[320] Welcome + First Lecture [reproducibility]
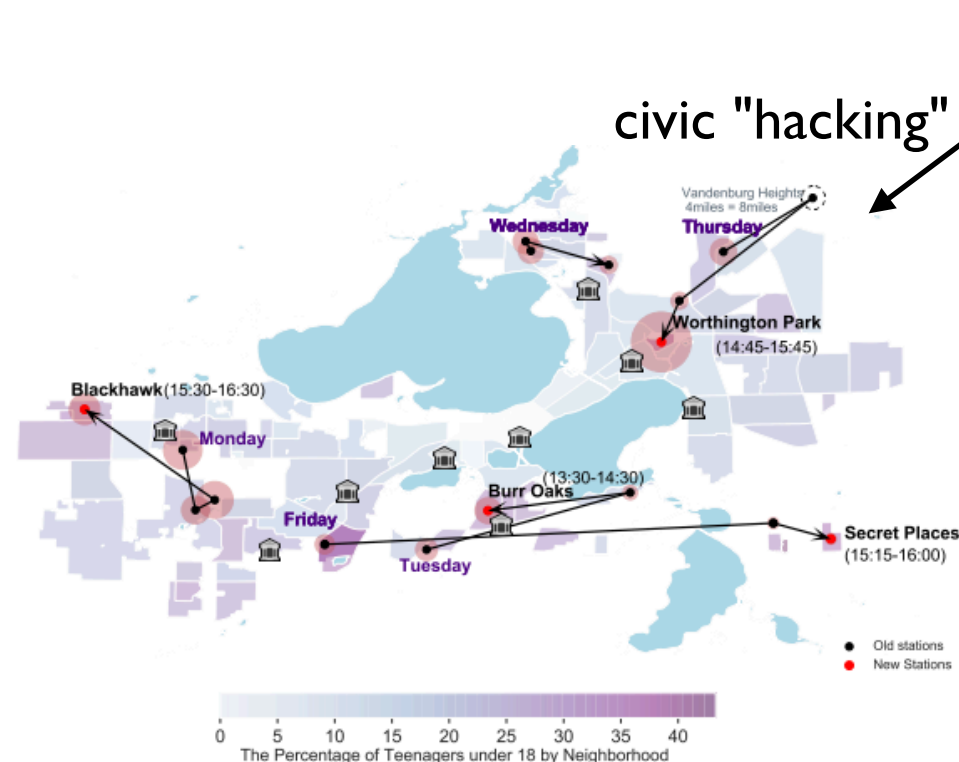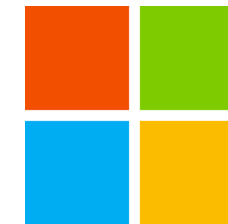
Tyler Caraza-Harter

Introductions

Tyler Caraza-Harter
- Long time Badger
- Email: tharter@wisc.edu
- Just call me "Tyler" (he/him)

Industry experience
- Worked at Microsoft on SQL Server and Cloud
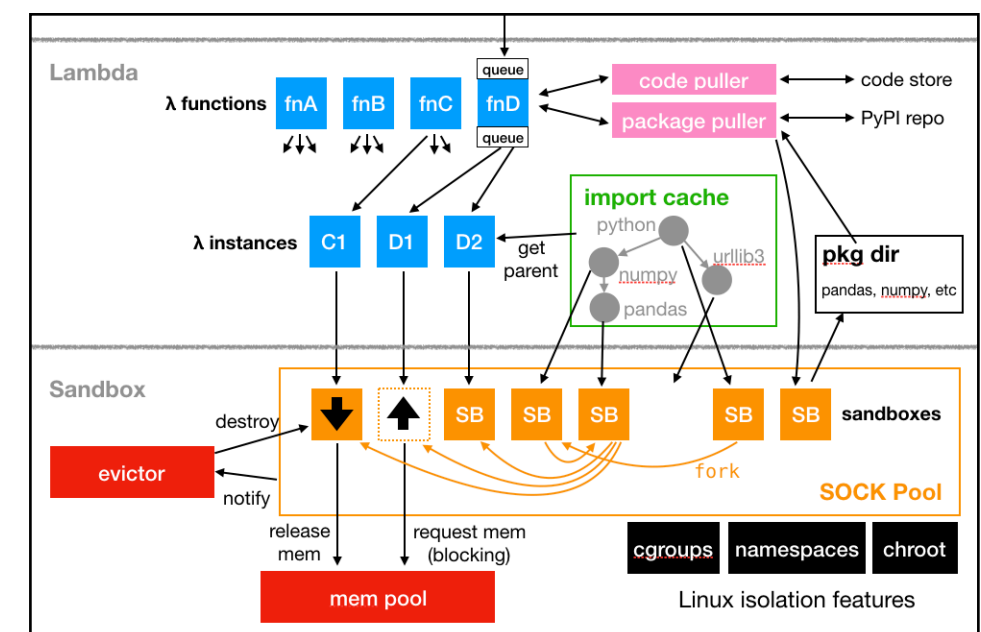- Other internships/collaborations: Qualcomm, Google, Facebook, Tintri

interests

civic "hacking"

Plot by Zishan Bai & Dingyi Zhou (previous students)

More: https://wisc-ds-projects.github.io/f20/

OpenLambda

Who are You?

Year in school?
- 1st year? 2nd? Junior/senior? Grad student?

Area of study
- Natural science, social science, engineering, business, statistics, data science, other?
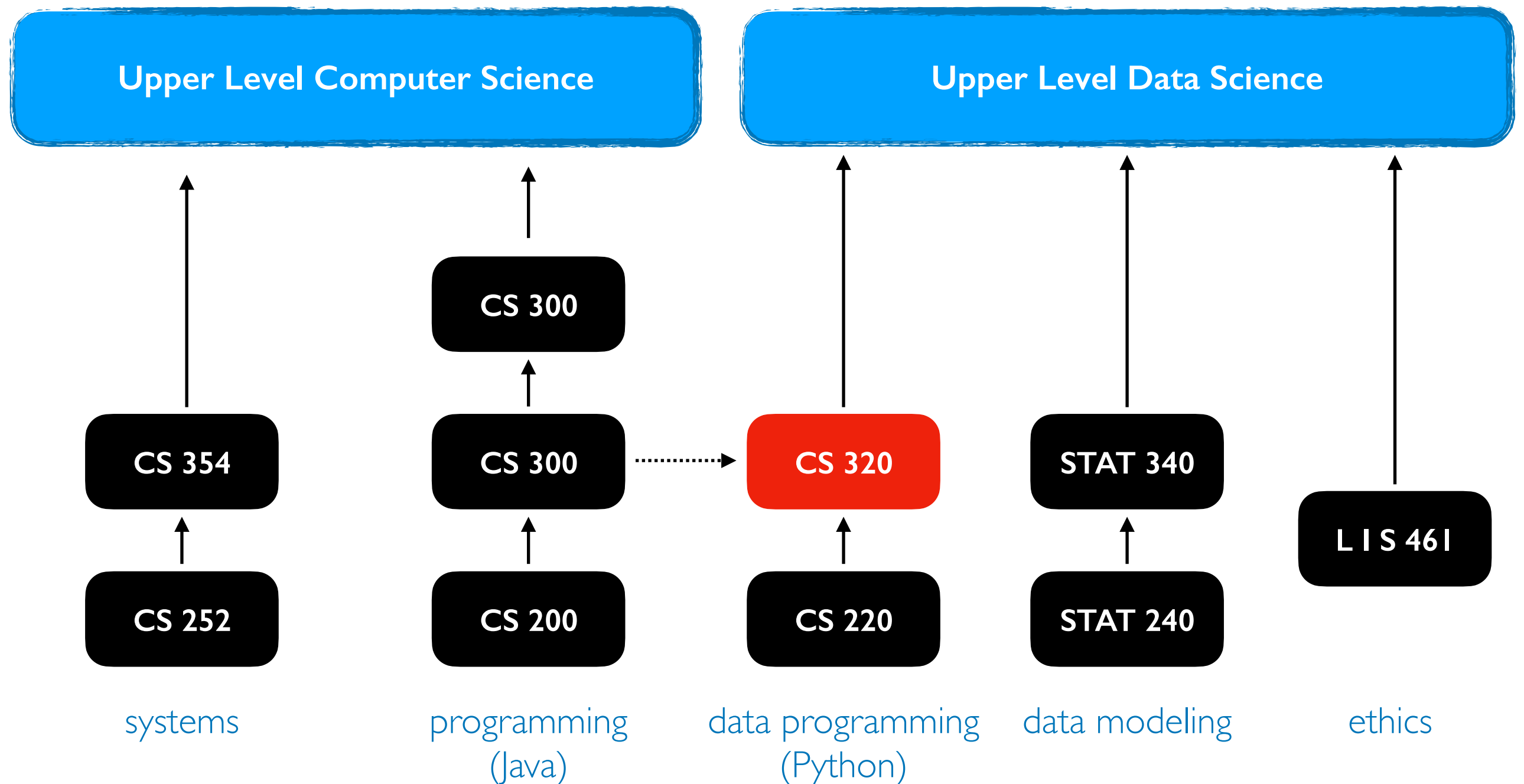
What CS courses have people taken before?
- CS 220/301 (the import one here)? CS 200? CS 300? CS 354?

