Lab Review

Lab 01 and 02

Lab 01

- Command Line Intro
- UNIX
- File Systems
- gedit

Command Line Pro’s & Con’s

Advantages

- Once you learn it, it’s faster
- Once you learn it, it’s “easier”
- No carpel tunnel syndrome
- Standardized – available on all UNIX

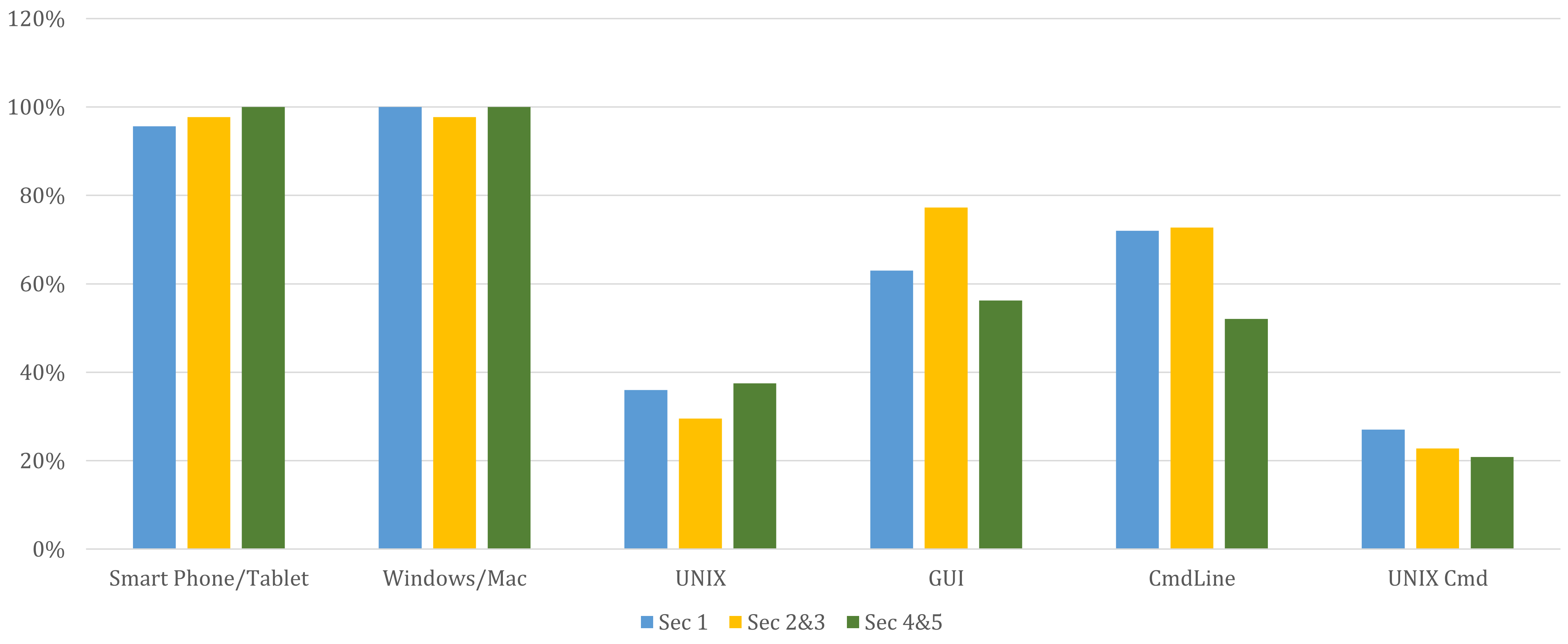
Disadvantages

- You need to learn a lot before you’re good at it
- If you make a mistake, you have to retype
- It’s easier to make a mistake
- Have to RTFM
- GUI is not standard in UNIX

