Introductory Computer Science

Week 1 — Introduction: What is Computer Science? Also, Git

Dann Sioson
dj.sioson@alum.utoronto.ca

June 8, 2019

Dedicated To Brandon

Table of Contents

Table of Contents

Introduction

- The content from the material provided is for people with no prior programming experience.
- We will start with Python to not discourage anybody as it tends to be one of the more intro-friendly languages.
- However, the intent of this course is to learn Computer Science using Python and not necessarily the other way around.
- To reinforce this, we may go into Java (depending on progression).
- Feel free to ask questions.

Tentative itinerary

1. Introduction, Git
2. Primitive Types, Strings, Sets, Tuples, Functions Dictionaries, Lists, if-statements, Loops
3. File Reading, Memory Model, and Debugging
4. Recursion*
5. Recursion*
6. Object Oriented Programming*
7. Object Oriented Programming*
8. Design OR Data Structures*

I will try my best to cover topics to your interests
We may bleed into August
* = Not Certain

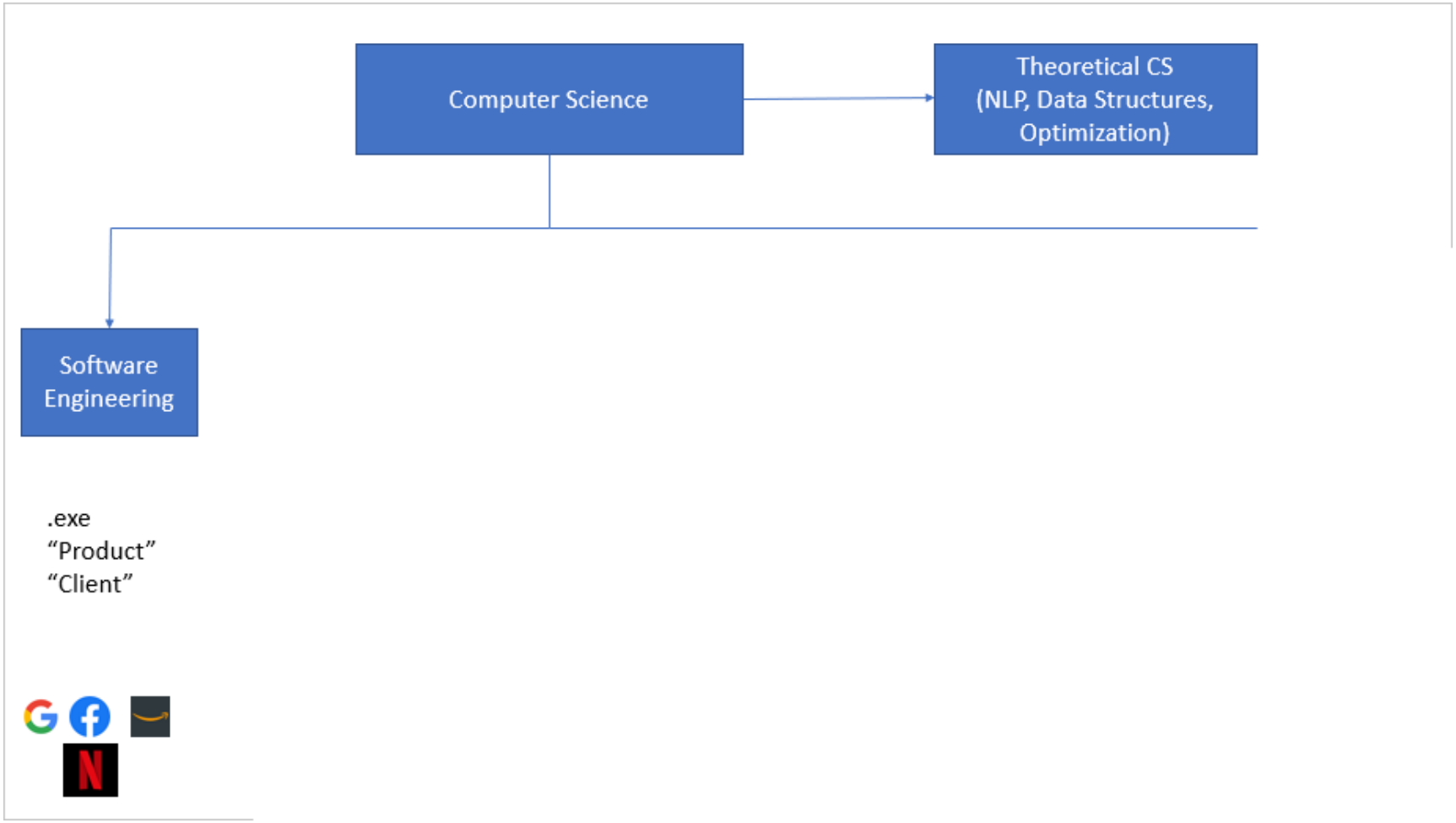
Table of Contents

What is Computer Science?

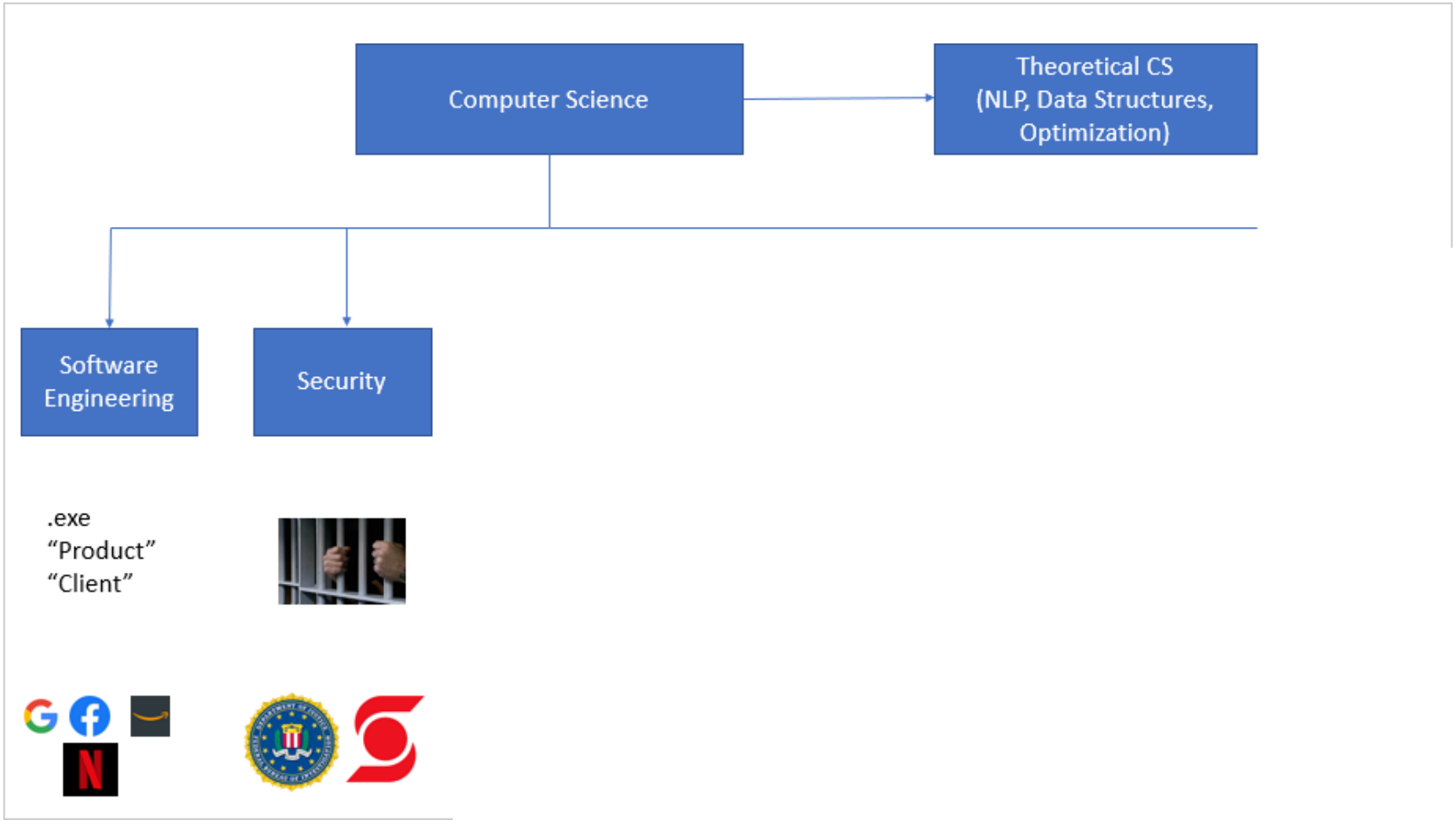
- Not a formal science
- Not necessarily about computers
- The study of algorithms and (virtual) data structures
- More a bit later...

