Introduction to Network Programming Part I

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

Today's Outline

1. Basic programming: Introduction to Python
2. Socket Programming
3. WWW information processing

1. Introduction to Python

Programming should be simple.

Preliminaries

- The Official Python Homepage
  - http://www.python.org
  - Free for all
  - Usually, it is already included with most Linux distributions
- Enthought Python Distribution (EPD)
  - http://www.enthought.com/products/epd.php
  - “Kitchen and sink included”
  - Free for academic (degree-granting institutes), but must pay for other uses

Some good references

- Python documentation, tutorial, and howto's
  - http://www.python.org/doc/
  - http://docs.python.org/tutorial/
  - http://docs.python.org/howto/
- "Tutorial on Network Programming with Python", N. Matloff, UC Davis
  - http://heather.cs.ucdavis.edu/~matloff/Python/PyNet.pdf

Some good references

- Book: there are many
  - search Google and/or Amazon

- Examples of codes
  - http://python.net/crew/wesc/cpp/

“Hello world” compared

- HelloWorld.py

```
print “Hello World”
```

- HelloWorld.c

```
#include <stdio.h>
void main(void)
{
  printf( “Hello World\n” );
}
```

“Hello world” compared

- HelloWorld.py

```
print “Hello World”
```

- HelloWorld.java

```
class HelloWorld
{
  public static void main(String args[])
      {
        System.out.println("Hello World!");
      }
}
```

Using Python's IDLE (Integrated Development Environment)

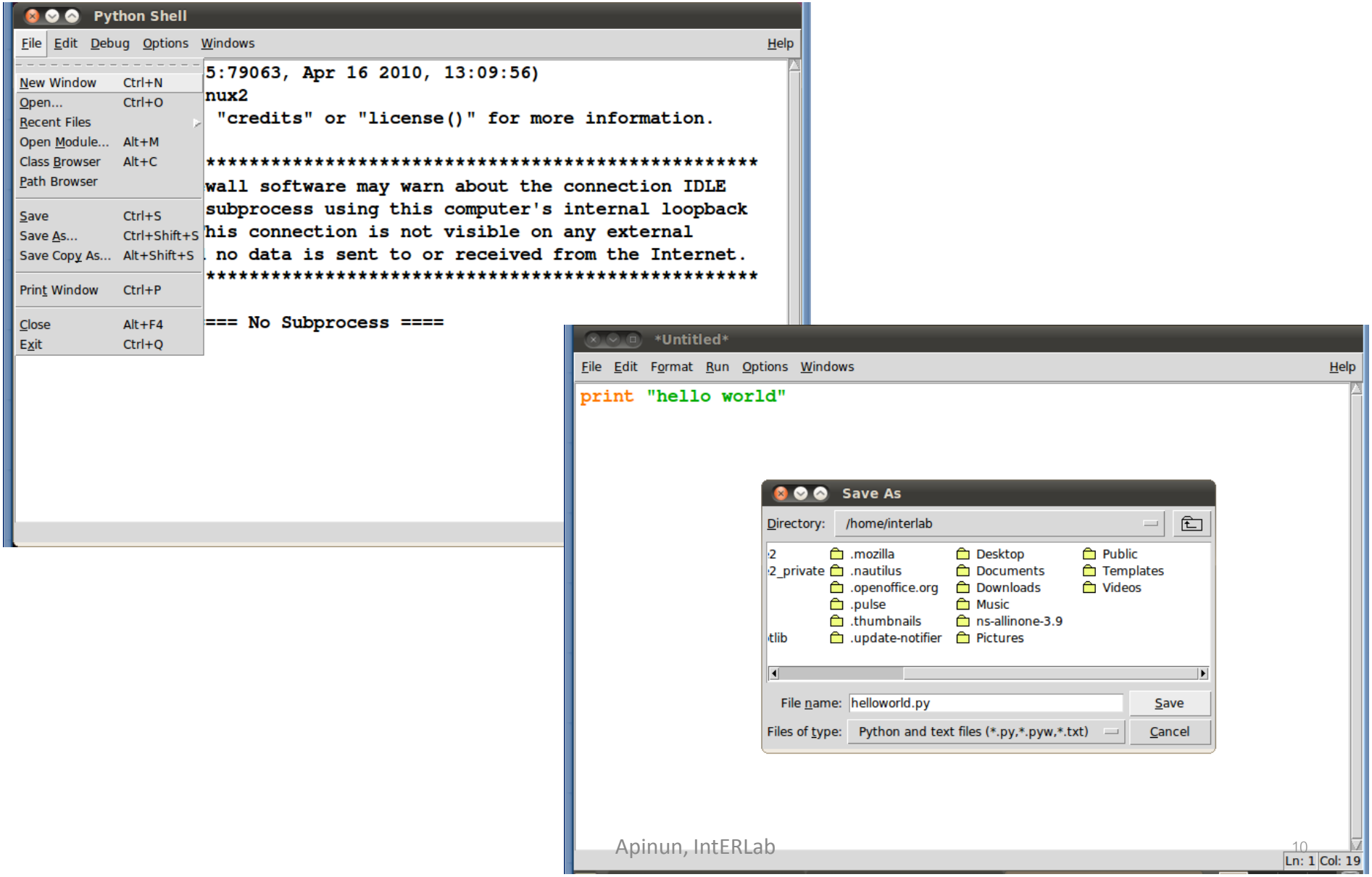
Python Shell

File Edit Debug Options Windows Help

```
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)
[GCC 4.4.3] on linux2
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.5      ==== No Subprocess ====
>>> print "Hello World"
Hello World
>>> print 'Hello World again'
Hello World again
>>> 1 + 2 + 4 + 8
15
>>> |
```

Ln: 19 Col: 4

Using Python's IDLE



C/C++ Like String Formatting

Python Shell

File Edit Debug Options Windows Help

```
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)
[GCC 4.4.3] on linux2
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.5      ==== No Subprocess ====
>>> print "%d %s 0x%02x" % (1, "two", 16)
1 two 0x10
>>> |
```

