Proxy: An interface for accessing a particular resource.

1. You are calling foo.var()
2. This assumes that foo is in the same process as bar()
3. What if later on you want to put all Foo-related operations into a separate process
   1. Can you avoid changing your code?
4. Proxy to the rescue
   1. Same interface, entirely different behavior
5. This is called a communication proxy
   1. Other types: logging, virtual, guarding
6. Proxy: A class that functions as an interface to a particular resource. That resource may be remote, expensive to construct, or may require logging or some other added functionality.

Protection Proxy:

1. In the example of car and driver I create a class for car called CarProxy that extends Car.
2. All the checks will be done in the CarProxy
3. This way you can change all the usage of Car to CarProxy and nothing will change in the code.
4. This way same API, different behaviour.
5. In dependency injection you can arrange the thing in a way that will give CarProxy when Car is created.

Property proxy

1. Doesn't really fall into proxy category
2. Replace fields of a class with PropertyProxy so you can have more control over them.
3. Direct initialization is not possible in this way like x = 5;

Dynamic Proxy for logging

1. With providing a LoggingClass that handles all method calls on the specified target, we can manage all function calls on a specified interface şn one place.
2. Example code important.

Proxy vs. Decorator

1. Proxy provide an identical interface; decorator provides an enhanced interface
2. Decorator typically aggregates (or has reference to) what it is decorating; proxy does not have to
3. Proxy might not even be working with a materialized object

Summary

1. A proxy has the same interface as the underlying object
2. To create a proxy, simply replicate the existing interface of an object
3. Add relevant functionality to the redefined member functions
4. Different proxies (communication, logging, caching etc.) have completely different behaviors.