Introduction to Network Programming Part II

Apinun Tunpan, Ph.D.
Internet Education and Research Lab
Asian Institute of Technology
September 2010

Today's Outline

1. Some useful Python modules
2. Bluetooth programming
3. GPS device interface
4. Further self studies

1. Some useful Python modules

1.1 Date & time module (time)

```
>>> import time
>>> dir( time )
['__doc__', '__name__', '__package__', 'accept2dyear', 'altzone', 'asctime
', 'clock', 'ctime', 'daylight', 'gmtime', 'localtime', 'mktime', 'sleep',
'strftime', 'strptime', 'struct_time', 'time', 'timezone', 'tzname', 'tzse
t']
>>> time.gmtime()
time.struct_time(tm_year=2010, tm_mon=9, tm_mday=15, tm_hour=8, tm_min=6,
tm_sec=3, tm_wday=2, tm_yday=258, tm_isdst=0)
>>> time.localtime()
time.struct_time(tm_year=2010, tm_mon=9, tm_mday=15, tm_hour=15, tm_min=6,
tm_sec=7, tm_wday=2, tm_yday=258, tm_isdst=0)
>>> time.asctime( time.gmtime() )
'Wed Sep 15 08:06:17 2010'
>>> time.asctime( time.localtime() )
'Wed Sep 15 15:06:24 2010'
>>> time.ctime()
'Wed Sep 15 15:06:29 2010'
```

This returns a struct_time representing UTC time.

This returns a struct_time representing local time.

time.asctime() converts the input struct_time to a readable string

time.ctime() returns a string showing local time.

time.sleep()

```
>>> import time
>>> def GoToSleep( n_sec ):
        print 'Going to sleep on ', time.ctime()
        time.sleep( n_sec )
        print 'Waking up on ', time.ctime()
```

```
>>> GoToSleep( 12 )
Going to sleep on  Wed Sep 15 15:22:50 2010
Waking up on  Wed Sep 15 15:23:02 2010
>>> GoToSleep( 15 )
Going to sleep on  Wed Sep 15 15:23:13 2010
Waking up on  Wed Sep 15 15:23:28 2010
```

Sleep for 12 and 15 seconds

1.2 System module (sys)

```
# SysArgv.py : showing how to access command line parameters

import sys
print sys.argv
```

```
File  Edit  View  Terminal  Help
interlab@ubuntu104:~/PythonNetworking$ python SysArgv.py Network Programming is fun
['SysArgv.py', 'Network', 'Programming', 'is', 'fun']
interlab@ubuntu104:~/PythonNetworking$
```

1.3 Operating system module (os)

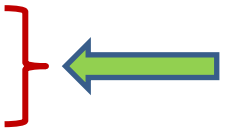
```
>>> import os
>>> os.getcwd()
'/home/interlab/PythonNetworking'
>>> os.listdir( '.' )
['helloworld.py', 'WebbrowserControl.py', 'SysArgv.py', 'RecvMesgUDP.py', 'RecvM
esgTCP.py', 'Part II', 'speak.py', 'SimpleHTMLParser.py', 'SendMesgUDP.py', 'Sen
dMesgTCP.py']
>>> os.stat( 'helloworld.py' )
posix.stat_result(st_mode=33188, st_ino=146737L, st_dev=2049L, st_nlink=1, st_ui
d=1000, st_gid=1000, st_size=20L, st_atime=1284451149, st_mtime=1284450857, st_c
time=1284450857)
>>> os.chdir( 'Part II' )
>>> os.getcwd()
'/home/interlab/PythonNetworking/Part II'
>>> os.mkdir( 'Section 2.1' )
>>> os.chdir( 'Section 2.1' )
>>> os.getcwd( )
'/home/interlab/PythonNetworking/Part II/Section 2.1'
```

Get the current working directory

List the contents of the current directory

Get the information of a specific file

Change the current working directory



Create a new directory and change the current directory to it.

1.3 Operating system module (os) continues

```
>>> import os
>>> p = os.popen( 'ls' )
>>> result = p.read()
>>> result
'helloworld.py\nPart II\nRecvMesgTCP.py\nRecvMesgUDP.py\nSendMesgTCP.py\nSendMesgU
DP.py\nSimpleHTMLParser.py\nspeak.py\nSysArgv.py\nWebbrowserControl.py\n'
>>> print result
helloworld.py
Part II
RecvMesgTCP.py
RecvMesgUDP.py
SendMesgTCP.py
SendMesgUDP.py
SimpleHTMLParser.py
speak.py
SysArgv.py
WebbrowserControl.py

>>> lines = result.split( '\n' )
>>> lines
['helloworld.py', 'Part II', 'RecvMesgTCP.py', 'RecvMesgUDP.py', 'SendMesgTCP.py',
'SendMesgUDP.py', 'SimpleHTMLParser.py', 'speak.py', 'SysArgv.py', 'WebbrowserCont
rol.py', '']
```

Open a pipe and execute the /bin/ls Linux command. Then read the result from the opened pipe.

The result is in fact a string received from the output of the /bin/ls command. Entries of the result are separated by '\n' (the newline character).

If we print this result, it looks nice. But do not forget that it is still a string which can be very long.

str.split() can break down a string into a list. Here we use '\n' as a field separator.

Exercise

- The iwlist command in linux can scan for WiFi access points and show their characteristics (e.g. channels, cell ids, signal strengths). Write a Python program to read the whole output of the iwlist command (e.g. from “iwlist scan”) and keep it as a (very long) string.

Solving a problem

- We are doing a Vehicle-to-Infrastructure (V-2-I) communication project. Suppose that our first step is to monitor and record WiFi access points, their channels and signal strengths, along the road while we drive. Can we build a simple automated tool for this purpose ?

We shall revisit this problem after we learn more about Regular Expressions.

