National Institute of Allergy and Infectious Diseases

Introduction to Programming for Biologists

R. Burke Squires, NIAID BCBB, richard.squires@nih.gov

NIAID

National Institute of Allergy and Infectious Diseases

Outline

- Intro
  - How Can Programming Help Us?
  - Why Do We Have To Program A Computer?
- How Do You Program?
- Where Do We Go From Here?

Why Learn To Program?

Challenges of Programming

- Picking a programming language
- Learning a programming language
- Knowing which piece of the language to use
- Debugging!!!
- Remembering what the program does when you come back to it.
- Keeping track of different programs

Opportunities

- Make your research:
  - FASTER - Repeat an entire analysis by just replacing the input
  - BETTER – Validate your analysis, use statistics
  - REPRODUCIBLE - Make your research (more) reproducible
  - MORE COMPETATIVE – many other researchers will be using these similar techniques

Learning Objectives

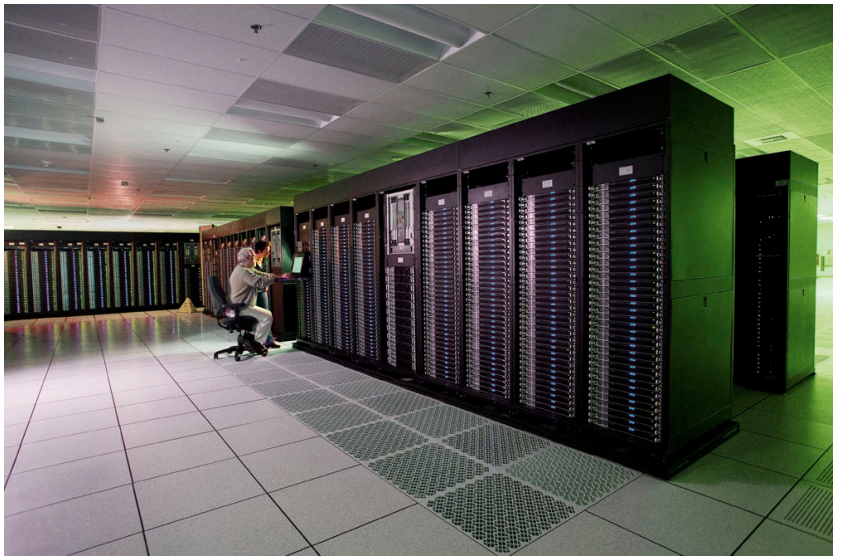
- To familiarize you with computers, programming, programming languages, scientific packages
- Enable you to communicate better with programmers, bioinformaticians
- Critique existing source code for your use
- Enable you to make your research more reproducible

What Are Some examples of How Programming Can Help?

- “Glue” program output and input together into a workflow
- Repeat an entire analysis to run on new data set(s)
- Test multiple parameters all at the same time
- Download data from a known location and process it…automatically
- Create, update, move, and rename files and folders
- …automate the boring stuff!

Computers

http://www.apple.com



Compute Cluster /
High-performance computer /
Supercomputer

https://computing.llnl.gov/tutorials/linux_clusters/

NIH National Institute of Allergy and Infectious Diseases

NIAID

How Do Computer Communicate?

- **Bi**nary digi**t**– 1, 0
  - Electrical impulses
  - “Bit”
- “Byte”
  - 8 bits
- “Word”
  - 64 bit (32 bit) – 8 bytes at a time
  - Width of communication path in central processing unit (CPU)
    - I7, i5, Xeon, Athlon

http://engineering.integrisgp.com/wp-content/uploads/2015/10/computer-speak.jpg

What Does This Mean For Us?

Programming Languages

- Computers “speak” 1, 0
- Human speak English, etc.

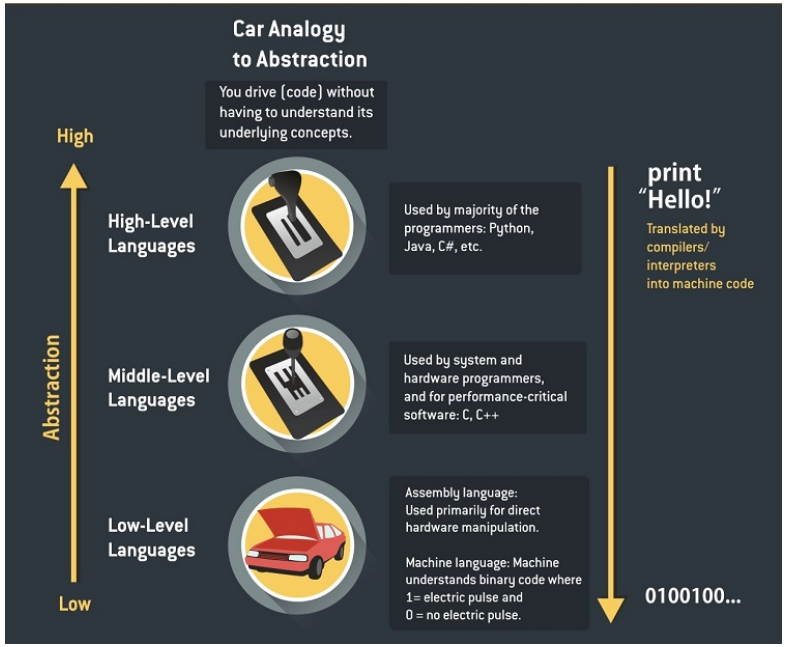
- Programming Language Levels

  - Machine languages
    - Interpreted directly in hardware

  - Assembly languages
    - Wrappers over machine language

  - High-level languages (C, Java)
    - Machine-independent language

  - Scripting languages (Python)
    - Extremely high-level and powerful

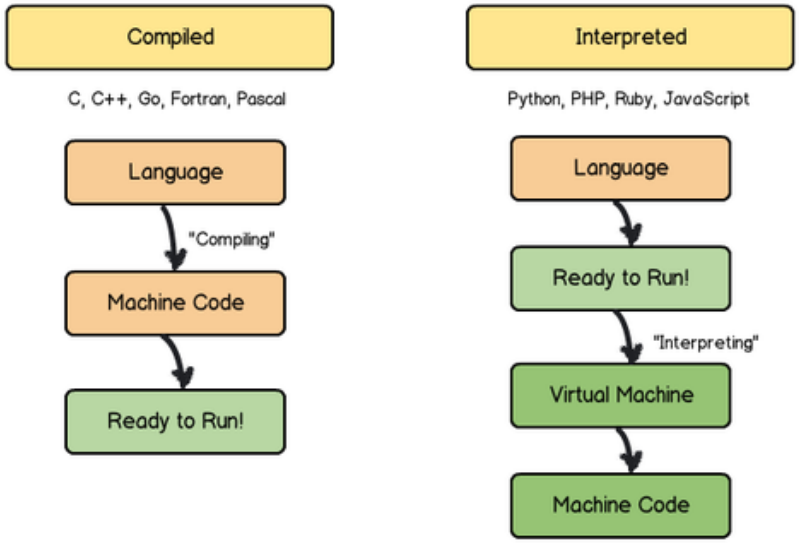


National Institute of Allergy and Infectious Diseases

http://carlcheo.com/what-is-programming-what-do-programmers-do

Interpreters / Compilers

- How do we go from high-level languages to assembly?
  - Interpreter (Python, R)
  - Compiler (C, C++, Java)
- We write program “source code”
  - Usually saved as a file
- We “run” or “execute” program



