UNIVERSITY OF Hull

Using Files

Wrestling with Python
Classes

Persisting Data

- At the moment the data in our programs is discarded when we exit the Python system
- Programs need a way of persisting data
- Python can store data using the file system of the computer
- This also means that Python programs can share data with other programs

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

- The variable f refers to a "file stream" that connects the program to a file

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

- The file is called 'Fred.txt'

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

- This is the filename which includes a language extension marking the file as text

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

- This request that the file be opened for writing – r would mean reading

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- These four lines of text create a file that contains two lines:

```
Line 1
Line 2
```

- This character sequence means "Take a New Line" here

Writing into a file

```
f=open('Fred.txt',mode='w')
f.write('Line 1\n')
f.write('Line 2')
f.close()
```

- It is **very important** that the file is closed when the writing has finished
- This is the point at which the text is actually written into the file

University of Hull

File Storage Locations

File Write Location

- If you just give the name of the file it will be created in the location where the program is running
- This might be a Python system directory
  - Your files will be very hard to find on the system
  - Your program might not be permitted to write there

Where files are stored

- The files are stored in different places depending on the system you are using
- Different computers have different places where the files are stored
  - PCs and Macs store files in different ways
  - This also depends on how the PC is configured
  - Some systems will store files on the network

The File Path

```
f=open('C:/users/rob/test.txt',mode='w')
```

- The file path is a string of text that specifies where on the system the file is located
- This would put a file called `test.txt` in the folder for user `rob`

The Drive letter

```
f=open('C:/users/rob/test.txt',mode='w')
```

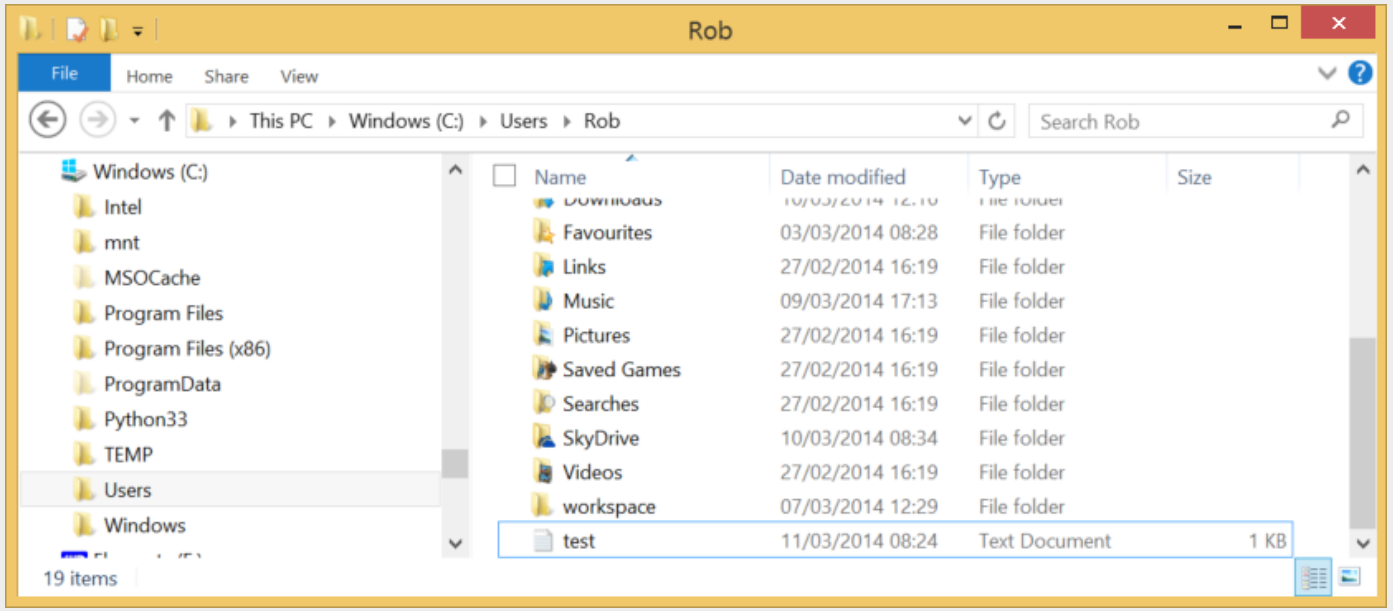
- This specifies the drive where the file is to be stored
- The drive is specified by a letter
- Drive c: is usually the main storage device in your computer
- Other disks may have different letters

The File Path separator

```
f=open('C:/users/rob/test.txt',mode='w')
```

- The / character separates each of the folders in the path to this file
- In Windows there is a folder that holds another folder for each user of the system
- The separator is different for systems

The Files on the Disk



- This is where the file ends up in my home directory

Reading from a File

Reading from a file

```
f=open('Fred.txt',mode='r')
line = f.readline()
print(line)
f.close()
```

- A file can be opened in read mode
- Each call of `readline` will fetch the next line of the file
- Again, you should close the file when finished

Detecting the end a file

```
x = f.readline()
if len(x) == 0:
    print('End of file')
```

- A program can detect when the end of a file has been reached by checking for a line length of zero
- An empty line in a file will have a length of 1, the linefeed character