1.4 Regular expression (re) module

- Matching or searching for a specific pattern (specified by a regular expression)
- Substituting one or more occurrences of a pattern (specified by a regular expression)
- Splitting the string based on a regular expression

Common uses of regular expressions

- Re.match() = match if the pattern occurs at the beginning of a string
- Re.search() = search for the first occurrence of a pattern in the whole string
- Re.findall() = search for every occurrence of a pattern in the whole string
- Re.sub() = replace all occurrences of the pattern with a different string.

re.match() vs. re.search()

```
>>> import re
>>> result1 = re.match( 'aa',   'aabbccddeeffaabbccddeeff' )
>>> print result1
<_sre.SRE_Match object at 0x937aa30>
>>> result2 = re.match( 'bb',   'aabbccddeeffaabbccddeeff' )
>>> print result2
None
>>> result3 = re.search( 'bb', 'aabbccddeeffaabbccddeeff' )
>>> print result3
<_sre.SRE_Match object at 0x937ae90>
>>> result1.span()
(0, 2)
>>> result3.span()
(2, 4)
>>> result3.start()
2
>>> result3.end()
4
```

re.match() sees if the pattern occurs at the beginning of a string.

Pattern 'bb' does not occur at the beginning.

Re.search() looks for the pattern inside the whole string.

When we look into a search result, we can see the index range where the pattern occurs in the string.

Note that re.search() returns only the first occurrence.

re.findall() vs. re.finditer()

```
>>> import re
>>> result4 = re.findall( 'cc', 'aabbccddeeffaabbccddeeff' )
>>> print result4
['cc', 'cc']
```

re.findall() returns all occurrences of the matches

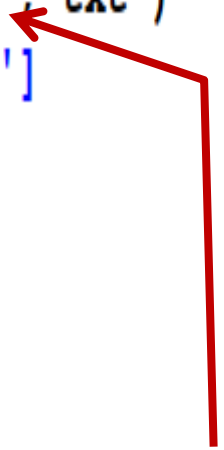
```
>>> iterator = re.finditer( 'cc', 'aabbccddeeffaabbccddeeff' )
>>> for i in iterator:
        print i.span()


(4, 6)
(16, 18)
```

re.finditer() returns an iterator on which you can run a loop (e.g. to find the spans of the occurrences)

Here ‘cc’ occurs twice, at index 4 and index 16.

Example

```
>>> import re
>>> txt = "John actively runs a science project. He probably discovers a new method."
>>> re.findall( '\w+ly', txt )
['actively', 'probably']
```



\w matches any alphanumeric character and the underscore (equivalent to [a-zA-Z0-9_])

\+ means one or more occurrences of the pattern ahead of it.

\w+ly then means one or more of \w instances which end with ly

Adapted from Python documentation: http://docs.python.org/howto/regex.html

Example

```
>>> import re
>>> txt = "John actively runs a science project. He probably discovers a new method."
>>> re.findall( '[rp]\w+', txt )
['runs', 'project', 'probably', 'rs']
>>>
>>> re.findall( '\s+[rp]\w+', txt )
[' runs', ' project', ' probably']
```

[rp] matches r or p.

\w+ matches one or more of any alphanumeric or the underscore.

\s+ matches one or more of whitespaces

Adapted from Python documentation: http://docs.python.org/howto/regex.html

re.sub() : pattern substitution

```
>>> import re
>>> text = 'This\t \tis\fa::very\v,weird\n\tstring.'
>>> text
'This\t \tis\x0ca::very\x0b,weird\n\tstring.'
>>> print text
This             is♩a::very♪,weird
        string.
>>> text1 = re.sub( '[\s:,]+', ' ', text )
>>> text2 = re.sub( 'weird', 'easy to read', text1 )
>>> print text2
This is a very easy to read string.
```

Here we replace every occurrence of this pattern with a single space.

Here we replace “weird” by “easy to read”.

re.split() : splitting a string using RE

```
>>> import re
>>> text = 'This\t \tis\fa::very\v,weird\n\tstring.'
>>> text
'This\t \tis\x0ca::very\x0b,weird\n\tstring.'
>>> print text
This             is♩a::very♪,weird
        string.
>>> re.split( '[\s:,.]+', text )
['This', 'is', 'a', 'very', 'weird', 'string', '']
>>>
```

[\s:,.]+ matches one or more of whitespace, colon, comma and dot.

In this case, we split the string text using this expression as field separator.

Further references on regular expressions

- http://docs.python.org/howto/regex.html
- http://docs.python.org/library/re.html

Revisiting our problem

- We are doing a Vehicle-to-Infrastructure (V-2-I) communication research. Suppose that our first step is to monitor and record WiFi access points, their channels and signal strengths, along the road while we drive. Can we build a simple automated tool for this purpose ?

One problem of RE – we can easily overdo it..

```
>>> import os, re
>>> txt = os.popen( 'iwlist scan' ).read()
>>> entries = re.split( '[\s.:=]+', txt )
>>> entries
['ra0', 'Scan', 'completed', 'Cell', '01', '-', 'Address', '00', '14', '6C', 'D9', '7A', '
46', 'ESSID', '"Interlab', 'AP-1B"', 'Mode', 'Managed', 'Channel', '6', 'Quality', '70/100
', 'Signal', 'level', '-62', 'dBm', 'Noise', 'level', '-81', 'dBm', 'Encryption', 'key', '
on', 'Bit', 'Rates', '36', 'Mb/s', 'IE', 'WPA', 'Version', '1', 'Group', 'Cipher', 'TKIP',
'Pairwise', 'Ciphers', '(1)', 'TKIP', 'Authentication', 'Suites', '(1)', 'PSK', 'Cell', '0
2', '-', 'Address', '00', '14', '6C', 'D9', '78', 'A8', 'ESSID', '"intERLab', 'AP-1C"', 'M
ode', 'Managed', 'Channel', '1', 'Quality', '65/100', 'Signal', 'level', '-64', 'dBm', 'No
ise', 'level', '-81', 'dBm', 'Encryption', 'key', 'on', 'Bit', 'Rates', '36', 'Mb/s', 'Cel
l', '03', '-', 'Address', '00', '25', '9C', 'DC', '24', 'E6', 'ESSID', '"Cam_AIT_A06"', 'M
ode', 'Managed', 'Channel', '3', 'Quality', '34/100', 'Signal', 'level', '-76', 'dBm', 'No
ise', 'level', '-81', 'dBm', 'Encryption', 'key', 'on', 'Bit', 'Rates', '11', 'Mb/s', 'Cel
l', '04', '-', 'Address', '00', '11', '33', '55', '77', '99', 'ESSID', '"TEST1234"', 'Mode
', 'Ad-Hoc', 'Channel', '11', 'Quality', '91/100', 'Signal', 'level', '-54', 'dBm', 'Noise
', 'level', '-81', 'dBm', 'Encryption', 'key', 'on', 'Bit', 'Rates', '18', 'Mb/s', 'Cell',
'05', '-', 'Address', '00', '25', '9C', 'DC', '02', '9C', 'ESSID', '"Cam_AIT_A5"', 'Mode',
'Managed', 'Channel', '13', 'Quality', '15/100', 'Signal', 'level', '-84', 'dBm', 'Noise',
'level', '-81', 'dBm', 'Encryption', 'key', 'on', 'Bit', 'Rates', '11', 'Mb/s', '']
```

