Solution to Homework 2
Spring 2015

1 Echo

Here's my code for `echo.py`. It's mostly comment, since I included my "diary" there.

```
# python program to read a file and echo its contents
#
# Ethan Bolker
# February 9, 2015
# Math 480 hw2
#
# The first time I tried this program I got an extra blank line
# between each line of the input file. That's because when python
# reads in a line from a file it includes the newline character
# at the end of the line. To get just what I want to print I need to
# delete that last character, since the print() function will supply the
# newline on output. I do that by slicing the line.
#
# Google found these answers for me:
# http://stackoverflow.com/questions/8009882/how-to-read-large-file-line-by-line-in-python
# http://stackoverflow.com/questions/15478127/remove-final-character-from-string-python

import sys

filename = sys.argv[1]

for line in open(filename):
    print( line[:-1] )
```

2 Average

The **open()** function opens a file and returns an object that you can loop over with **for**. Before I tried this I wondered whether I would have to strip the newline from the end of each line. I was lucky: the **float()** function strips white space (blanks and tabs and newlines) at the beginning and the end of its argument.

```
# python program to read a file of floats and calculate their average
#
# Ethan Bolker
# February 9, 2015
# Math 480 hw2

import sys
filename = sys.argv[1]

count = 0
total = 0
for line in open(filename):
    count += 1
    total += float(line)
print("average: " + str(total/count))
```

Here's a test, pasted from Windows `powershell`. The leading blanks on the second line don't matter. The fourth line shows that the **float()** function can parse scientific notation.

```
PS C:\eb\python\hw2> type floats.txt
100
```

```
    200
300
4e2
PS C:\eb\python\hw2> python average.py floats.txt
average: 250.0
```

3 Robust Average

Our main focus in this course is quick-and-dirty software to answer mathematical questions we're curious about. If our programs break from time to time, so what? But applications intended for general use must not crash. To keep that from happening you have to anticipate all the things that might go wrong, and guard against them. When something does go wrong you have to warn the user and continue on, or exit the program.

You can detect problems either by testing in advance with an **if:** statement, or you can just **try:** something and deal with any problem in the **except:** clause.

Here are the tests for `robust_average.py`.

First look for errors telling the program what file to use – there might be no files named, or more than one, or a file that isn't there. (The file might be there but not be readable. I haven't tested for that, but could.)

```
PS C:\eb\python\hw2> python robust_average.py
usage: robust_average.py <filename>
PS C:\eb\python\hw2> python robust_average.py nosuchfile
File nosuchfile not found
PS C:\eb\python\hw2> python robust_average.py floats.txt floats.txt
usage: robust_average.py <filename>
```

If the file is really there, test for bad file contents, and warn the user when a line can't be parsed by **float()**.

```
PS C:\eb\python\hw2> type empty.txt
PS C:\eb\python\hw2> python robust_average.py empty.txt
Warning: no floats to average
PS C:\eb\python\hw2> type .\floaterrors.txt
100
next line is blank

200
2..0
-100
2e2
1f2
average should be 100
PS C:\eb\python\hw2> python robust_average.py  .\floaterrors.txt
Warning: float error on line next line is blank
Warning: float error on line
Warning: float error on line 2..0
Warning: float error on line 1f2
Warning: float error on line average should be 100
average: 100.0
```

```
# python program to read a file of floats and calculate their average
#
# This version has lots of error checking.
# Error messages are sent to stderr, not stdout
# (that's where ordinary print() statements go).
#
# Note how error checking accounts for a very large fraction
# of the code!
#
```

```
# http://stackoverflow.com/questions/5574702/how-to-print-to-stderr-in-python
#
# Ethan Bolker
# February 10, 2015
# Math 480 hw2

import sys

# see if there's exactly one filename
if (len(sys.argv) != 2):
    print("usage: " + sys.argv[0] + " <filename>", file=sys.stderr)
    exit(1) # leave program with a return value different from 0

filename = sys.argv[1]

# make sure the file is there and can be opened
try:
    filecontents = open(filename)
except FileNotFoundError:
    print("File " + sys.argv[1] + " not found", file=sys.stderr)
    exit(1)

count = 0
total = 0

for line in open(filename):
    # perhaps the line doesn't contain a float
    try:
        total += float(line)
    except ValueError:
        print("Warning: float error on line " + line[:-1])
        continue # back to top of for loop
    count += 1

if count == 0:
    print("Warning: no floats to average")
else:
    print("average: " + str(total/count))
```

