Tutorial 2: MAC addresses

First of all you need to pull the various MAC related classes and functions into your namespace.

Note: Do this for the purpose of this tutorial only. In your own code, you should be explicit about the classes, functions and constants you import to avoid name clashes.

```
>>> from netaddr import *
```

You can reasonably safely import everything from the netaddr namespace as care has been taken to only export the necessary classes, functions and constants.

Always hand pick your imports if you are unsure about possible name clashes.

Basic operations

Instances of the EUI class are used to represent MAC addresses.

```
>>> mac = EUI('00-1B-77-49-54-FD')
```

Standard repr() access returns a Python statement that can reconstruct the MAC address object from scratch if executed in the Python interpreter.

```
>>> mac
EUI('00-1B-77-49-54-FD')
```

Accessing the EUI object in the string context.

```
>>> str(mac)
'00-1B-77-49-54-FD'
>>> '%s' % mac
'00-1B-77-49-54-FD'
```

Here are a few other common properties.

```
>>> str(mac), str(mac.oui), mac.ei, mac.version
('00-1B-77-49-54-FD', '00-1B-77', '49-54-FD', 48)
```

Numerical representations

You can view an individual IP address in various other formats.

```
>>> int(mac) == 117965411581
True
>>> hex(mac)
'0x1b774954fd'
>>> oct(mac)
'01556722252375'
>>> mac.bits()
'00000000-00011011-01110111-01001001-01010100-11111101'
>>> mac.bin
'0b110110111011101001001010101011111101'
```

Formatting

It is very common to see MAC address in many different formats other than the standard IEEE EUI-48.

The EUI class constructor handles all these common forms.

```
>>> EUI('00-1B-77-49-54-FD')
EUI('00-1B-77-49-54-FD')
```

IEEE EUI-48 lowercase format

```
>>> EUI('00-1b-77-49-54-fd')
EUI('00-1B-77-49-54-FD')
```

Common UNIX format

```
>>> EUI('0:1b:77:49:54:fd')
EUI('00-1B-77-49-54-FD')
```

Cisco triple hextet format

```
>>> EUI('001b:7749:54fd')
EUI('00-1B-77-49-54-FD')
>>> EUI('1b:7749:54fd')
EUI('00-1B-77-49-54-FD')
>>> EUI('1B:7749:54FD')
EUI('00-1B-77-49-54-FD')
```

Bare MAC addresses (no delimiters)

```
>>> EUI('001b774954fd')
EUI('00-1B-77-49-54-FD')
>>> EUI('01B774954FD')
EUI('00-1B-77-49-54-FD')
```

PostreSQL format (found in documentation)

```
>>> EUI('001B77:4954FD')
EUI('00-1B-77-49-54-FD')
```

It is equally possible to specify a selected format for your MAC string output in the form of a 'dialect' class. Its use is similar to the dialect class used in the Python standard library csv module.

```
>>> mac = EUI('00-1B-77-49-54-FD')
>>> mac
EUI('00-1B-77-49-54-FD')
>>> mac.dialect = mac_unix
>>> mac
EUI('0:1b:77:49:54:fd')
>>> mac.dialect = mac_unix_expanded
>>> mac
EUI('00:1b:77:49:54:fd')
>>> mac.dialect = mac_cisco
>>> mac
EUI('001b.7749.54fd')
>>> mac.dialect = mac_bare
>>> mac
EUI('001B774954FD')
>>> mac.dialect = mac_pgsql
>>> mac
EUI('001b77:4954fd')
```

You can, of course, create your own dialect classes to customise the MAC formatting if the standard ones do not suit your needs.

Here's a tweaked UNIX MAC dialect that generates uppercase, zero-filled octets.

```
>>> class mac_custom(mac_unix): pass
>>> mac_custom.word_fmt = '%.2X'
>>> mac = EUI('00-1B-77-49-54-FD', dialect=mac_custom)
>>> mac
EUI('00:1B:77:49:54:FD')
```

Querying organisational information

EUI objects provide an interface to the OUI (Organisationally Unique Identifier) and IAB (Individual Address Block) registration databases available from the IEEE.

Here is how you query an OUI with the EUI interface.

```
>>> mac = EUI('00-1B-77-49-54-FD')
>>> oui = mac.oui
>>> oui
OUI('00-1B-77')
>>> oui.registration().address
[u'Lot 8, Jalan Hi-Tech 2/3', u'Kulim  Kedah  09000', u'MY']
>>> oui.registration().org
u'Intel Corporate'
```

You can also use OUI objects directly without going through the EUI interface.

A few OUI records have multiple registrations against them. I'm not sure if this is recording historical information or just a quirk of the IEEE reigstration process.

This example shows you how you access them individually by specifying an index number.

```
>>> oui = OUI(524336)    #   OUI constructor accepts integer values, too.
>>> oui
OUI('08-00-30')
>>> oui.registration(0).address
[u'2380 N. ROSE AVENUE', u'OXNARD  CA  93010', u'US']
>>> oui.registration(0).org
u'NETWORK RESEARCH CORPORATION'
>>> oui.registration(0).oui
'08-00-30'
>>> oui.registration(1).address
[u'GPO BOX 2476V', u'MELBOURNE  VIC  3001', u'AU']
>>> oui.registration(1).org
u'ROYAL MELBOURNE INST OF TECH'
>>> oui.registration(1).oui
'08-00-30'
>>> oui.registration(2).address
[u'CH-1211 GENEVE 23', u'SUISSE/SWITZ', u'CH']
>>> oui.registration(2).org
u'CERN'
>>> oui.registration(2).oui
'08-00-30'
>>> for i in range(oui.reg_count):
...     str(oui), oui.registration(i).org
...
('08-00-30', u'NETWORK RESEARCH CORPORATION')
('08-00-30', u'ROYAL MELBOURNE INST OF TECH')
('08-00-30', u'CERN')
```

Here is how you query an IAB with the EUI interface.

```
>>> mac = EUI('00-50-C2-00-0F-01')
>>> mac.is_iab()
True
>>> iab = mac.iab
>>> iab
IAB('00-50-C2-00-00-00')
>>> iab.registration()
{'address': [u'1241 Superieor Ave E', u'Cleveland  OH  44114', u'US'],
 'iab': '00-50-C2-00-00-00',
 'idx': 84680704,
 ...
 'org': u'T.L.S. Corp.',
 'size': 537}
```