This was supposed to be a string:
"Interlab AP-1B" but it got split !!

ApScanV1.py

```
# ApScanV1.py : access point scanning

import os

txt = os.popen( 'iwlist scan' ).read()
lines = txt.split( '\n' )
xlines = map( lambda i : i.strip(), lines )
print xlines
```

Output of ApScanV1.py

```
IDLE 2.6.4
>>> ========================== RESTART ==========================
>>>
['ra0       Scan completed :', 'Cell 01 - Address: 00:14:6C:D9:7A:46', 'ESSID:"I
nterlab AP-1B"', 'Mode:Managed', 'Channel:6', 'Quality:65/100  Signal level:-64
dBm  Noise level:-81 dBm', 'Encryption key:on', 'Bit Rates:36 Mb/s', 'IE: WPA Ve
rsion 1', 'Group Cipher : TKIP', 'Pairwise Ciphers (1) : TKIP', 'Authentication
Suites (1) : PSK', 'Cell 02 - Address: 00:14:6C:D9:78:A8', 'ESSID:"intERLab AP-1
C"', 'Mode:Managed', 'Channel:1', 'Quality:70/100  Signal level:-62 dBm  Noise l
evel:-81 dBm', 'Encryption key:on', 'Bit Rates:36 Mb/s', 'Cell 03 - Address: 00:
25:9C:DC:24:E6', 'ESSID:"Cam_AIT_A06"', 'Mode:Managed', 'Channel:3', 'Quality:15
/100  Signal level:-84 dBm  Noise level:-81 dBm', 'Encryption key:on', 'Bit Rate
s:11 Mb/s', 'Cell 04 - Address: 00:11:33:55:77:99', 'ESSID:"TEST1234"', 'Mode:Ad
-Hoc', 'Channel:11', 'Quality:76/100  Signal level:-60 dBm  Noise level:-81 dBm'
, 'Encryption key:on', 'Bit Rates:18 Mb/s', 'Cell 05 - Address: 00:25:9C:DC:02:9
C', 'ESSID:"Cam_AIT_A5"', 'Mode:Managed', 'Channel:13', 'Quality:15/100  Signal
level:-84 dBm  Noise level:-81 dBm', 'Encryption key:on', 'Bit Rates:11 Mb/s', '
', '']
```

Well, it seems good, but we'll need some re-formatting effort

ApScanV2.py (Part 1 of 2)

```
# ApScanV2.py : access point scanning
WLAN = 'ra0'
import os, re, time

def ParseLines( lines ):
    cells = dict()
    c = None
    for l in lines:
        if re.match( '^Cell', l ) is not None:
            c = l.split(' ')[4]
            cells[c] = dict()
            continue
        if c is not None and re.match( '^ESSID', l ) is not None:
            essid = l.split(':')[1]
            cells[c]['ESSID'] =  essid
            continue
        if c is not None and re.match( '^Quality', l ) is not None:
            qsn = re.sub( 'Quality[:=]','Q=', l )
            qsn = re.sub( 'Signal [Ll]evel[:=]', 'S=', qsn )
            qsn = re.sub( 'Noise level[:=]', 'N=', qsn )
            qsn = re.sub( '\s+', ' ', qsn )
            cells[c]['QSN'] = qsn
            continue
        if c is not None and re.match( '^Channel', l ) is not None:
            ch = re.sub( 'Channel[:=]', '', l )
            cells[c]['CH'] = ch
            continue
    return cells
```

Cell ID

ESSID

Quality
Signal Level
Noise

Channel

ApScanV2.py (Part 2 of 2)

```
def DoScan():
    txt = os.popen( 'iwlist ' + WLAN + ' scan' ).read()
    lines = txt.split( '\n' )
    xlines = map( lambda i : i.strip(), lines )
    cells = ParseLines( xlines )
    fmt = '%-17s | %-2s | %-30s | %s'

    print '\n' + time.ctime()
    print fmt % ( 'Cell', 'CH', '(Q)uality & (S)ignal & (N)oise', 'ESSID' )

    for c in cells.keys():
        print fmt%(c,cells[c].get('CH'),cells[c].get('QSN'),cells[c].get('ESSID'))

if __name__ == '__main__':
    while True:
        DoScan()
        time.sleep( 3 )
```

