```
#!/usr/bin/python
"""
FileStats.py
P. TenHoopen, 11/21/08

This program reads the statistics of files from a specified directory and subdirectories.
It displays the number of files, types and number of files, average size, smallest size, largest size, and the
name of the largest file.
"""

# Import Modules
import os
import pygtk
pygtk.require("2.0")
import gtk
import gtk.glade

# Global Variables
extensions = {}         # dictionary to hold extensions and counts
largestFileSz = 0       # size of largest file
largestFile = ""        # name of largest file
numFiles = 0            # number of files found
sumFileSizes = 0.0      # total size of files used to calc average file size
userDir = ""            # path entered by user

# Classes

class SetupGTKInterface:
    """This provides the GTK interface."""

    def __init__(self):

        # set the Glade file
        self.gladefile = "FileStats3.glade"
        self.wTree = gtk.glade.XML(self.gladefile)

        # create events dictionay and connect it
        eventDict = {
            "on_btnOK_clicked" : self.btnOK_clicked,
            "on_btnCancel_clicked" : self.btnCancel_clicked,
            "on_FileStatsWindow_destroy" : gtk.main_quit,
            "on_mnuQuit_activate" : self.mnuQuit_clicked,
            "on_mnuAbout_activate" : self.mnuAbout_clicked }
        self.wTree.signal_autoconnect(eventDict)

        # set up the results window - import the widget
        self.textviewResults = self.wTree.get_widget("textviewResults")
        # set the text buffer
        self.textbufferResults = self.textviewResults.get_buffer()

        # import the text entry widget
        self.entryPath = self.wTree.get_widget("entryPath")

    def btnOK_clicked(self, widget):
        """
        This function is called when the user clicks the OK button.
        It gets the path the user entered in the Path box and calls the analyze function.
        """

        # use global variable defined above
        global userDir

        # get the text from the Path box
        userDir = self.entryPath.get_text()
        analyze()

    def btnCancel_clicked(self, widget):
        """
        This function is called when the user clicks the Cancel button.
        It exits the program.
        """

        gtk.main_quit()

    def mnuQuit_clicked(self, widget):
        """
        This function is called when the user selects Quit from the File menu.
```

```
        It exits the program.
        """

        gtk.main_quit()

    def mnuAbout_clicked(self, widget):
        """
        This function is called when the user selects About from the Help menu.
        It shows the about dialog box.
        """

        # create and show an about dialog box
        aboutDialog = AboutWindow()

class AboutWindow:
    """This class shows an About dialog box."""

    def __init__(self):

        # set the Glade file
        self.gladefile = "FileStatsAbout.glade"
        self.wTree = gtk.glade.XML(self.gladefile)

        # import the about dialog widget
        aboutDlg = self.wTree.get_widget("aboutdialogFS")

        # show the dialog box
        aboutDlg.run()

        # destroy it after it is closed
        aboutDlg.destroy()


# Functions

def recordExtension(thisExtension):
    """This function keeps track of the extensions that are found."""

    # add to dictionary if new, or update count if existing
    if thisExtension not in extensions:
        extensions[thisExtension] = 1
        #print "New file type '%s' was added" % thisExtension  # debug
    else:
        extensions[thisExtension] += 1
        #print "Count of %s is %s" % (thisExtension, extensions[thisExtension] )  # debug

    return 1

def collectFileInfo(fileName):
    """This function examines file statistics to keep track of largest file and total file sizes."""

    # use global variables defined above
    global largestFileSz
    global largestFile
    global sumFileSizes

    # collect file statistics
    fileStats = os.stat(fileName)
    # get file size
    size = fileStats.st_size
    # keep size of all files
    sumFileSizes += size
    # is this the largest file found so far?
    if size > largestFileSz:
        largestFileSz = size
        largestFile = fileName

    return 1

def displayStats():
    """This function displays the collected statistics in a textview widget on the window."""

    results = "There were %d files found.\n\n" % numFiles
    results += "Count \t File Ext\n"
    results += "------- \t --------\n"
    for extension in extensions:
        results += "%d \t\t %s\n" % (extensions[extension], extension)
```

```
    results += "\nLargest file:  %s at %d bytes.\n" % (largestFile, largestFileSz)
    results += "Total size of files is %d bytes.\n" % (sumFileSizes)
    results += "Average size of files is %.2f bytes." % (sumFileSizes / numFiles)

    # put the results in the GTK textview object (reference the window object - fswg)
    fswg.textbufferResults.set_text(results)

    return 1

def analyze():
    """This function is called when the user clicks the OK button.  It analyzes the files specified in the
topDir variable."""

    # variables
    topDir = "."   # directory to search, default to current directory

    # use global variables defined above
    global numFiles
    global userDir

    # get top directory to analyze from user
    if (len(userDir) > 0):  # weak error checking!
        topDir = userDir    # as long as the user enters something, use it; otherwise use "."

    # walk through the files
    walklist = os.walk(topDir)

    # os.walk returns {dirpath, dirnames, filenames}

    for root, dirs, files in walklist:
        # examine each file found
        for eachFile in files:
            numFiles += 1
            # look for a extension delimiter (.) in the file name
            if "." in eachFile:
                # get the file extension using a string slice
                # extract all text after the period
                extension = eachFile[eachFile.index(".")+1:]
            else:
                # no period delimiter so no extension can be determined
                extension = "none"

            # keep track of extensions that are found
            recordExtension(extension)

            # get file name and path
            fileName = os.path.join(root, eachFile)

            # examine files stats
            collectFileInfo(fileName)

    # display the results of the analysis
    displayStats()

    # reset the stats
    resetStats()

    return 1

def resetStats():
    """ This function reset the stats."""

    # use global variables defined above
    global userDir
    global extensions
    global largestFileSz
    global largestFile
    global numFiles
    global sumFileSizes

    userDir = ""
    extensions.clear()
    largestFileSz = 0
    largestFile = ""
    numFiles = 0
    sumFileSizes = 0.0
```

```
    return 1

# This initiates window setup and calls the 'main' GTK function when this script is executed.
if __name__ == '__main__':

    # fswg = file stats window gtk handle
    fswg = SetupGTKInterface()
    gtk.main()
```