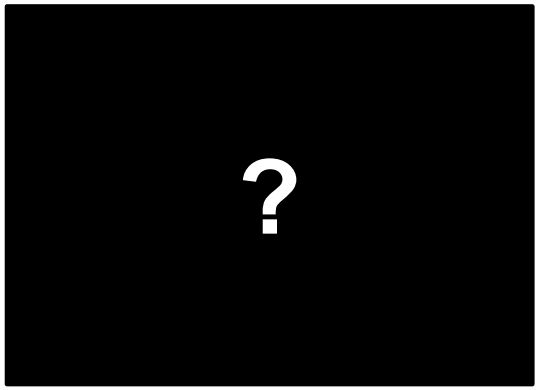
National Institute of Allergy and Infectious Diseases

http://www.hackersclub.io/single-post/2016/04/06/Scripting-vs-Coding-vs-Programming

NIAID

Algorithm

Input =>



=> Output

How Do We Program?

- Think About Your Workflow (Algorithm)
  - Write Pseudocode
  - Break down our task into smaller tasks
  - Describe the major components of our program
  - Does this sound difficult?
  - Let's look at an example that I suspect most of us are familiar with…
- Convert pseudo code into a program or "source code"
  - Write in a text file

An Example Workflow: Coffee

- Who drinks coffee?
- Who makes coffee?
- Who buys coffee?

- What does it take to make a pot or cup of coffee?

- (I'm a French Press type of coffee drinker)

AeroPress…

Workflow…or…

Steps or

Recipe or

Protocol or

Flowchart…

Ultimately this is an algorithm

National Institute of Allergy and Infectious Diseases

NIAID

Coffee

- What if I want to make different type of coffee drinks?
- What is different about my recipe?
- What is the same about my recipe?
- Is reproducibility important?
- What happens if you gave those exact instructions to a three year old?

NIAID

A New Way Of Seeing The World

- What we are really teaching is a new way of looking at a problem
- See repetition in your work; have fun with it

- Where have you all seen repetition in your work?
  - Find repetition even when it does not look like it; where is repetition hiding?

- Reproducibility / reuse of code

Writing Our Program

- Source code
  - Written in plain text
- Write in
  - Text editor
  - Integrated development environment(IDE)
  - Jupyter Notebook

```
#!/usr/bin/python

“This programs sums numbers given to it”

a = 1
s = 0

print('Enter Numbers to add to the sum.')

print('Enter 0 to quit.')

while a != 0:
    print('Current Sum:', s)
    a = float(input('Number? '))
    s = s + a

print('Total Sum =', s)
```

Writing Our Program

- All most all programming languages share the same concepts
  - Data types
  - Data structures
  - Conditionals
  - Loops
  - Functions
  - Files input/output

Data Types

- Integer
  - 1, 2, 3, 999
- Character
  - “a”, “b”, “c”, “A”, “B”, “C”
- Boolean
  - True, False
- Floating-point number, “Float”
  - 3.14
- String
  - “ATCG”

Data Structures

- Sequence Data Types
  - Array (List)
    - Ordered; duplicates allowed; can change
    - In python - [1, 2, 3], [“A”, “B”, “C”]
  - Tuple
    - Ordered; duplicates allowed; cannot change
    - In python – tuple: (“a”, “t”, “g”, “c”);
- Sets Data Types
  - Sets
    - Unordered; no duplicate elements
    - In python - Set(['John', 'Jane', 'Jack', 'Janice'])
- Mapping Data Types
  - Hash (Dictionary)
    - Key / value pair
    - Unordered; no duplicate keys, can change
    - In python – { “BRCA1”: “cancer gene” }

Variables make use of data types and structures

Variables

- A data item that may take on more than one value during the runtime of a program
- Refer to a place in memory where the value is held
- Example
  - conc = 10

Keywords / Reserved Words (python)

- and
- exec
- not
- assert
- finally
- or
- break
- for
- pass
- class
- from
- print
- continue
- global
- raise
- def
- if
- return
- del
- import
- try
- elif
- in
- while
- else
- is
- with
- except
- lambda
- yield

Operators

Definition

- An object that is capable of manipulating a value or operator.
- For example, in "1 + 2", the "1" and "2" are the operands and the plus symbol is the operator.

Order of Operators

Operator	Name	Example
=	Equals	a = b
==	Equals	a == b
!=	Not equal	a != b
+	Plus	a + b
-	Minus	a - b
/	Divide	a / b
*	Times	a * b

Decision Statements

Definition

Example(s)

```
If conc == 10:
  print(
```

Loops

- For loop…
  - Execute a known number of times
- While loop
  - Execute while some condition is met

Functions

Definition

- A basic task of a computer, especially one that corresponds to a single instruction from the user.
- When an task needs to be performed more then once, it is a good opportunity to convert that code into a function

Example

```
def multiply(num1, num2):
    answer = num1 * num2
    return answer
```

File I/O

- Read a file
  - Text files
  - Binary files
- Write a file
  - Text file
  - Binary file

Sockets / Network I/O

- Read from a socket / network connection
- Write to a socket / network connection

Debugging

- Syntax – wrong grammar, i.e., breaking the rules of how to write the language
  - Forgetting punctuation, misspelling keyword
  - The program will not run at all with syntax errors
- Logic - the program runs, but does not produce the expected results.
  - Using an incorrect formula, incorrect sequence of statements, etc.

How Do We Get Started?

