Python and file handling

- file manipulation primitives are by default available
  - no need to import library to, read, write files
- creating a simple text file
-

```
#!/usr/bin/python

file = open("newfile.txt", "w")
file.write("hello world in the new file\n")
file.write("and another line\n")
file.close()
```

-

```
#!/usr/bin/python

file = open("newfile.txt", "w")
file.writelines(["hello world in the new file\n",
                 "and another line\n"])
file.close()
```


Python and file handling

-

```
#!/usr/bin/python

file = open("newfile.txt", "r")
for line in file.readlines():
    print line
```

- many ways to read a file
  - `file.read()` returns a string containing all characters in the file
  - `file.read(N)` returns a string containing next N characters
  - `file.readline()` returns a string containing characters up to \n
  - `file.readlines()` returns the complete file as a list of strings each separated by \n

Further Python Networking

- many python modules which give access to application layer networking services
  - ftp, http, telnet, etc
- sometimes you may have to implement your own application layer protocol
- in which case you use `sockets` (a transport layer service)

client.py (TCP)

-

```
#!/usr/bin/python

import sys
from socket import *
serverHost = "localhost"
serverPort = 2000

# create a TCP socket
s = socket(AF_INET, SOCK_STREAM)

s.connect((serverHost, serverPort))
s.send("Hello world")
data = s.recv(1024)
print data
```

server.py (TCP)

```
#!/usr/bin/python

from socket import *
myHost = ""
myPort = 2000

# create a TCP socket
s = socket(AF_INET, SOCK_STREAM)
# bind it to the server port number
s.bind((myHost, myPort))
# allow 5 pending connections
s.listen(5)

while True:
    # wait for next client to connect
    connection, address = s.accept()
    while True:
        data = connection.recv(1024)
        if data:
            print "received data", data
            connection.send("echo -> " + data)
        else:
            break
    connection.close()
```


udpserver.py Python UDP server

- the UDP server

```
#!/usr/bin/python

from socket import *
myHost = ''
myPort = 2000

# create a UDP socket
s = socket(AF_INET, SOCK_DGRAM)
# bind it to the server port number
s.bind((myHost, myPort))

while True:
    data, address = s.recvfrom(1024)
    print "UDP server:", data, "from", address
    if data:
        s.sendto('echo -> ' + data, address)
    else:
        break
```


udpclient.py Python UDP client

- and the client

```
#!/usr/bin/python

import sys
from socket import *
serverHost = 'localhost'
serverPort = 2000

# create a UDP socket
s = socket(AF_INET, SOCK_DGRAM)

s.connect((serverHost, serverPort))
s.send('Hello world')
data = s.recv(1024)
print data
```


IMAP library

```
#!/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()
```

- tutorial read the Python documentation ⟨`../../../python/html/index.html`⟩ and modify the imap example above to delete your read email messages

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" 012345678@localhost
this is the body of the message
^D
$ exit
```


Tutorial

- command line arguments familiarity, copy this program and call it, `arguments.py`

```
#!/usr/bin/python

import sys, getopt

def Usage ():
    print "arguments.py [-v][-p][-h]"

try:
    optlist, list = getopt.getopt(sys.argv[1:],
                                  ":vph")
except getopt.GetoptError:
    Usage()
    print "incorrect argument given"
    sys.exit(1)

for opt in optlist:
    if opt[0] == "-h":
        Usage()
    if opt[0] == "-v":
        print "verbose found"
    if opt[0] == "-p":
        print "probeonly found"
```


Tutorial

- run this program using a terminal and try the following commands:

```
python arguments.py -v
python arguments.py -h
python arguments.py -p
python arguments.py -p -v
python arguments.py -pv
python arguments.py -pvh
python arguments.py -pv -h
python arguments.py -vp -h
python arguments.py -v -p -h
python arguments.py -h -phv
```

- notice that the order of the arguments are irrelevant

Tutorial

- copy the following program and call it `client.py`

```
#!/usr/bin/python

import sys
from socket import *
serverHost = "localhost"
serverPort = 2000

# create a TCP socket
s = socket(AF_INET, SOCK_STREAM)

s.connect((serverHost, serverPort))
s.send("Hello world")
data = s.recv(1024)
print data
```

now copy this program and call it server.py

Tutorial

```
#!/usr/bin/python

from socket import *
myHost = ""
myPort = 2000

# create a TCP socket
s = socket(AF_INET, SOCK_STREAM)
# bind it to the server port number
s.bind((myHost, myPort))
# allow 5 pending connections
s.listen(5)

while True:
    # wait for next client to connect
    connection, address = s.accept()
    while True:
        data = connection.recv(1024)
        if data:
            print "received data", data
            connection.send("echo -> " + data)
        else:
            break
    connection.close()
```


Tutorial

- now run `server.py` by opening up a terminal and typing:
- `python server.py`

Tutorial

- now open up another terminal and type:
- `python client.py`

Tutorial

- now modify your `client.py` file so that you connect to your neighboughs machine (hint you can find the name of your neighbough machine by opening up a terminal and typing):
- `hostname`
- and adding 1 or subtracting 1 from the end value
- this machine name needs to replace the string `localhost` in `client.py`
- now rerun your `client.py` program

Tutorial

- now modify your server program so that it uses the string to attempt to read a file and return its contents (choose the filename `README`)
  - using an editor create the file `README` (notice python is case sensitive on GNU/Linux so use capitals for this example)
  - hint see earlier section on file handling
- now stop your local server from running and restart it
- modify your client program so that it connects to `localhost` and not your neighboughs machine
  - and debug your code
- when your code works modify your client to connect to your neighboughs machine