Please fill this form: https://forms.gle/LsApPYT5PWa1NBNF6. Why?
- Help me get to know you
- Get participation credit

Related courses

Upper Level Computer Science

Upper Level Data Science

CS 300

CS 354   **CS 300** ⇢ **CS 320**   **STAT 340**

CS 252   **CS 200**   **CS 220**   **STAT 240**   **L I S 461**

systems   programming (Java)   data programming (Python)   data modeling   ethics

P1 (Project 1) and other resources will help 320-to-300 students.

Welcome to Data Programming II, in person!

Builds on CS ~~301~~ 220. https://stat.wisc.edu/undergraduate-data-science-studies/

CS 220	CS 320
getting results	getting **reproducible** results
writing correct code	writing **efficient** code
using objects	designing **new types** of objects
functions: `f(obj)`	**methods**: `obj.f()`
lists+dicts	graphs+trees
analyzing datasets	**collecting**+analyzing datasets
plots	animated visualizations
tabular analysis	**simple machine learning**

CS 301 content (for review): https://tyler.caraza-harter.com/cs301/fall19/schedule.html

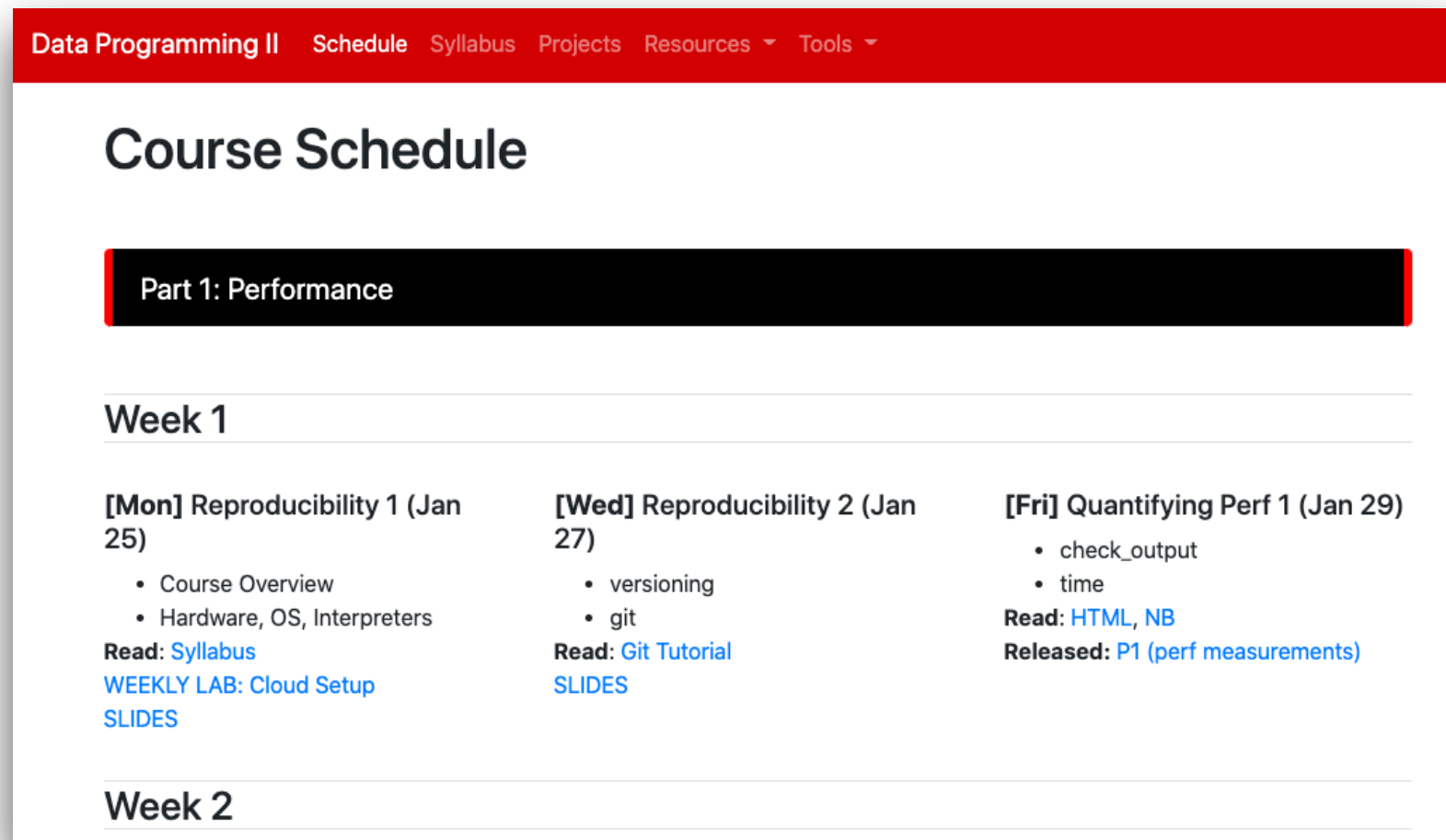# Course Logistics

Scheduled Activities

Lectures

- 3 times weekly

- feel free to bring a laptop

- will generally be recorded+posted online
  (questions will be recorded -- feel free to save until after if you aren't comfortable being recorded)

- Fill "Reflect on Lecture" each time:
  https://tyler.caraza-harter.com/cs320/f21/surveys.html

Lab

- Weekly on Mondays

- Bring a laptop

- Mostly self guided (320 staff will be there to answer questions)

- Focus is on project prep

- Feel free to use extra time to collaborate with team

Course Website

It's here: https://tyler.caraza-harter.com/cs320/f21/schedule.html

read syllabus carefully
and checkout other content

I'll also use Canvas for four things:
- general announcements
- quizzes
- office hours
- simple grade summaries (not feedback or exam answers)

Class organization: People

Teams

- you'll be randomly assigned to a team of 4-7 students
- teams will last the whole semester
- some types of collaboration with team members are allowed (not required) on graded work, such as projects+quizzes
- most collaboration with non-team members in not allowed

Staff

1. Instructor
2. Teaching Assistants (grad students)
3. Mentors (undergrads)

we all provide office hours, and you can attend any that you prefer!

Class organization: People

Teams

- you'll be randomly assigned to a team of 4-7 students

- teams will last the whole semester

- some types of collaboration with team members are allowed (not required) on graded work, such as projects+quizzes

- most collaboration with non-team members in not allowed

Staff

1. Instructor

2. Teaching Assistants

   head TA: in charge of projects
   team TA: primary contact for team, same whole semester
   grader TA: reviews projects (rotates weekly)

3. Mentors

   team mentor: meets weekly with your team, same whole semester

we all provide office hours, and you can attend any that you prefer!

Other Communication

Piazza

- find link on site
- don't post >5 lines of project-related code (considered cheating)
- pinned post will list office hours (me, TAs, mentors)

Forms

- https://tyler.caraza-harter.com/cs320/f21/surveys.html
- Who are you? Feedback Form. Thank you!
  Reflect on Lecture. Grading Issues.