- Create a workflow of what you want your program to do
- Usually call this “pseudocode”
- Pseudocode consists of
  - Clear natural human readable language
  - Three rules
    - sequence
    - choice (if)
    - repeat (while)
- Later we will create pseudocode then convert it to actual code

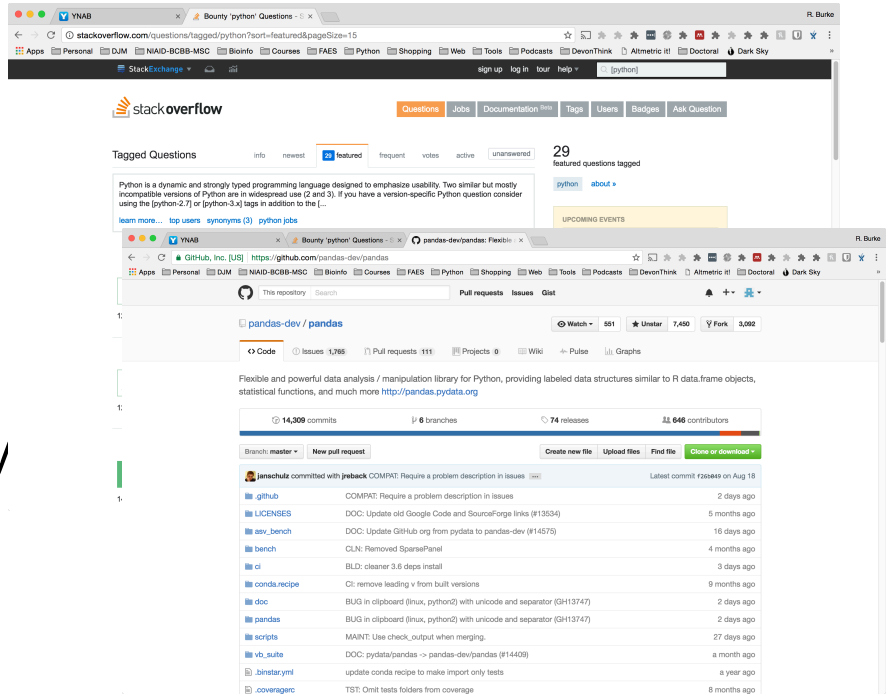
How Do We Convert Pseudocode to an Actual Program?

- Write your own code, all from scratch
- Write your own code but use packages
- Modify existing code
- Search for, adapt code written by some one else

- NOTE: Code is read more often then it is written!

Where To Find Package or Code?

- Your Previous Code
- Other lab members code
- Google Search
- StackOverflow
- GitHub (Open source, source code repository)
- Programing language repository (python – pip, conda; R – cran)



National Institute of Allergy and Infectious Diseases

NIAID

How To Write Your Own Code?

NIH National Institute of Allergy and Infectious Diseases

NIAID

How To Write Your Own Code?

- Picking a Programming language
  - Usually the programming language most comfortable with
  - If you are new to programming, pick one of the languages that we see today
  - Pick the language that best fits the situation

- Process
  - Start with pseudo code
  - Transform pseudo code into real code, one line at a time

Scientific Computing Packages

Python

- Matplotlib - a MATLAB-like plotting library
- Numpy – fast matrix calculations
- Pandas – scientific data analysis
- Plotly, a web-based scientific plotting library
- SageMath is a large mathematical software application for linear algebra, combinatorics, numerical mathematics, calculus, and more
- SciPy - a large library of scientific tools
- Scikit-learn – Machine learning
- Stats-model – Statistical analysis
- Jupyter – Analysis notebook / collaboration
- Conda / bioconda - installers

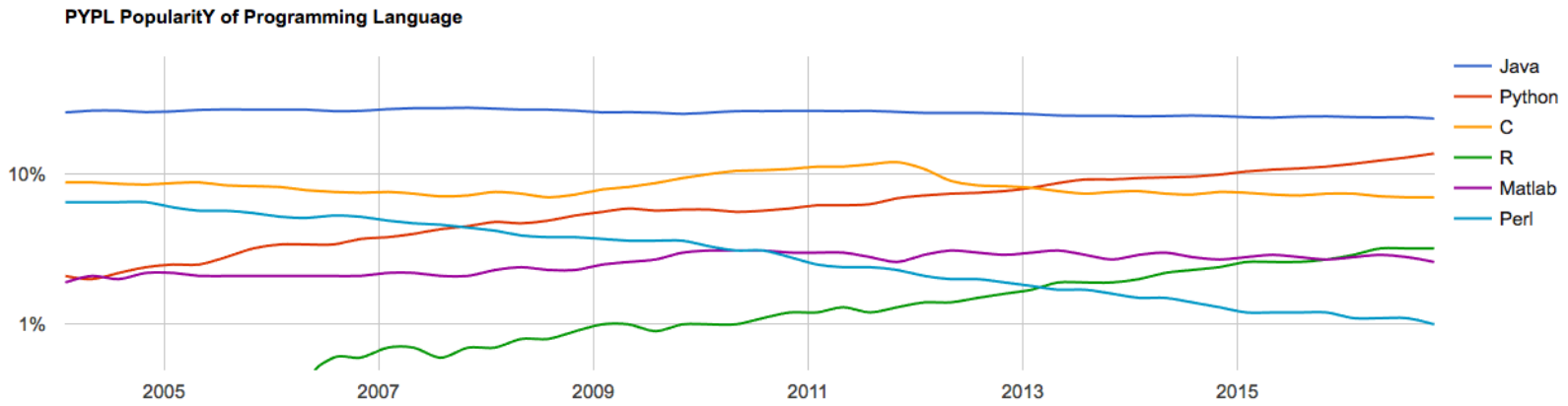
R

- Bioconductor
- dplyrdata wrangling, data analysis
- ggplot2 - data visulization
- plyr - data aggregation
- shiny - turn R data into interactive Web applications

NIH National Institute of Allergy and Infectious Diseases

NIAID

Language Popularity



NIH National Institute of Allergy and Infectious Diseases

NIAID

Printing “Hello World!”

C++

```
#include <iostream>
int main()
{
    std::count << "Hello World” << std::endl;
    return 0;
}
```

Java

```
public class HelloWorld {
    public static void main (String[] args) {
        System.out.println("Hello World!");
    }
}
```