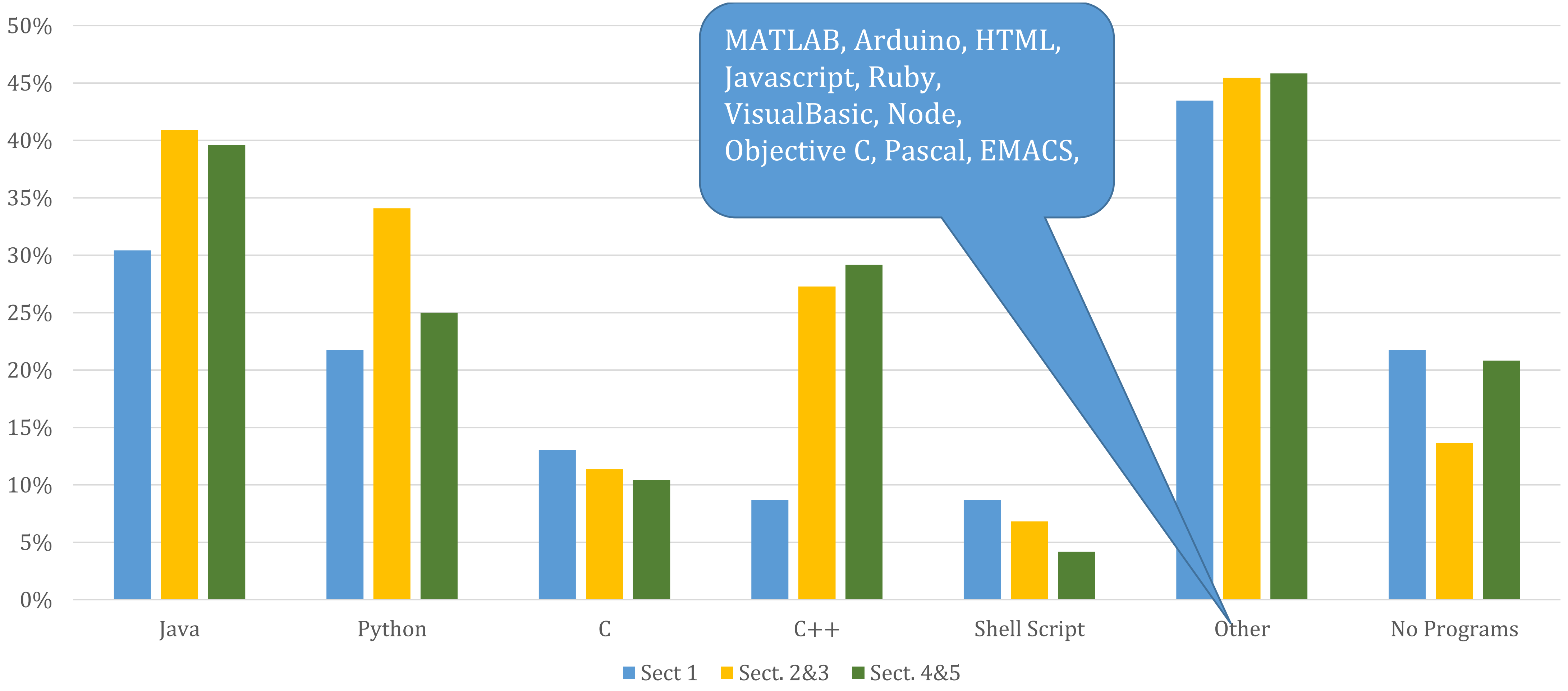
rmdir fails if directory is not empty

- Good: Protects you from deleting something you wanted.
- Bad: Requires extra commands if you want to delete directory AND contents
  - Allternative: “rm –r junk”

Class Computer Usage



Programming Experience



Programming Applications - School

- Engineering
  - calculate how much material is needed to support a given weight.
- Mathematics
  - solve math problems. You would put in the problems and the computer program would output the answer.
  - My differential equations homework.
- Science (Chemistry/Physics/Biology)
  - you have to use your data to calculate comparable answers-plugging in several different "experimental tries" into the same mathematical equations. A computer program could alculate your results for you.
- General
  - enter my grades for each class and then output the GPA instead of having to use a calculator