Output of ApScanV2.py

```
Fri Sep 17 19:47:56 2010
Cell               | CH | (Q)uality & (S)ignal & (N)oise | ESSID
00:14:6C:D9:7A:46 | 6  | Q=60/100 S=-66 dBm N=-81 dBm   | "Interlab AP-1B"
00:25:9C:DC:24:E6 | 3  | Q=24/100 S=-80 dBm N=-81 dBm   | "Cam_AIT_A06"
00:11:33:55:77:99 | 11 | Q=81/100 S=-58 dBm N=-81 dBm   | "TEST1234"
00:14:6C:D9:78:A8 | 1  | Q=55/100 S=-68 dBm N=-81 dBm   | "intERLab AP-1C"
00:25:9C:DC:02:9C | 13 | Q=15/100 S=-84 dBm N=-81 dBm   | "Cam_AIT_A5"

Fri Sep 17 19:47:59 2010
Cell               | CH | (Q)uality & (S)ignal & (N)oise | ESSID
00:14:6C:D9:7A:46 | 6  | Q=60/100 S=-66 dBm N=-81 dBm   | "Interlab AP-1B"
00:25:9C:DC:24:E6 | 3  | Q=24/100 S=-80 dBm N=-81 dBm   | "Cam_AIT_A06"
00:11:33:55:77:99 | 11 | Q=81/100 S=-58 dBm N=-81 dBm   | "TEST1234"
00:14:6C:D9:78:A8 | 1  | Q=55/100 S=-68 dBm N=-81 dBm   | "intERLab AP-1C"
00:25:9C:DC:02:9C | 13 | Q=15/100 S=-84 dBm N=-81 dBm   | "Cam_AIT_A5"
```

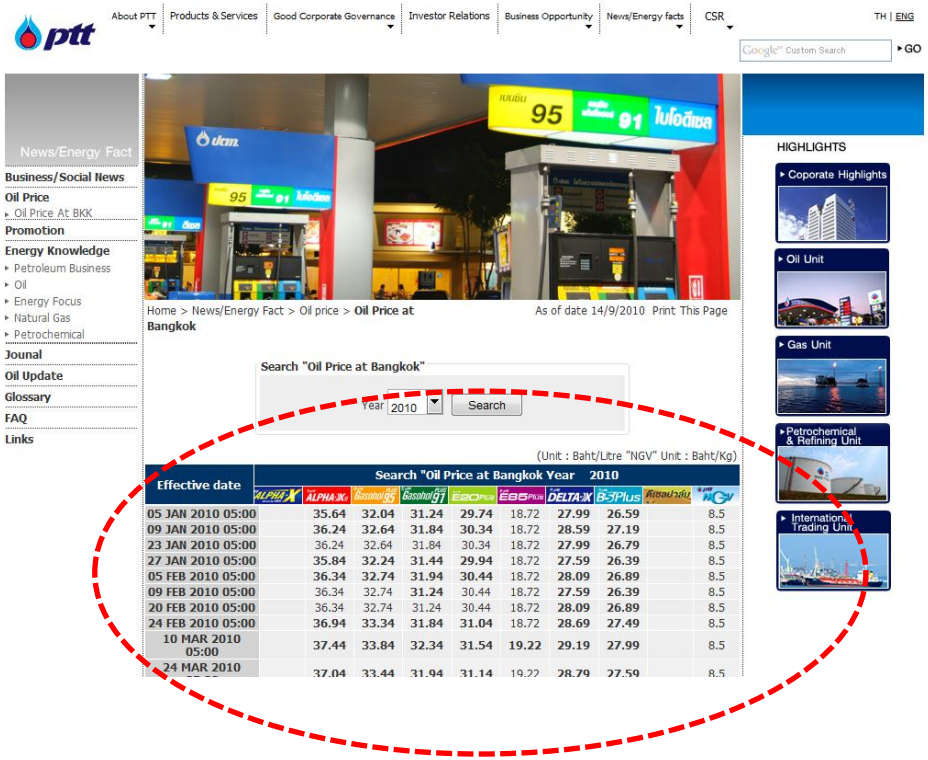
You can move your laptop around and observe signal quality.

Exercise

- Extend ApScanV2.py so that it can display
  - Network mode (“managed” vs. “ad-hoc”)
  - Whether the network is encrypted

Exercise

- Recall the HTML parser example from last time. We would like to make it smarter.




```
Table data found: ♩

Table data found: 22 AUG 2010 05:00
Table data found: 35.04
Table data found: 31.24
Table data found: 29.74
Table data found: 28.94
Table data found: 18.82
Table data found: 28.19
Table data found: 26.99
Table data found: 8.5
Table data found: ♩

Table data found: ♩

Table data found: 26 AUG 2010 05:00
Table data found: 34.44
Table data found: 30.64
Table data found: 29.14
Table data found: 28.34
Table data found: 18.42
Table data found: 27.79
Table data found: 26.59
Table data found: 8.5
Table data found: ♩
```

Say, we would like to have the lists of oil prices, kept and indexed in a dictionary by their dates.

2. Bluetooth Programming

Bluetooth basics

- Personal Area Network (PAN) 2.402-2.480 GHz ISM
- There are 3 classes of Bluetooth devices:
  - Class 1: Max power 100mW (range ~100 meters)
  - Class 2: Max power 2.5mW (range ~10 meters)
  - Class 3: Max power 1mW (range ~1 meter)
- There are several versions of Bluetooth
  - Bluetooth v1.2 (~1Mbps)
  - Bluetooth v2.0+EDR (~3 Mbps)
  - Bluetooth v2.1+EDR (e.g. “secure simple pairing”, SSP)
  - Bluetooth v3.0+HS (~24Mbps)

Bluetooth modules for Python

- Bluez (Pybluez)
  - Homepage: http://www.bluez.org/
  - Ubuntu: sudo apt-get install bluez python-bluez
  - Tutorial:
    - http://people.csail.mit.edu/albert/bluez-intro/
- Lightblue
  - Homepage: http://lightblue.sourceforge.net/
  - Ubuntu: sudo apt-get install python-lightblue
  - Tutorial: See the lightblue's homepage

lightblue.finddevices() : search for nearby BT devices

```
>>> import lightblue
>>> devices = lightblue.finddevices()
>>> devices
[('00:1B:C1:02:F0:8B', 'HOLUX_M-1000', 7936), ('00:21:FC:FC:F5:91', 'Good Old No
kia', 5898756)]
>>> for d in devices:
        print d


('00:1B:C1:02:F0:8B', 'HOLUX_M-1000', 7936)
('00:21:FC:FC:F5:91', 'Good Old Nokia', 5898756)
>>>
```