Python

```
print(“Hello World!”)
```

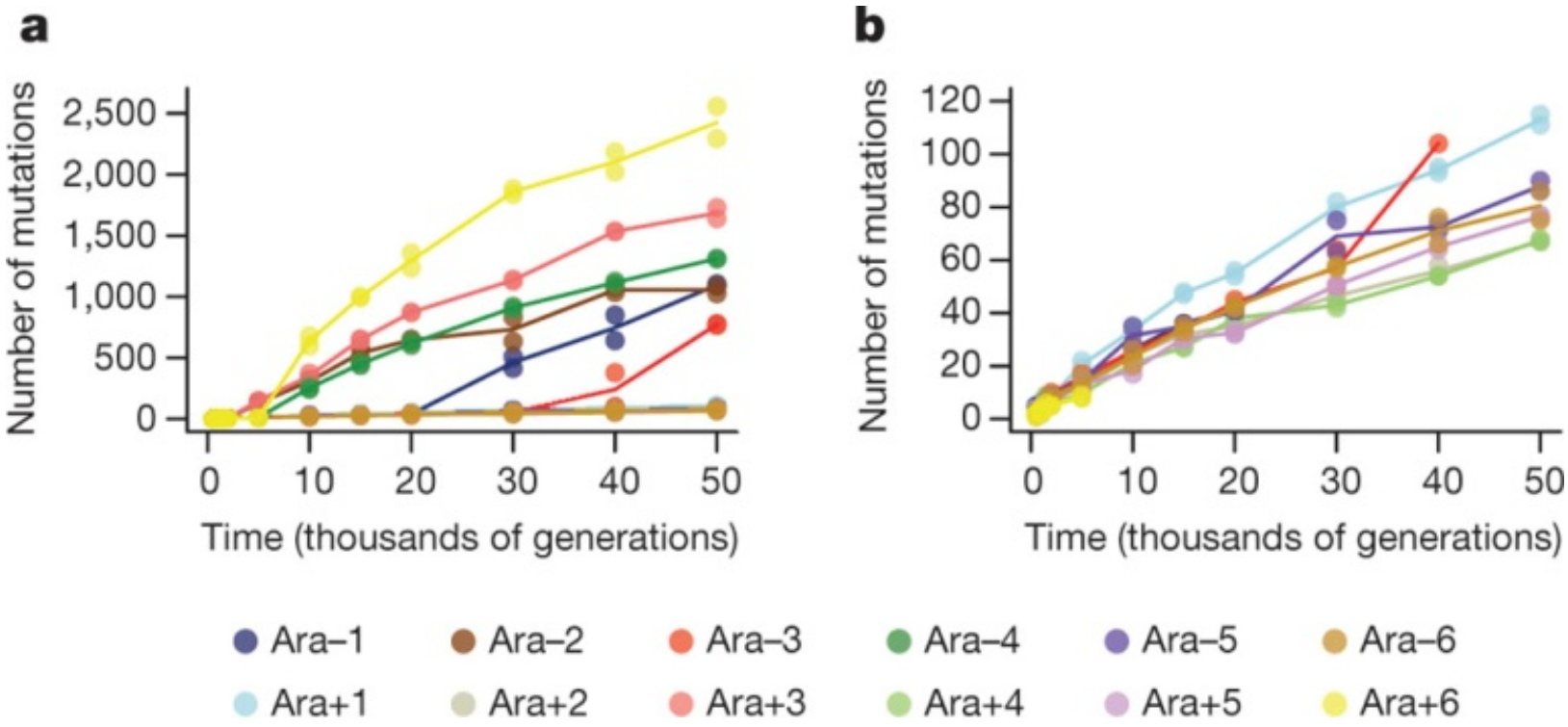
R

```
print("Hello World!", quote = FALSE)
```

Pseudocode Experiment

- Long Term Evolution in E. coli Experiment
- Analyze evolution in E. coli over 50,000+ generations
- Hypothesis
  - All 12 strains will evolve at the same rate
- To test our hypothesis we have sequences many generations and we now need a program to compare the sequences against each

Total number of mutations over time in the 12 LTEE populations



O Tenaillon *et al. Nature* 1–6 (2016) doi:10.1038/nature18959

NIH National Institute of Allergy and Infectious Diseases

nature

NIAID

Pseudocode
What do we want our program to do?

- Assumptions
  - We have a reference sequence (in fasta)
  - We have three sequences for each strain / year (in fasta)
  - Organize by strain name, year
    - Ara-1, 2, 3, 4, 5, 6
    - Ara+1, 2, 3, 4, 5, 6

- Hints
  - The EMBOSS suite has program to compute the differences between two sequences

Pseudocode
What do we want our program to do?

- Get reference sequence

- Loop
  - Read data file(s)
  - Calculate mutations
  - Average among replicates
  - Save data

- Plot data

Pseudocode
What do we want our program to do?

- Define input files and output files
- Use the reference genome to get gene positions and genome sequence
- Use the simplified gene file to compute for each position in the genome a coding status
- First read of data to identify all mutations with a unique tag per population
- Attribute to each mutation a coding status and filter
- Count mutations per categories at any given sampling time
- Print an array for R to build a phylogeny

Advanced Concepts

National Institute of Allergy and Infectious Diseases

NIAID

Advanced Concepts

- Data Structures
- Algorithms
  - Big O Notation
  - P vs NP Problem
  - Dynamic Programming
  - Greedy Programing
- Development Methodologies
  - Waterfall
  - Agile Development
- Parallel Computing
  - Embarrassingly Parallel
  - Graphical Processing Unit (GPU)
  - Computing
- Machine Learning
  - Deep Learning
- DevOps
  - Version Control
  - Continuous Integration
  - Containers (Docker)
  - Cloud Computing
- Other
  - Application program interface (API)?
  - Common Workflow Language (?)

Data Structures

- Array (or list)
- Hash Table (dictionary)
- Heap
- Graph
- Linked List
- List
- Queue
- Stack
- Tree
  - Binary search tree

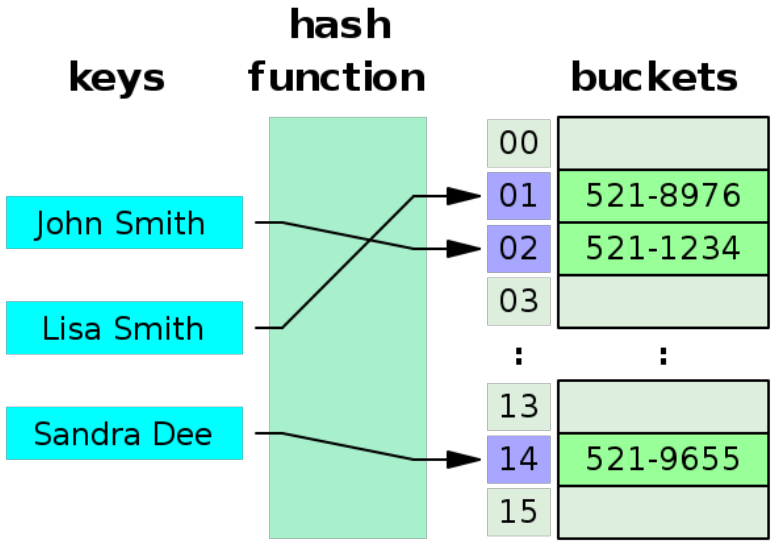
- Table comparing data structures
  - Mutable, immutable
  - Ordered, unordered
  - What they are called in different programming languages
- For animations of data structures, please see here:
- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