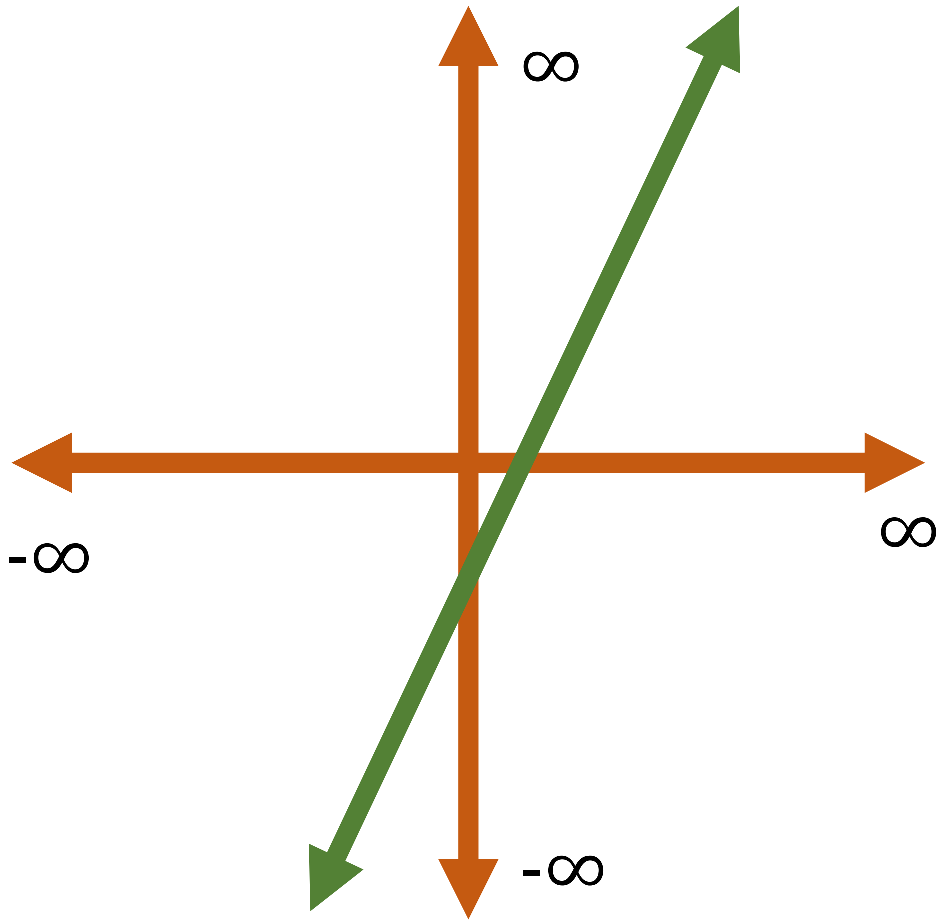
Programming Applications - Other

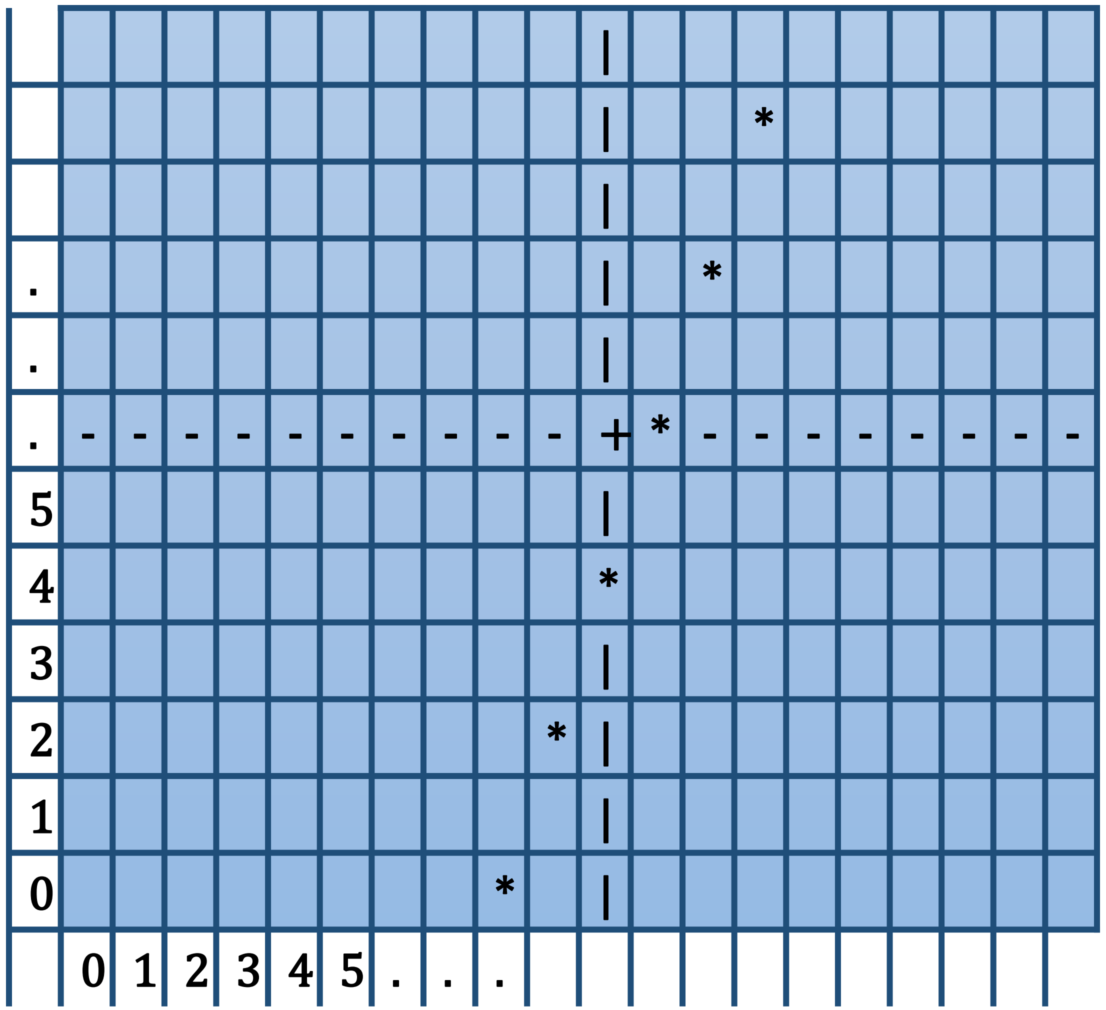
- if I want the browser window to play from my speakers and another program to play audio from my headphones. Input data would be the programs and the audio outputs I would like to assign them to.
- I hate shopping, so I wish my computer could do it for me…
- My schedule for lifeguarding was not the most fairly organized. There are multiple guards on duty at a time and many different chairs to sit in. …There could be a computer program designed to find an optimal strategy to organize the guards.