The ‘for’ Loop

Python Shell

File Edit Debug Options Windows Help

```
IDLE 2.6.5      ==== No Subprocess ====
>>> for i in range(5):
        print i


0
1
2
3
4
>>> range(5)
[0, 1, 2, 3, 4]
>>> for i in ['One', 'Two', 'Three']:
        print i


One
Two
Three
>>> |
```

The dir() function allows you to see what're available
The 'import' keyword allows you to use more modules

Python Shell

File Edit Debug Options Windows Help

```
IDLE 2.6.5      ==== No Subprocess ====
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'ma
in']
>>> import math
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'ma
in', 'math']
>>> dir( math )
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh
', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degre
es', 'e', 'exp', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum'
, 'hypot', 'isinf', 'isnan', 'ldexp', 'log', 'log10', 'log1p', 'modf',
'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>> math.sin( math.pi / 2  )
1.0
>>> math.factorial( 5 )
120
>>> math.e
2.7182818284590451
```

Two different ways to import

- Approach 1

  import *modulename*

- Approach 2

  (2.1) from *modulename* import *item1, item2,…*

  (2.2) from *modulename* import *

Import approach 1 : OO style

```
IDLE 2.6.5      ==== No Subprocess ====
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main']
>>> import math
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main', 'math
']
>>> dir( math )
['__doc__', '__name__', '__package__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'exp', 'fa
bs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'hypot', 'isinf', 'isnan', '
ldexp', 'log', 'log10', 'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
>>> math.pi
3.1415926535897931
>>> math.cos( math.pi / 4 )
0.70710678118654757
>>> math.e
2.7182818284590451
>>> math.log( math.e/2  )
0.30685281944005466
>>>
```

This is a clean object-oriented approach. When you use any object, you refer to its parent.

Import approach 2.1 : selective import

```
IDLE 2.6.5      ==== No Subprocess ====
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main']
>>> from math import pi, e, cos, log
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'cos', 'e', '
log', 'main', 'pi']
>>> dir( math )
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    dir( math )
NameError: name 'math' is not defined
>>> cos( pi / 4 )
0.70710678118654757
>>> log( e/2 )
0.30685281944005466
>>>
>>>
>>>
>>>
```

There is no parent object in this case. It may be easier to write a program, if you know the scope of the objects and do not accidentally overwrite the imported objects !!!

Import approach 2.2 : import everything

```
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main']
>>> from math import *
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'acos', 'acos
h', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh'
, 'degrees', 'e', 'exp', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'hypot', 'isinf', 'isnan', 'ldexp', 'log', 'log10', 'log1p', 'main', 'modf', 'pi
', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
>>> pi
3.141592653589793
>>> cos( pi / 4 )
0.70710678118654757
>>> sin( pi / 4 )
0.70710678118654746
>>> sin( pi / 2 )
1.0
>>> tan( pi / 4 )
0.99999999999999989
>>> e
2.7182818284590451
>>> log( e / 2 )
0.30685281944005466
>>> log10( e / 2 )
0.13326448623927062
```

Be careful !! You get everything from the module at the top level. And if you import many modules to the same level, chances of having naming conflicts can be high.

As you might guess, the command dir(__builtins__) lists all built-in functions

```
>>> dir( __builtins__ )
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BufferError', 'BytesWarning',
 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FloatingPoint
Error', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError'
, 'IndexError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'No
tImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'Referen
ceError', 'RuntimeError', 'RuntimeWarning', 'StandardError', 'StopIteration', 'SyntaxError', 'SyntaxWar
ning', 'SystemError', 'SystemExit', 'TabError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecod
eError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning'
, 'ValueError', 'Warning', 'ZeroDivisionError', '_', '__debug__', '__doc__', '__import__', '__name__',
'__package__', 'abs', 'all', 'any', 'apply', 'basestring', 'bin', 'bool', 'buffer', 'bytearray', 'bytes
', 'callable', 'chr', 'classmethod', 'cmp', 'coerce', 'compile', 'complex', 'copyright', 'credits', 'de
lattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'execfile', 'exit', 'file', 'filter', 'float', 'f
ormat', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'int
ern', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'long', 'map', 'max', 'mi
n', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'raw_input', '
reduce', 'reload', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 's
tr', 'sum', 'super', 'tuple', 'type', 'unichr', 'unicode', 'vars', 'xrange', 'zip']
>>> |
```

help( ) can give you information …

Python Shell

File Edit Debug Options Windows Help

```
IDLE 2.6.5      ==== No Subprocess ====
>>> import math
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main'
, 'math']
>>> help( math )
Help on built-in module math:

NAME
    math

FILE
    (built-in)

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured in radians) of x.

    acosh(...)
```

Some of Python’s basic data types

- Integer
- Long integer
- Floating point
- String
- Complex
- Boolean
- List
- Dictionary
- Set

Integer, floating point and string data types

Python Shell

File Edit Debug Options Windows

```
>>> type( 1 )
<type 'int'>
>>> type( 2.0 )
<type 'float'>
>>> type( 'Hello IntERLab' )
<type 'str'>
>>> a = 1
>>> b = 2.0
>>> c = 'Hello IntERLab'
>>> type( a )
<type 'int'>
>>> type( b )
<type 'float'>
>>> type( c )
<type 'str'>
>>> a + b
3.0
>>> type( a + b )
<type 'float'>
>>> c + a
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    c + a
TypeError: cannot concatenate 'str' and 'int' objects
```

The type() function tells you the object's type.

Types of variables automatically follow the types of their values. No need to declare the variables ;-)

Type conversions, if possible, occur automatically.