Email

- me: tharter@wisc.edu
- TAs: https://tyler.caraz-harter.com/cs320/f21/contact.html

Course Etiquette

Meetings

1. office hours are drop-in (no need to reserve)
2. email me to schedule individual meetings

Email

3. let us know your NetID (if not from netid@wisc.edu)
4. don't start new email thread if topic is the same
5. CC team members when appropriate
6. unless urgent, please give me 48 hours to respond before following up (I'll try to be faster usually)
7. use your judgement about whether to email me or TA first
8. if general question, consider using piazza instead

Graded Work: Projects+Participation

Participation - **3%**

- class surveys
- project specification fixes
- timely reporting of grading issues

7 Projects - **8%** each

- **format**: notebook, module, or program
- part 1: you can optionally collaborate with team
- part 2: must be individualy (only help from 320 staff)
- still a `tester.py`, but more depends on TA evaluation (more plots)
- ask for specific feedback (giving constructive criticism is a priority in CS 320)

Time Commitment
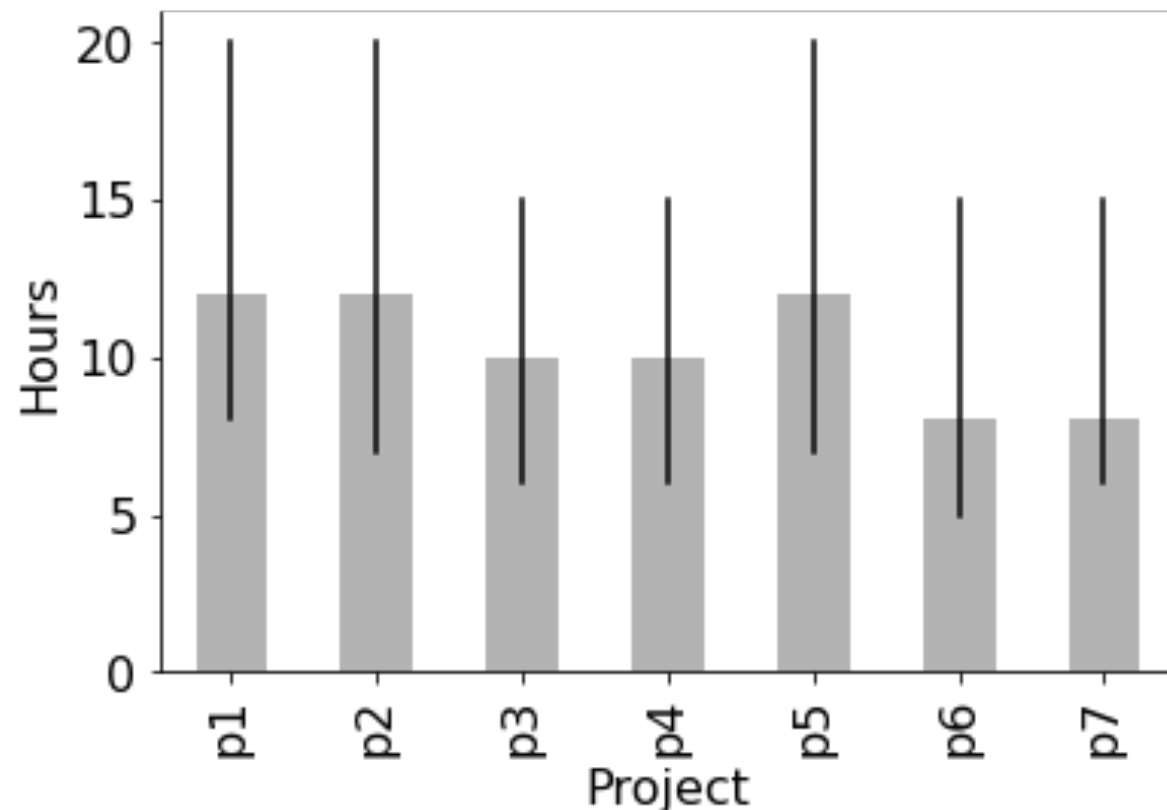
Observations
- 10-12 hours per project is typical
- 20% of students sometimes spend 20+ hours on some
- students who were faster earlier were less likely to complete the course
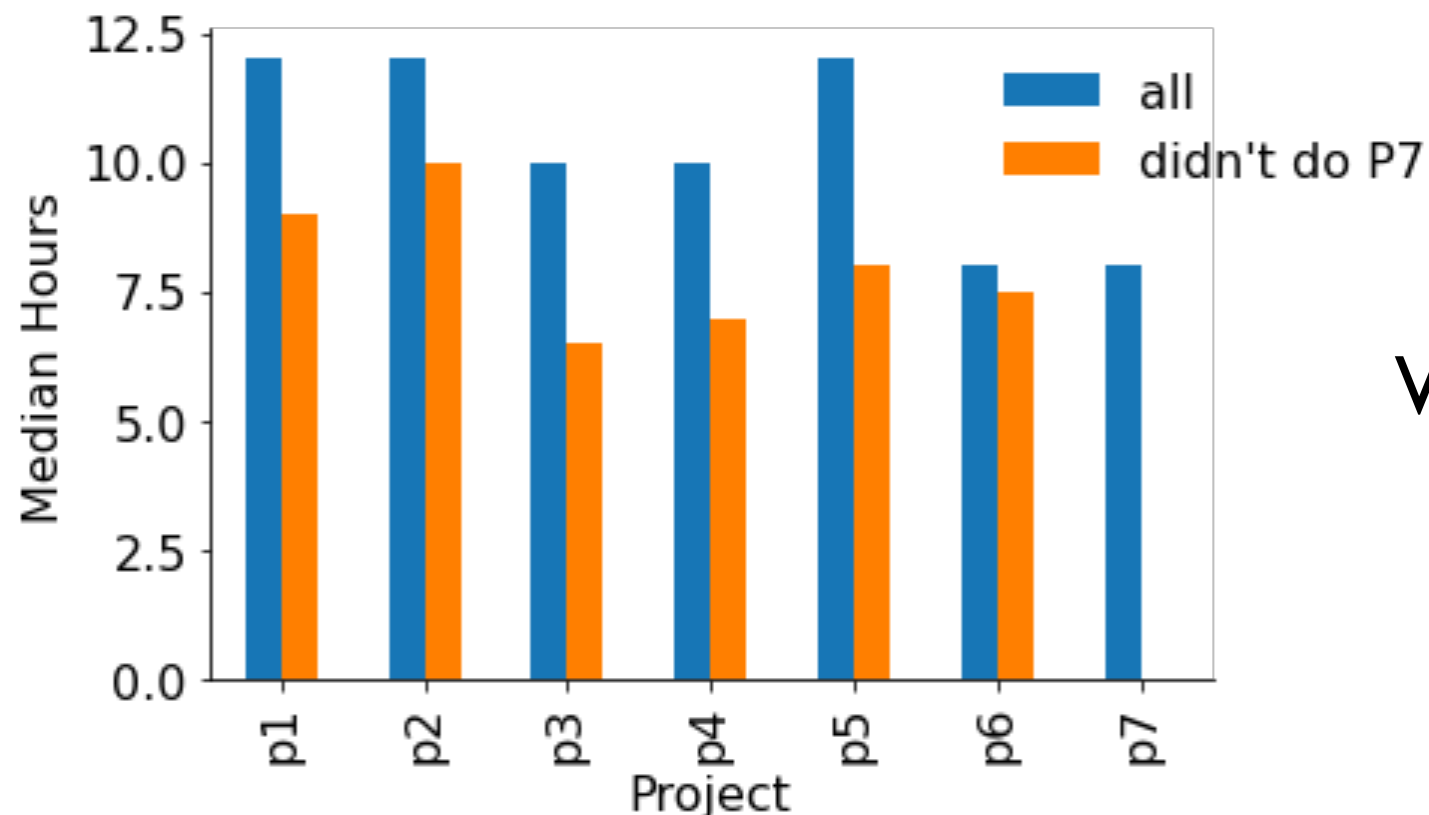
Weekly Expectations
- 4 hours - lecture/lab
- 6 hours - project coding
- 2 hours - reading/quizzes/etc

Graded Work: Exams/Quizzes

Eleven Online Quizzes - 1% each
- focus on material about one week old
- no time limit
- on Canvas, open book/notes
- can take together AT SAME TIME
  with team members (no other human help)

One Midterm - 10%
- individual, multi-choice, 40 minutes
- one page notes, both sides
- in class

One Final - 20%
- individual, multi-choice, 2 hours
- one page notes, both sides

Academic Misconduct

Read syllabus to make sure you know what is and isn't OK.

It's not obvious!

Since Fall 2019, I have made the following misconduct reports:
- **33** students for cheating on projects
- **2** past students for sharing solutions from past semesters
- **7** students for cheating on exams

How we'll keep the class fair
- run MOSS on submissions
- randomize exam question order

Please talk to me if you're feeling overwhelmed with 320 or your semester in general!