Data Structures

Array

- [x, y, z]
- An ordered set of data
- A set of variables that each store an item
- After deleting an element:
  - [x, , z]

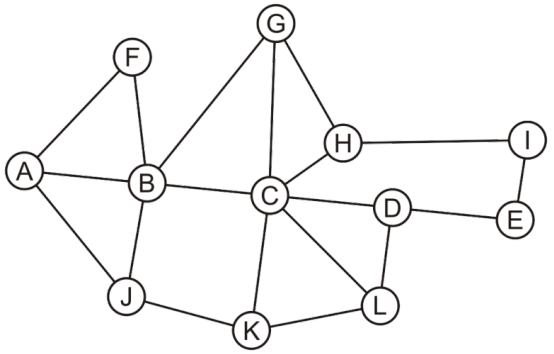
Hash Table



Data Structure Visualizations: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

Data Structures

Graph



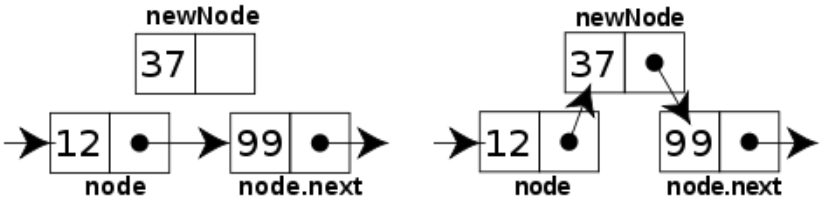
undirected graph

List

- [1, 2, 3, 4, 5, 6]
- An ordered set of data
- A set of items
- After deleting an element:
  - [1, 2, 4, 5, 6]