Lab 2 – Calling Functions

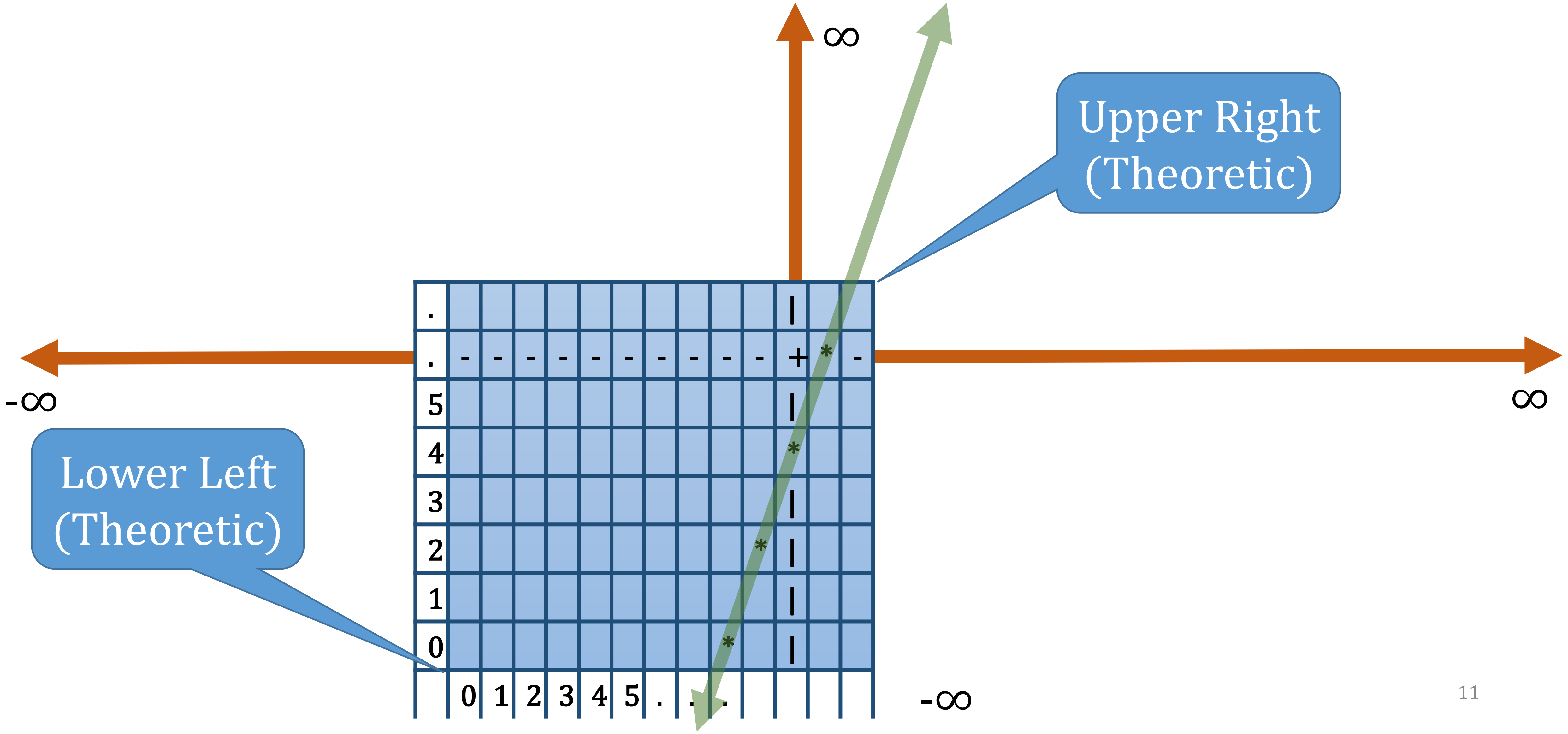
- Objective: Get some experience dealing with C function calls
- Objective: Write some simple code
- Objective: Introduce functions treated as variables
- Objective: Introduce #define
- Objective: Introduce some of the trade-offs in writing application code

Theoretic vs. Concrete Coordinates





Theoretic vs. Concrete Coordinates



Why are ROWS/COLS #define'd?

- ROWS/COLS define the CONCRETE output size
- The assumption is that now matter how we scale, we want a fixed output.
- Like choosing paper size for a printer
- Choose ROWS/COLS to fit your screen
- #define allows a single change to propagate throughout the code
  - Change once… not many places
  - Less error prone… can't forget to change some instances of 20 to 25

Q4: Why are ll/ur parameters?

- Enable programmer to choose the frame that maps to the concrete display.
- Allows things like zoom/pan

Q1: Will Float Improve the Graph?

- No… Float works in the theoretic world, but does not change the concrete!
- As designed, graphFn maps 1 concrete unit (character) to 1 or more theoretic units

```
int x_unit=(tx-fx)/COLS;
if (x_unit==0) x_unit=1;
```

- Even with more precise theoretical units, concrete units remain integers!

UNLESS and Until…

Enable small (float) concrete units!

- Allow x_unit and y_unit to be float
- Enables x_unit and y_unit less than 1.
- See Examples, lab2