Otherwise, you get an 'Exception'

Long integer and complex data types

Python Shell

File Edit Debug Options Windows Hel

```
>>> (1024 * 1024 * 1024) * 2
2147483648L
>>> 2147483647
2147483647
>>> 2147483648
2147483648L
>>> type( 2147483647 )
<type 'int'>
>>> type( 2147483648 )
<type 'long'>
>>> type( (1+1j) )
<type 'complex'>
>>> (0+1j) * (0+1j)
(-1+0j)
>>> 2147483648 + (1+1j)
(2147483649+1j)
>>>
```

Conversion to long integer is automatic

(0+1j) is the square root of -1

Boolean data type

Python Shell

File Edit Debug Options Windows He

```
>>> type( True )
<type 'bool'>
>>> True or False
True
>>> True and False
False
>>> 0 or False
False
>>> 1 and True
True
>>> True and 1
1
>>> bool( 1 )
True
>>> bool( 0 )
False
>>> bool( 'False' )
True
>>> 1 == True
True
>>> 0 == True
False
>>> True == 1
True
>>> False == 0
True
```

It is a bit tricky here, we better stick with the 'True' and 'False' notations.

The bool() type conversion interprets 0 as False, but anything else as True.

Less tricky than the above

Explicit type conversions

Python Shell

File Edit Debug Options Windows He

```
IDLE 2.6.5      ==== No Subprocess ====
>>> float( 15 )
15.0
>>> int( 12.354 )
12
>>> str( 20.3453 )
'20.3453'
>>> float( '3.141593' )
3.1415929999999999
>>> str( 1.2345 ) +  '6'
'1.23456'
>>> list( 20.3213 )
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    list( 20.3213 )
TypeError: 'float' object is not iterable
>>> list( 'Hello World' )
['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
>>> str( ['2', 0, 1.0, '3' ] )
"['2', 0, 1.0, '3']"
>>> type( True )
<type 'bool'>
>>> str( True )
'True'
```

When convert to float, be careful with hardware's floating point precision !!

If you cannot explicitly convert it, you will know.

You can make a list from a string, but the reverse may not be the same!

Ln: 232 Col

List objects

- A list consists of ordered python objects
- You can mix objects of different types in a single list

```
>>> [1, 2, 3, 'Hello', 'World', 0.2 ]
```

```
[1, 2, 3, 'Hello', 'World', 0.2]
```

Methods applicable to list objects

- Append
- Count
- Extend
- Index
- Insert
- Pop
- Remove
- Reverse
- Sort

The list.append() method

```
>>> a = list()
>>> a.append('1')
>>> a.append(['2','3','4'])
>>> a
['1', ['2', '3', '4']]
>>> a.append([['x','y','z'],['Hello', 'World']] )
>>> a
['1', ['2', '3', '4'], [['x', 'y', 'z'], ['Hello', 'World']]]
```

The list.extend() method

```
>>> b = list()
>>> b.extend('1')
>>> b.extend(['2','3','4'])
>>> b
['1', '2', '3', '4']
>>> b.extend( [['x','y','z'],['Hello', 'World']] )
>>> b
['1', '2', '3', '4', ['x', 'y', 'z'], ['Hello', 'World']]
```

The list.insert() method

```
>>> x = ['A', 'B', 'C', 'D', 'E' ]
>>> x.insert( 2, 'Hello' )
>>> x
['A', 'B', 'Hello', 'C', 'D', 'E']
>>> x.insert( 4, 'World' )
>>> x
['A', 'B', 'Hello', 'C', 'World', 'D', 'E']
```

The list.reverse() method

```
>>> x = ['a', 'b', 'c', 'd', 'e']
>>> x
['a', 'b', 'c', 'd', 'e']
>>> x.reverse()
>>> x
['e', 'd', 'c', 'b', 'a']
```

The list.sort() method

```
>>> y = [1, 4, 2, 6, 8, 0, 9]
>>> y.sort()
>>> y
[0, 1, 2, 4, 6, 8, 9]
```

The list.pop() method

```
>>> x = [101, 102, 103, 104, 105, 106]
>>> x
[101, 102, 103, 104, 105, 106]
>>> x.pop()
106
>>> x
[101, 102, 103, 104, 105]
>>> x.pop()
105
>>> x
[101, 102, 103, 104]
```

The list.pop() method

```
>>> x = ['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six' ]
>>> x.pop(4)
'Four'
>>> x
['Zero', 'One', 'Two', 'Three', 'Five', 'Six']
>>> x.pop(5)
'Six'
>>> x
['Zero', 'One', 'Two', 'Three', 'Five']
>>> x.pop(1)
'One'
>>> x
['Zero', 'Two', 'Three', 'Five']
```

The list.count() method

```
>>> x = [1, 2, 3, 'a', 'b', 'c', 'a', 2, 2]
>>> x
[1, 2, 3, 'a', 'b', 'c', 'a', 2, 2]
>>> x.count( 2 )
3
>>> x.count( 'a' )
2
```

Dictionary objects

- A dictionary is a series of modifiable { key:object } entries
- Each key is associated with a specific object
  - A key must be one of the hash-able types
    - integer/long, string, float, complex
  - An object can be of any Python type

```
>>> {1:'Peter', 2:'John', 3:'Rose', 'h':['Get','Help']}
```

Dictionary Example

```
IDLE 2.6.5      ==== No Subprocess ====
>>> d = dict()
>>> d[(1+2j)] = 'Hello'
>>> d['three'] = 3
>>> d[4] = ['Number', 'Four']
>>> d[10000000000L] = True
>>> d.keys()
[10000000000L, 4, 'three', (1+2j)]
>>> d.values()
[True, ['Number', 'Four'], 3, 'Hello']
>>> d['nodes'] = ['192.168.0.1', '128.0.0.1', '10.2.1.2' ]
>>> d.keys()
[10000000000L, 'nodes', 4, 'three', (1+2j)]
>>> d.values()
[True, ['192.168.0.1', '128.0.0.1', '10.2.1.2'], ['Number', 'Four'], 3, 'Hello']
>>> d['nodes']
['192.168.0.1', '128.0.0.1', '10.2.1.2']
>>> d['nodes'][0]
'192.168.0.1'
>>> d['nodes'].append( '192.155.32.1' )
>>> d['nodes']
['192.168.0.1', '128.0.0.1', '10.2.1.2', '192.155.32.1']
>>>
```

You can add a new entries to the dictionary in the form of key → object

dict.keys() and dict.values() methods

This simplifies record management effort and our lives !!