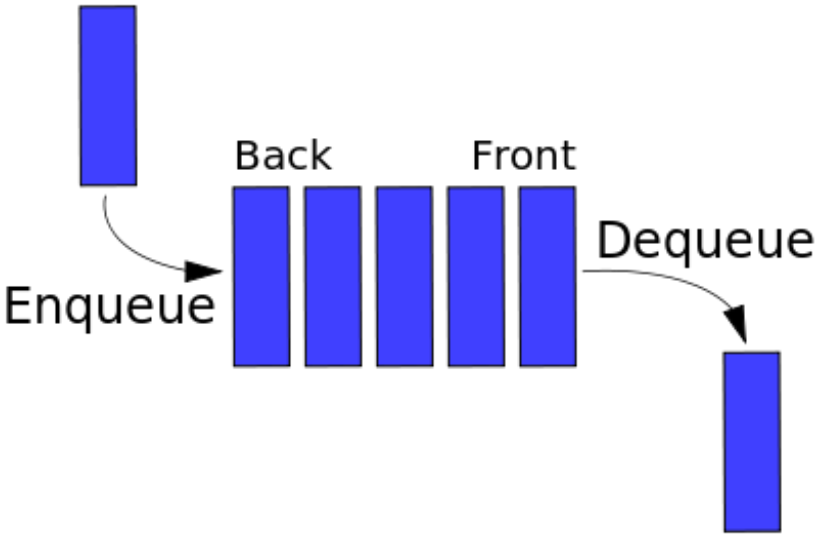
Data Structures

Linked List



Source: https://en.wikipedia.org/wiki/Linked_list

Queue



First in, First Out Queue

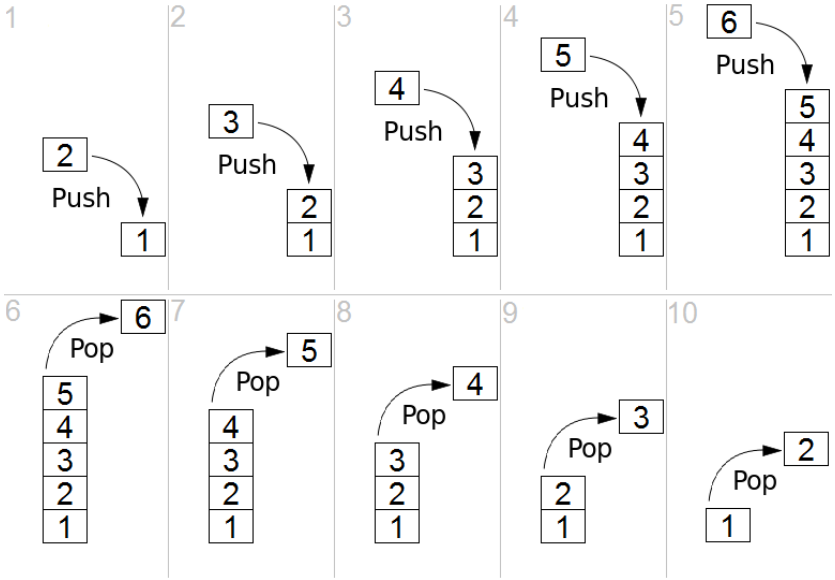
https://en.wikipedia.org/wiki/Queue

NIH National Institute of Allergy and Infectious Diseases

NIAID

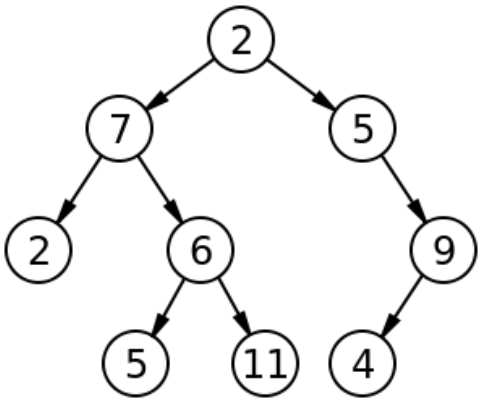
Data Structures

Stack



Last in, First Out

Tree

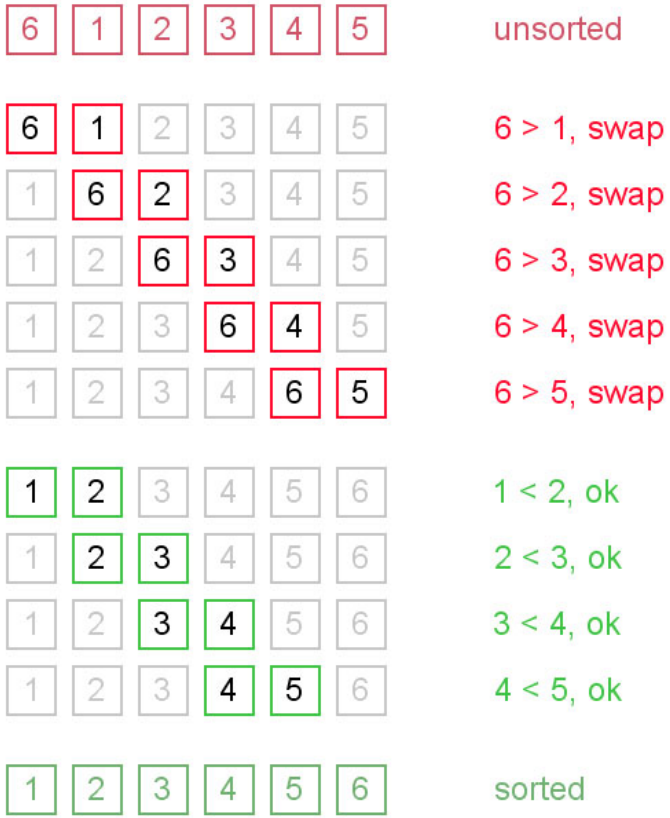


Algorithms

What Is An Algorithm

- Definition
  - Noun
  - A step-by-step procedure for solving a problem or accomplishi some end especially by a comput <a search algorithm> (Merriam-Webster)
  - In other words…a “recipe” or “protocol” for the computer

- Bubble sort algorithm
  - Compare two numbers, if numbe on left is bigger, swap
  - Repeat

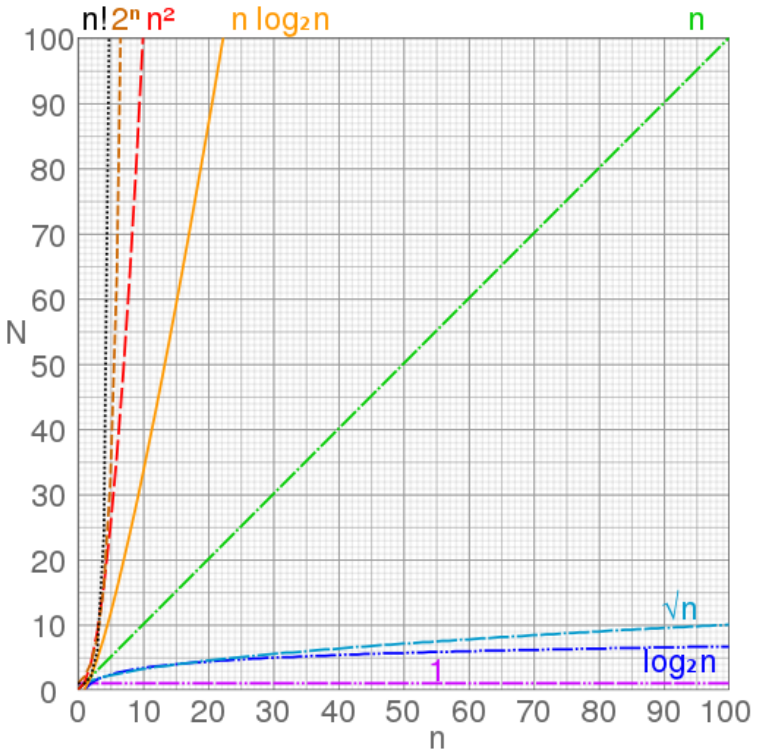


NIH National Institute of Allergy and Infectious Diseases

NIAID

Big O Notation

- Pick a number between 1 and 10
  - I will tell you if the number if too high or two low
  - How many guesses do you need to get the number? Usually 3 – 4 or nearly log(10)
  - Now how many guesses do you think it would take if the options were to a million?
- Big O Notation
  - Approximately how long an algorithm will take to run



National Institute of Allergy and Infectious Diseases

NIAID

Big O Notation

Growth Rate	Name	Code Example	Description
1	Constant	a= b + 1;	statement (one line of code)
log(n)	Logarithmic	while(n>1){ n=n/2; }	Divide in half (binary search)
n	Linear	for(c=0; c<n; c++) { a+=1; }	Loop
n*log(n)	Linearithmic	Mergesort, Quicksort, …	Effective sorting algorithms

n^2	Quadratic	for(c=0; c<n; c++) {
		for(i=0; i<n; i++) {
		a+=1; } }
			Double loop

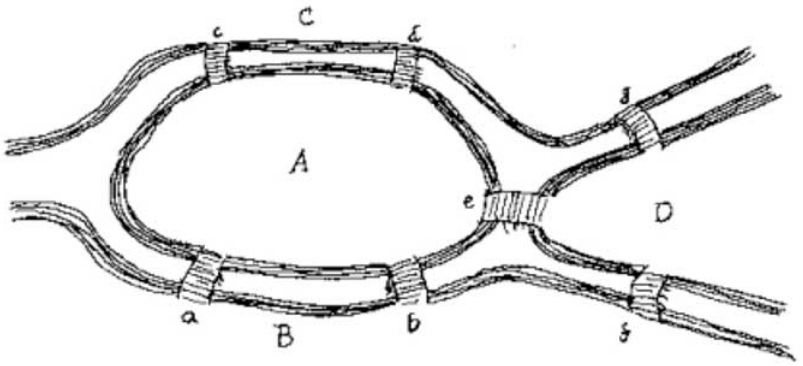
n^3	Cubic	for(c=0; c<n; c++){
		for(i=0; i<n; i++){
		for(x=0; x<n; x++){
		a+=1; } } }
			Triple loop

2^n	Exponential	Trying to break a password generating all possible combinations	Exhaustive search

http://adrianmejia.com/blog/2014/02/13/algorithms-for-dummies-part-1-sorting/

Algorithms
P vs NP Problem

- Polynomial versus nondeterministic polynomial time
- How long it takes to solve a problem versus how long it takes to verify there is a solution
- NP-complete problems
  - Harder to compute than verify
- Example
  - 7 Bridges
  - Euler wondered if a person could walk across each of the seven bridges once and only once to touch every part of the town.



