How it Works

The doctest module works by examining the docstrings in the module, from the module level to the function, class and method levels. It will not look at imported modules though. For the most part, you can copy-and-paste an interactive console session in and doctest will work fine with it.

doctest will look for output following the final >>> or … line that contains code. Said output will only extend to the next >>> or all-whitespace line.

Here are a few gotcha's to watch out for.

> - If you continue a line using a backslash, then you will want to use a raw docstring (i.e. r""“This is raw”"") to preserve the backslash. Otherwise it will be considered a part of the string and will probably raise an error.
> - Tracebacks usually contain exact file path and line numbers which change when you're developing and will the paths will certainly change across machines. So this will cause issues too. Fortunately, doctest is pretty flexible with tracebacks. You only need the Traceback line and the actual Exception line to make the test pass. Just change the example from before to this:

```
def add(a, b):
    """
    Return the addition of the arguments: a + b

    add(1, 2)
    #3
    add(-1, 10)
    #9
    add('a', 'b')
    #'ab'
    add(1, '2')
    #Traceback (most recent call last):
    #TypeError: unsupported operand type(s) for +: 'int' and 'str'
    """
    return a + b
```

> - As you know dictionaries don’t have a specific ordering, so if you try to print one in your doctest, it will probably fail. Instead, if you must use a dictionary, you should do a comparison of dictionaries as that should return a bool and be much more reliable.
> - The same issue arises when you print an **id** or an instance of something, like a class. Since these are dynamic, they won’t work well with doctest.