Small overlook of fields in Computer Science

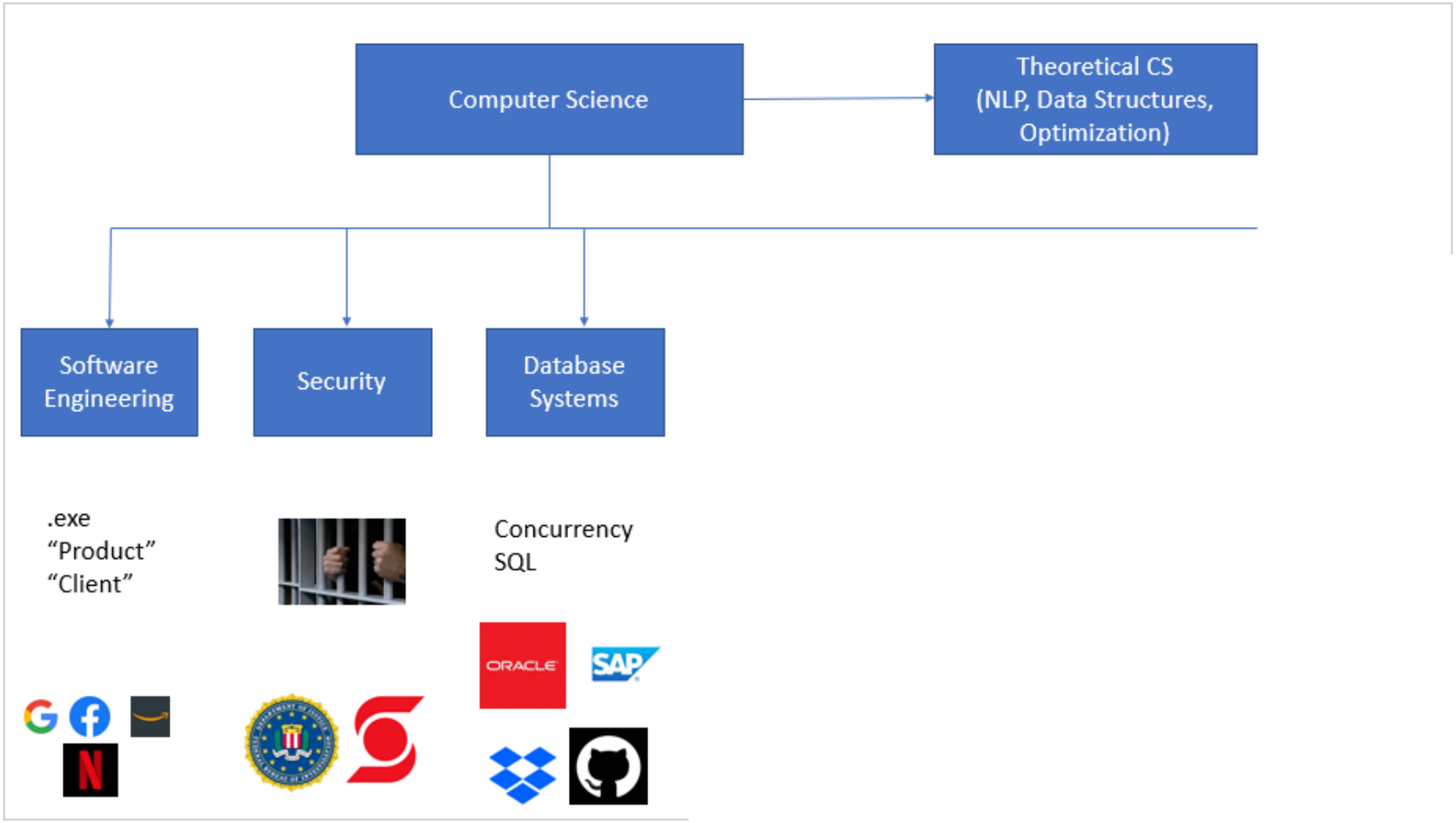
Small overlook of fields in Computer Science