Returns a list of tuples

lightblue.findservices() : search for nearby BT services

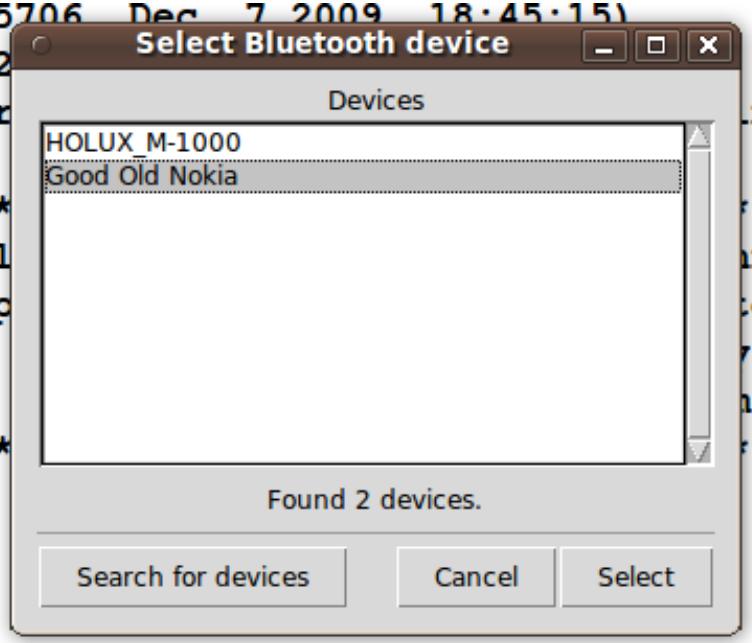
```
>>> import lightblue
>>> services = lightblue.findservices()
>>> services
[('00:1B:C1:02:F0:8B', 1, 'SPP slave'), ('00:21:FC:FC:F5:91', 1, 'Dial-up networking'), ('
00:21:FC:FC:F5:91', 15, 'Nokia PC Suite'), ('00:21:FC:FC:F5:91', 3, 'COM 1'), ('00:21:FC:F
C:F5:91', 13, 'Voice Gateway'), ('00:21:FC:FC:F5:91', 12, 'Audio Gateway'), ('00:21:FC:FC:
F5:91', 21505, None), ('00:21:FC:FC:F5:91', 14, None), ('00:21:FC:FC:F5:91', 22529, None),
('00:21:FC:FC:F5:91', 15, 'Network Access Point Service'), ('00:21:FC:FC:F5:91', 24, None)
, ('00:21:FC:FC:F5:91', 9, 'OBEX Object Push'), ('00:21:FC:FC:F5:91', 10, 'OBEX File Trans
fer'), ('00:21:FC:FC:F5:91', 7, 'Nokia SyncML Server'), ('00:21:FC:FC:F5:91', 11, 'SyncML
Client'), ('00:21:FC:FC:F5:91', 25, 'Music-Player'), ('00:21:FC:FC:F5:91', 23, 'Media Play
er'), ('00:21:FC:FC:F5:91', 23, 'Media Player'), ('00:21:FC:FC:F5:91', 4, 'SIM ACCESS')]
>>> for s in services:
        print s


('00:1B:C1:02:F0:8B', 1, 'SPP slave')
('00:21:FC:FC:F5:91', 1, 'Dial-up networking')
('00:21:FC:FC:F5:91', 15, 'Nokia PC Suite')
('00:21:FC:FC:F5:91', 3, 'COM 1')
('00:21:FC:FC:F5:91', 13, 'Voice Gateway')
('00:21:FC:FC:F5:91', 12, 'Audio Gateway')
```

The SPP slave service is available on Holux M-1000 GPS

These are the services available on my Nokia phone

lightblue.selectdevice() : device selection GUI

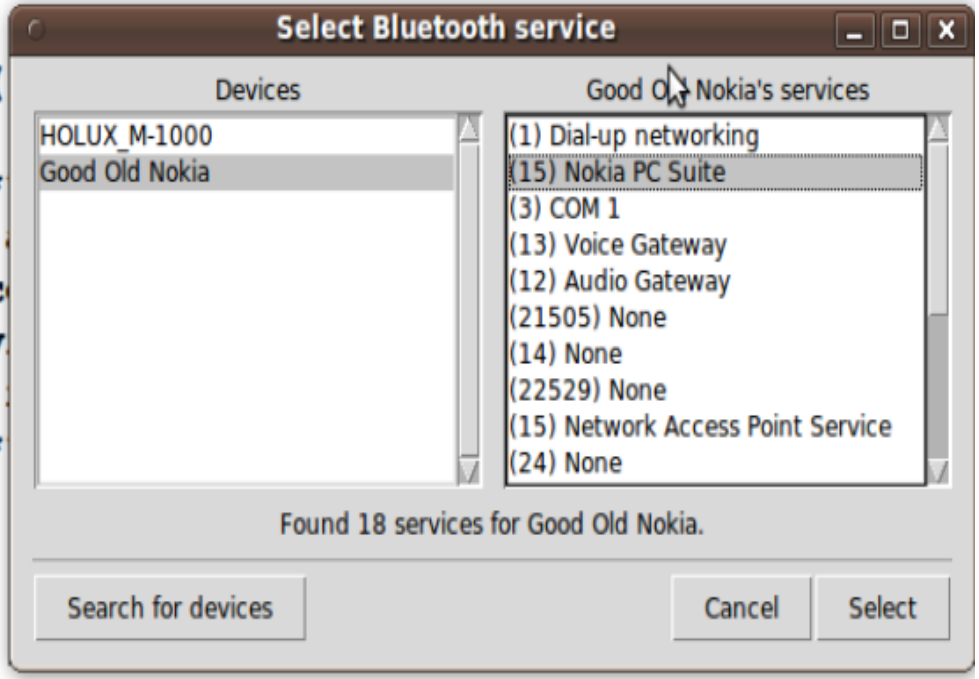
```
Python 2.6.4 (r264:75706, Dec  7 2009, 18:45:15)
[GCC 4.4.1] on linux2
Type "copyright", "cr                                  nformation.

    ****************                                   ***************
    Personal firewall                                  nection IDLE
    makes to its subp                                  ternal loopback
    interface.  This                                   external
    interface and no                                   the Internet.
    ****************                                   ***************

IDLE 2.6.4      ====
>>> import lightblue
>>> dir( lightblue )
['BluetoothError', 'L2CAP', 'OBEX', 'RFCOMM', '__builtins__', '__doc__', '__file
__', '__name__', '__package__', '__path__', '_docstrings', '_lightblue', '_light
bluecommon', '_obex', '_obexcommon', 'advertise', 'finddevicename', 'finddevices
', 'findservices', 'gethostaddr', 'gethostclass', 'obex', 'selectdevice', 'selec
tservice', 'socket', 'splitclass', 'stopadvertise']
>>> lightblue.selectdevice()
```