The dict.get() method

```
>>> x = { 1:'One', 2:'Two', 3:'Three', 4:'Four' }
>>> x.get( 2 )
'Two'
>>> x[2]
'Two'
>>> x.get(5)
>>> x[5]
```

Notice that there's no error here

This causes a KeyError exception

```
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    x[5]
KeyError: 5
```

The dict.iterkeys() method

```
>>> x = { 1:'One', 2:'Two', 3:'Three', 4:'Four' }

>>> ik = x.iterkeys()
>>> for k in ik:
        print k

1
2
3
4
```

Notice that ik can be used for one-time only

The dict.itervalues() method

```
>>> x = { 1:'One', 2:'Two', 3:'Three', 4:'Four' }

>>> iv = x.itervalues()
>>> for v in iv:
        print v

One
Two
Three
Four
```

Notice that iv can be used for one-time only

Set objects

- Allow basic set operations
  - Union
  - Intersection
  - Difference

Example of Sets

```
>>> myNeighbors =set(['192.168.0.1','192.168.0.2','192.168.0.3','192.168.0.4'])
>>> SamNeighbors=set(['192.168.8.5','192.168.0.2','192.168.9.7','192.168.0.4'])
>>> U1 = myNeighbors.union( SamNeighbors )
>>> U2 = SamNeighbors.union( myNeighbors )
>>> U1
set(['192.168.9.7', '192.168.8.5', '192.168.0.2', '192.168.0.3', '192.168.0.1',
'192.168.0.4'])
>>> U2
set(['192.168.9.7', '192.168.8.5', '192.168.0.2', '192.168.0.3', '192.168.0.1',
'192.168.0.4'])
>>> U1 == U2
True
>>> U1 == myNeighbors
False
>>> D1 = myNeighbors.difference( SamNeighbors )
>>> D2 = SamNeighbors.difference( myNeighbors )
>>> D1
set(['192.168.0.3', '192.168.0.1'])
>>> D2
set(['192.168.9.7', '192.168.8.5'])
>>> SamNeighbors.intersection( myNeighbors )
set(['192.168.0.2', '192.168.0.4'])
>>> myNeighbors.intersection( SamNeighbors )
set(['192.168.0.2', '192.168.0.4'])
```

In terms of network programming …

- Which Python data type(s) would you use to
  - keep the history of the URLs (websites) that you have visited ?
  - count how often you visit particular URLs ?
  - discover and maintain connectivity with your neighbor nodes ?
  - implement a DNS-like name lookup service. (e.g. mapping ‘www.google.com’ to ‘79.14.254.104’) ?
  - monitor available services (e.g. ftp, http, ssh) on a website ?

Function definition

```
IDLE 2.6.5      ==== No Subprocess ====
>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main']
>>> def my_add( x, y ):
        'Adds objects x and y together'
        return x + y

>>> dir()
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'main',
'my_add']
>>> help( my_add )
Help on function my_add in module __main__:

my_add(x, y)
    Adds objects x and y together

>>> my_add( 5, 2 )
7
>>> my_add( 'Hello ', 'World' )
'Hello World'
>>> my_add( 'www', '.google.com' )
'www.google.com'
>>> my_add( 2.0, (1+3j) )
(3+3j)
```

Use 'def' keyword to define a function

Documentation is easy. You SHOULD document your code for readability.

Type conversions occur automatically, if x and y are compatible with '+'

Exception handling

```
>>> my_add( 'Web ', 2.0 )
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    my_add( 'Web ', 2.0 )
  File "<pyshell#4>", line 3, in my_add
    return x + y
TypeError: cannot concatenate 'str' and 'float' objects
```

This is a bad way for a program to die..

```
>>> def my_smart_add( x, y ):
        'smarter way to add x and y together'
        try:
                return x + y
        except Exception as e:
                print "Error detected:", e
                return None
```

We use the 'try' and 'except' construct to handle exceptions like this

```
>>> result = my_smart_add( 'Web ', 2.0 )
Error detected: cannot concatenate 'str' and 'float' objects
>>> print result
None
>>> result is None
True
```

Applying a function to all list elements

```
>>> def square( i ):
        'computes the square of i'
        return i * i

>>> x = range( 10 )
>>> y = list()
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> y
[]
>>> for i in x:
        y.append( square( i ) )


>>> y
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Say, we want to compute the squares of all elements in x

This is a slow way to do it.

Can we make it faster & shorter ?

Applying a function to all list elements