Reading: same as 220/301 and some others...
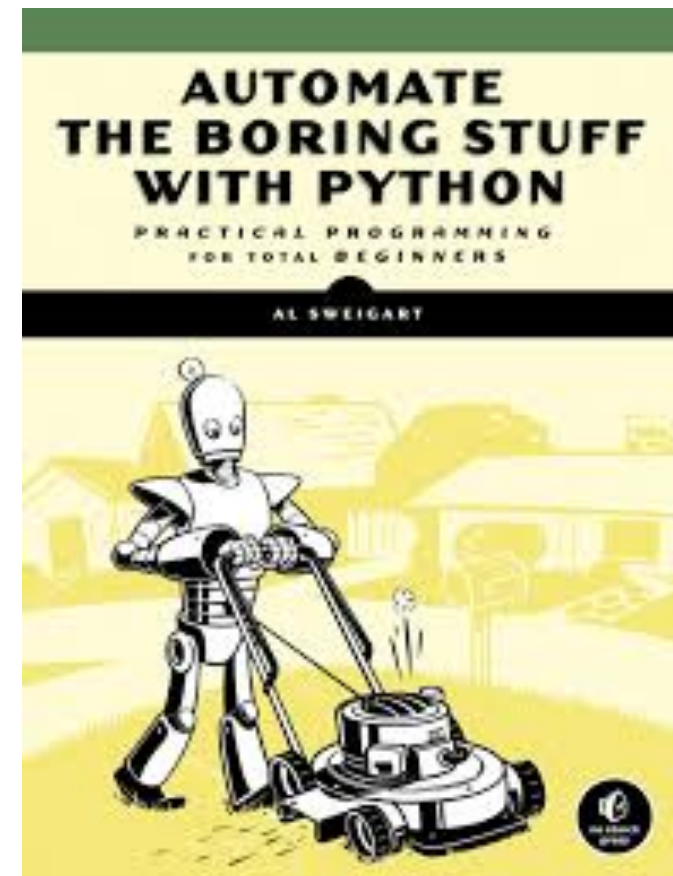
I'll post links to other online articles and my own notes

Lectures don't assume any reading prior to class

Tips for 320 Success

1. Just show up!
    ➡ Get 100% on participation and don't miss quizzes

2. Use office hours
    ➡ we're idle after a project release and swamped before a deadline

3. Do labs before projects

4. Take the lead on group collaboration

5. Learn debugging

6. Run the tester often

7. If you're struggling, reach out -- the sooner, the better

Any questions?

Today's Lecture:
Reproducibility

Discuss: *how might we define "reproducibility" for a data scientist?*

15 new terms to learn today...

reproducibility: others can run our analysis code and get same results

process:

byte:

process memory:

address:

encoding:

CPU:

instruction set:

how many terms do you know already?

operating system:

resource:

allocation:

abstraction:

virtual machine:

cloud:

ssh:

Big question: *will my program run on someone else's computer?*
(not necessarily written in Python)

Things to match:

1 Hardware

2 Operating System

3 Dependencies ← next lecture

CPU
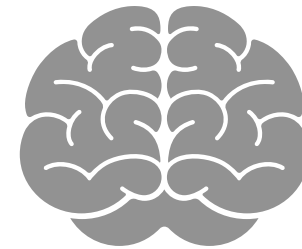
Hardware: Mental Model of Process Memory

Imagine...

- one huge list, **per each** ~~running program~~ process
- every entry in the list is an integer between 0 and 255 (aka a "byte")
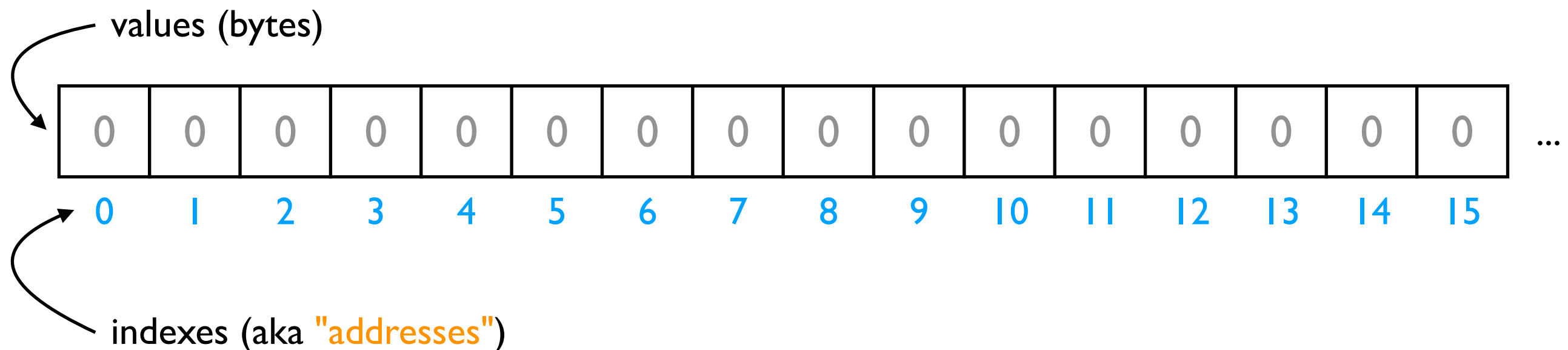
values (bytes)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

indexes (aka "addresses")

How can we use one giant list to handle the following?
- multiple lists
- variables and other references
- strings
- code

> data

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ...

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Is this really all we have for state?

How can we use one giant list to handle the following?
- multiple lists
- variables and other references
- strings
- code

0	0	0	0	0	0	0	0	3	20	0	0	11	22	33	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
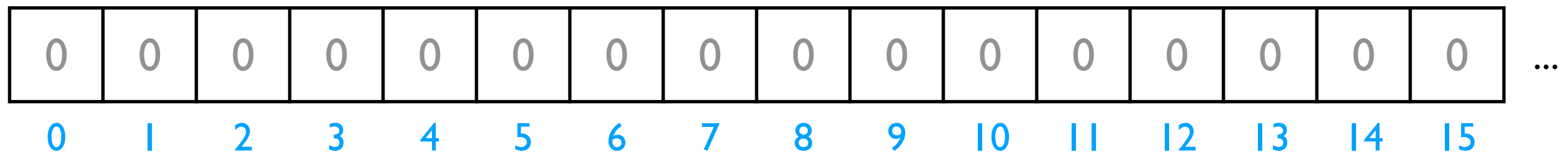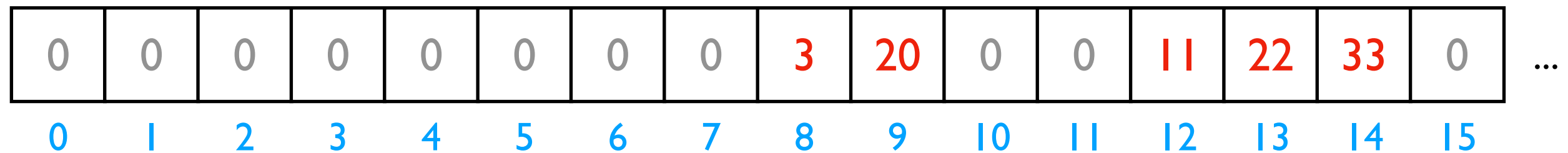
the [3,20] list starts at ~~index~~ address 8 in the giant list

the [11,22,33] list starts at address 12 in the giant list

How can we use one giant list to handle the following?

- multiple lists
- variables and other references
- strings
- code

0	0	0	0	0	0	0	0	3	20	0	0	11	22	33	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

```
# fast
L2.append(44)
```

implications for performance...

How can we use one giant list to handle the following?

- **multiple lists**
- variables and other references
- strings
- code

| ↓ |
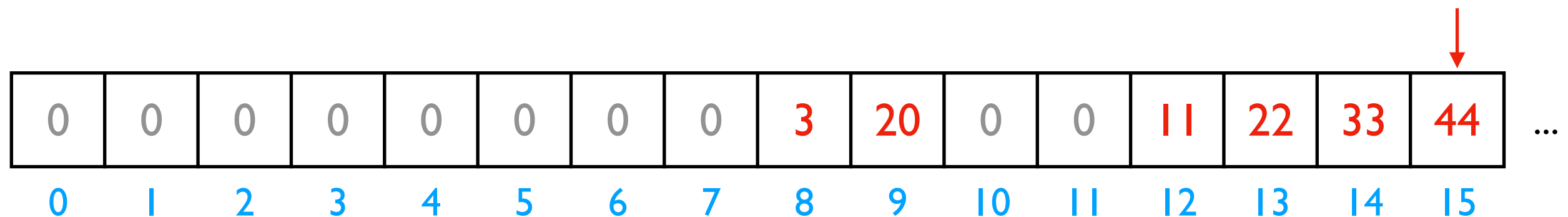| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 20 | 0 | 0 | 11 | 22 | 33 | 44 | ...

0   1   2   3   4   5   6   7   8   9   10 11 12 13 14 15

implications for performance...

```
# fast
L2.append(44)
```

How can we use one giant list to handle the following?

