10.6. Using `with` for Files

Note

This section is a bit of an advanced topic and can be easily skipped. But with statements are becoming very common and it doesn't hurt to know about them in case you run into one in the wild.

Now that you have seen and practiced a bit with opening and closing files, there is another mechanism that Python provides for us that cleans up the often forgotten close. Forgetting to close a file does not necessarily cause a runtime error in the kinds of programs you typically write in an introductory programing course. However if you are writing a program that may run for days or weeks at a time that does a lot of file reading and writing you may run into trouble.

Python has the notion of a context manager that automates the process of doing common operations at the start of some task, as well as automating certain operations at the end of some task. For reading and writing a file, the normal operation is to open the file and assign it to a variable. At the end of working with a file the common operation is to make sure that file is closed.

The Python with statement makes using context managers easy. The general form of a with statement is:

```
with <create some object that understands context> as <some name>:
    do some stuff with the object
    ...
```

When the program exits the with block, the context manager handles the common stuff that normally happens at the end, in our case closing a file. A simple example will clear up all of this abstract discussion of contexts. Here are the contents of a file called "mydata.txt".

Data file: `mydata.txt`

```
1 2 3
4 5 6
```

Save & Run Original - 1 of 1

```
with open('mydata.txt', 'r') as md:
    for line in md:
        print(line)
# continue on with other code

```

```
1 2 3

4 5 6

```

Activity: 1 -- ActiveCode (ac9_12_1)

The first line of the with statement opens the file and assigns it to the variable `md`. Then we can iterate over the file in any of the usual ways. When we are done we simply stop indenting and let Python take care of closing the file and cleaning up. The final line `print(md)`

This is equivalent to code that specifically closes the file at the end, but neatly marks the set of code that can make use of the open file as an indented block, and ensures that the programmer won't forget to include the .close() invocation.

Save & Run Original - 1 of 1

```
md = open('mydata.txt', 'r')
for line in md:
    print(line)
md.close()
# continue with other code

```

```
1 2 3

4 5 6
```

Activity: 2 -- ActiveCode (ac9_12_2)

10.5. Finding a File in your Filesystem">

nding a File in your Filesystem">

10.7. Recipe for Reading and Processing a File">

You have attempted 3 of 2 activities on this page

| Back to top