IST 1025

Introduction to Programming

Files

Data Storage

- Data (and programs) are loaded into *primary memory* (RAM) for processing

- From where?
  - From input devices (keyboard, microphone)
  - From *secondary memory* - a hard disk, a flash stick, a CD, or a DVD

What Is a File?

- A *file* is a software object that allows a program to represent and access data stored in secondary memory (hard disk, flash disk, CD, DVD)

- Two basic types of files:
  - Text files: for storing and accessing text (characters)
  - Binary files: executable programs and their data files (such as images, sound clips, video)

Text Files

- A text file is logically a sequence of characters
- Basic operations are
  - input (read characters from the file)
  - output (write characters to the file)
- There are several flavors of each type of operation

File Input

- We want to bring text in from a file for processing

- Three steps:
  - Open the file for input
  - Read the text and save it in a variable
  - Process the text

Opening a File

```
<a variable> = open(<a file name>, <a flag>)
```

<a flag> can be

- **'r'** - used for input, to *read* from an existing file
- **'w'** - used for output, to *overwrite* an existing file
- **'a'** - used for output, to *append* to an existing file

Example: Read Text from a File

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

myfile = open(filename, 'r')

text = myfile.read()

print(text)
```

The file name must either be in the current directory or be a pathname to a file in a directory

Python raises an error if the file is not found

`text` refers to one big string

Read Lines of Text from a File

```
filename = input('Enter a file name: ')
myfile = open(filename, 'r')

for line in myfile:
    print(line)
```

The variable **`line`** picks up the next line of text on each pass through the loop

`line` will contain the newline character, so the echo will print extra blank lines

Read Lines of Text from a File

```
filename = input('Enter a file name: ')
myfile = open(filename, 'r')

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

Extract a substring up to but not including the last character (the newline)

This will produce an exact echo of the input file

Alternatively, Use **`readline`**

```
filename = input('Enter a file name: ')
myfile = open(filename, 'r')

while True:
    line = myfile.readline()
    if line == '':
        break
    print(line[:-1])
```

The **`readline`** method reads a line of text and returns it as a string, including the newline character

This method returns the empty string if the end of file is encountered

Count the Words

```
filename = input('Enter a file name: ')
myfile = open(filename, 'r')
wordcount = 0

for line in myfile:
    wordcount += len(line.split())

print('The word count is', wordcount)
```

File Output

- We want to save data to a text file

- Four steps:
  - Open the file for output
  - Covert the data to strings, if necessary
  - Write the strings to the file
  - Close the file

Example: Write Text to a File

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

myfile = open(filename, 'w')

myfile.write('Two lines\nof text')

myfile.close()
```

If the file already exists, it is overwritten; otherwise, it is created in the current directory or path

The **`write`** method expects a string as an argument

Failure to close the file can result in losing data

Example: Write Integers to a File

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

myfile = open(filename, 'w')

for i in range(1, 11):
    myfile.write(str(i) + '\n')

myfile.close()
```

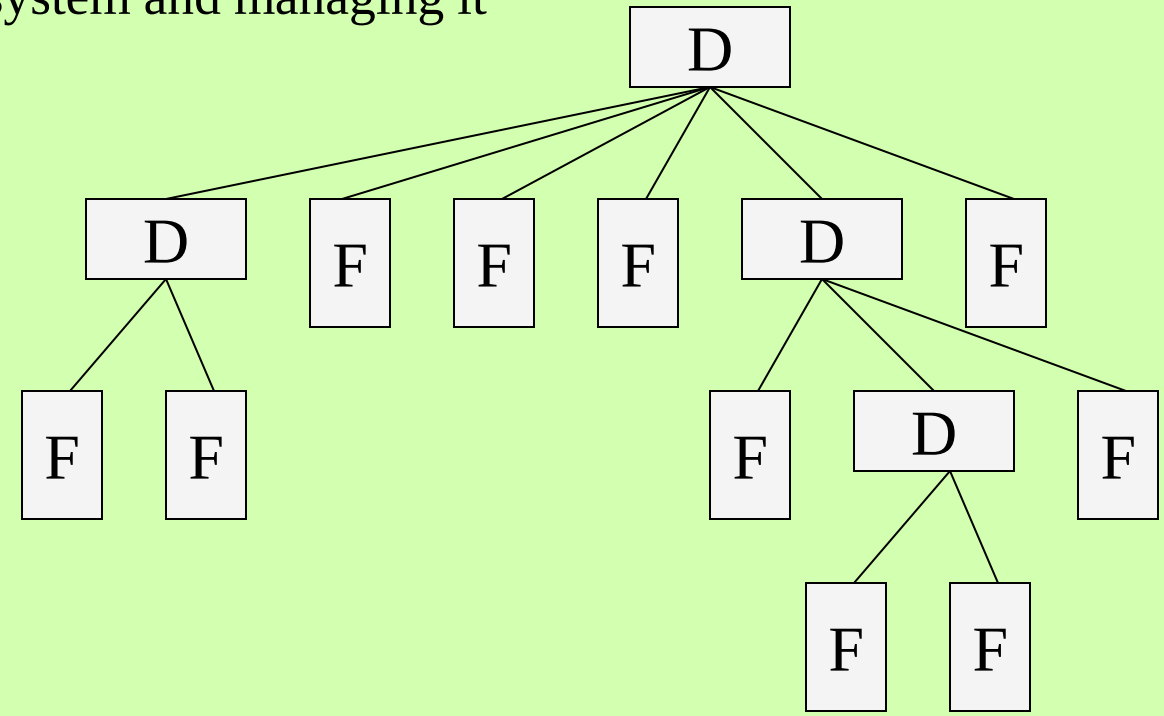
`write` can be called 0 or more times

Data values must be converted to strings before output

Separators, such as newlines or spaces, must be explicitly written as well

Managing Directories

- Directories are organized in a tree-like structure
- Python's **`os`** module includes many functions for navigating through a directory system and managing it



`os` Functions

Function	What It Does
`os.getcwd()`	Returns the current working directory (string)
`os.chdir(path)`	Attaches to the directory specified by path
`os.listdir(path)`	Returns a list of the directory's contents
`os.remove(name)`	Removes a file at path
`os.rename(old, new)`	Resets the old path to the new one
`os.removedirs(path)`	Removes the directory (and all subdirectories) at path

`os.path` Functions

Function	What It Does
`os.path.exists(path)`	Returns **`True`** if **`path`** exists or **`False`** otherwise.
`os.path.isdir(path)`	Returns **`True`** if **`path`** names a directory or **`False`** otherwise.
`os.path.isfile(path)`	Returns **`True`** if **`path`** names a file or **`False`** otherwise.
`os.path.getsize(path)`	Returns the size of the object named by **`path`** in bytes.

Example: Does the File Exist?

```
import os.path

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

if not os.path.exists(filename):
    print('Error: the file not not exist!')
else:
    myfile = open(filename, 'r')
    print(myfile.read())
    myfile.close()
```