Important modules: Biopython, SQL & COM

Information sources

- python.org
  - tutor list (for beginners), the Python Package index, on-line help, tutorials, links to other documentation, and more.
- biopython.org (and mailing list)
- newsgroup comp.lang.python

Biopython

- www.biopython.org
- Collection of many bioinformatics modules
- Some well tested, some experimental
- Check with biopython.org before writing new software. It may already exist.
- Contribute your code (even useful scripts) to them.

The Seq object

```
>>> from Bio import Seq
>>> seq = Seq.Seq("ATGCATGCATGATGATCG")
>>> print seq
Seq('ATGCATGCATGATGATCG', Alphabet())
>>>
```

Alphabet? What's that?

Python doesn't know that you gave it DNA.
(It could be a strange protein.)

Alphabets

```
>>> from Bio import Seq
>>> from Bio.Alphabet import IUPAC
>>> protein = Seq.Seq("ATGCATGCATGC", IUPAC.protein)
>>> dna = Seq.Seq("ATGCATGCATGC", IUPAC.unambiguous_dna)
>>> protein[:10]
Seq('ATGCATGCAT', IUPACProtein())
>>> protein[:10] + protein[::-1]
Seq('ATGCATGCATCGTACGTACGTA', IUPACProtein())
>>> dna[:6]
Seq('ATGCAT', IUPACUnambiguousDNA())
>>> dna[0]
'A'
>>> protein[:10] + dna[:6]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "/usr/local/lib/python2.3/site-packages/Bio/Seq.py", line 45, in __add__
    raise TypeError, ("incompatable alphabets", str(self.alphabet),
TypeError: ('incompatable alphabets', 'IUPACProtein()', 'IUPACUnambiguousDNA()')
>>>
```

Translation

```
>>> from Bio import Seq
>>> from Bio.Alphabet import IUPAC
>>> from Bio import Translate
>>>
>>> standard_translator = Translate.unambiguous_dna_by_id[1]
>>> seq = Seq.Seq("GATCGATGGGCCTATATAGGATCGAAAATCGC",
...               IUPAC.unambiguous_dna)
>>> standard_translator.translate(seq)
Seq('DRWAYIGSKI', HasStopCodon(IUPACProtein(), '*'))
>>>
```

Reading sequence files

We've put a lot of work into reading common bioinformatics file formats. As the formats change, we update our parsers. There's (almost) no reason for you to write your own GenBank, SWISS-PROT, ... parser!

Reading a FASTA file

```
>>> from Bio import Fasta
>>> parser = Fasta.RecordParser()
>>> infile = open("ls_orchid.fasta")
>>> iterator = Fasta.Iterator(infile, parser)
>>> record = iterator.next()
>>> record.title
'gi|2765658|emb|Z78533.1|CIZ78533 C.irapeanum 5.8S rRNA gene
and ITS1 and ITS2 DNA'
>>> record.sequence
'CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGGAATAAACGATCGAG
TGAATCCGGAGGACCGGTGTACTCAGCTCACCGGGGGCATTGCTCCCGTGGTGACCCTGATTTGTTGTT
GGGCCGCCTCGGGAGCGTCCATGGCGGGTTTGAACCTCTAGCCCGGCGCAGTTTGGGCGCCAAGCCATA
TGAAAGCATCACCGGCGAATGGCATTGTCTTCCCCAAAACCCGGAGCGGCGGCGTGCTGTCGCGTGCCC
AATGAATTTTGATGACTCTCGCAAACGGGAATCTTGGCTCTTTGCATCGGATGGAAGGACGCAGCGAAA
TGCGATAAGTGGTGTGAATTGCAAGATCCCGTGAACCATCGAGTCTTTTGAACGCAAGTTGCGCCCGAGGCCATCAGGCTAAGGGCACG
CCTGCTTGGGCGTCGCGCTTCGTCTCTCTCCTGCCAATGCTTGCCCGGCATACAGCCAGGCCGGCGTGGTGCGGATGTGAAAGATTGGC
CCCTTGTGCCTAGGTGCGGCGGGTCCAAGAGCTGGTGTTTTGATGGCCCGGAACCCGGCAAGAGGTGGACGGATGCTGGCAGCAGCTGC
CGTGCGAATCCCCCATGTTGTCGTGCTTGTCGGACAGGCAGGAGAACCCTTCCGAACCCCAATGGAGGGCGGTTGACCGCCATTCGGAT
GTGACCCCAGGTCAGGCGGGGGCACCCGCTGAGTTTACGC'
```

