Plan for October 26-30

- Structuring Data and Code for Efficiency
  - Computer time, how expensive is it?
  - Data storage, how expensive is it?
  - Programmer time, how expensive is it?
- Problems that need solving, how to start?
  - You write APTs, your function is called
  - You may need to use other libraries, call other functions

Where does data come from?

- Files on your computer
  - How to open, read, manage
  - What about CSV files? JSON? XML?
    - We'll see this later, parsing isn't always pretty
- URLs
  - Different resource, programatically similar
- One person programs, Apps, Servers, …
  - Manage resources as if they're scarce?
  - Memory, open files, open connections …
  - Close what you open!

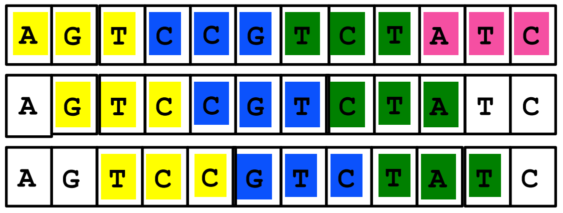
Genomic Data

- FASTA format for data, store as string, ignore first line
  - Read lines of file? Read as string?
  - Attentive to newlines '\n'
  - See ORFinder.py

```
>gi|53794092|gb|AY751490.1| Homo sapiens breast …
ATGGATTTATCTGCTCTTCGCGTTGAAGAAGTACAAAATGTCATTAATGCTATGCAGAAAATCTTAGAG
GTCCCATCTGTCTGGAGTTGATCAAGGAACCTGTCTCCACAAAGTGTGACCACATATTTTGCAAATTTT
CATGCTGAAACTTCTCAACCAGAAGAAAGGGCCTTCACAGTGTCCTTTATGTAAGAATGATATAACCAA
AGGAGCCTACAAGAAAGTACGAGATTTAGTCAACTTGTTGAAGAGCTATTGAAAATCATTTGTGCTTTT
AGCTTGACACAGGTTTGGAGTATGCAAACAGCTATAATTTTGCAAAAAAGGAAAATAACTCTCCTGAAC
TCTAAAAGATGAAGTTTCTATCATCCAAAGTATGGGCTACAGAAACCGTGCCAAAAGACTTCTACAGAG
GAACCCGAAAATCCTTCCTTGCAGGAAACCAGTCTCAGTGTCCAACTCTCTAACCTTGGAACTGTGAGA
CTCTGAGGACAAAGCAGCGGATACAACCTCAAAAGACGTCTGTCTACATTGAATTGGGATCTGATTCTT
TGAAGATACCGTTAATAAGGCAACTTATTGCAGTGTGGGAGATCAAGAATTGTTACAAATCACCCCTCA
GGAACCAGGGATGAAATCAGTTTGGATT
```

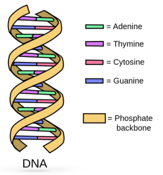
Genomic Data

- Sequenced/digital data, where does DNA/genomic data "start"? Codon is 3 bps
  - Sequence/snapshot, did we miss 0,1,2 basepairs?
  - Also read in reverse, 3 or 6 *open reading frames*



Look for "clues" for Reading Frame

- Start Codons/Stop Codon pairs
  - Codon is a triple "ATG" and more
  - Protein coding regions important, codon codes for an amino acid
- Rudimentary code in ORFinder.py
  - Counts start/stop codons ATG, TAG, TAA, TGA
  - Could look for ATG...TAG
    - Better for really finding ORF
  - Start to illustrate concepts



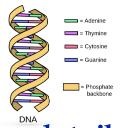
Finding counts for all Codons

- We could keep list of pairs

`[ ["CGA", 3], ["TAG", 4],…["GGC", 7],…]`

- We could keep two lists `count[k]` is number of times `codon[k]` occurs

`["CGA", "TAG", … "GGC", …]`
`[3,4,…,7, …]`



- See `ORFinder.codon_counts` for details
  - See also: `Timings.py`

Answer Questions

http://bit.ly/101fall15-oct27-data

Eugene (Gene) Myers

Lead computer scientist/software engineer a Celera Genomics, then at Berkeley, now at Max-Planc Institute

- BLAST and WG-Shotgun

"What really astounds me is the architecture of life. The system is extremely complex. It's like it was designed. … There's a huge intelligence there."

