Outline

Files

Writing to a file

Reading from a file

Common functions for accessing files

Using files

Files

- Every computer system uses files to store data
- This allows information to be saved from one computation to another
- Each operating system (eg Unix, Linux, Windows, MAC OS, Android, . . .) comes with its own file system
- A file system has operations for creating, accessing, reading from, writing to and deleting files
- Accessing a file is done by using a file handle

File handle

- Consider the following Python statement:

  `fileHandle = open('names.txt', 'w')`
- This invocation of the `open` function instructs the operating system to create a file with the name `names.txt` and returns a file handle for that file that is bound to the variable `fileHandle`
- The second argument to the `open` function, "`w`", indicates that the file is opened for writing
- Any previous contents of the file will be overwritten

  - take care not to destroy an existing file!

Common functions for accessing files (1)

- The following are some of the common functions for accessing files:
- `open(fn, ’w’) fn` is a string representing a file name. Creates a file for writing and returns a file handle
- `open(fn, ’r’) fn` is a string representing a file name. Opens an existing file for reading and returns a file handle
- `open(fn, ’a’) fn` is a string representing a file name. Opens an existing file for appending and returns a file handle
- `fh.close()` closes the file associated with the file handle `fh`

Writing to a file (1)

```
# Program to demonstrate the use of files
# Prompts the user for a given name and a family na


# Open the file for reading
fileHandle = open('names.txt', 'w')

# Prompt the user for a given name
givenname = input('Enter a given name: ')
fileHandle.write(givenname)

# Prompt the user for a family name
familyname = input('Enter a family name: ')
fileHandle.write(familyname)

fileHandle.close()
```

Writing to a file (2)

- Running this program with the following interaction:

  ```
  >>>
  Enter a given name: John
  Enter a family name: Dunnion
  >>>
  ```

  The contents of the file `names.txt` are as follows:

  ```
  JohnDunnion
  ```

- If we want different strings to appear on different lines in the file, we must include a newline character when writing each string to the file

Writing to a file (3)

```
# Program to demonstrate the use of files
# Prompts the user for a given name and a family na
# and writes them to a file with newlines

# Open the file for writing
fileHandle = open('names.txt', 'w')

# Prompt the user for a given name
givenname = input('Enter a given name: ')
fileHandle.write(givenname + '\n')

# Prompt the user for a family name
familyname = input('Enter a family name: ')
fileHandle.write(familyname + '\n')

fileHandle.close()
```

Reading from a file (1)

- To read from a file, we must call the `open` function with a second argument of “`r`”, indicating that the file is opened for reading

```
fh1 = open(’names.txt’, ’r’)
```

The function **`readline`**`()` reads a line from a file e.g.

```
line = fh1.readline()
```

`readline()` returns the empty string "" if the file is empty or when you have reached the **end of the file**

```
# Program to demonstrate the use of files
# Reads names from a file and prints them out

# Open the file for reading

fh1 = open('names.txt', 'r')

line = fh1.readline() # read 1st line from file

while line != "":      # "" means end of file reached
    print(line, end = "")
    line = fh1.readline()  # read next line

fh1.close()                 # close file
```

```
# Program to demonstrate the use of files
# Reads names from a file and prints them out
# Prompt the user for a file name

 filename = input('Enter a file name: ')

 # Open the file for reading
 fh = open(filename, 'r')

line = fh.readline() # read 1st line from file

while line != "":      # "" means end of file reached
    print(line, end = "")
    line = fh.readline()  # read next line

fh.close()                     # close file
```

Reading from a file (4)

- Assume `names.txt` contains 2 lines

```
John
Dunnion
```

- The output of running the program is the following:

```
Enter a file name: names.txt
John
Dunnion
```

Common functions for accessing files (2)

- `fh.readline()` returns the next line in the file associated with the file handle `fh`

  If a blank line is read, a newline (`\n`) will be returned.