- **multiple lists**
- variables and other references
- strings
- code

0	0	0	0	0	0	0	0	3	20	0	0	11	22	33	44	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

implications for performance...

```
# fast
L2.append(44)

# slow
L2.pop(0)
```

How can we use one giant list to handle the following?

- multiple lists
- variables and other references
- strings
- code

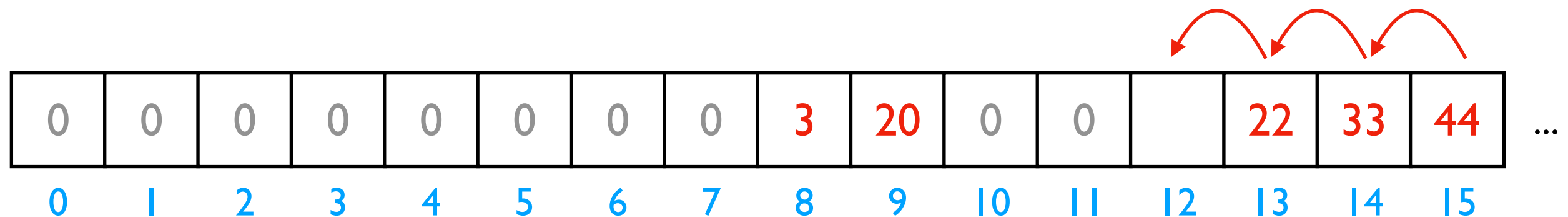| 0	0	0	0	0	0	0	0	3	20	0	0		22	33	44	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

implications for performance...

```
# fast
L2.append(44)

# slow
L2.pop(0)
```

How can we use one giant list to handle the following?

- **multiple lists**
- variables and other references
- strings
- code

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 20 | 0 | 0 | 22 | 33 | 44 | 0 | ... |

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

We'll think more rigorously about performance in CS 320 (big-O notation)

```
# fast
L2.append(44)

# slow
L2.pop(0)
```

How can we use one giant list to handle the following?

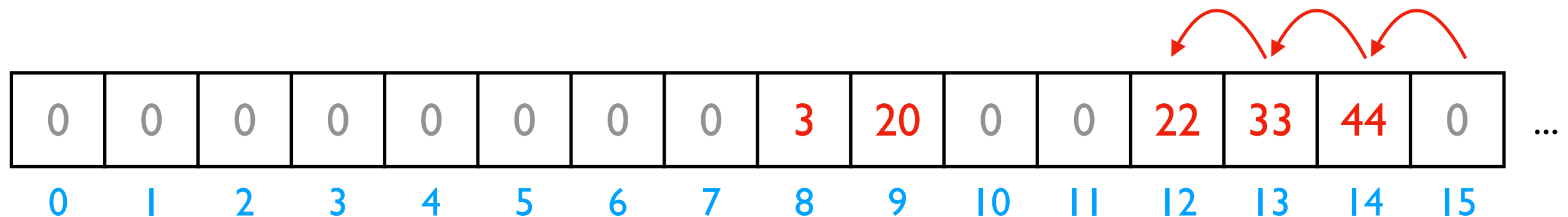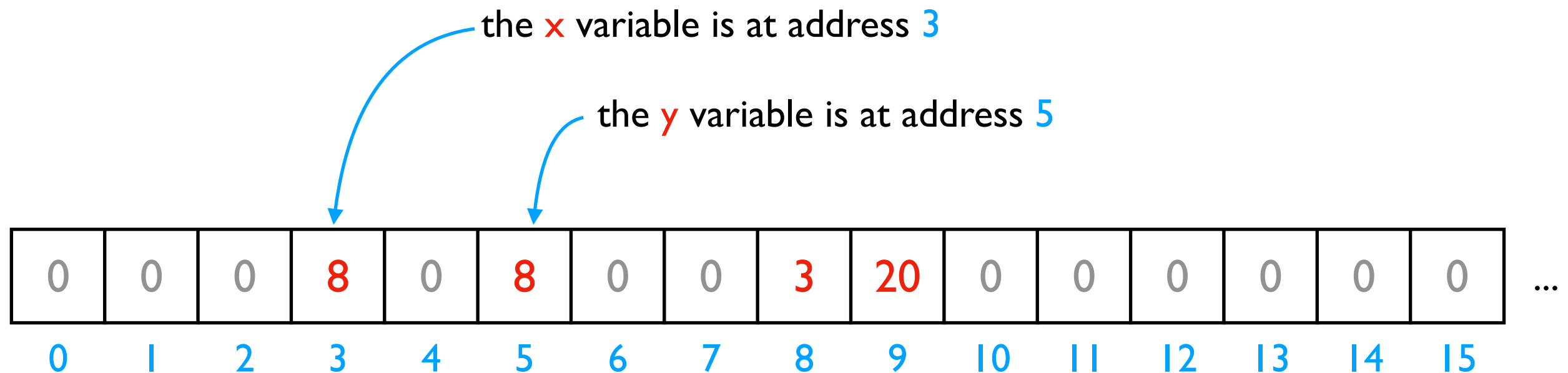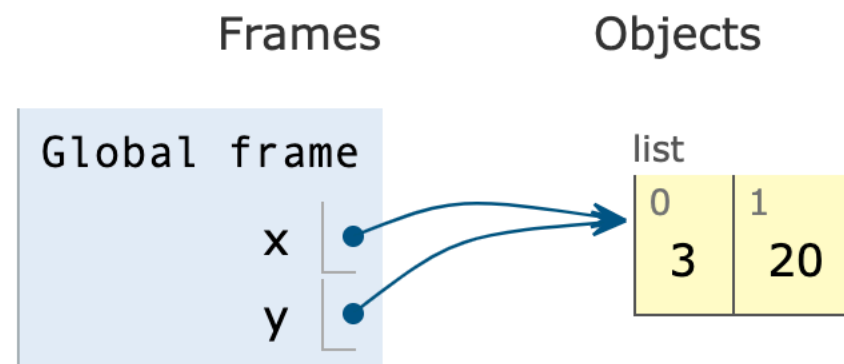- multiple lists
- variables and other references
- strings
- code

the x variable is at address 3

the y variable is at address 5

0	0	0	8	0	8	0	0	3	20	0	0	0	0	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Python 3.6

```
1  x = [3, 20]
2  y = x
```

Edit this code

Frames   Objects

Global frame   list
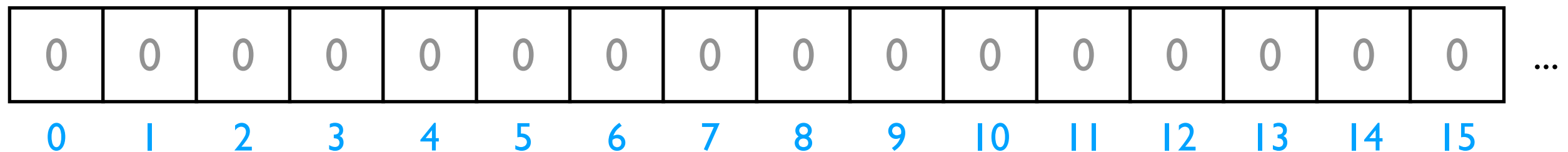
    x   0   1
    y   3   20

PythonTutor's visualization

How can we use one giant list to handle the following?
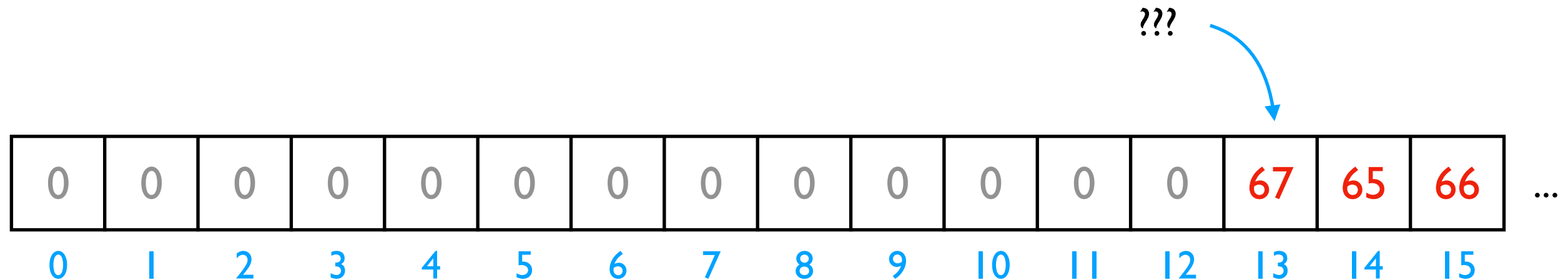- multiple lists
- variables and other references
- strings   discuss: how?
- code

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Is this really all we have for state?

