Multithreading

OOSD

School of Information Technology
Otago Polytechnic
Dunedin, New Zealand

Enter and run this program

```
import threading
import time

 def timer(name, delay, repeat):
     print("Timer " + name + " starting.")
     for i in xrange(repeat):
         time.sleep(delay)
         print(name + " " + time.ctime(time.time()))
     print("Timer " + name + " finishing.")

 if __name__ == "__main__":
     t1 = threading.Thread(target=timer, args=("Timer 1", 2, 5))
     t2 = threading.Thread(target=timer, args=("Timer 2", 4, 5))
     t1.start()
     t2.start()

     print("Main complete")
```

Threads

- Threads are the smallest unit that can be scheduled by the operating system.
- Every process has at least one thread.
- Threads can share memory

Syncronisation with `join`

```
t1 = threading.Thread(target=timer, args=("Timer 1", 2, 5))
t2 = threading.Thread(target=timer, args=("Timer 2", 4, 5))
t1.start()
t2.start()
print("Do stuff while threads run")
t1.join()
t2.join()
print("All threads complete")
```

Uses for threads

- *Parallel processing*: We can break a task into parts that can be performed in parallel. This can speed up the process.
- emphAsynchonous processing: We carry out a time consuming task in a seperate thread while the main process remains responsive.

Thread class

In the following examples we will use the `Thread` class from the `threading` module. We do this to create objects that run in their own threads.

Parallelisation example

```
import os, re, threading

class PingCheck(threading.Thread):
    received_packages = re.compile(r"() received")

    def __init__ (self,ip):
        threading.Thread.__init__(self)
        self.ip = ip
        self._successful_pings = -1
```

Parallelisation example

```
def run(self):
    ping_out = os.popen("ping -q -c2 "+self.ip,"r")
    while True:
        line = ping_out.readline()
        if not line: break
        n_received = re.findall(self.received_packages,line)
        if n_received:
            self._successful_pings = int(n_received[0])
```

Parallelisation example

```
def status(self):
    if self._successful_pings == 0:
        return "no response"
    elif self._successful_pings == 1:
        return "alive, but 50 % package loss"
    elif self._successful_pings == 2:
        return "alive"
    else:
        return "shouldn’t occur"
```

Parallelisation example

```
if __name__ == "__main__":
    check_results = []
    for suffix in range(1,255):
        ip = "10.25.1."+str(suffix)
        current = PingCheck(ip)
        check_results.append(current)
        current.start()

    for el in check_results:
       el.join()
       print "Status from ", el.ip,"is",el.status()
```

Asynchronous processing example

```
class FileSaver(threading.Thread):
    def __init__(self, text, filename):
        threading.Thread.__init__(self)
        self.text = text
        self.filename = filename

    def run(self):
        f = open(self.filename, ’w’)
        f.write(self.text)
        f.close()
        time.sleep(10)
```

Asynchronous processing example

```
if __name__ == "__main__":
    files = []
    for i in xrange(3):
        msg = raw_input("Enter some text: ")
        fname = "file" + str(i) + ".txt"
        save = FileSaver(msg, fname)
        save.start()
        files.append(save)

    for f in files:
        f.join()
        print(f.filename + " was saved.")
```