Reading all the lines a file

```
f=open('Fred.txt',mode='r')
for line in f:
    print(line)
```

- A file can be used as the source of a for loop iteration
- The content of the loop will then be performed for each of the lines in the file

Line endings

```
>>> f=open('Fred.txt',mode='r')
>>> a = f.readline()
>>> a
'Line 1\n'
```

- When you read a line from a file you will get the line ending character in the string
- This can cause problems with printing as you will get a blank line after each line

Stripping

```
>>> a = a.strip()
>>> a
'Line 1'
```

- All string variables provide some methods that can act on them and return the result
- The strip method removes non-printing characters from the start and the end of a string

Writing Numbers

```
score = 6
f.write(str(score))
```

- When we write a number value to a file it must be converted into a string before it can be written
- When the value is read back the program will have to use the `int` function to convert the string back into a value

Exceptions

```
try:
    f=open('Fred.txt',mode='r')
    line = f.readline()
    print(line)
    f.close()
except:
    print("Error reading from file")
```

- When you use files you should capture exceptions that may be raised

Writing Structured Data

Writing out structured data

- You often have to write out the contents of a list of items
- You can use foreach loop to work through each item in the list
- It is a good idea to write out the number of items in the list first

Writing a list of golf rounds

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This code writes out the golf rounds to a file

Opening the file

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This opens the file for output
- The name is a property of the round

Getting the number to write

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This gets the length of the list of rounds that are to be stored

Saving the number being written

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This writes the length out into the file

Writing each round

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This writes out each round
- Note that the round is able to write itself

Closing the file

```
file = open(self.fileName,"w")
numOfRounds = len(self.rounds)
file.write(str(numOfRounds)+'\n')
for round in self.rounds:
    round.writeToFile(file)
file.close()
```

- This closes the file

Reading Structured Data

Reading the rounds back

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- The behaviour to read the rounds file back is very similar

Open the file for reading

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Open the file for reading

Find out how many rounds there are in the file

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Retrieve the number of lines that were written

Read each round in turn

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Go round the loop once for each round

Create an empty round

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Make an empty round

Read the round from the file

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Get the round to read itself from the file

Store the newly read round

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Put the new round on the end of the list

Close the file

```
file = open(self.fileName,"r")
numOfRounds = int(file.readline())
for i in range(numOfRounds):
    round = Round()
    round.readFromFile(file)
    self.rounds.append(round)
file.close()
```

- Close the file when you have finished

Using split to break up input text

```
t = "1 2 3 4"
u = t.split()
print(u)
['1', '2', '3', '4']
```

- You can ask a string to split itself into a list of the items in it

Using split to break up input text

```
t = "1 2 3 4"
u = t.split()
print(u)
['1', '2', '3', '4']
```

- This is very useful if you are reading items from a file

Using split to break up input text

```
t = "1 2 3 4"
u = t.split()
print(u)
['1', '2', '3', '4']
```

- The items are split on any “whitespace” character

Using split to break up input text

```
t = "1 2 3 4"
u = t.split(' ')
print(u)
['1', '2', '3', '4']
```

- You can specify the character to split on if you like

Save methods in objects

Division of Purpose

- It is sensible to make the data classes responsible for saving and loading their own data
- That way, if the design of the data changes we only have to update the behaviours in the classes themselves

Round Save Method

```
def writeToFile(self, file):
  try:
     for hole in self.scoreCard:
        s = str(hole.par)+' '+str(hole.shots)
        file.write(s+'\n')
  except:
     print('Error writing to file',file.name)
```

- This method saves a round to a file
- The file to save to is passed as a parameter

Passing a file as a parameter

```
def writeToFile(self, file):
   try:
      for hole in self.scoreCard:
         s = str(hole.par)+' '+str(hole.shots)
         file.write(s+'\n')
   except:
      print('Error writing to file',file.name)
```

- The method does not open the file to save into
- The file has already been opened

Passing a file as a parameter

```
def writeToFile(self, file):
   try:
      for hole in self.scoreCard:
         s = str(hole.par)+' '+str(hole.shots)
         file.write(s+'\n')
   except:
      print('Error writing to file',file.name)
```

- This will append the round information onto the end of an already open file

Two items on one line

```
def writeToFile(self, file):
   try:
      for hole in self.scoreCard:
         s = str(hole.par)+' '+str(hole.shots)
         file.write(s+'\n')
   except:
      print('Error writing to file',file.name)
```

- The par and the shots values are written on the same line of the file
- The read method uses the split function

Filename in error reports

```
def writeToFile(self, file):
   try:
      for hole in self.scoreCard:
         s = str(hole.par)+' '+str(hole.shots)
         file.write(s+'\n')
   except:
      print('Error writing to file',file.name)
```

- You can find out the name of a file that is being used

Improving the golf club programs

- Warren has created worked solutions for the Golf club tasks 1-3
- The customer for the program likes them very much
- They would like an additional behaviour to be added…

Saving/Loading Game Results

- The customer would like to be able to save the results of games (Task 2)
- The scores for each player and who won the game must the stored in a text file which can be loaded later
- You can use the save and load behaviours from Task 3 as a good place to look for ideas