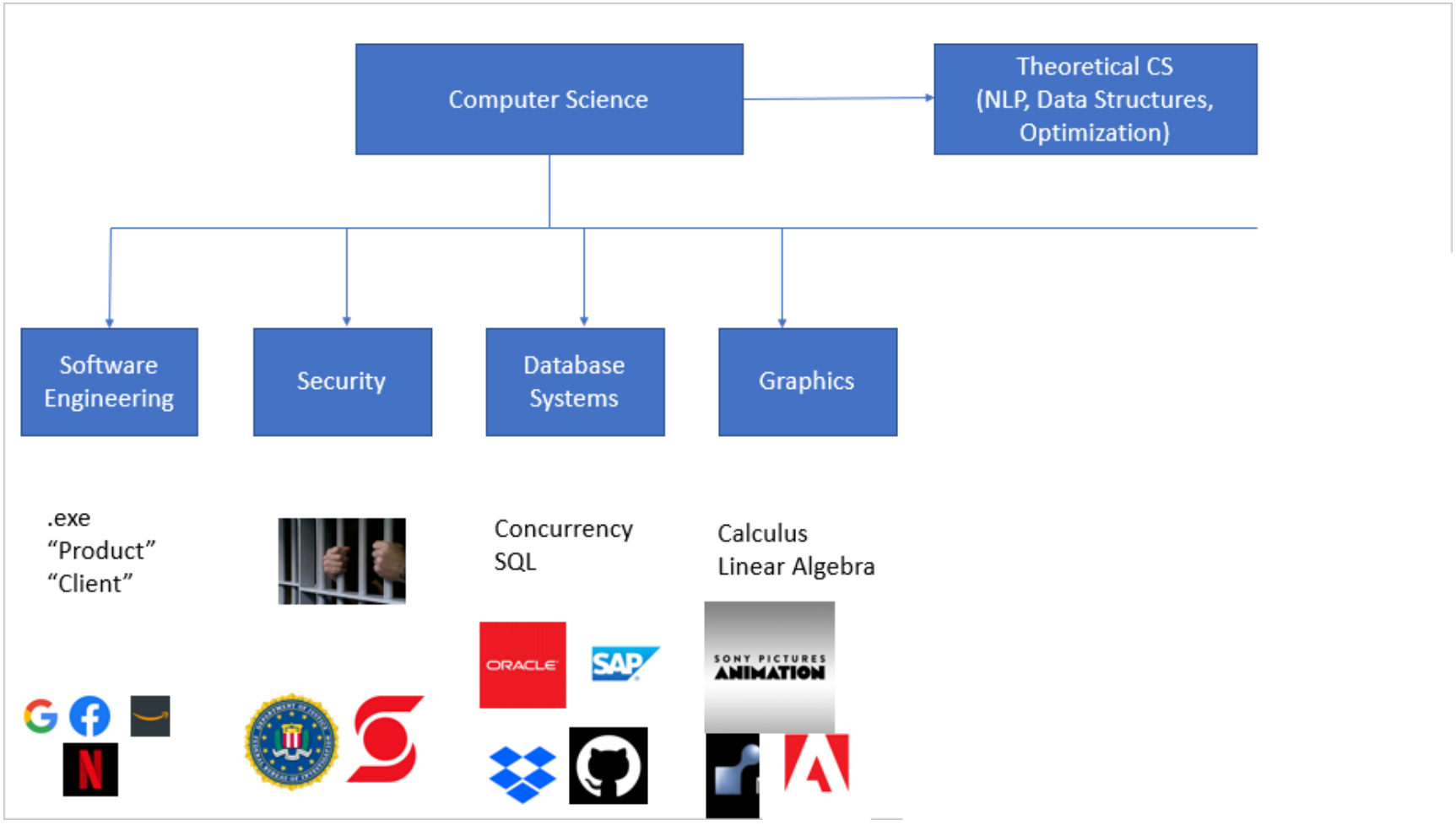
Small overlook of fields in Computer Science



