

# **Programming for GCSE**

## **Topic 10.1: Files**

William Marsh

School of Electronic Engineering and Computer Science  
Queen Mary University of London

---

# Aims

- Be able to develop simple programs for text I/O to files
  - Understanding teaching issues for files
    - Output is easier than input
    - Need to 'design' a file format
-

# Outline

- File I/O Operation
  - Example programs
  - Issues of I/O: asymmetry of input and output
    - Output: you know
    - Input: you don't know
  - Input problem
    - Designing a file format
-



# BASIC FILE OPERATIONS

---

# Operations on Files

- open
    - File name: a string
    - Mode: 'r' (read) or 'w' (write)
    - Creates a 'file object'
  - write
    - Write strings: lines end with `\n`
  - readline
    - Read a line
  - close
-

# What is a file?

- Sequence of bytes
    - *What do these bytes represent?*
  - A text file has strings
    - Often organised in lines
    - Example line: "This is a line\n"
  - What does the string represent?
    - Could be a number
-

# Where is the File – Buffering

- File is on disk
  - BUT
    - Not every character causes a write to disk
    - *Why not?*
    - File is 'buffered'
  - Usually, this is invisible
    - Can lead to strange errors
-



# EXAMPLE PROGRAMS

---



# Open – Write – Close

```
import io

f = open("hello.txt", 'w')
f.write("This is a line\n")
f.write("This is a string ")
f.write("followed by more on the line\n")
f.close()
```

This is a line

This is a string followed by more on the line

---

# Writing Numbers

- First convert the number to a string

```
import io

f = open("numbers.txt", 'w')
f.write(str(10) + "\n")
f.write(str(100) + "\n")
f.close()
```

---

# Reading – readline

- Line from file becomes string
  - Includes the "\n"

```
import io

f = open("numbers.txt", 'r')
line1 = f.readline()
line2 = f.readline()
print("Line 1:", line1)
print("Line 2:", line2)
f.close()
```

# Reading in a Loop

- The following pattern is very important
  - Read all lines in a loop

```
import io
f = open("hello.txt")

l = f.readline()
while len(l) > 0 :
    print(l)
    l = f.readline()

f.close()
```

Read first line

Stop when line blank

Read next line  
at end of loop

# End of the file?

- In Python
    - `f.readline()` gives an empty string at the end of a file
    - *What about blank lines?*
    - Blank lines are NOT empty: a blank line contains a single `"\n"` character
      - Note: `"\n"` is a line separator; there may not be one at the end of the last line
  - Other languages use different methods to signal the end of file
-



DESIGN A FILE

---

# Problem: Phone Numbers

- Phone numbers are held in a file
    - Name
    - Number
  - A program reads the file and searches for the name
    - Prints the number if name is found
  - How should the file be organised?
-

# Ideas

```
Alice,0207001111  
Bob,0207002222  
Charlie,0207003333
```

```
Alice  
0207001111  
Bob  
0207002222  
Charlie  
0207003333
```

```
Alice  
Bob  
Charlie  
0207001111  
0207002222  
0207003333
```

---



# Using Split

```
Alice,0207001111  
Bob,0207002222  
Charlie,0207003333
```

- Multiple values on a line have to be separated
- 'split' method: string → list of strings

```
>>> astring = "Bob,12345,SW13 4NN"  
  
>>> astring.split(",")  
['Bob', '12345', 'SW13 4NN']  
  
>>> astring.split("3")  
['Bob,12', '45,SW1', ' 4NN']  
>>>
```

# Reading Numbers

- We use split to discard the "\n" character at the end of lines containing a number

```
f = open("numFile.txt", 'r')
line1 = f.readline()
f.close()

num = line1.split("\n")[0]
print("Double the number is", int(num)*2)
```

---



# TEACHING ISSUES

---

# Issues for I/O

- Must understand representation:
    - e.g. number versus text
    - e.g. lines
  - File operation are object-based
    - Open returns an object
    - Do operations on the object
-

# Issues for I/O

- Input is difficult:
    - How many lines?
    - What is on the lines?
  - Design file contents so it is easy to read
  - Error handling
    - E.g. file name wrong
    - Needs 'exceptions'
-



# SYLLABUS

---

### 2.3.1 Programming techniques

Candidates should be able to:

- (a) identify and use variables, operators, inputs, outputs and assignments
- (b) understand and use the three basic programming constructs used to control the flow of a program: Sequence; Conditionals; Iteration
- (c) understand and use suitable loops including count and condition controlled loops
- (d) use different types of data including Boolean, string, integer and real appropriately in solutions to problems
- (e) understand and use basic string manipulation
- (f) understand and use basic file handling operations: open, read, write and close**
- (g) define and use arrays as appropriate when solving problems.

# Summary

- Files
    - Small number of operations
    - Combines other aspects of programming:
      - Loops
      - Representation: including lines
  - Input challenge
  - Some issues
    - Object-based library
-