Quiz Today

Covers:

- Concepts from lab 6
- Material from lecture 6

Time:
Programming for Biologists and Biochemists

Lecture 7: BioPython

Brian Kidd
May 10, 2010
Overview

1. Accessing Online Information
2. Object-oriented Programming
3. BioPython
4. Design Review
Overview

1. Accessing Online Information
2. Object-oriented Programming
3. BioPython
4. Design Review
Accessing Online Information

HTML pages

Public Databases

microArrays

PDB

PubMed

Queries and downloads
Getting Info Online

Go direct from python

    # tools for directly accessing urls
    import urllib

    # download a page and save it in a file. this will download
    # the homepage of the new york times online.
    urllib.urlretrieve("http://nytimes.com", "nytHomepage.html")

Simple html grab and parse script

    # Desc:
    #
    # This script grabs the source html from the nyt homepage and searches
    # this page for a keyword. At the moment this program is quite simple,
    # doing minimal parsing of the HTML code. This script could be modified
    # to extract more information from the page using regular expressions.
search_url.py
Online Genetic Databases

**NCBI** – National Center for Biotechnology Information*

GEO** – functional genomics data repository
GenBank – genetic sequence data bank
BLAST† – find similarity between sequences
OMIM‡ – human genes & genetic phenotypes
Pubmed – 19 million citations for biomedical articles

**EMBL** – European Molecular Biology Lab

**DDBJ** – DNA Data Bank of Japan

**Gene Expression Omnibus**
†Basic Local Alignment Search Tool
‡Online Medelian Inheritance in Man

*Division of the National Library of Medicine at the NIH
Online Protein Databases

**PDB** — experimentally determined structures of biomolecules

**UnitProt** — curated information about protein function, classification and cross-reference

**ExPASY** — domains, families and functional sites

*Protein Data Bank
**Universal Protein Resource
†Expert Protein Analysis System
Getting Info From Websites

http://www.soe.ucsc.edu/classes/bme060/Spring11/project/projectResources.pdf
Overview

1. Accessing Online Information
2. Object-oriented Programming
3. BioPython
4. Design Review
Procedural Programming

Break down problem into sequence of steps, and then write code to carry out each step.

So far this is the how we’ve approached solving programming problems in class.

This programming style is called **procedural programming**.
Object Oriented Programming

Sometimes a procedural programming approach doesn’t work on a problem.

Example: model the growth behavior of a colony of bacteria

As a first approximation, you can model the behavior sequentially.
Object Oriented Programming

Suppose that you want to model the growth behavior at the level of an individual cell.

You don’t know what is going to happen to each cell, but you can think of each bacteria as an “object” with “data”, like the nucleus, mitochondria, etc. and “methods” like birth, replication, metabolism.

This programming style is called object oriented programming (OOP).
Brief History of OOP

In the early 80's, Bjarne Stroustrup created C++, which was the first widely used OOP language.

Since then, Java and other OOP languages have been developed, and existing languages have been modified to fit with the OOP paradigm.

Python is an OOP language.

We've already used OOP in this class, e.g. `string.lower()`. We'll use objects even more with BioPython.
Basic Idea of OOP

Objects are data structures

Objects contain data and methods

Objects:
- CELL
  - len_dna = value
  - size = value
  - replication()
  - metabolism()

Variables:
- CELL1.len_dna
- CELL2.replication()
Specific Objects in Python

Objects are data structures

Objects contain data and methods

```
str
lower()
upper()
count()
find()
split()
...
```

```
str1.lower()
str2.split()
```
Objects in Python

Create (define) an object with a name assignment

Access the data and methods of an object with the dot syntax:

```
object.data = math.pi
object.method() = str.upper()
```
Object Basics in Python

Object – reference to a data type that belongs to a class.

Class – package containing methods to create and manipulate objects.

Method – functions defined within the class.
Classes and Objects

A **class** is the description of an **object**

Protein = class
Hemoglobin = object

A **class** has **methods** and **data**

Proteins can fold = method
Proteins can bind = method
Proteins have amino acids = data
Proteins have coordinates (x, y, z) = data

A **class** inherits methods and data from parents
Class Inheritance

Objects **inherit** methods and data from parents

- Hemoglobin inherits amino acids from Proteins
- Hemoglobin inherits binding strength from Proteins

Objects can define their own specialized methods and data

- Hemoglobins have a heme molecule that Proteins does not have
Pros and Cons of Using Objects

Pros:

- Reusable
- Modular
- Generally easier to understand and maintain

Cons:

- More difficult to write
- Need to do more research (reading of other people’s code) rather than writing code

In this class, we’ve been using built-in objects from Python. Now we'll use BioPython objects that other developers have written.
How to Get Python Objects

Before you can use a Python object, you need to obtain it:

Write your own (not recommended for class)

Number of general purpose objects are included in the standard Python distribution

Other general purpose objects are available from http://www.python.org (or elsewhere online)

Collection of objects for bioinformatics are called BioPython: http://biopython.org
Overview

1. Accessing Online Information
2. Object-oriented Programming
3. BioPython
4. Design Review
BioPython

"The BioPython Project is an international association of developers of freely available Python tools for computational molecular biology."