  If an empty string (“”) is returned, the end of file (EOF) has been reached

- `fh.write(s)` writes the string `s` to the end of the file associated with the file handle `fh`

Checking for a file's existence (1)

- To program defensively/carefully/sensibly(!!), we should make sure that a file exists before we open it for reading

```
Enter a file name: names1.txt

Traceback (most recent call last):
  File "/home/john/Documents/dept/comp10280/2015
    fh1 = open(filename, 'r')
IOError: [Errno 2] No such file or directory:
```

- We might also want to check whether a file exists before opening it for writing
- Why?

Checking for a file's existence (2)

- To check for a file's existence, we can use a number of techniques
- One technique is to use the function `os.path.isfile(path)`
- This returns `True` if `path` is an existing regular file and returns `False` otherwise
- We need to include the line **import os** to access this function e.g.

```
import os
```

```
# Checks that the file exists first
import os
# Prompt the user for a file name
filename = input(’Enter a file name: ’)
# Check whether the file exists

if not os.path.isfile(filename):
   print('File:' + filename + ' does not exist')
else:
   fh1 = open( filename, 'r’)

   line = fh1.readline() # read 1st line from file
   while line != "":
      print(line, end = "")
      line = fh1.readline()

fh1.close()
```

Write a program to read daily rainfall amounts and store them in a file

```
# write_rain.py: Create file  to store daily rainfall amounts in mm

fname = input("\nEnter filename to be created: ")

fout = open( fname, "w") # Create new file

text = input("\nEnter raifall amount or Press Enter to quit: ")

while text != "":
    fout.writelines( [text, "\n"] )
    text = input("\nEnter rainfall amount or Enter to quit: ")

fout.close() # Close the file

print("\nFile: ", fname, "created \n")
```

Running write_rain.py

```
Enter filename to be created: rain.txt

Enter raifall amount or Press Enter to quit: 12

Enter rainfall amount or Enter to quit: 10

Enter rainfall amount or Enter to quit: 5

Enter rainfall amount or Enter to quit:

File:  rain.txt created
```

Check the contents of rain.txt

```
% cat rain.txt

12
10
5
%
```

Write a program to calculate and display average, minimum and maximum daily rain fall using data in file created by write_rain.py

```
# read_rain.py:  Calculate and display average, minimum and maximum
import os
import sys

fname = input("\nEnter filename containing rainfall data:  ")

if not os.path.isfile(fname):
   print('File: ' + fname + ' does not exist \n’)
   print('\nQuitting ..\n’)
   sys.exit()

fin = open(fname, "r")

line = fin.readline()            # Read 1st line
if line != "" :                  # Check there was data in the file
   daily_rain = float( line)
   min_rain = daily_rain            # Set minimum rainfall in a day
   max_rain = daily_rain            # Set max rain in a day
   total_rain = 0    # Average = total  amount/ number of days
num_days = 0         # Number of days rain in file
```

Running write_rain.py

```
while line != "": # while line not empty - not end of file
    total_rain = total_rain + daily_rain
    num_days = num_days + 1
    if (daily_rain > max_rain):
        max_rain = daily_rain
    if (daily_rain < min_rain):
        min_rain = daily_rain
    line = fin.readline() # read next line from file

    if line != "":
        daily_rain = float( line )    # end of while loop

fin.close()

if num_days > 0:                    # if there was data in the file
    average_rain = total_rain / num_days
    print("\nAverage daily rain:   ", average_rain)
    print("\nMinimum daily rain: ", min_rain)
    print("\nMaximum daily rain: ", max_rain)
    print("\nNumber of days: ", num_days, "\n\n")
else:
     print("No data in: ", fname, "\n")
```

Running read_rain.py

```
Enter filename containing rainfall data:  rain.txt


Average daily rain:    9.0


Minimum daily rain:  5.0


Maximum daily rain:  12.0


Number of days:  3
```