dependency Injection is one way to accomplish this.