

Arguments in Python

- as we have seen in bash, `getopts`, provides a useful method for handling arguments
 - in fact many languages have adopted `getopts`
 - C, C++, bash and python
- `autoftp` has three which had three options:
 - `-v` verbose on
 - `-p` probeonly
 - `-h` help

Autoftp arguments in python

- ```
#!/usr/bin/python
import sys, getopt

def Usage ():
 print "autoftp [-v][-p][-h]"
 sys.exit(0)

optlist, list = getopt.getopt(sys.argv[1:], ':vphf:')
print "optlist =", optlist
print "list =", list
for opt in optlist:
 print opt
 if opt[0] == '-h':
 Usage()
 if opt[0] == '-f':
 print "file found"
 if opt[0] == '-v':
 print "verbose found"
 if opt[0] == '-p':
 print "probeonly found"
```

## Autoftp arguments in python

- notice that the script fails if an unsupported option is issued

- ```
./autoftp2.py -x
...
getopt.GetoptError: option -x not recognised
```

Better argument handling

- so we need a way to trap these errors
 - python uses an exception handler for this

- ```
#!/usr/bin/python
import sys, getopt

def Usage ():
 print "autoftp [-v][-p][-h]"
 sys.exit(0)

try:
 optlist, list = getopt.getopt(sys.argv[1:], ':vphf:')
except getopt.GetoptError:
 Usage()
 print "called exception"
 sys.exit(1)

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

## Better argument handling

- when run it yields the following

```
./autoftp3.py -x
autoftp [-v][-p][-h]
```

## When is a module not a module?

- it is often useful to create a module
  - for yourself and others to use in the future
  - to subdivide the large problem set into a number of smaller modules
- sometimes a module might be able to operate as a stand alone program
  - consider autoftp could be organised as a module

## When is a module not a module?

```
if __name__ == "__main__":
 main()
```

- which means run the function `main` if this module is explicitly invoked by the user
  - note that it is not run if this module was imported

## Example times module

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

def multiplyby10(value):
 return value+'0'

if __name__ == "__main__":
 if len(sys.argv) == 2:
 print 'testing the times module'
 print multiplyby10(sys.argv[1])
```

- the above code needs to be saved in file `times.py`

## Example program

```
#!/usr/bin/python
import times, sys

if len(sys.argv) == 2:
 print 'importing the times module'
 print times.multiplyby10(sys.argv[1])
```

- note that the module times takes a string and adds a '0' to the left hand side
  - effectively multiply by 10
- note it also uses the `if __name__ == "__main__":` condition which only calls the multiply routine if this module was invoked as the main program by the user

## Example program

## Example program

```
./prog.py 12
importing the times module
120
```

```
./times.py 12
testing the times module
120
```

- exercise for the reader, add a function to perform divide and modulus of a numerical integer string

**Tutorial**

- cut and paste the following program into a file called: lab7.py

```
#!/usr/bin/python
import sys, getopt

optlist, list = getopt.getopt(sys.argv[1:], ':vphf:')
print "optlist =", optlist
print "list =", list
```

**Tutorial**

- run the program from the command line

```
$ python lab7.py -v
$ python lab7.py -h
$ python lab7.py -p
$ python lab7.py -pvh
$ python lab7.py -p -vh
$ python lab7.py -p -vf foobar
```

- ensure that you understand what the `optlist` values mean and how they relate to the options on the command line

**Tutorial**

- now improve the program by adding the `for` loop as mentioned in the lecture

**Tutorial**

```
#!/usr/bin/python
import sys, getopt

optlist, list = getopt.getopt(sys.argv[1:], ':vphf:')
print "optlist =", optlist
print "list =", list
for opt in optlist:
 print "opt[0] = ", opt[0], "opt[1] = ", opt[1]
 if opt[0] == '-h':
 print "usage"
 if opt[0] == '-f':
 print "file found"
 if opt[0] == '-v':
 print "verbose found"
 if opt[0] == '-p':
 print "probeonly found"
```

**Tutorial**

- and test it again using the command line above
  - for which option is `opt[1]` useful?
  - what does `opt[0]` contain?

**SMTP module**

- ```
#!/usr/bin/python
import smtplib, string, sys, time

mailserver = "localhost"

From = string.strip(raw_input('From: '))
To = string.strip(raw_input('To: '))
Subject = string.strip(raw_input('Subject: '))

Date = time.ctime(time.time())
Header = ('From: %s\nTo: %s\nDate: %s\nSubject: %s\n\n'
         % (From, To, Date, Subject))

Text = "my message"
server = smtplib.SMTP(mailserver)
failed = server.sendmail(From, To, Header + Text)
server.quit()
if failed:
    print 'failed to send mail'
else:
    print 'all done..'
```

SMTP module

- cut and paste the above code into a file `mymail.py`
- run the program from the command line and enter the from and to fields as your GNU/Linux username
- now alter the program so that `mailserver` is set to `mailserv1.uni.glam.ac.uk`

SMTP module

- now run the program again and this time set the `To:` field and `From:` field to your University email address
- check that your program has sent the email

SMTP module

- finally if you have time, modify the program to read in a sequence of lines to make up the `Text` component of the email message
 - when a blank line is read from the keyboard the message is complete