By the people, for the people!
BioPython

Collection of modules for dealing with biological data in Python

Open source project – http://www.biopython.org
Currently set up as a Wiki

Includes information about:
Installation
Documentation
Development
HowTo ...
Using BioPython

Modules written to solve a wide range of computational biology tasks:

- Sequence objects
- Sequence manipulation and IO
- Parsing program results (e.g. BLAST, GenBank)
- Accessing databases
- Running external programs (e.g. BLAST)

Many of the tasks that you will want for your projects or research can be performed using BioPython modules
Breakdown of BioPython

Today we'll cover the basics of sequences, records, parsing and sequence I/O [sections 1-5] of the tutorial.

In lecture 9 we'll cover BLAST, MSAs, Entrez, Swiss-Prot, and PDB [sections 6-10] of the tutorial.
It's all about Bio

The "main" module is Bio

import Bio

However, there are MANY sub modules or packages (~75)

A BioPython Object Example

Create a BioPython “object” to store a DNA sequence

```python
from Bio.Seq import Seq
first_seq = Seq('GCTATGCAGC')

len(first_seq)
print(first_seq)
```

At first pass, this looks like a more complicated version of strings, but there's more.
A BioPython Object Example

Create a BioPython “object” to store sequences explicitly

```python
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

dna_seq = Seq('GCTATGCAGC', IUPAC.unambiguous_dna)
rna_seq = Seq('GCUAUGCAGC', IUPAC.unambiguous_rna)

len(dna_seq)
print dna_seq

dna_seq + rna_seq ???
```
Learning About BioPython

http://www.biopython.org/DIST/docs/tutorial/tutorial/Tutorial.html

help(module)

Search Google
But what should you do if you don't know in which object a specific method is defined?

This can happen because objects inherit methods from other objects (and the relevant documentation will be stored in the object from which the method was inherited.)

http://biopython.org/wiki/Seq

http://biopython.org/wiki/SeqIO
Finding Method Definitions II

Use the BioPython documentation at:

http://biopython.org/wiki/Documentation
BioPython Sequence Objects

BioPython has several different objects for handling protein, DNA, and RNA sequence data

http://biopython.org/wiki/Seq

Seq

Principal sequence object in BioPython
Essentially a string with an alphabet

Alphabet

Specifies type of sequence encoding for letters (DNA, RNA, Protein)
Follows International Union of Pure and Applied Chemistry (IUPAC) standards
from Bio import Seq
help(Seq)

from Bio.Seq import Alphabet
help(Alphabet)

from Bio.Alphabet import IUPAC
help(IUPAC)
BioPython's SeqIO objects make sequence data-format conversion simple:

Data formats currently supported: Fasta, EMBL, Genbank, Swissprot, PIR and Fastq

SeqIO can read a stream of sequences - located in a single or in multiple files

Once the sequence data has been read in with SeqIO, it is available to BioPython in the form of Seq objects
from Bio import SeqIO

help(SeqIO)

http://biopython.org/wiki/SeqIO
Once a sequence is available to BioPython in the form of a sequence object, many standard bioinformatics calculations can be performed on the sequence including:

- extracting subsequences
- performing reverse complementation
- translating a DNA sequence
- identifying restriction enzyme sites
- identifying signal cleavage sites
- obtaining molecular weights and sequence statistics
BioPython's sequence translation method incorporates several features that can be very useful when translating "real life" DNA sequences:

- Selecting any one of the six translation frames
- Selecting alternate translation tables (eg for mitochondrial DNA)
- Checking for proper coding initiation and termination signals
Homework

Reading:

BioPython tutorial

http://biopython.org/wiki/Main_Page

Quiz:

No quiz next week

Design Review:

In class next week
Reminder

Project Check Point 1 Due

Updated flow chart and pseudo code of your program at that time (preferably .ppt(x), .key, or .pdf)

Be prepared to walk us through your flow chart (~90 sec) and have one question that you’re currently wrestling with.

Due in dropbox by 11:00 p.m. on Friday May 13, 2011
Code Review Format

1. 10-15 seconds to summarize problem
   Specific research problem are you addressing

2. Walk us through your flow chart

3. Ask for help on one item you’re currently stumped with

4. You talk for 2 minutes (MAX)