Note: this program may freeze IDLE. You should run it from linux command line..

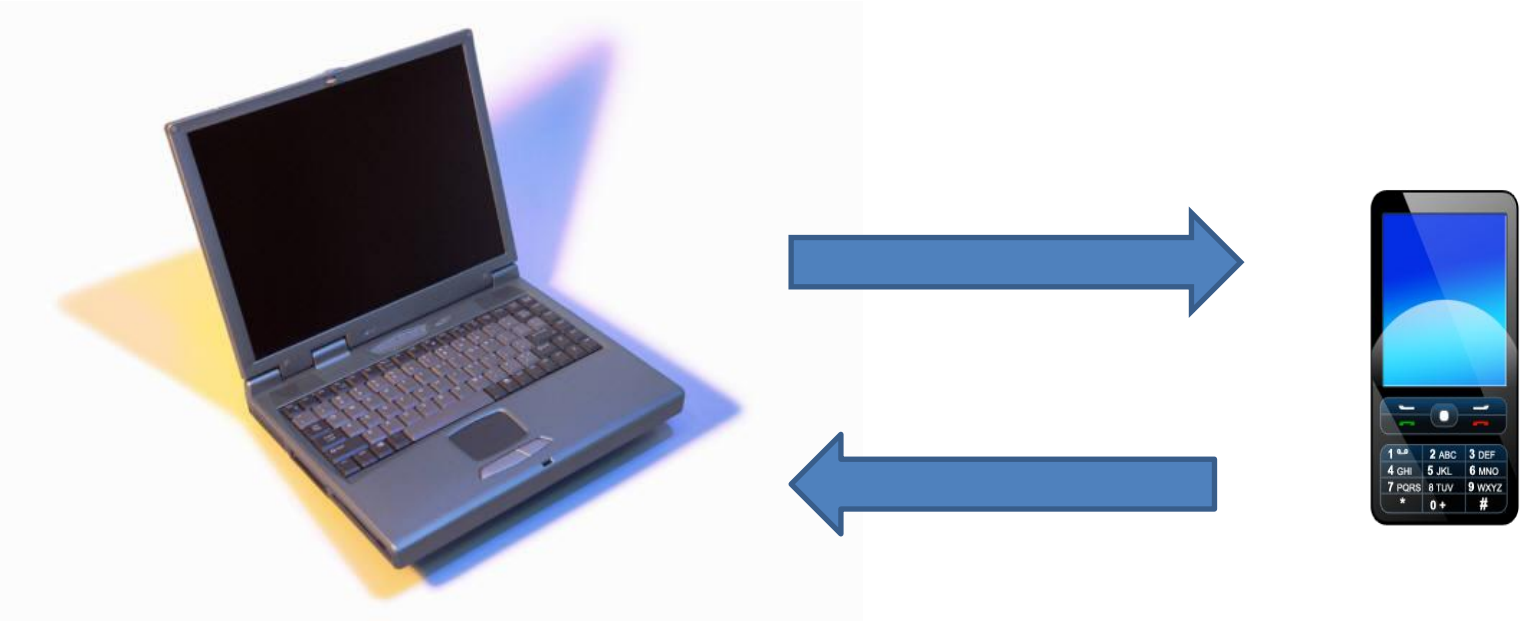
lightblue.selectservice() : service selection GUI

```
Python 2.6.4 (r264:75706, Dec  7 2009, 18:45:15)
[GCC 4.4.1] on linux2
Type "copyright", "credits" or "license(

    **********************************
    Personal firewall software may warn
    makes to its subprocess using this c
    interface.  This connection is not v
    interface and no data is sent to or
    **********************************

IDLE 2.6.4      ==== No Subprocess ====
>>> import lightblue
>>> choice = lightblue.selectservice()
```



Note: this program may freeze IDLE. You should run it from linux command line..

Sending objects via Object Exchange (OBEX)