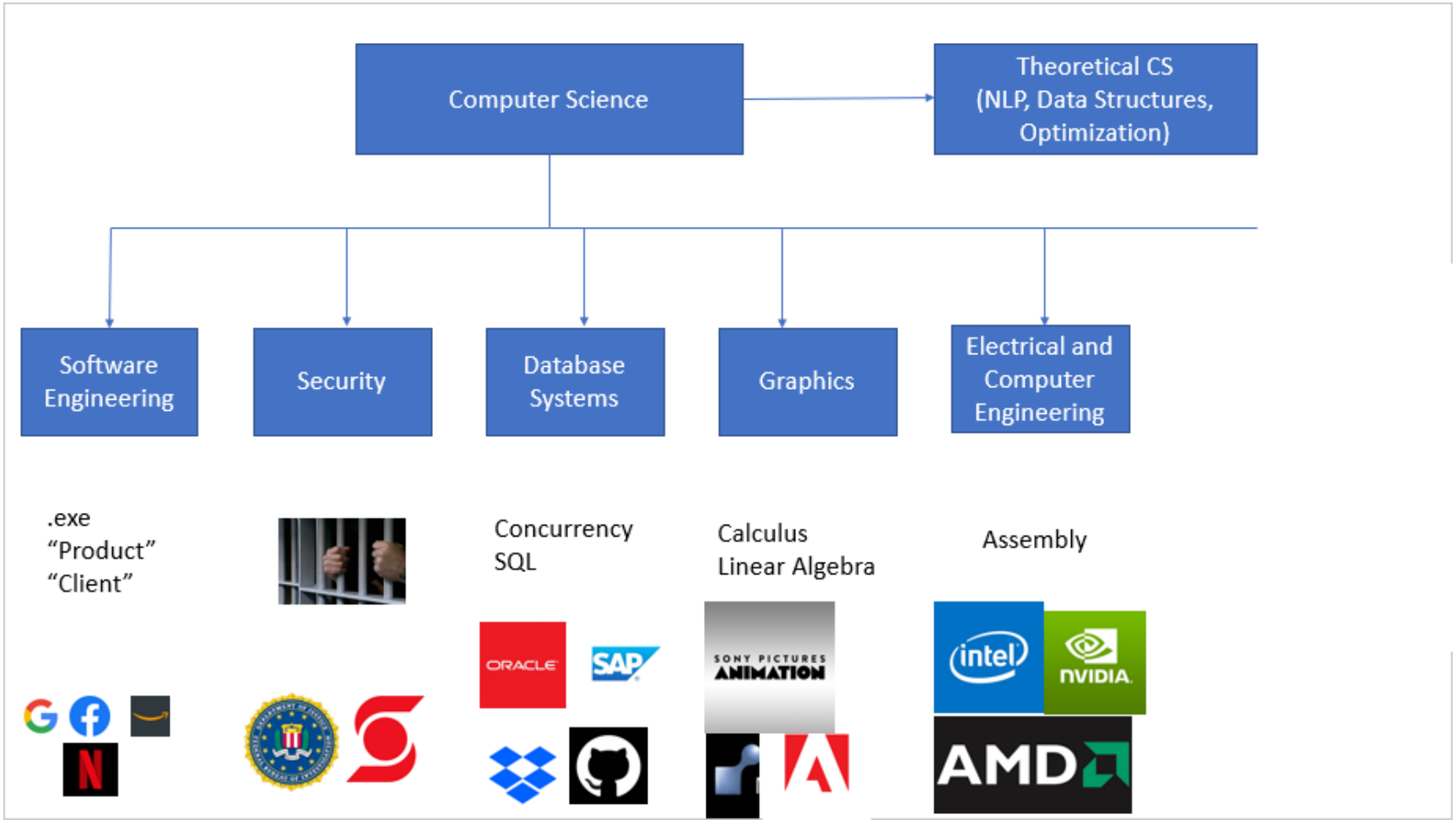
Small overlook of fields in Computer Science



