7.2 Writing Files

To open a file for writing you must use either

```
F = open(<file name>, "w")
```

or

```
F = open(<file name>, "a")
```

The value returned by the **open( )** statement is a *file object*; we need this object in order to manipulate the file. The "w" mode causes a new file to be created. If a file with this name already exists in the current folder, it will be destroyed and replaced by the new file. The "a" mode will append onto the end of an existing file, or create a new one if no file with the given name already exists. In either case each write statement writes onto the end of the file.

You can write a string to file object F with

```
F.write( <string> )
```

Note that this does not pad the string with any spaces, and it does not terminate any line in the file. The statements

```
F.write("26")
F.write("35")
```

will put "2635" into the file. You need to add any separators that you want as you write the data. If you wish to have the file organized into lines of text, as most text files are, you need to directly write the line breaks: "\n" is the symbol we use for this. So to write the string "This is a line." into file object F we would say

```
F.write("This is a line.\n")
```

Finally, note that you can only write a string into a text file. If you want to output numbers, you can write them with formatted strings, as in

```
F.write("%d\n"%value)
```

or you can use **str()**, as in

```
F.write( str(value)+"\n" )
```

Here is a simple program that reads a list of names from the user and writes these names into a file called "names.txt".

```
def main():
    F = open( "names.txt", "w" )
    done = False
    while not done:
        name=input("Enter a name, or a blank to exit: ")
        if name == "":
            done = True
        else:
            F.write(name+"\n")
main()
```

Program 7.2.1: Saving strings to a file

The next program builds a file that contains a table of prime numbers. This time we build up the data in a list, and print the list to a file at the end. Rather than trying to divide all numbers into a given candidate, we will use the fact that all smaller primes have already been entered into our list; these are the only potential divisors that we need to check. The resulting program finds all prime numbers up to one million within a few seconds.

The important function here is WriteToFile(L), which writes list L to the file "primes.txt". This starts by opening the file, then writing the entries one at a time into the file. We use a formatted string:

```
    F.write( "%10d" % entry )
```

to guarantee that there will be white space around each number: this uses 10 spaces per entry, and the entries have no more than 6 digits. Since we use multiple write statements per line, the formatted print statement also guarantees that our numbers come out in neat columns.

Unless we want thousands of numbers to be printed on the same line, at some point we need to give line breaks. Eight entries per line makes for a nice table. We have a variable lineCount that keeps track of how many entries are on the current line. When this gets up to 8 we terminate the line with

```
    F.write( "\n" )
```

and reset lineCount to 0.

```
# This builds a table of primes numbers.

def isPrime(x, L):
    # This returns True is x is prime,
    # where L is the list of all primes
    # less than x.
    for d in L:
        if x%d == 0:
            return False
        elif d*d > x:
            return True
    return True

def FindPrimes( L ):
    # This puts all primes less than 1000000 into L
    for x in range(2, 1000000):
        if isPrime(x, L):
            L.append(x)

def WriteToFile( L ):
    # This writes the entries of L to a file "primes.txt",
    # printing 8 entries per line
    F = open( "primes.txt", "w")
    lineCount = 0
    for entry in L:
        F.write( "%10d" % entry )
        lineCount = lineCount + 1
        if lineCount == 8:
            F.write( "\n" )
            lineCount = 0

def main():
    Primes = []
    FindPrimes( Primes )
    WriteToFile( Primes )

main()
```

Program 7.2.2: Saving prime numbers to a file