```
>>> def square( i ):
        'computes the square of i'
        return i * i

>>> x = range( 10 )
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> y = map( square, x )
>>> y
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>>
```

This is a shorter and faster way to do it. The map( ) function replaces the for loop in previous example.

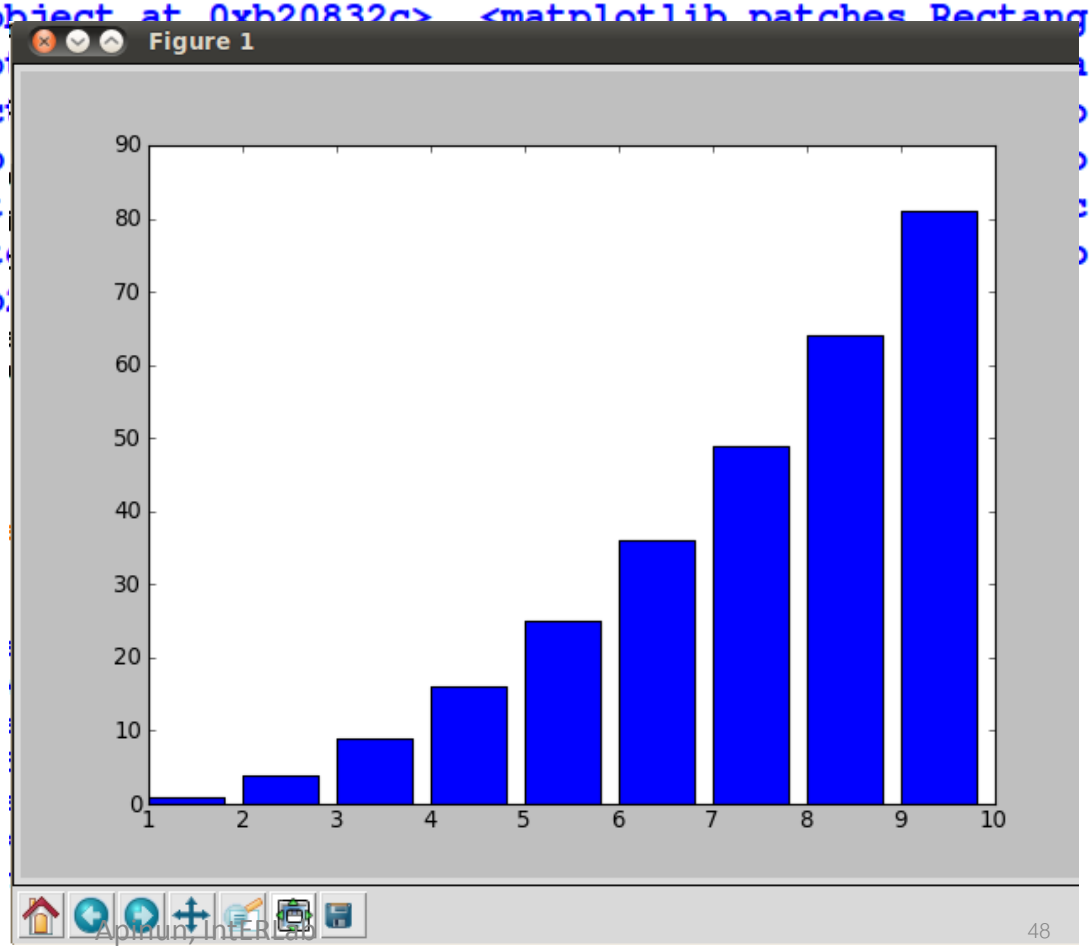
Applying a function to all list elements

```
>>> x = range( 10 )
>>> x
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> y = map( lambda i: i*i, x )
>>> y
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>>
```

This is another shorter and faster way to do it. We define a lambda function (i.e. a function without an explicit name), right inside map( ).

Let’s plot

```
>>> x = range( 10 )
>>> y = map( lambda i: i*i, x )
>>> from pylab import bar, show
>>> bar(x, y)
[<matplotlib.patches.Rectangle object at 0xb20832c>, <matplotlib.patches.Rectang
le object at 0xb20868c>, <matplo
tplotlib.patches.Rectangle objec
bject at 0xb208eec>, <matplotlib
tlib.patches.Rectangle object at
t at 0xb20e8cc>, <matplotlib.pat
.patches.Rectangle object at 0xb
>>> show()
```



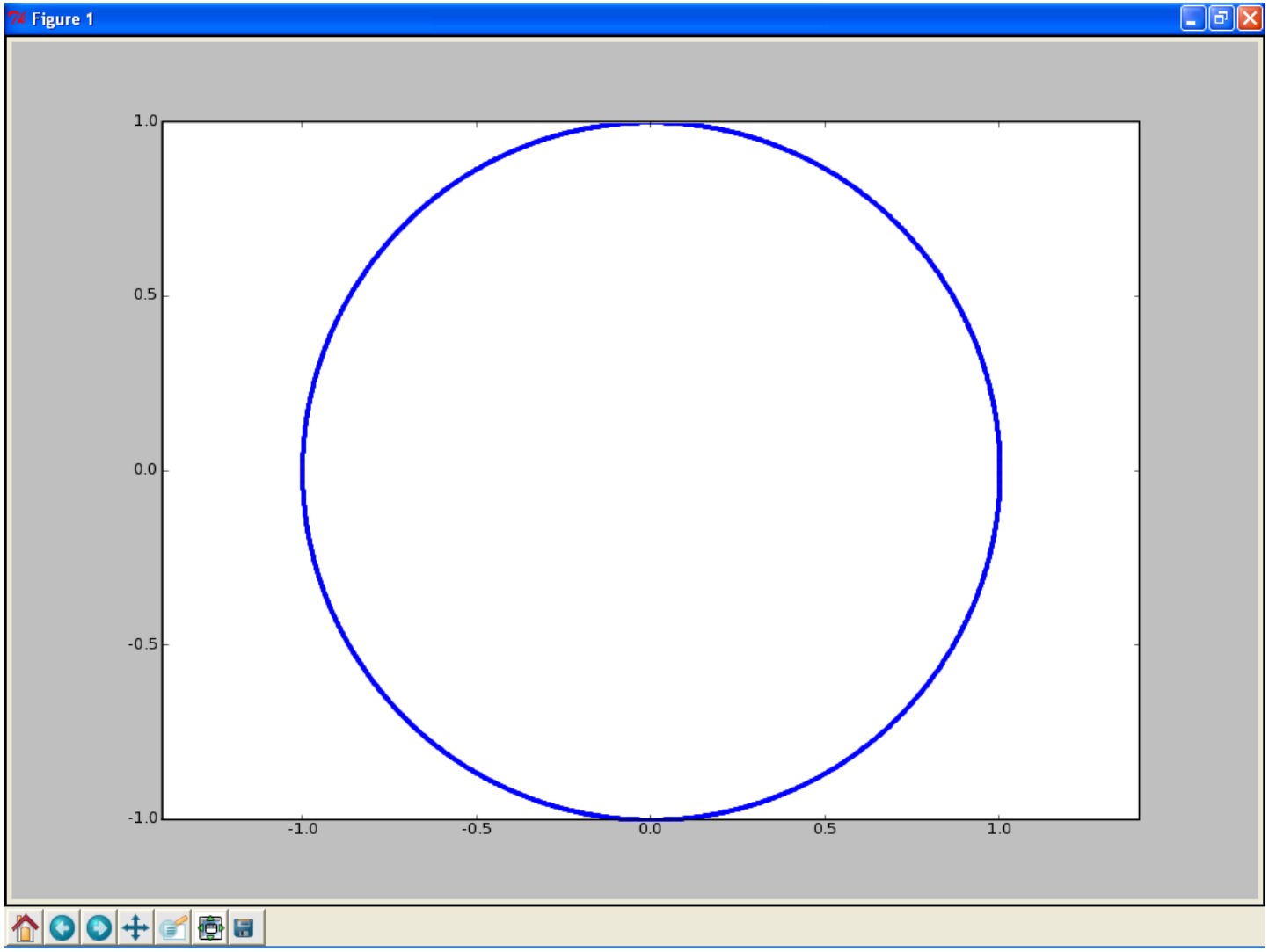
Plotting multiple series