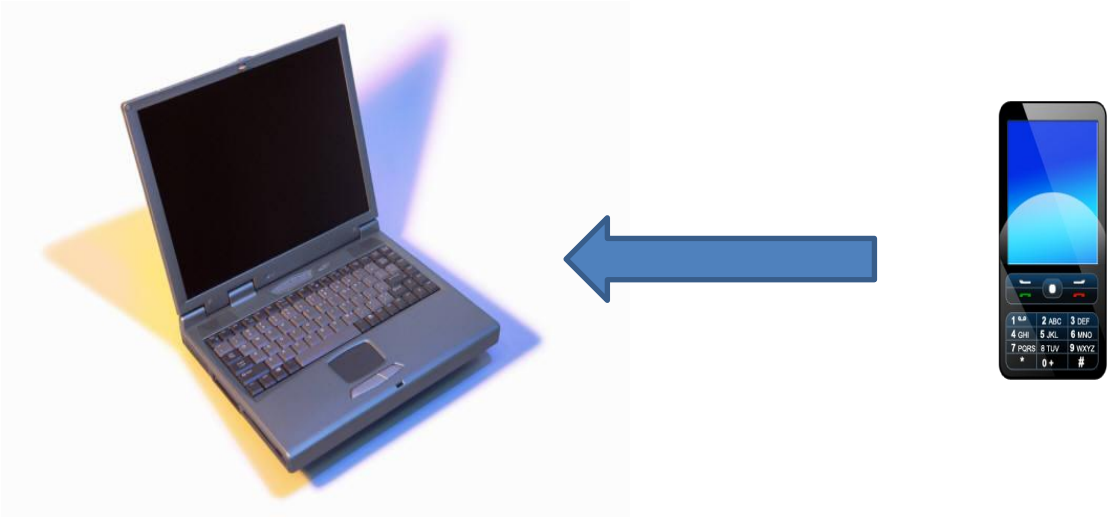


Object Exchange (OBEX) Push Example

```
# LightblueSend.py: Apinun Tunpan, intERLab, AIT

import lightblue

print "Finding Devices.."
device = lightblue.selectdevice()

if device is not None:
  print "Device selected was: ", device
else:
  print "No device was selected. Exiting.."
  exit()

obex_push = None
print "Finding OBEX Object Push service..."

for s in lightblue.findservices( addr = device[0] ):
  print s
  if s[2] == 'OBEX Object Push':
    obex_push = s[1]

if obex_push is not None:
  client = lightblue.obex.OBEXClient( str(device[0]), obex_push )
  print client.connect()
  print client.put( {"name":"Myfile.jpg"}, file("Myfile.jpg","rb" ) )
  client.disconnect()
else:
  print "OBEX Object Push service was not listed on the device. Exiting.
```

Exercise

- Implement a program to receive a file from mobile phone via OBEX push



3. GPS device interface

Bluetooth GPS

Ideas for interfacing with GPS

- Connect (e.g. via bluetooth) to the GPS
- Read data lines
- Get to the right line (e.g. $GPGGA)
- Split & Decode the line
  - To get UTC, Lat, Lon, Height, Validity…
- Disconnect

Connecting with a Bluetooth device

- This time we need to transfer stream of data
- There are two protocols:
  - RFCOMM : reliable, stream-based
  - L2CAP : best-effort, datagram

More information at

http://people.csail.mit.edu/albert/bluez-intro/x95.html

The bluez (bluetooth) module

```
>>> import bluetooth
>>> services = bluetooth.find_service()
>>> for  s in services:
        print s
```

GPSBlue1.py : basic GPS reading

```
import sys, time, bluetooth
deviceAddress = '00:1B:C1:02:F0:8B' # Change this line to your device's MAC

def main():

  # Find RFCOMM port
  services = bluetooth.find_service(address=deviceAddress)
  port = None
  for svc in services:
    if svc["name"] == "SPP slave":
      port = svc["port"]
      break

  if port is None:
    print "Could not find RFCOMM port for SPP slave service."
    sys.exit(4)

  # Create bluetooth socket
  bluesock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
  bluesock.connect((deviceAddress, port))

  while True:
    gpsdata = bluesock.recv(2048)
    gpslines = gpsdata.splitlines()
    print '**************************   Printing raw GPS data: **************'
    for l in gpslines:
      print l
    time.sleep( 1 )

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

Finding a right port to connect to

Making an RFCOMM connection to the GPS module.

Reading the data

Sample output from GPSBlue1.py

```
**************************   Printing raw GPS data: **********************************
$GPGGA,042230.000,1404.6657,N,10036.7734,E,1,10,0.88,34.5,M,-28.0,M,,*7F
$GPGSA,A,3,23,17,03,19,07,20,13,28,11,08,,,1.51,0.88,1.23*0E
$GPGSV,4,1,14,13,56,236,33,19,51,024,28,11,50,169,34,07,44,343,36*75
$GPGSV,4,2,14,23,41,192,38,24,30,078,,03,20,035,35,28,13,290,25*70
$GPGSV,4,3,14,08,12,326,18,17,11,226,23,20,08,171,24,06,07,039,*7F
$GPGSV,4,4,14,32,02,154,,45,,,*4E
$GPRMC,042230.000,A,1404.6657,N,10036.7734,E,0.00,268.33,210910,,,A*6E
$GPZDA,042230.000,21,09,2010,,*58
**************************   Printing raw GPS data: **********************************
$GPGGA,042231.000,1404.6657,N,10036.7734,E,1,10,0.88,34.5,M,-28.0,M,,*7E
$GPGSA,A,3,23,17,03,19,07,20,13,28,11,08,,,1.51,0.88,1.23*0E
$GPRMC,042231.000,A,1404.6657,N,10036.7734,E,0.01,268.33,210910,,,A*6E
$GPZDA,042231.000,21,09,2010,,*59
**************************   Printing raw GPS data: **********************************
$GPGGA,042232.000,1404.6657,N,10036.7734,E,1,10,0.88,34.5,M,-28.0,M,,*7D
$GPGSA,A,3,23,17,03,19,07,20,13,28,11,08,,,1.51,0.88,1.23*0E
$GPRMC,042232.000,A,1404.6657,N,10036.7734,E,0.01,268.33,210910,,,A*6D
$GPZDA,042232.000,21,09,2010,,*5A
**************************   Printing raw GPS data: **********************************
$GPGGA,042233.000,1404.6657,N,10036.7733,E,1,10,0.88,34.5,M,-28.0,M,,*7B
$GPGSA,A,3,23,17,03,19,07,20,13,28,11,08,,,1.51,0.88,1.23*0E
$GPRMC,042233.000,A,1404.6657,N,10036.7733,E,0.01,268.33,210910,,,A*6B
$GPZDA,042233.000,21,09,2010,,*5B
```

Some of NMEA GPS data types

- $GPGAA – Fix information (2D or 3D)
- $GPGSA – Satellite status
- $GPGSV – Satellites in view
- $GPRMC – Recommended minimum
- $GPZDA -- UTC date and time

For more information, search Google™ , or visit:
http://www.gpsinformation.org/dale/nmea.htm

Understanding the lat/lon formats

- Common formats
  - Decimal degrees (e.g. 100.12345)
  - Degrees, decimal minutes (e.g. 100 7.407’)
  - Degrees, minutes, decimal seconds (e.g. 100 7’ 24.42”)

With a straight forward conversion:

Decimal degrees

= Degrees + minutes/60 + seconds/3600

GPSBlue2.py (part 1 of 2)

