

Class 05: The Central Dogma (2)

Held: Thursday, 10 September 2009

Summary: We begin to build our library of procedures that help us ask interesting (and not so interesting) bioinformatics questions.

Related Pages:

- EBoard.

Notes:

- Believe it or not, but we're a little ahead of schedule. We'll see what happens today.
- For Tuesday, you should start reading Chapter 3. Note that there are some spots that are rough going.
- For a week from Tuesday, you should work on the On-your-own project from Section 2.6. (Of course, "on-your-own" means "in a group of two-to-four".
- Don't forget to sign up for the Bio picnic!
- EC/Support: Watch Ben run at the Les Duke invitational on Saturday.
- CS Table tomorrow (noon, PDR A): A Blind Person's Interactions with Technology.

Overview:

- Web Exploration, Discussed
- Some more Python
- Guided Project 2.5

Web Exploration, Discussed

- Compared to the first Web exploration we gave you, these instructions were much more detailed, and much less open ended.
 - Was that good, bad, ...?
- Here are some notes, thoughts, and questions
 - Did this exploration help you find better ways to look for DNA?
 - Does the FASTA file for HBB use the template sequence or the coding sequence? How do you know?
 - Can you find typos in the GenBank record?
 - Stupid tidbit: Sam is likely to have an HBB mutation
 - What software can you use to compare sequences?
- What procedures would we have to write to make all of this easier to do?

Detour: Some Python

- There had been questions about the `map` procedure.
 - `map` is a procedure of two parameters, a procedure and a list.
 - It builds a new list by applying the procedure to each element of the list.
 - We'll do some examples
- Although I haven't mentioned it explicitly, in many cases, you can treat strings as if they were already lists.
 - You can use `map` with strings.
 - You can use `for val in sequence:` with strings
 - You can index the elements of strings
 - ...
- We'll be working a lot with individual elements of lists and strings.
 - You get a particular element with `lst[position]`.
 - You get a range of elements with `lst[start:after]`
 - You get the first n elements with `lst[:n]`
 - You get all but the first n elements with `lst[n:]`

Guided Exercise

- We're going to try a Python variant of guided project 2.5 to get you ready for on-your-own project 2.6
- What is our overall goal? Simplified sequence alignment
- What do we mean by that?
 - Find the best alignment of a short sequence to a long sequence, allowing one deletion.
 - That is, break the short sequence into no more than two parts, and align them in order. Find the best combination of alignments.
- What are our related goals?
 - Think about algorithm design
 - Build a library of useful procedures related to sequence alignment and the central dogma
 - Learn more Python!
- Since you're building a library, you should open a single file in which you're going to put all of your Python code.

Copyright © 2009 Vida Praitis and Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.