How can we use one giant list to handle the following?

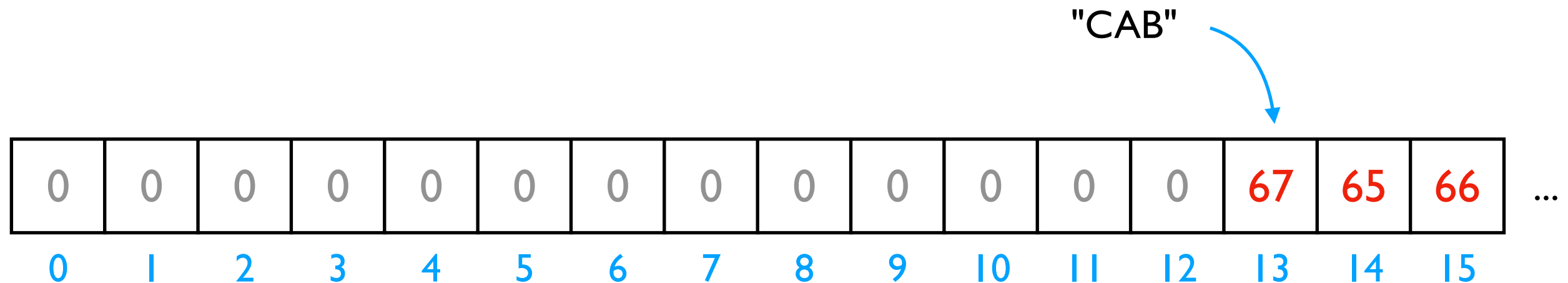- multiple lists
- variables and other references
- strings
- code

???

0	0	0	0	0	0	0	0	0	0	0	0	0	67	65	66	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

encoding:

code	letter
65	A
66	B
67	C
68	D
...	...

```
f = open("file.txt", encoding="utf-8")
```

How can we use one giant list to handle the following?
- multiple lists
- variables and other references
- strings
- code

"CAB"

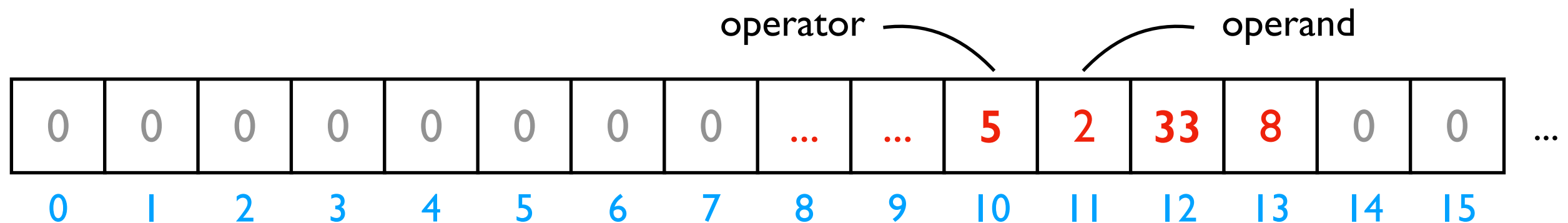| 0	0	0	0	0	0	0	0	0	0	0	0	0	67	65	66	...

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

encoding:

code	letter
65	A
66	B
67	C
68	D
...	...

```
f = open("file.txt", encoding="utf-8")
```

How can we use one giant list to handle the following?

- multiple lists
- variables and other references
- strings
- code

```
while ????:
    i += 2
        # what line next?
```
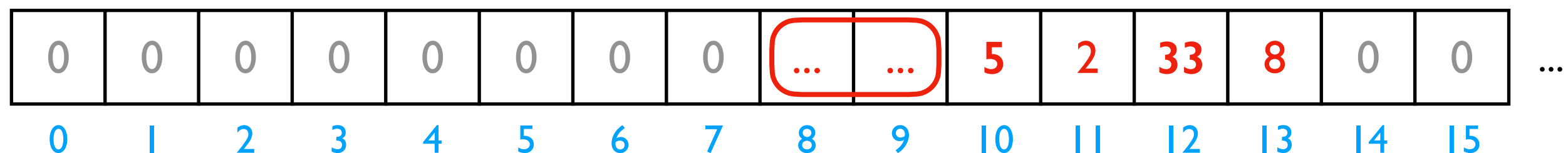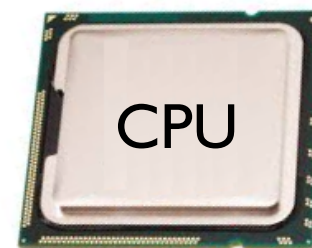
operator — operand

0	0	0	0	0	0	0	0	...	...	5	2	33	8	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Instruction Set

code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:
- keep track of what instruction we're on
- understand instruction codes
- much more

CPU

0	0	0	0	0	0	0	0	…	…	5	2	33	8	0	0	…
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Write code in [Python 3.6 ⇕]
    (drag lower right corner to resize code editor)

➡ 1 ━━━━━━━
  2 ━━━━━━━━━
  3 ━━━━━━━━

➡ line that just executed
➡ next line to execute

Instruction Set

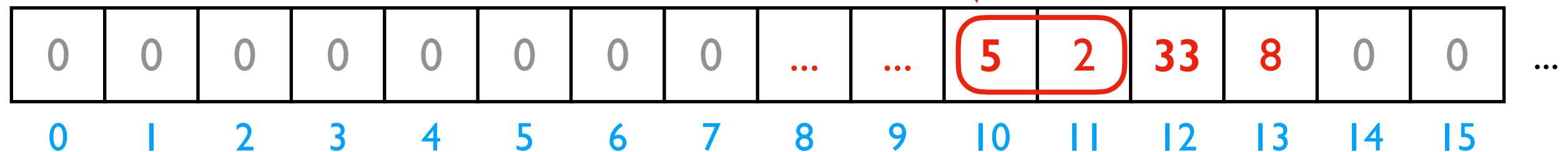| code	operation
5	ADD
8	SUB
33	JUMP
…	…

Hardware: Mental Model of CPU

CPUs interact with memory:
- keep track of what instruction we're on
- understand instruction codes
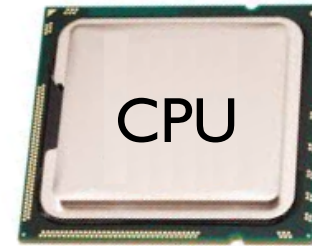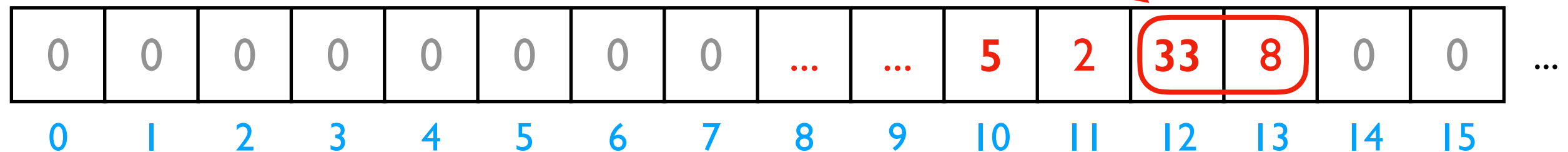- much more

CPU

add 2 to variable

0	0	0	0	0	0	0	0	...	...	5	2	33	8	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Instruction Set

code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:
- keep track of what instruction we're on
- understand instruction codes
- much more
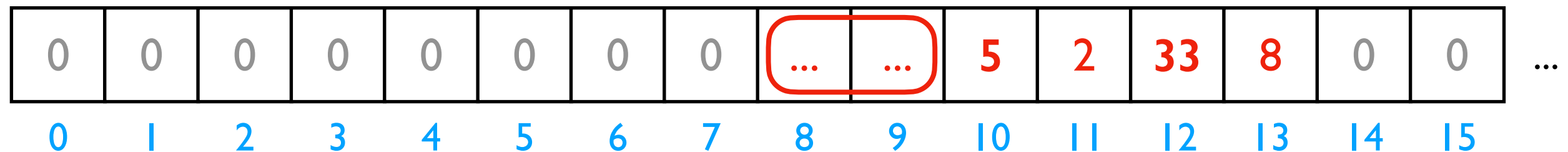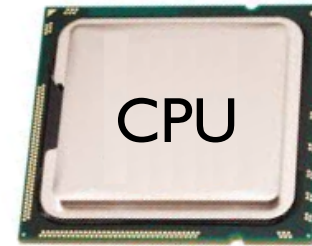
CPU

go back to top of loop

0	0	0	0	0	0	0	0	...	...	5	2	33	8	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

	code	operation
	5	ADD
Instruction Set	8	SUB
	33	JUMP
	...	...