```
>>> from pylab import *
>>> x = arange( 0, 10 )   # a list  [0,1,...,9]
>>> y1 = x * 2
>>> y2 = x ** 2
>>> plot( x, y1, 'ro:', x, y2, 'gd-' )
>>> show()
```

More Matplotlib examples

- http://matplotlib.sourceforge.net/gallery.html

Exercise : plot a unit circle (radius 1)



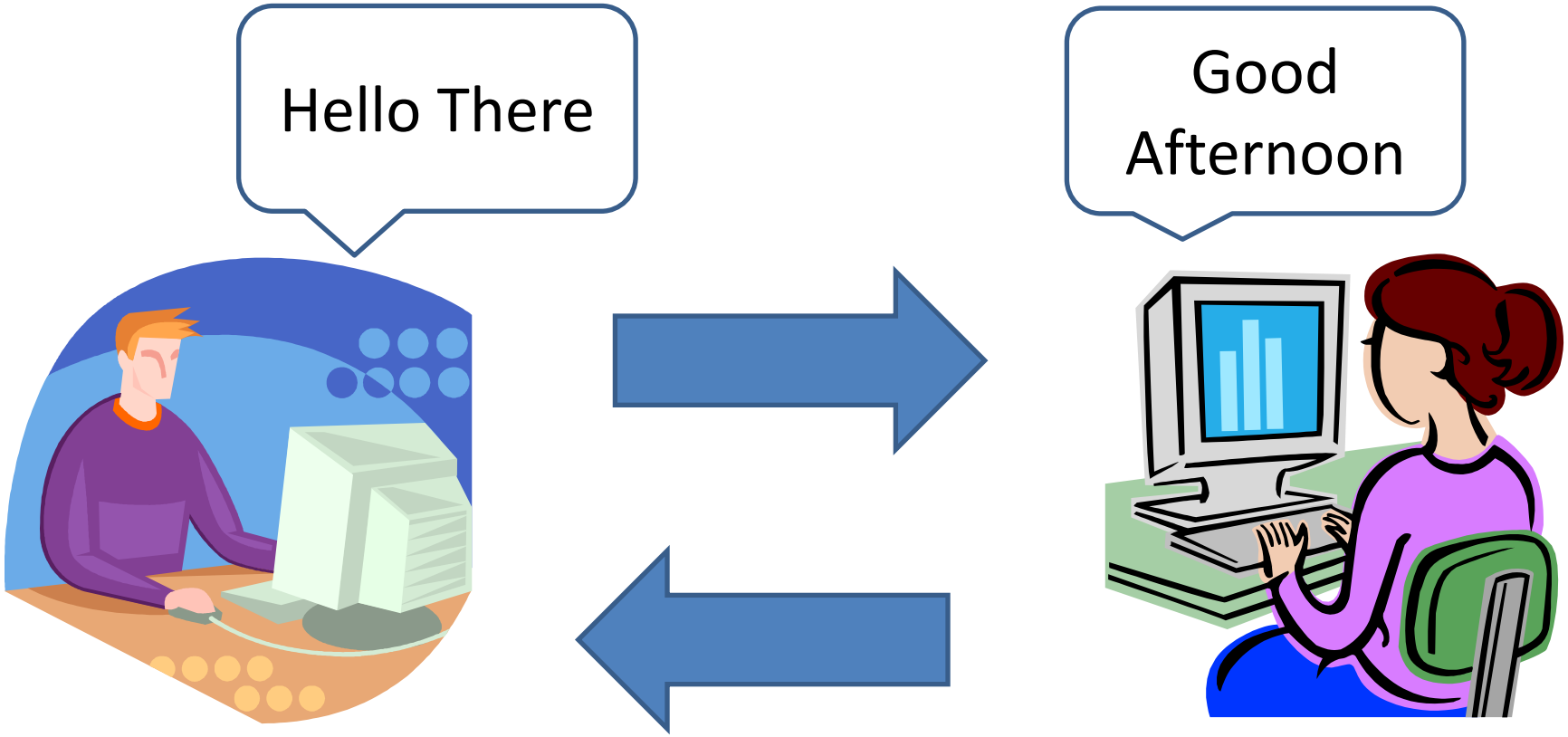
Make your computer speak

```
#
# Example: Make your computer speak.
# Apinun Tunpan, intERLab

import speechd

def main():
    sp = speechd.Speaker( 'MySpeaker' )
    sp.set_language( 'en' )
    sp.set_synthesis_voice( 'us' )
    sp.set_voice( 'FEMALE1' )
    sp.set_rate( 1 )
    sp.set_punctuation( speechd.PunctuationMode.SOME )
    sp.speak( 'Hello, how are you' )
    sp.close()