Reading all records

```
>>> from Bio import Fasta
>>> parser = Fasta.RecordParser()
>>> infile = open("ls_orchid.fasta")
>>> iterator = Fasta.Iterator(infile, parser)
>>> while 1:
...   record = iterator.next()
...   if not record:
...     break
...   print record.title[record.title.find(" ")+1:-1]
...
C.irapeanum 5.8S rRNA gene and ITS1 and ITS2 DN
C.californicum 5.8S rRNA gene and ITS1 and ITS2 DN
C.fasciculatum 5.8S rRNA gene and ITS1 and ITS2 DN
C.margaritaceum 5.8S rRNA gene and ITS1 and ITS2 DN
C.lichiangense 5.8S rRNA gene and ITS1 and ITS2 DN
C.yatabeanum 5.8S rRNA gene and ITS1 and ITS2 DN
```

.... additional lines removed ....

Reading a GenBank file

```
>>> from Bio import GenBank
>>> parser = GenBank.RecordParser()
>>> infile = open("input.gb")
>>> iterator = GenBank.Iterator(infile, parser)
>>> record = iterator.next()
>>> record.locus
'10A19I'
>>> record.organism
'Oryza sativa (japonica cultivar-group)'
>>> len(record.features)
31
>>> record.features[0].key
'source'
>>> record.features[0].location
'1..99587'
>>> record.taxonomy
['Eukaryota', 'Viridiplantae', 'Streptophyta', 'Embryophyta',
'Tracheophyta', 'Spermatophyta', 'Magnoliophyta', 'Liliopsida',
'Poales', 'Poaceae', 'Ehrhartoideae', 'Oryzeae', 'Oryza']
>>>
```

Only changed the format name

Get data over the web

Python includes the 'urllib2' (successor to urllib) to fetch data given a URL. It can handle GET and POST requests for HTTP and HTTPS, do ftp, and read local files.

Several web servers have an interface for programs to get data directly from the server instead of going through the HTML page meant for humans. But most of the time we have to do some screen-scraping.

Live demos!

I'm off the net here at Dan's house on a Mac so I can't test the following examples. Hopefully I've had a time to prepare them during the break.

NCBI's EUtils

Designed for software to access NCBI's databases directly.
Can query literature and sequence databases.
Biopython includes a library for working with it.
Sadly, it's poorly documented.

The module for this is Bio.EUtils

Remote BLAST (at NCBI)

Biopython includes a library to run a job on NCBI's BLAST server. To your program it looks just like a normal function call.

The module for this is Bio.BLAST.NCBIWWW

BLAST (locally)

If you instead want to use a local installation of BLAST you can use Bio.Blast.NCBIStandalone

Microsoft/COM/Excel

Python runs on Unix, Macs, Microsoft Windows, and more.
Python support for Windows is very good.
"The most Microsoft compliant language outside Redmond."

COM is how one program can communicate with another under Windows. It's very easy to have Python talk to Excel to load or get data from a spreadsheet.

For more information, see the "Win32 Programming in Python" book by Mark Hammond.

SQL

Python can connect to different databases (like MySQL, PostgreSQL, and Oracle). Usually there are two ways to talk to the database; directly using the database-specific interface or indirectly through a Python adapter which tries to hide the differences between the databases.

There is a standard open-source database schema for bioinformatics called BioSQL. Python supports it.