Dynamic Programming

- Method for solving a complex problem by breaking it down into a collection of simpler sub-problems
- Solving each of those sub-problems just once, and storing their solutions
- The next time the same sub-wwwwwwwwwwwwwwwproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution

Dynamic programming matrix:

i		j → (sequence **y**)	0	1 T	2 G	3 C	4 T	5 C	6 G	7 T	8 = N A
0			0	−6	−12	−18	−24	−30	−36	−42	−48
1	T		−6	5	−1	−7	−13	−19	−25	−31	−37
2	T		−12	−1	3	−3	−2	−8	−14	−20	−26
3	C		−18	−7	−3	8	2	3	−3	−9	−15
4	A		−24	−13	−9	2	6	0	1	−5	−4
5	T		−30	−19	−15	−4	7	4	−2	6	0
M = 6	A		−36	−25	−21	−10	1	5	2	0	11

(sequence **x**)

Optimum alignment scores 11:

T	–	–	T	C	A	T	A
T	G	C	T	C	G	T	A
+5	−6	−6	+5	+5	−2	+5	+5

NIH National Institute of Allergy and Infectious Diseases

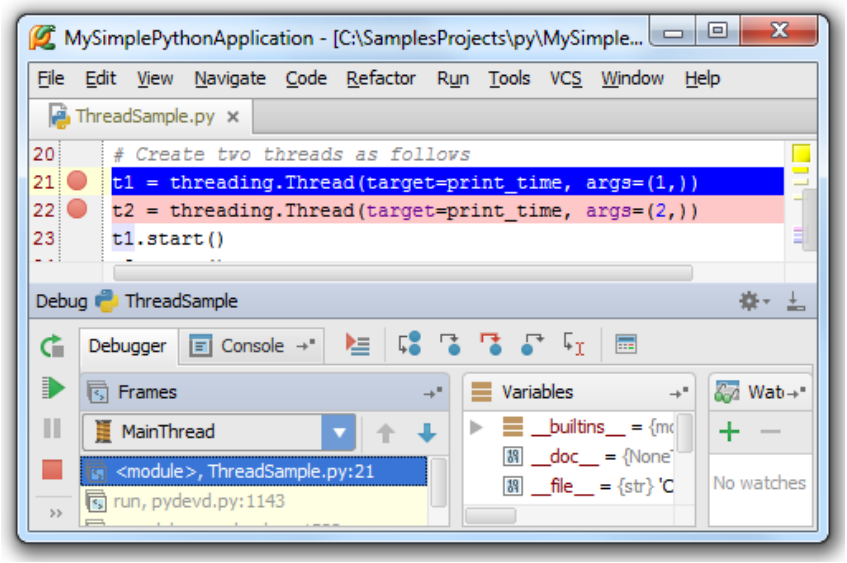
NIAID

Greedy Programing

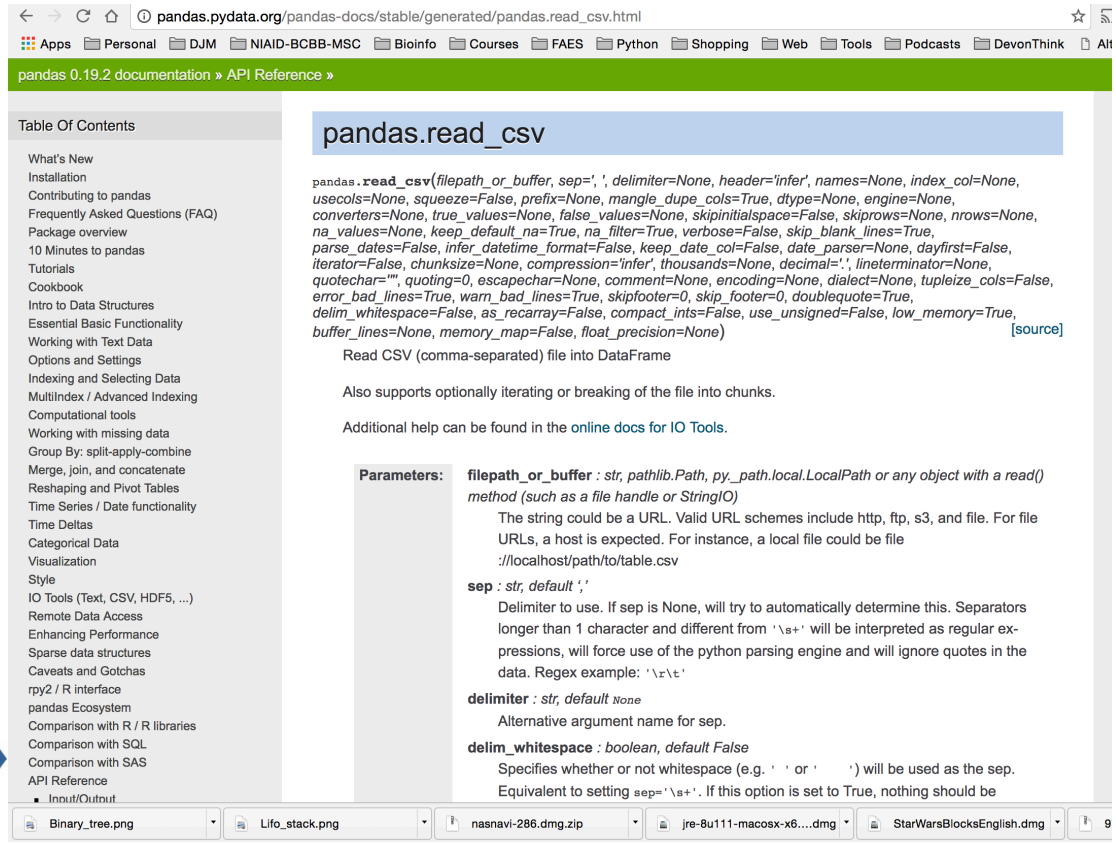
- Problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum

Debugging

- Majority of programming time
  - 75 – 80% of programming
- How
  - Can use `print` statements or
  - Integrated development environment (IDE) software
    - Software to write software
  - Set break-points
  - See what is happening in program during execution



Application Program Interface (API)



NIH

NIAID

Test-driven Development

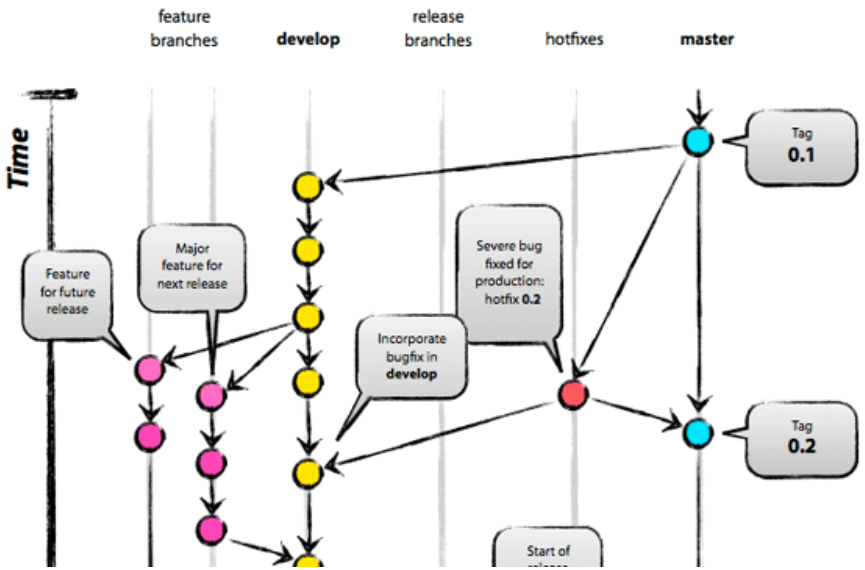
- Testing your software
- How
  - Unit test
  - Assert
- TDD Procedure
  - Write test before writing code
  - Only write enough cost to pass test
  - Write another test
  - Repeat

```
# content of test_sample.py
def func(x):
        return x + 1

def test_answer():
        assert func(3) == 5
```