```
# GPSBlue.py:  reading and decoding from Holux M-1000 BT GPS
# IntERLab, September, 2010. Based on the code by Dr.Mahtab and Dr.Apinun.

import sys, time, bluetooth
deviceAddress = '00:1B:C1:02:F0:8B'  # Change this line to your device's MAC

def DecodeGPS(line):
  'From a $GPGGAA sentence, decode lat, lon, alt and utc'
  fields = line.split(',')
  if fields[0] != '$GPGGA' or len(fields) < 6 or int(fields[6]) < 1:
    return None

  else:
    utc      = fields[1]

    lat_deg = int(float(fields[2]))/100
    lat_min = (float(fields[2]) - lat_deg*100)/60.0
    lat = lat_deg + lat_min

    lon_deg = int(float(fields[4]))/100
    lon_min = (float(fields[4]) - lon_deg*100)/60.0
    lon = lon_deg + lon_min

    if fields[3] != 'N': lat = lat * -1    # Adjust for the sign

    if fields[5] != 'E': lon = lon * -1    # Adjust for the sign

    altitude = fields[9]

    quality = fields[6]

    return (utc,lat,lon,altitude,quality)
```

Transforming the format of lat, lon from DDMM.MMMM to DD.DDDDD

Altitude

Quality of the GPS fix.

GPSBlue2.py (part 2 of 2)

```
def main():

  # Find RFCOMM port
  services = bluetooth.find_service(address=deviceAddress)
  port = None
  for svc in services:
    if svc["name"] == "SPP slave":
      port = svc["port"]
      break

  if port is None:
    print "Could not find RFCOMM port for SPP slave service."
    sys.exit(4)

  # Create bluetooth socket
  bluesock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
  bluesock.connect((deviceAddress, port))

  while True:
    gpsdata = bluesock.recv(2048)
    gpslines = gpsdata.splitlines()
    for l in gpslines:
      data = DecodeGPS(l)
      if data is not None:
        mesg="utc= %s lat= %f lon= %f alt= %s q= %s" % data
        print mesg
    time.sleep( 1 )

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

Sample Output of GPSBlue2.py

```
utc= 080557.000 lat= 14.077713 lon= 100.612975 alt= 13.5 q= 1
utc= 080558.000 lat= 14.077715 lon= 100.612975 alt= 13.4 q= 1
utc= 080559.000 lat= 14.077717 lon= 100.612970 alt= 13.2 q= 1
utc= 080600.000 lat= 14.077720 lon= 100.612972 alt= 13.2 q= 1
utc= 080601.000 lat= 14.077725 lon= 100.612963 alt= 13.2 q= 1
utc= 080602.000 lat= 14.077725 lon= 100.612965 alt= 13.2 q= 1
utc= 080603.000 lat= 14.077727 lon= 100.612968 alt= 13.2 q= 1
utc= 080604.000 lat= 14.077728 lon= 100.612968 alt= 13.2 q= 1
utc= 080605.000 lat= 14.077732 lon= 100.612958 alt= 13.1 q= 1
utc= 080606.000 lat= 14.077730 lon= 100.612947 alt= 13.0 q= 1
utc= 080607.000 lat= 14.077730 lon= 100.612940 alt= 12.9 q= 1
utc= 080608.000 lat= 14.077730 lon= 100.612935 alt= 12.9 q= 1
utc= 080609.000 lat= 14.077732 lon= 100.612923 alt= 12.8 q= 1
utc= 080610.000 lat= 14.077732 lon= 100.612918 alt= 12.7 q= 1
utc= 080611.000 lat= 14.077733 lon= 100.612910 alt= 12.7 q= 1
utc= 080612.000 lat= 14.077735 lon= 100.612903 alt= 12.4 q= 1
utc= 080613.000 lat= 14.077735 lon= 100.612905 alt= 12.4 q= 1
utc= 080614.000 lat= 14.077735 lon= 100.612907 alt= 12.3 q= 1
utc= 080615.000 lat= 14.077733 lon= 100.612908 alt= 12.3 q= 1
utc= 080616.000 lat= 14.077732 lon= 100.612910 alt= 12.2 q= 1
utc= 080617.000 lat= 14.077730 lon= 100.612910 alt= 12.2 q= 1
utc= 080618.000 lat= 14.077728 lon= 100.612912 alt= 12.1 q= 1
utc= 080619.000 lat= 14.077728 lon= 100.612913 alt= 12.1 q= 1
```

Exercises

- Extend GPSBlue.py so that it displays the current location of your PC + GPS on Google Maps™

- Add GPS coordinate reading to the Access Point scan (ApScanV2.py) so that it displays the lat/lon of the reading.

4. Further self studies

What makes Python full of useful modules and libraries?

- The answer: language wrappers and bindings.

- http://www.swig.org/exec.html
- http://code.google.com/p/pybindgen/

For your research

Python may already have the modules that make your life easier:

- Graph Theory
  - http://networkx.lanl.gov/
  - http://code.google.com/p/python-graph/
- Scientific Computing & Optimization
  - http://www.scipy.org/
  - http://mdp-toolkit.sourceforge.net/
  - http://cvxmod.net/

For your research

- GIS
  - http://trac.gispython.org/lab
  - http://trac.gispython.org/lab/wiki/OwsLib
  - http://trac.osgeo.org/gdal/wiki/GdalOgrInPython
- Social Network
  - http://code.google.com/p/python-twitter/
  - http://code.google.com/p/pyfacebook/

Thank you for your attention

Hope you have fun and learn a lot from this class.