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

2. Socket Programming

Connecting computers…

Sending information between computers



RecvMesgUDP.py (UDP/IP)

```
PORT = 10000
MAX_SIZE = 1024

import socket

def main():
    s = socket.socket( socket.AF_INET, socket.SOCK_DGRAM )
    s.bind( ('', PORT ) )
    while True:
        print "Waiting for a message..."
        mesg, source = s.recvfrom( MAX_SIZE )
        print "Received from IP %s Port %d: %s" %( source[0],source[1], mesg )


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

SendMesgUDP.py (UDP/IP)

```
PORT = 10000

import socket

def main():
    s = socket.socket( socket.AF_INET, socket.SOCK_DGRAM )

    destination = raw_input( 'Destination IP :' )

    while True:
        mesg = raw_input( 'Message to send: ' )
        if len( mesg ) > 0:
            s.sendto( mesg, (destination, PORT))

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

Exercise

- Modify RecvMesg.py so that it reads the received message to you out loud…

Exercise : Identify the problems associated with SendMesgUDP.py and RecvMesgUDP.py

- What happens if :
  - You send a message to someone who is not running RecvMesgUDP.py ?
  - You send while your network interface is down ?
  - You send to an IP which is in the same subnet ?
  - You send to an IP which is in a different subnet ?
  - MAX_SIZE is too small ?

- Do you experience any other breakdowns ?

RecvMesgTCP.py (TCP/IP)

```
# RecvMesgTCP.py  (TCP/IP)
PORT = 10000
MAX_SIZE = 1024

import socket

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind( ( '',PORT ) )
    print "Listening for incoming connection request..."
    s.listen(1)
    conn, addr = s.accept()
    print "Connection established, client is at IP %s Port %d" % (addr[0],addr[1])

    while True:
        print "Waiting for a message ..."
        mesg = conn.recv( MAX_SIZE )
        if len(mesg) > 0:
            print "Message: %s" % (mesg)
        else:
            break

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

SendMesgTCP.py (TCP/IP)

```
# SendMesgTCP.py

PORT = 10000

import socket

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM )

    destination = raw_input( 'Destination IP :' )

    s.connect( (destination, PORT) )

    while True:
        mesg = raw_input( 'Message to send : ' )
        if len( mesg ) > 0:
            s.send( mesg )

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

Exercise : Identify the problems associated with SendMesgTCP.py and RecvMesgTCP.py

- What happens if :
  - You send a message to someone who is not running RecvMesgTCP.py ?
  - RecvMesgTCP.py dies ?
  - SendMesgTCP.py dies ?
  - MAX_SIZE is too small ?

- Do you experience any other breakdowns ?

socket.gethostbyname() and socket.gethostbyname_ex()

```
>>> help( socket.gethostbyname_ex )
Help on built-in function gethostbyname_ex in module _socket:

gethostbyname_ex(...)
    gethostbyname_ex(host) -> (name, aliaslist, addresslist)

    Return the true host name, a list of aliases, and a list of IP addresses,
    for a host.  The host argument is a string giving a host name or IP number.

>>> socket.gethostbyname( 'google.com' )
'72.14.254.104'
>>> socket.gethostbyname_ex( 'google.com' )
('google.com', [], ['72.14.254.104'])
>>> socket.gethostbyname( 'yahoo.com' )
'209.191.122.70'
>>> socket.gethostbyname_ex( 'yahoo.com' )
('yahoo.com', [], ['72.30.2.43', '98.137.149.56', '209.191.122.70', '67.195.160.
76', '69.147.125.65'])
>>> socket.gethostbyname_ex( 'www.google.com' )
('www.l.google.com', ['www.google.com'], ['72.14.254.104'])
```

Further self study

- What are blocking vs. non-blocking sockets ?
  - How can use make use of them ?

3. WWW information processing

Web browser control

```
>>> import webbrowser
>>> webbrowser.open_new( 'http://www.google.com' )
>>> webbrowser.open_new_tab( 'http://www.yahoo.com' )
```

Exercise : Interfacing with Google maps™

- The Google maps™ service allows us to open a map using a URL similar to the following

http://maps.google.com/maps?f=q&source=s_q&hl=en&q=14.077458,100.612863

Write a program that asks the latitude and longitude of a location and show such a location on Google maps™