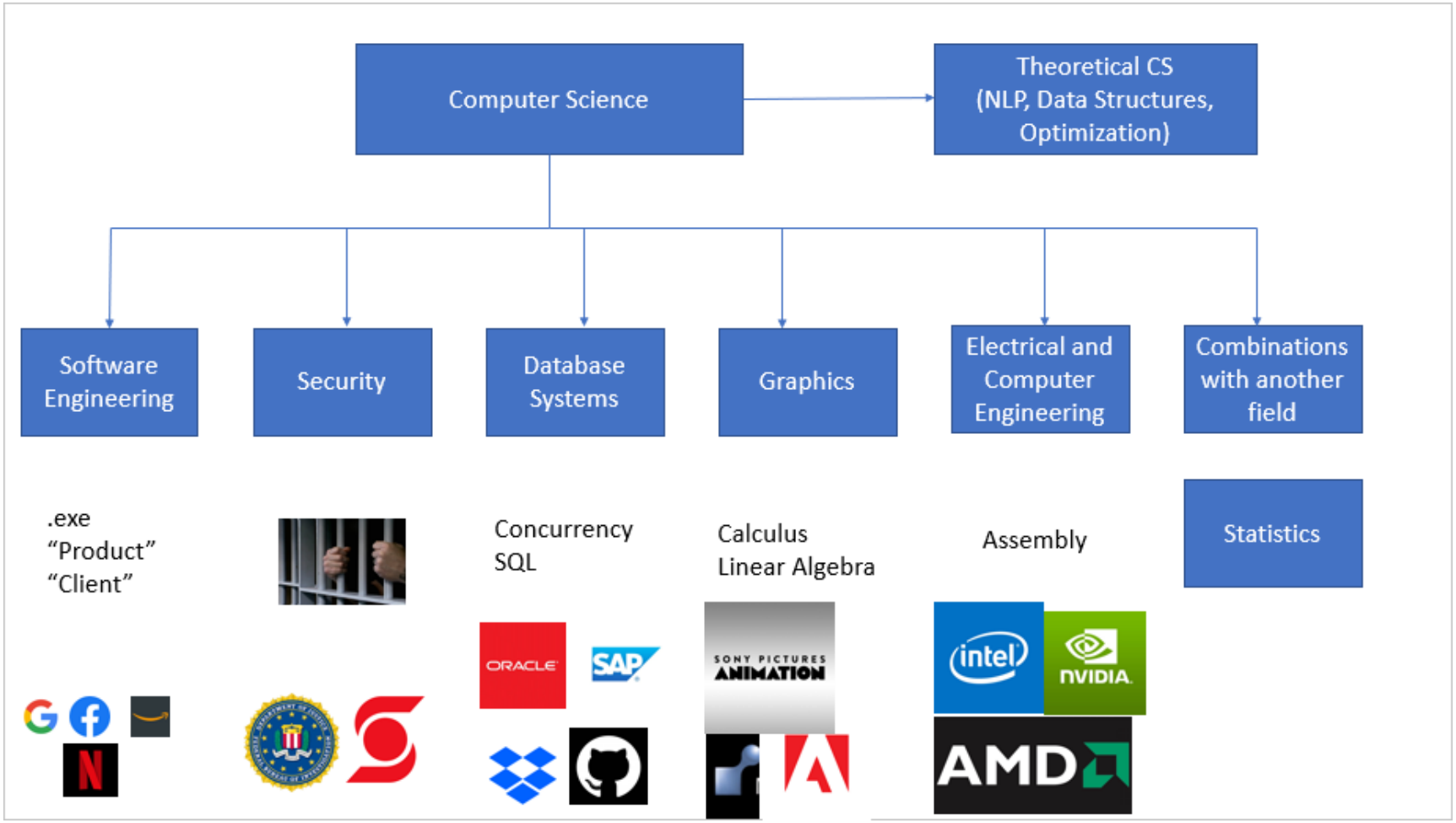
Small overlook of fields in Computer Science



