[Index | Exercise 3.5 | Exercise 3.7]

Exercise 3.6

Objectives:

- Learn how to customize the behavior of objects by redefining special methods.
- Change the way that user-defined objects get printed
- Make objects comparable
- Create a context manager

Files Created: None

Files Modified: `stock.py`

(a) Better output for representing objects

All Python objects have two string representations. The first representation is created by string conversion via `str()` (which is called by `print`). The string representation is usually a nicely formatted version of the object meant for humans. The second representation is a code representation of the object created by `repr()` (or simply by viewing a value in the interactive shell). The code representation typically shows you the code that you have to type to get the object. Here is an example that illustrates using dates:

```
>>> from datetime import date
>>> d = date(2008, 7, 5)
>>> print(d)              # uses str()
2008-07-05
>>> d    # uses repr()
datetime.date(2008, 7, 5)
>>>
```

There are several techniques for obtaining the `repr()` string in output:

```
>>> print('The date is', repr(d))
The date is datetime.date(2008, 7, 5)
>>> print(f'The date is {d!r}')
The date is datetime.date(2008, 7, 5)
>>> print('The date is %r' % d)
The date is datetime.date(2008, 7, 5)
>>>
```

Modify the `Stock` object that you've created so that the `__repr__()` method produces more useful output. For example:

```
>>> goog = Stock('GOOG', 100, 490.10)
>>> goog
Stock('GOOG', 100, 490.1)
>>>
```

See what happens when you read a portfolio of stocks and view the resulting list after you have made these changes. For example:

```
>>> import stock, reader
>>> portfolio = reader.read_csv_as_instances('Data/portfolio.csv',
stock.Stock)
>>> portfolio
[Stock('AA', 100, 32.2), Stock('IBM', 50, 91.1), Stock('CAT', 150, 83.44),
Stock('MSFT', 200, 51.23),
 Stock('GE', 95, 40.37), Stock('MSFT', 50, 65.1), Stock('IBM', 100, 70.44)]
>>>
```

(b) Making objects comparable

What happens if you create two identical `Stock` objects and try to compare them? Find out:

```
>>> a = Stock('GOOG', 100, 490.1)
>>> b = Stock('GOOG', 100, 490.1)
>>> a == b
False
>>>
```

You can fix this by giving the `Stock` class an `__eq__()` method. For example:

```
class Stock:
    ...
    def __eq__(self, other):
        return isinstance(other, Stock) and ((self.name, self.shares,
self.price) ==
                                             (other.name, other.shares,
other.price))
    ...
```

Make this change and try comparing two objects again.

(c) A Context Manager

In Exercise 3.5, you made it possible for users to make nicely formatted tables. For example:

```
>>> from tableformat import create_formatter
>>> formatter = create_formatter('text')
>>> print_table(portfolio, ['name','shares','price'], formatter)
      name     shares      price
---------- ---------- ----------
        AA        100       32.2
       IBM         50       91.1
       CAT        150      83.44
      MSFT        200      51.23
        GE         95      40.37
      MSFT         50       65.1
       IBM        100      70.44
>>>
```

One issue with the code is that all tables are printed to standard out (`sys.stdout`). Suppose you wanted to redirect the output to a file or some other location. In the big picture, you might modify all of the table formatting code to allow a different output file. However, in a pinch, you could also solve this with a context manager.

Define the following context manager:

```
>>> import sys
>>> class redirect_stdout:
        def __init__(self, out_file):
            self.out_file = out_file
        def __enter__(self):
            self.stdout = sys.stdout
            sys.stdout = self.out_file
            return self.out_file
        def __exit__(self, ty, val, tb):
            sys.stdout = self.stdout
```

This context manager works by making a temporary patch to `sys.stdout` to cause all output to redirect to a different file. On exit, the patch is reverted. Try it out:

```
>>> from tableformat import create_formatter
>>> formatter = create_formatter('text')
>>> with redirect_stdout(open('out.txt', 'w')) as file:
        tableformat.print_table(portfolio, ['name','shares','price'],
formatter)
        file.close()

>>> # Inspect the file
>>> print(open('out.txt').read())
      name     shares      price
---------- ---------- ----------
        AA        100       32.2
       IBM         50       91.1
```

```
        CAT        150      83.44
       MSFT        200      51.23
         GE         95      40.37
       MSFT         50       65.1
        IBM        100      70.44
>>>
```

[Solution | Index | Exercise 3.5 | Exercise 3.7]

>>> Advanced Python Mastery
. . . A course by dabeaz