Programming for Biologists and Biochemists

Lecture 9: BioPython II and Wrap-up

Brian Kidd
May 24, 2010
Overview

1. argv
2. BioPython II
3. Final Review
Overview

1. argv
2. BioPython II
3. Final Review
Passing Arguments to Python

You can pass arguments to Python from the command line (doesn't work with IDLE)

Analogous to passing arguments to a function
ARGV – The Argument Vector

A list of strings typed at the command line after the program's name. Made available through sys module.

```python
from sys import argv

e.g. typing the command:

    argTester.py dna rna protein

argv now contains

    argv[0] = 'argTester.py'
    argv[1] = 'dna'
    argv[2] = 'rna'
    argv[3] = 'protein'
```

*See Chapter 6, page 213.*
ARGV Continued

Information stored as a list that can be accessed by index

Example pieces of code for reading argv

1. print argv

2. for i, a in enumerate(argv):
   print "The argument %s is %s" % (i, a)
Checking Inputs in Python: Ex1

ARGV list stores information from the command line

Suppose we expect the two input arguments to be passed to our program
1: input filename, 2: output filename

How might we set this up?

from sys import argv

iFilename = argv[1]
oFilename = argv[2]
More elaborate input checking

Suppose we expect the two input arguments to be passed to our program:
1: input filename, 2: output filename

Create some help information for someone to use your program

How might we set this up?
from sys import argv, exit

def Help():
    print "This program reads in data and writes"
    print "it to a file.\n"
    print "usage: %s <inFile> <outFile>" % argv[0]
    exit()

if len(argv) != 3:
    Help()

iFilename = argv[1]
oFilename = argv[2]
More elaborate input checking

Suppose we expect the two input arguments to be passed to our program
1: input filename, 2: output filename

Now we want to check if the first argument is a file and the filename for the second argument already exists

How might we set this up?
import os.path
from sys import argv, exit

if len(argv) != 3:
    Help()    # assumes this has been defined

iFilename = argv[1]
oFilename = argv[2]

if not os.path.isfile(iFilename):
    print "\n%s is not a file\n" % iFilename
    exit()

if os.path.exists(oFilename):
    print "\n%s already exists\n" % oFilename
    exit()

*See Chapter 6, page 231.*
Terminal Emulator

Easy on mac (X11 or Terminal)

On a PC (putty or cygwin or others*)

http://its.ucsc.edu/software/ssh/

*Talk with the TA*
Overview

1. argv

2. BioPython II

3. Final Review
"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

In lecture 7, we covered the basics of sequences, records, parsing and sequence I/O [sections 1-5] of the tutorial.

Today we'll cover BLAST, MSAs, Entrez, Swiss-Prot, and PDB [sections 6-10] of the tutorial.
BioPython II

Multiple Sequences Alignment objects

Basic Local Alignment Search Tool

NCBI Entrez

Swiss-Prot and ExPASy

Protein Data Bank
Multiple Sequence Alignments

Align module contains code for dealing with alignments (Alignment class)

- object stores sequence alignments
- not meant for making alignments
- assumed that sequences are aligned
MSA example 1

Align module

```
MHQAIFIYQIGYPLKSGYIQSIRSPPEYDNW
```

```
MH--IFIYQIGYALKSGYIQSIRSPPEY--NW
```

`msa01.py`

`msa02.py`
BLAST

Sequence similarity search program for comparing a user's query to a database of sequences

Query Sequence

Alignment 1

Alignment 2

Alignment 3

HSP 1 (Alignment 3)  HSP 2 (Alignment 3)
BLAST

Sequence similarity search program for comparing a user's query to a database of sequences

- track the discovery of genes between species
- find functional and evolutionary relationships between sequences
- identify consensus regulatory patterns
- infer structure from previously crystallized proteins
- identify members of gene families

BLAST
BLAST with Biopython

Can be run locally or remotely on NCBI webserver

blast01.py
blast02.py
blast03.py
blast04.py
Entrez

Search engine for NCBI databases

http://www.ncbi.nlm.nih.gov/books/NBK3831/

obtain articles from Pubmed
sequence databases
structure databases
expression data
genomes and taxonomic data
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>efetch</td>
<td>Retrieves records in the requested format from a list of one or more primary IDs or from the user’s environment.</td>
</tr>
<tr>
<td>einfo</td>
<td>Provides field index term counts, last update, and available links for each database.</td>
</tr>
<tr>
<td>egquery</td>
<td>Provides Entrez database counts in XML for a single search using Global Query.</td>
</tr>
<tr>
<td>elink</td>
<td>Checks for the existence of an external or Related Articles link from a list of one or more primary IDs.</td>
</tr>
<tr>
<td>epost</td>
<td>Posts a file containing a list of primary IDs for future use in the user’s environment to use with subsequent search strategies.</td>
</tr>
<tr>
<td>esearch</td>
<td>Searches and retrieves primary IDs (for use in EFetch, ELink, and ESummary).</td>
</tr>
<tr>
<td>espell</td>
<td>Retrieves spelling suggestions.</td>
</tr>
<tr>
<td>esummary</td>
<td>Retrieves document summaries from a list of primary IDs or from the user’s environment.</td>
</tr>
<tr>
<td>read</td>
<td>Parses the XML results returned by any of the above functions.</td>
</tr>
</tbody>
</table>
Entrez with Biopython

Information can be queried through eUtils

entrez01.py
entrez02.py
entrez03.py
ExPASy and Swiss-Prot
ExPASy with Biopython

prosite01.py
swissprot01.py
swissprot02.py
PDB

Database of structural information
PDB with Biopython

Can be grab and parse structures

pdb01.py
Overview

1. argv
2. BioPython II
3. Final Review
Course Overview

6 Quizzes
7 Labs
8 Lectures
Course Description

Objectives:

Become familiar with programming tools for solving biological problems*

Learn programming skills that are useful for careers in biology and the life sciences

Learn problem solving skills that apply to computational and experimental disciplines

*Expected to write your own Python program from scratch by the end.
Course Topics

Introduction to computing and python programming

Manipulating Data Types
Flow Control
Files and Functions
Searching Text and Pattern Matching
Large Datasets and Statistics
Biopython
Suggestions for Studying

Go over quizzes
Review lecture notes
Homework

Reading:

  BioPython tutorial
  
  [http://biopython.org/wiki/Main_Page](http://biopython.org/wiki/Main_Page)

Quiz:

  No quiz next week

Project Presentations:

  In class next week
Reminder

Project Check Point 2 Due

Turn in your code that demonstrates a complete main program with:

- ability to read in your input data
- proper flow of control through the program
- two working functions for your program

Due in dropbox by 11:00p.m. on Friday May 27, 2011
Beta Testers

Each student will beta test 2 programs from your fellow classmates

Code will be de-identified

Each student is required to provide feedback

Due in dropbox by **11:00p.m.** on Sunday May 29, 2011
Reminder II

Draft report Due

Draft of your final report (.doc(x) or .pdf)

Due in dropbox by 11:00 p.m. Monday May 30, 2011
Presentation Format

1. You talk for 3 minutes (MAX)

2. You get three slides that must follow the provided template
   - background for specific research problem
   - results from program
   - next steps for extending your program
Other Important Dates

Final report Due

Final report (.doc(x) or .pdf)

Due in dropbox by **11:00p.m.** Monday June 6, 2011

Final exam

Wednesday June 8, 2011

8:00a.m. – 11:00a.m. Social Sciences II Room 71
Final Report

Follows scientific article format

- title
- abstract
- introduction
- methods
- results
- discussion/conclusions

Template provided (~1600 words)
Project QA