Mocking Strategies

Why Use Mocks?

Mocks improve *isolation*, by letting you by pass component dependencies. What that gives you:

- Run code that depends on other, complex objects, without actually setting up those objects
- Simulate error conditions (without jumping through hoops)
- Simplify the testing environment, and reduce setup work. Sometimes by a lot
- Better control starting state
- Avoid undesired side effects (e.g. network API calls)
- Increase speed (by avoiding running expensive code)

And some downsides too.

Downsides of Mocks

- Can create hard-to-change and hard-to-maintain tests
- Can create fragile tests, tied to details of current implementation
- Can make tests harder to understand
- Can dig yourself into a hole where your arch is over-dependent on mocks for unit tests
- Mocking tends to work by leveraging more "magic" features. So they sometimes conflict with your own advanced code
- Can get out of sync with the object it's mocking, if you don't use autospec

Given this, avoid mocks when they are not needed. **But don't hesitate to use them when they are.**