# Abstraction | Coursera



## Abstraction

## Abstraction

One of the key techniques for designing any large system is *abstraction*. As we previously discussed, abstraction is the separation of interface from implementation. The *interface* of something is *what* it does, while the *implementation* is *how* it does it.

Consider this example of abstraction: driving a car versus knowing how it works under the hood. The car provides a simple interface (turning the steering wheel turns the car, pushing the accelerator makes the car go faster, pushing the brake slows it down, ...). However, the implementation (how everything works under the hood) is quite complex. Of course, you do not actually have to know how the car works to drive it—the implementation is hidden from you. You only need to know the interface.

A similar principle applies in designing programs—you should be able to use a function if you know *what* it does, without having to know *how* it does it. In fact, in your programming so far, you have routinely used a wide variety of C library functions without knowing their implementation details (even ones that you may have written an equivalent for as a practice exercise may be implemented in a much more optimized manner in the real C library).

### Completed