Hardware: Mental Model of CPU

CPUs interact with memory:
- keep track of what instruction we're on
- understand instruction codes
- much more

CPU

0	0	0	0	0	0	0	0	...	...	**5**	**2**	**33**	**8**	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Instruction Set

code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

discuss: what would happen if a
CPU tried to execute an
instruction for a different CPU?

0	0	0	0	0	0	0	0	...	...	5	2	33	8	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Instruction Set for CPU X

code	operation
5	ADD
8	SUB
33	JUMP
...	...

Instruction Set for CPU Y

code	operation
5	SUB
8	ADD
33	undefined
...	...

Hardware: Mental Model of CPU

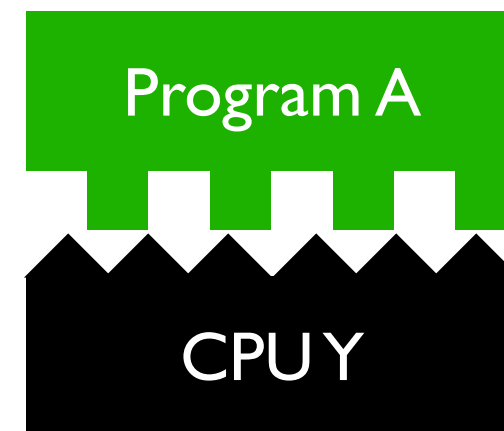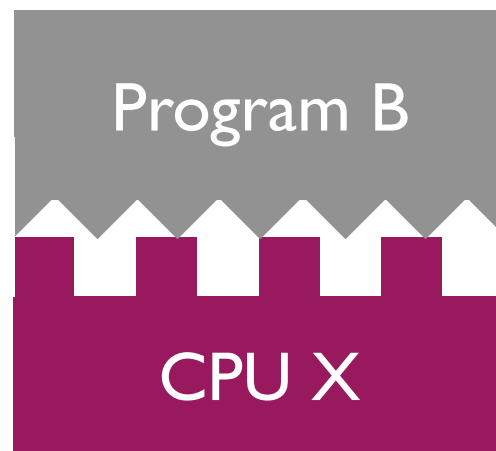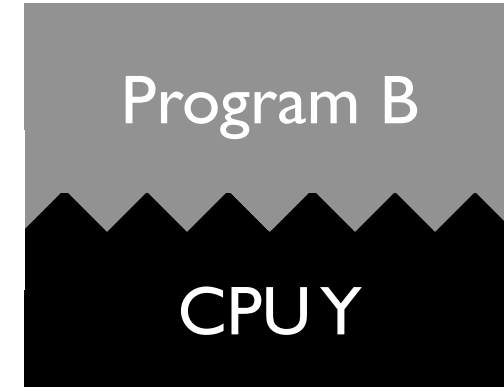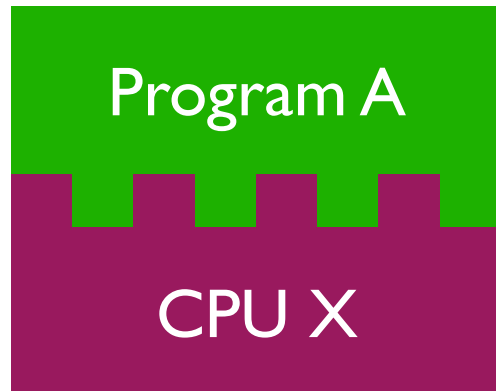a CPU can only run programs that use instructions it understands!
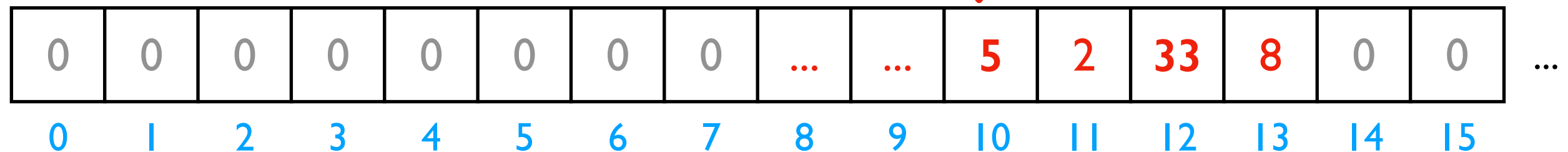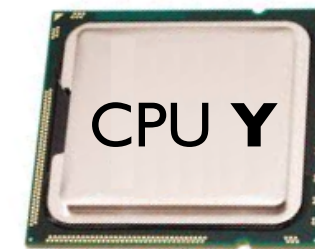
CPU **Y**

0	0	0	0	0	0	0	0	...	...	5	2	33	8	0	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

Instruction Set for **CPU X**

code	operation
5	ADD
8	SUB
33	JUMP
...	...

Instruction Set for **CPU Y**

code	operation
5	SUB
8	ADD
33	undefined
...	...

A Program and CPU need to "fit"

A Program and CPU need to "fit"

Program A

CPU X

Program B

CPU Y

why haven't we noticed this yet
for our Python programs?

Interpreters

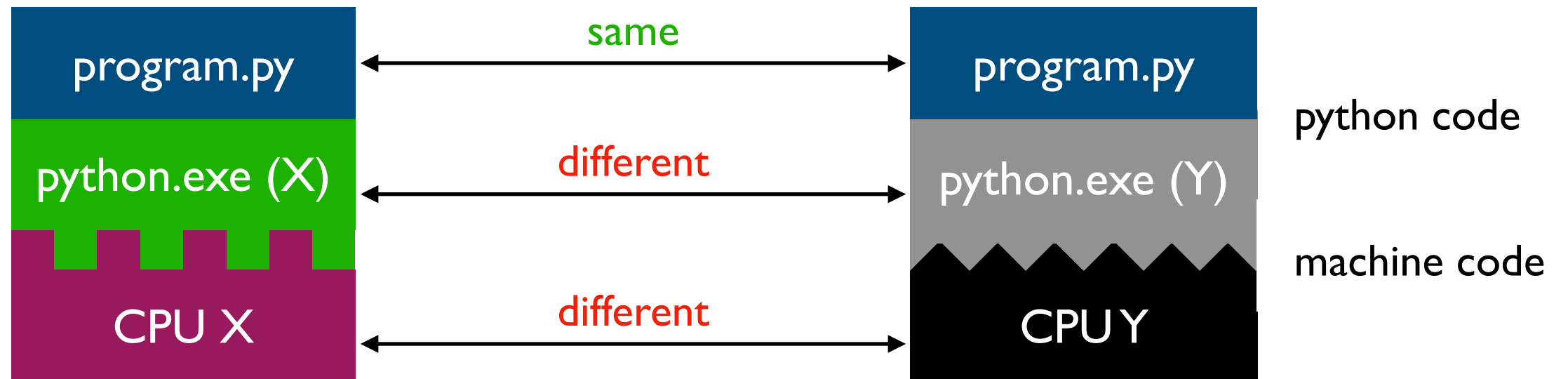|		
program.py	same	program.py
python.exe (X)	different	python.exe (Y)
CPU X	different	CPU Y

python code
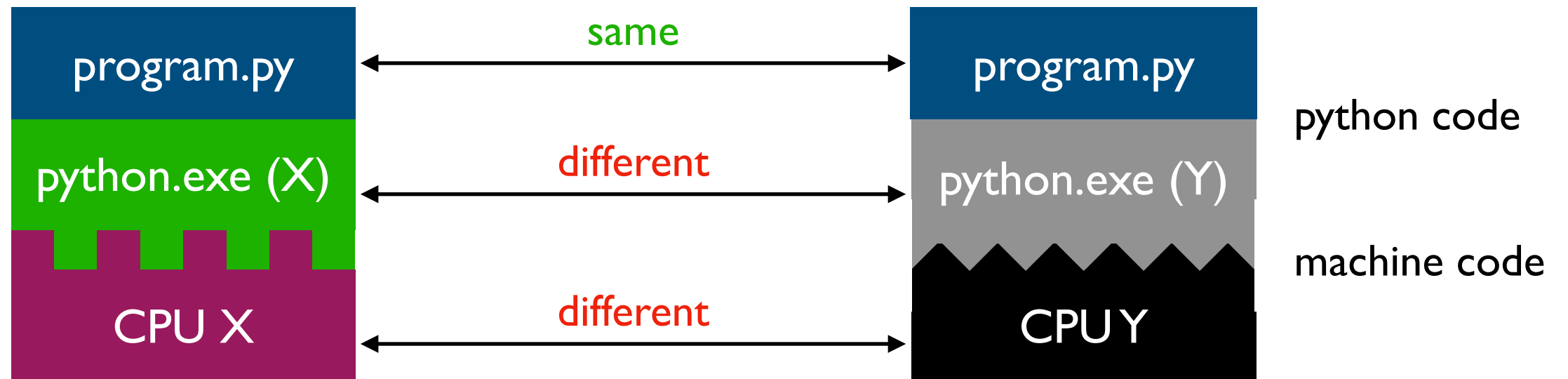
machine code

Interpreters (such as python.exe) make it easier to run the same code on different machines

