# Reading and Writing Files

1.

```
filename = input('Which file would you like to back-up? ')
new_filename = filename + '.bak'
backup = open(new_filename, 'w')

for line in open(filename):
    backup.write(line)

backup.close()
```

2.

```
alkaline_metals = []
for line in open('alkaline_metals.txt'):
    alkaline_metals.append(line.strip().split(' '))
```

3.

We could read the file contents into a data structure, such as a `list`, and then iterate over the `list` from end (last line) to beginning (first line).

4.

```
def process_file(reader):
    """ (file open for reading) -> NoneType

    Read and print the data from reader, which must start with a single
    description line, then a sequence of lines beginning with '#', then a
    sequence of data.
    """

    # Find and print the first piece of data.
    line = skip_header(reader).strip()
    print(line)

    # Read the rest of the data.
    print(reader.read())
```

5.

```
import time_series

def smallest_value_skip(reader):
    """ (file open for reading) -> number or NoneType

    Read and process reader, which must start with a time_series header.
    Return the smallest value after the header.  Skip missing values, which
    are indicated with a hyphen.
    """

    line = time_series.skip_header(reader).strip()

    # Only execute this code, if there is data following the header.
    if line != '':
        smallest = int(line)

        for line in reader:
            line = line.strip()
            if line != '-':
                value = int(line)
                smallest = min(smallest, value)

        return smallest

if __name__ == '__main__':
    with open('hebron.txt', 'r') as input_file:
        print(smallest_value_skip(input_file))
```

6.

```
import time_series

def smallest_value_skip(reader):
    """ (file open for reading) -> NoneType

    Read and process reader, which must start with a time_series header.
    Return the smallest value after the header.  Skip missing values, which
    are indicated with a hyphen.
    """

    line = time_series.skip_header(reader).strip()

    # Now line contains the first data value; this is also the smallest value
    # found so far, because it is the only one we have seen.
    smallest = int(line)

    for line in reader:
        line = line.strip()
        if line == '-':
            continue

        value = int(line)
        smallest = min(smallest, value)

    return smallest

if __name__ == '__main__':
    with open('hebron.txt', 'r') as input_file:
        print(smallest_value_skip(input_file))
```

7.

```
def read_molecule(reader):
    """ (file open for reading) -> list or NoneType
    Read a single molecule from reader and return it, or return None to
signal
    end of file.  The first item in the result is the name of the compound;
    each list contains an atom type and the X, Y, and Z coordinates of that
    atom.
    """

    # If there isn't another line, we're at the end of the file.
    line = reader.readline()
    if not line:
        return None

    if not (line.startswith('CMNT') or line.isspace()):
        # Name of the molecule: "COMPND name"
        key, name = line.split()

        # Other lines are either "END" or "ATOM num atom_type x y z"
        molecule = [name]
    else:
        molecule = None

    reading = True
    while reading:
        line = reader.readline()
        if line.startswith('END'):
            reading = False
        elif not (line.startswith('CMNT') or line.isspace()):
            key, num, atom_type, x, y, z = line.split()
            if molecule == None:
                molecule = []
            molecule.append([atom_type, x, y, z])

    return molecule
```

8.

```
def read_molecule(reader):
    """ (file open for reading) -> list or NoneType

    Read a single molecule from reader and return it, or return None to
signal
    end of file.  The first item in the result is the name of the compound;
    each list contains an atom type and the X, Y, and Z coordinates of that
    atom.
    """

    # If there isn't another line, we're at the end of the file.
    line = reader.readline()
    if not line:
        return None

    # Name of the molecule: "COMPND   name"
    key, name = line.split()

    # Other lines are either "END" or "ATOM num atom_type x y z"
```

```
molecule = [name]
reading = True

serial_number = 1
while reading:
    line = reader.readline()
    if line.startswith('END'):
        reading = False
    else:
        key, num, atom_type, x, y, z = line.split()
        if int(num) != serial_number:
            print('Expected serial number {0}, but got {1}'.format(
                serial_number, num))
        molecule.append([atom_type, x, y, z])
        serial_number += 1

return molecule
```