Retrieving html data from a web page

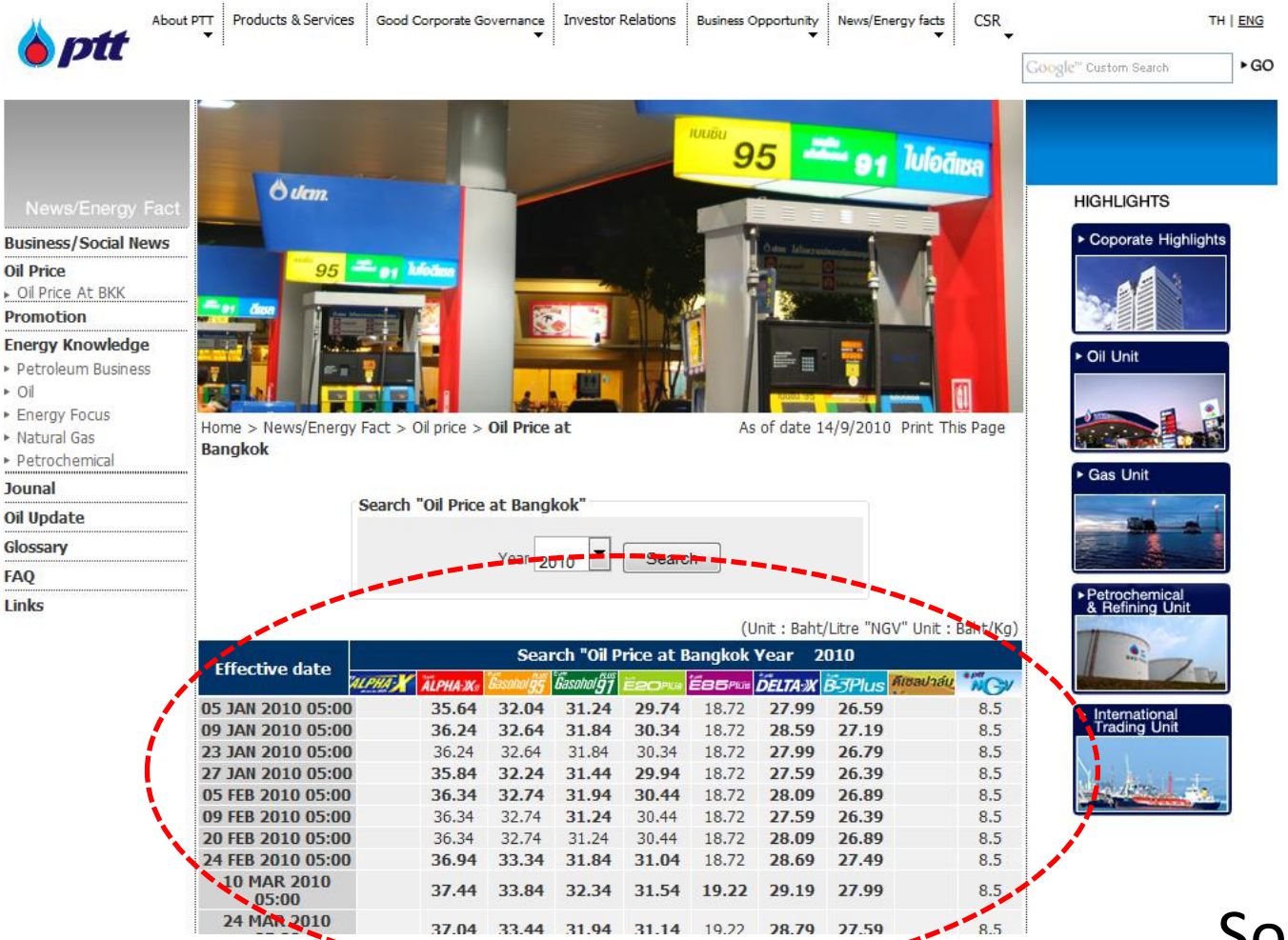
```
>>> import urllib
>>> web  = urllib.urlopen('http://cnn.com')
>>> text = web.read()
>>> print "read", len(text), "bytes"
>>> print text
```

You will get the whole html document in a single string

XML DOM Parsing

```
>>> from xml.dom import minidom
>>> import urllib

>>> url = 'http://www.w3schools.com/xml/simple.xml'
>>> web = urllib.urlopen( url )
>>> xmldoc = minidom.parse( web )
>>> web.close()

>>> print xmldoc.toxml()
```

If you drive, then fuel price is one of your concerns



ptt

About PTT | Products & Services | Good Corporate Governance | Investor Relations | Business Opportunity | News/Energy facts | CSR

TH | ENG

Google™ Custom Search ▸GO

News/Energy Fact

- Business/Social News
- Oil Price
  - Oil Price At BKK
- Promotion
- Energy Knowledge
  - Petroleum Business
  - Oil
  - Energy Focus
  - Natural Gas
  - Petrochemical
- Jounal
- Oil Update
- Glossary
- FAQ
- Links

Home > News/Energy Fact > Oil price > **Oil Price at Bangkok**

As of date 14/9/2010 Print This Page

Search "Oil Price at Bangkok"

Year 2010 Search

(Unit : Baht/Litre "NGV" Unit : Baht/Kg)

Effective date	Search "Oil Price at Bangkok Year 2010								
	ALPHA-X	ALPHA-X	Gasohol 95	Gasohol 91	E20	E85	DELTA-X	B-5 Plus	NGV
05 JAN 2010 05:00		35.64	32.04	31.24	29.74	18.72	27.99	26.59	8.5
09 JAN 2010 05:00		36.24	32.64	31.84	30.34	18.72	28.59	27.19	8.5
23 JAN 2010 05:00		36.24	32.64	31.84	30.34	18.72	27.99	26.79	8.5
27 JAN 2010 05:00		35.84	32.24	31.44	29.94	18.72	27.59	26.39	8.5
05 FEB 2010 05:00		36.34	32.74	31.94	30.44	18.72	28.09	26.89	8.5
09 FEB 2010 05:00		36.34	32.74	31.24	30.44	18.72	27.59	26.39	8.5
20 FEB 2010 05:00		36.34	32.74	31.24	30.44	18.72	28.09	26.89	8.5
24 FEB 2010 05:00		36.94	33.34	31.84	31.04	18.72	28.69	27.49	8.5
10 MAR 2010 05:00		37.44	33.84	32.34	31.54	19.22	29.19	27.99	8.5
24 MAR 2010		37.04	33.44	31.94	31.14	19.22	28.79	27.59	8.5

HIGHLIGHTS

- Coporate Highlights
- Oil Unit
- Gas Unit
- Petrochemical & Refining Unit
- International Trading Unit

Source: PTT

An HTML parser example

```
from sgmllib import SGMLParser
import urllib

class MyTableParser( SGMLParser ):
    def __init__(self):
        SGMLParser.__init__(self)
        self.inTable = False

    def start_table(self, attrs):
        self.inTable = True

    def handle_data(self, data):
        if self.inTable == True:
            print "Table data found: " + str( data )

    def end_table(self):
        self.inTable = False

def main():
    web = urllib.urlopen('http://www.pttplc.com/en/news-energy-fact-oil-price-bangkok.aspx')
    content = web.read()
    MyParser = MyTableParser()
    MyParser.feed( content )

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

Sample output

```
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: ♩
```

Exercises

- Think about how to make the oil price html parser smarter
- There are some other web sites which offer much nicer web services, can you identify these sites and how to use them ?

Thank you for your attention