

## Generic file reading function

- often we want to read a file and process it a line at a time
  - we could write a function easily enough..
  - but we do this many times
- another way is to generate a library function which reads in the file, and calls another function a line at a time

## Generic file reading function

```
#!/usr/bin/python
from sys import argv

def scanner(name, function):
    file = open(name, 'r')
    line = file.readline()
    while line:
        function(line)
        line = file.readline()
    file.close()

def processLine(line):
    print line

filename = 'ftplotg.data'
scanner(filename, processLine)
```

## imap library and Python

- the Internet Message Access Protocol (commonly known as IMAP or IMAP4) is an application layer Internet protocol that allows a local client to access email on a remote server
- in Python we can use the `imaplib` to download email

## imap library and Python

```
#!/usr/bin/python
import getpass, imaplib, string

m = imaplib.IMAP4_SSL('moppsy.comp.glam.ac.uk')
m.login(getpass.getuser(), getpass.getpass())
m.select()
typ, data = m.search(None, 'ALL')
print "typ = ", typ
print "data = ", data
for num in string.split(data[0]):
    typ, data = m.fetch(num, '(RFC822)')
    print 'Message %s\n%s\n' % (num, data[0][1])
m.logout()
```

## imap library and Python

- in this program the `m.search` function returns two values: `typ` and `data`
- `typ` is a status code and `data` is a list of strings, the first string is a space separated list of numbers indicating the available messages
- hence the construct `for num in string.split(data[0]):` extracts each number in sequence and assigns it to `num`

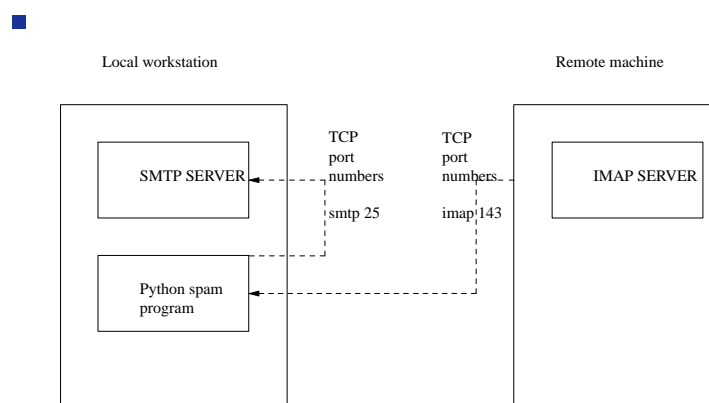
## Spam and email

- your task is to write a spam filter in Python
- this will use networking principles as discussed in lectures
- and use Python modules to draw on the services which sit above TCP

## Spam and email

- your program needs to download email messages from `moppsy.comp.glam.ac.uk`
  - perform a spam test on the message
  - if it is not spam then it needs to send it to a local smtp port

## Spam and email



- finally you need to document your program and suggest improvements

## moppsy

- is a Debian GNU/Linux client and server and it will run an imap service
  - you can ssh into `moppsy.comp.glam.ac.uk` and you can email yourself a test message

```
ssh moppsy.comp.glam.ac.uk
Login: u012345678
Password:
$ mail -s "this is a test" u012345678@localhost
this is the body of the message
.
$ exit
```

## Skeleton solution

- you can use this as a basis for your submission `skeleton <tostudents.data>`

## Strings

- the string module is very important
  - many tasks involve string handling
- network protocols also requires good string manipulation
  - http is string based: GET, PUT etc
  - smtp is also string based: Subject, From, To, fields etc
- string module provides functions to
  - split strings into words
  - strip newlines off the end of a string
  - find substrings
- consider reading through the [online Python documentation <../../../../python/html/index.html>](#) and examine the string module

## Coursework skeleton snippet

```
■ #
# isSpam - returns 1 if the message looks like spam
#         returns 0 if the message looks ok
#
def isSpam(message):
    total = 0
    htmlFound = 0
    for line in StringIO(getBody(message)).readlines():
        print line
    # incomplete
```

## Coursework skeleton snippet

```
#
# postEmail - sends message to the smtp port
#
def postEmail(message):
    print "message from", getField(message, "From:")
    print "subject is", getField(message, "Subject:")
    # incomplete
```

## Coursework skeleton snippet

```
#
# handleEmail - tests whether the message is spam or not
#                 If spam then it adds it to a file
#                 otherwise post it to the smtp port
#
def handleEmail(message):
    if (isSpam(message)):
        print "message is spam"
    else:
        print "message is good"
        postEmail(message)
    # incomplete
```

## Coursework skeleton snippet

```
#
# collectEmail - retrieves the email from the imap server:
#
def collectEmail():
    m = imaplib.IMAP4_SSL('moppsy.comp.glam.ac.uk')
    m.login(getpass.getuser(), 'yourpasswordgoeshere')
    m.select()
    typ, data = m.search(None, 'ALL')
    print data
    for num in string.split(data[0]):
        print num
        typ, data = m.fetch(num, '(RFC822)')
        handleEmail(data[0][1])
    m.logout()
collectEmail()
```

## Tutorial Assignment support

- divide and conquer is the classic Computer Science method for solving large projects
  - in our case the spam filter coursework could be split up into three areas
    - downloading the emails from the remote server
    - detecting for the presence of spam (>=60% of lines contain an HTML tag)
    - posting non spam emails locally (using smtp - see last weeks tutorial)

## Downloading emails from the remote server

- note be careful, the following examples use **bold** font for your keyboard input and normal courier for terminal output
  - obviously you need to substitute **u012345678** for your real enrolment number
- open up a terminal and login to the imap server `moppsy.comp.glam.ac.uk` and send your self a test email message. You can do this by:

## Downloading emails from the remote server

```

$ ssh moppsy.comp.glam.ac.uk
login: u012345678
Password:
$ mail -s "this is a test" u012345678@localhost
this is the body of the message
.
CC: press the enter key here
$ exit

```

## Downloading emails from the remote server

- now test whether your email message is resident on the server by:

```

ssh moppsy.comp.glam.ac.uk
login: u012345678
Password:
$ mail
Mail version 8.1.2 01/15/2001. Type ? for help.
"/var/mail/u012345678": 1 message 1 new
N 1 u012345678@localhost Wed Nov 15 13:30 this is a tes
& x
$ exit

```

## Downloading emails from the remote server

- now cut and paste this program into a file called `testimap.py`

```

#!/usr/bin/python

import getpass, imaplib, string

m = imaplib.IMAP4_SSL('moppsy.comp.glam.ac.uk')
m.login(getpass.getuser(), getpass.getpass())
m.select()
typ, data = m.search(None, 'ALL')
print "typ = ", typ
print "data = ", data
for num in string.split(data[0]):
    typ, data = m.fetch(num, '(RFC822)')
    print 'Message %s\n%s\n' % (num, data[0][1])
m.logout()

```

## Downloading emails from the remote server

- check it works..
- now download the `skeleton` `<tostudents.data>` program and run it
- check that it prints your test message