Version Control

- Keep track of changes (versions)
- Always have a working copy
- Create branches to add or test features
- Merge branches
- Enables multiple programmers to work at the same time



Development Methodologies

Waterfall

- Do all planning
- Write all code
- Perform all tests
- Release product

- Concern
  - Very difficult to do all planning in beginning and anticipate all problems

Agile Development

- Iterate
  - Plan one part
  - Write code to part
  - Test one part
  - Release one part
- Repeat

Parallel Computing

- Breaking a problem into smaller subparts
- Running each smaller part on separate core or computer
- Single data stream
  - SISD, MISD
- Multiple data streams
  - SIMD, MIMD, SPMD, MPMD
- Also
  - Embarrassingly Parallel

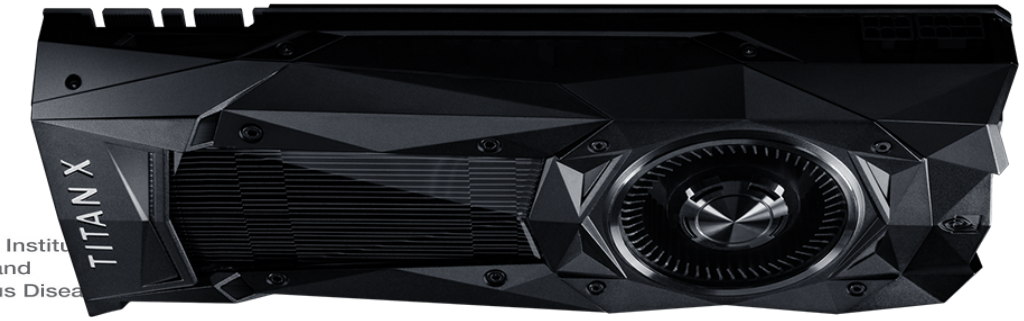
Graphical Processing Unit (GPU)

Central Processing Unit

- Intel Processor – Pentium, i7, Xeon
- “Brain” of laptop and servers
- 2, 4, or more “cores” / little brains

Graphical Processing Unit

- Graphics cards / Video cards
  - Up to 3584 NVIDIA CUDA® Cores
  - Up to 12 GB RAM
- Must write specific code to take advantage of



NIH National Institu Allergy and Infectious Disea

NIAID

Machine Learning

Machine Learning

- “Learning”
- Supervised learning
  - Support vector machines
  - Artificial Neural network
  - Bayesian methods
  - Random Forests
- Unsupervised learning
  - Clustering
  - Dimensionality reduction
  - Structure prediction

Deep Learning

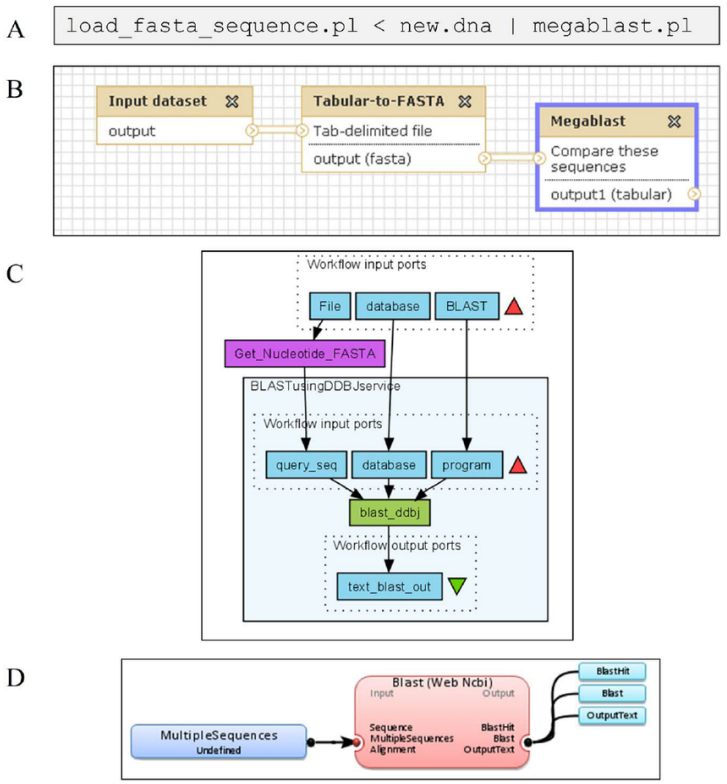
- Uses artificial neural network
  - Multiple hidden layers in network
- Tries to model the way the human brain processes light and sound into vision and hearing
- Needs training
- Has been shown to out perform experts with minimal training

Development - Operations (DevOps)

- Combination of programming (development) and operations
- Operations
  - Hardware, software that are used to run websites, software
- Utilizes
  - Version Control
  - Continuous Integration
    - When new update to software is detected (using version control), automatically runs test, compiles and puts into production
  - Containers (Docker)
    - An abstraction of hardware (software pretending to be hardware)
    - Allows one to download a single “file” with everything needed to run some doftware

Workflow / Pipeline

- The combining of multiple programs to steps into a single process
- Enable
  - Automation
  - Scaling
  - Reusable
  - Reporting
- Common Workflow Language
  - New standard for writing workflow instructions



Armadillo 1.1: An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and SimulationsArticle in PLoS ONE 7(1):e29903 2012

National Institute of Allergy and Infectious Diseases

NIAID

Getting Started

National Institute of Allergy and Infectious Diseases

NIAID

Learning Objectives

- To familiarize you with computer programming, programming languages, scientific packages
- Enable you to communicate better with programmers, bioinformaticians
- Critique existing source code for your use
- Assess which language that bests fits your needs
- Facilitate you becoming a reproducible scientist

Resources

- https://www.manning.com/books/grokking-algorithms
- https://www.toptal.com/developers/sorting-algorithms/

- LTEE
  - http://datadryad.org/resource/doi:10.5061/dryad.6226d