Here is the LaTeX source for this document. You can cut it from the pdf and use it to start your answers. I used the `\jobname` *macro for the source file name, so you can call your file by any name you like.*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\documentclass[10pt]{article}
\usepackage[textheight=10in]{geometry}

\usepackage{verbatim}
\usepackage{amsmath}
\usepackage{amsfonts} % to get \mathbb letters

\usepackage[utf8]{inputenc}
\DeclareFixedFont{\ttb}{T1}{txtt}{bx}{n}{9} % for bold
\DeclareFixedFont{\ttm}{T1}{txtt}{m}{n}{9}  % for normal
% Defining colors
\usepackage{color}
\definecolor{deepblue}{rgb}{0,0,0.5}
\definecolor{deepred}{rgb}{0.6,0,0}
\definecolor{deepgreen}{rgb}{0,0.5,0}

\usepackage{listings}

%Python style from
%http://tex.stackexchange.com/questions/199375/problem-with-listings-package-for-python-syntax-colo
\newcommand\pythonstyle{\lstset{
  language=Python,
  backgroundcolor=\color{white}, %%%%%%%
  basicstyle=\ttm,
  otherkeywords={self},
  keywordstyle=\ttb\color{deepblue},
  emph={MyClass,__init__},
  emphstyle=\ttb\color{deepred},
  stringstyle=\color{deepgreen},
  commentstyle=\color{red},  %%%%%%%%
  frame=tb,
  showstringspaces=false,
  numbers=left,numberstyle=\tiny,numbersep =5pt
}}
\usepackage{hyperref}

\begin{document}

\pythonstyle{}

\begin{center}
\Large{
Solution to Homework 2 \\
Spring 2015
}
\end{center}

\section{Echo}
Here's my code for \lstinline!echo.py!. It's mostly comment, since I
included my ``diary'' there.

\lstinputlisting{echo.py}
\section{Average}

The \lstinline!open()! function opens a file and returns an object that
```

```
you can loop over with \lstinline!for!. Before I tried this I wondered
whether I would have to strip the newline from the end of each line. I
was lucky: the \lstinline!float()! function strips white
space (blanks and tabs and newlines) at the beginning and the end of
its argument.

\lstinputlisting{average.py}

Here's a test, pasted from Windows \verb!powershell!. The leading
blanks on the second line don't matter. The fourth line shows that the
\lstinline!float()! function can parse scientific notation.

\begin{verbatim}
PS C:\eb\python\hw2> type floats.txt
100
    200
300
4e2
PS C:\eb\python\hw2> python average.py floats.txt
average: 250.0
\end{verbatim}

\section{Robust Average}

Our main focus in this course is quick-and-dirty software to answer
mathematical questions we're curious about. If our programs break from
time to time, so what? But applications intended for general use must
not crash. To keep that from happening you have to anticipate all the
things that might go wrong, and guard against them. When something
does go wrong you have to warn the user and continue on, or exit the
program.

You can detect problems either by testing in advance with an
\lstinline!if:! statement, or you can
just \lstinline!try:! something and deal with any problem in the
\lstinline!except:! clause.

Here are the tests for \lstinline!robust_average.py!.

First look for errors telling the program what file to use -- there
might be no files named, or more than one, or a file that isn't there.
(The file might be there but not be readable. I haven't tested for
that, but could.)

\begin{verbatim}
PS C:\eb\python\hw2> python robust_average.py
usage: robust_average.py <filename>
PS C:\eb\python\hw2> python robust_average.py nosuchfile
File nosuchfile not found
PS C:\eb\python\hw2> python robust_average.py floats.txt floats.txt
usage: robust_average.py <filename>
\end{verbatim}

If the file is really there, test for bad file contents, and warn the
user when a line can't be parsed by \lstinline!float()!.

\begin{verbatim}
PS C:\eb\python\hw2> type empty.txt
PS C:\eb\python\hw2> python robust_average.py empty.txt
Warning: no floats to average
```

```
PS C:\eb\python\hw2> type .\floaterrors.txt
100
next line is blank

200
2..0
-100
2e2
1f2
average should be 100
PS C:\eb\python\hw2> python robust_average.py  .\floaterrors.txt
Warning: float error on line next line is blank
Warning: float error on line
Warning: float error on line 2..0
Warning: float error on line 1f2
Warning: float error on line average should be 100
average: 100.0
\end{verbatim}

\lstinputlisting{robust_average.py}

\newpage
\emph{
Here is the \LaTeX{} source for this document. You can cut it from the
pdf and use it to start your answers. I used the} \verb!\jobname!
\emph{macro for the source file name, so you can call your file by any
  name you like.}
\verbatiminput{\jobname}

\end{document}
```