APT AnagramFree

- How do you know "spear" and "pears" are anagrams?
  - Sort the words and see if sorted form the same
  - What is returned by `sorted("spear")`?
  - What type is `''.join(sorted("spear"))`
  - Can we use `''` or `' '` or `':'` or `'|'`
- How do you know whether there are many words that are anagrams? Can sets help?

APT Cryptography

- Value returned by encrypt has type long
  - We've used int and float
  - Limitations on values in int because of size
    - Roughly 2 billion, 2^{31}-1, need negatives too!
- The type long "fixes" this
  - No limit on maximum size of integer values
  - Create using 0L and then voila, works
  - Not needed in Python 3

New Python Concept: Dictionary

- Lists are slow to search through, but they work with indexes (can keep parallel list)
  - Sets are fast, but items in sets are immutable!
  - Can't change item in set, can't index set
- Dictionary offers alternative
  - Very fast, very easy for associating keys with values (search on key, find value)
  - Example: word and # occurrences
  - Example: codon and list of indexes in DNA

Lookup in Timings.py

- Create (word,count) pairs in dictionary
  - Start with empty dictionary, {}
  - Query if word is a key in dictionary
- data[w] access
  - Value for w

```
def fast_lookup(source):
    data = {}
    for w in source:
        if w in data:
            data[w] += 1
        else:
            data[w] = 1
    return data
```

From 10,000 ft to 1 km: Dictionaries

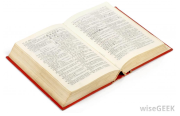
- What is a dictionary? Associate two things for quick lookup. AKA: map, hash
  - 152.3.140.1 is www.cs.duke.edu
  - 157.166.224.26 is cnn.com
  - 68.71.209.235 is espn.go.com
- A collection of (key,value) pairs
  - Look up a key, get an associated value
  - Update the value associated with a key
  - Insert a (key,value) pair
  - Loop over the keys, access pairs or value

A Python view of dictionaries

- A collection of (key,value) pairs that is similar syntactically to a list
  - A list can be accessed by index: a[3]
  - A dictionary can be accessed by key: d["cat"]
- The key in a dictionary must be immutable
  - Essentially because key converted to number and number used as index (to find value)
- Finding the value associated with a key is very fast
  - Essentially doesn't depend on # keys!

Python syntax for dictionaries

- Create a dictionary, assign values to keys:
  - `d = {}`
  - `d = {"apple":3, "guava":37}`
  - `d = dict([("owen",62.5),("bob",73.9)])`
  - `d["apple"] = 5`
- Keys are a set, support fast lookup
  - Tuples can be keys, lists cannot be keys



Python syntax for dictionaries

- Coding with Dictionaries
  - Error to access d[key] for update if key not in d

Dictionary Syntax/Function	Meaning
`d.items()`	List of (key,value) tuples
d.keys()	List of keys
d.values()	List of values
d.get(key)	Like d[key], no error
d	Query like d.keys()

Answer Questions

http://bit.ly/101fall15-oct27-1

DictionaryTimings.py

- Updating (key,value) pairs in structures
  - Search through unordered list
  - Search through ordered list
  - Use dictionary
- Why is searching through ordered list fast?
  - Guess a number from 1 to 1000, first guess?
  - What is 2^{10}? Why is this relevant? 2^{20}?
  - Dictionary is faster! But not ordered

danah boyd

Dr. danah boyd is a Senior Researcher at Microsoft Research, ... a Visiting Researcher at Harvard Law School, …Her work examines everyday practices involving social media, with specific attention to youth engagement, privacy, and risky behaviors. She heads Data & Society (Research Institute) and recently authored *It's Complicated: The Social Lives of Networked Teens.*

"we need those who are thinking about social justice to understand technology and those who understand technology to commit to social justice."

http://bit.ly/1GuB9x2