Small overlook of fields in Computer Science



Small overlook of fields in Computer Science



Sorry for the Tangents, let's get started!

Starter pack:

- Git
- GitHub
- Python
- Notepad ++

Table of Contents

Some UNIX commands

- `pwd` # Give me my current path
- `ls` # Show what is in my current path
- `cd some_dir` # Change directory
- `cd ..` # Change directory to parent path

Why Git?

- Typically the industry standard for version control
- If your computer crashes, you're capable of knowing where you left off in your progression
- Something to easily put on your CV
- What we are going to do our work on

Git: Getting ready

- `git clone <some_repository.git>` # Copy a repository to your current path
- `cd <some_repository>`
- `git status` # check status of your repository

Git: Staging files onto GitHub

- `git add file_1.txt`
- `git add file_1.txt file_2.txt` # adding multiple files by space
- `git add *` # add everything in the current path
- `git add *.txt` # add all text files in the current path

Git: Committing

Commits must have meaningful messages pertaining to what you did to your code. Particularly for any changes after you stage your files:

- `git commit -m "Believed completion of flargenschpoodle class"`

Git: Pushing

Put your changes onto GitHub

- `git push`

Vim

- A lightweight text editor
- Not initially “user friendly”
- Will be used to deal with conflicts (unless you configure it)

Vim: Getting in and out

- `vim file.txt`
- i = INSERT MODE
- <ESC> :wq = “write, then quit”

Conflicts: Prepending and Appending

- Open the `all/git_practice/practice_file.txt` file in Notepad++
- One person writes something above the existing written line, another person writes something below the existing written line
- Both commit and push

Conflicts: Realistic conflicts

- Update your local repository: `git pull`
- Open the `all/git_practice/practice_file.txt` file in Notepad++
- Both edit the existing line
- Both attempt to commit and push

Branching: Getting Started

Allows features to be worked on, on a modular level. Think of it as a timeline. Branching allows contributors of the repository to contribute without messing around with the live product

- `git branch <BRANCH NAME> # Create a branch`
- `git branch -a # View all branches`
- `git checkout <BRANCH NAME> # go to branch in the repository`
- `git push --set-upstream origin <BRANCH NAME> # put the branch onto GitHub`

Branching: Merging

All commits are not recorded until you merge it into the master branch. However, a dev-branch is made so that no errors occur on the live product

- `[dev-branch] $ git merge <YOUR BRANCH NAME> # Vim may pop up, and may ask you to clean any conflicts`
- `[dev-branch] $ git checkout master`
- `[master] $ git merge dev-branch`

Table of Contents

Going back... What is Computer Science?

- Not a formal science
- Not necessarily about computers
- The study of algorithms and data structures

What is an Algorithm?

- A set of instructions
- Broken up atomically
- Put together in sequential order

Algorithms

Think of Algorithms analogously to...

- Cooking Recipes
- GPS directions

Machines Are...

- Used to automate repetitive tasks that can make a person insane
- They can follow simple directions
- Can only follow specific instructions, not necessarily what Siri or Alexa could do
- Example: "Get dressed"
- "First put on your undies. Okay Second, put on your under shirt. Okay third, put on your pants. Okay, and finally, put on your shirt"
- Computer Science can be thought of turning low granular ideas (that tend to be ambiguous) into high granularity algorithms (which tends to be straight-forward)

General Theme in Computer Science

- You will be given many problems that seem to be large and overwhelming
- However, an objective skill to develop is to break down the problem into smaller parts
- Making it easy for yourself to understand

Now, if time permits, let's get started on Python!