A compiler is another tool for running the same code on different CPUs

Interpreters

Interpreters (such as python.exe) make it easier to run the same code on different machines

Discuss: *if all CPUs had the instruction set, would we still need a Python interpreter?*

Big question: *will my program run on someone else's computer?*
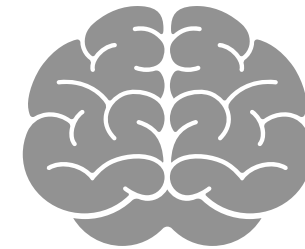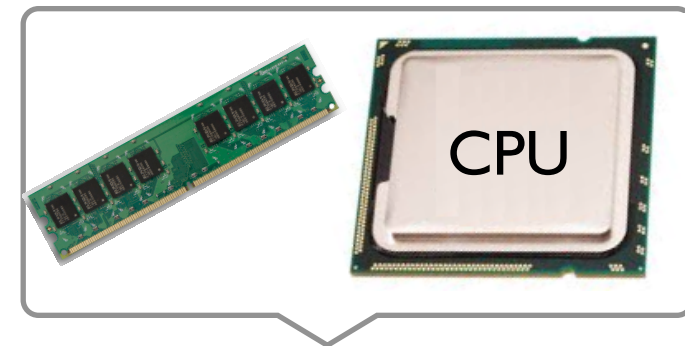(not necessarily written in Python)

Things to match:

1 Hardware

2 Operating System

3 Dependencies ←——— next lecture

Big question: *will my program run on someone else's computer?*
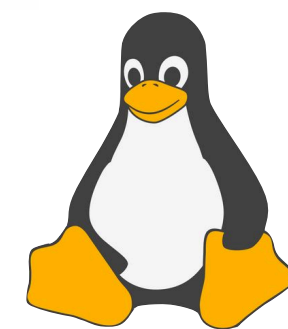(not necessarily written in Python)

Things to match:

1 Hardware

2 Operating System

3 Dependencies ← next lecture

Windows

macOS®

Linux

many others...

Red Hat

android

ubuntu

[this semester]

OS jobs: Allocate and Abstract Resources

[like CPU, hard drive, etc]

1 Allocation

2 Abstraction

Process B

...

waiting

Process Z

OS decides

running

Process A

CPU X

only one process can run on CPU at a time
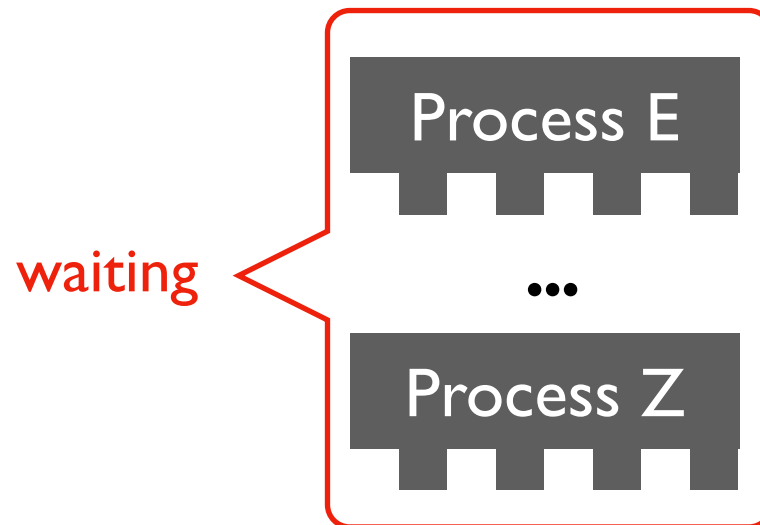
```
f = open("file.txt")
data = f.read()
f.close()
```

convenient

Operating System

inconvenient

ignorant of files/directories

Parallelism -- more later this semester...

waiting

Process E

...

Process Z

running processes

Proc A
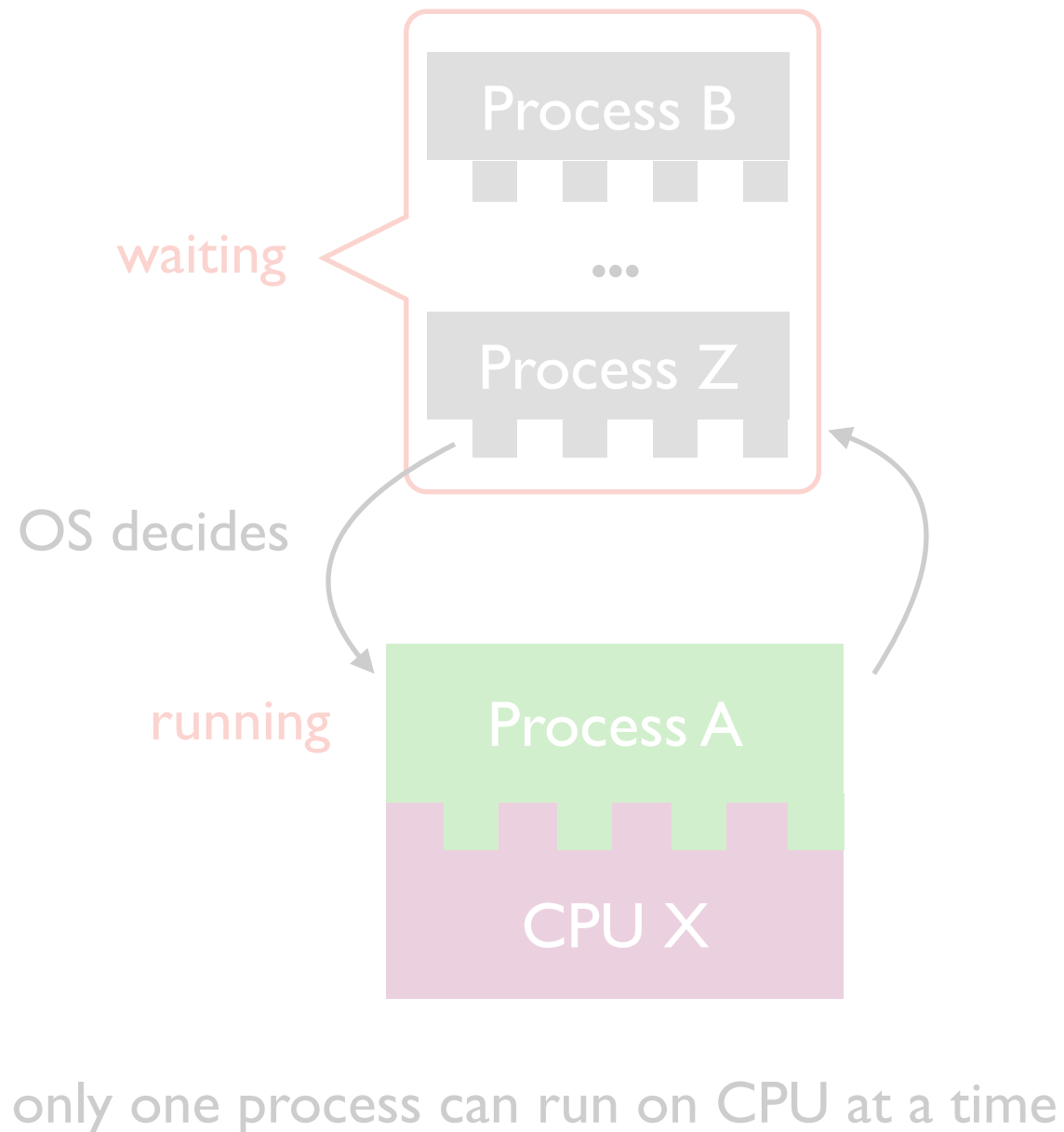Core 1

Proc B
Core 2

Proc C
Core 1

Proc D
Core 2
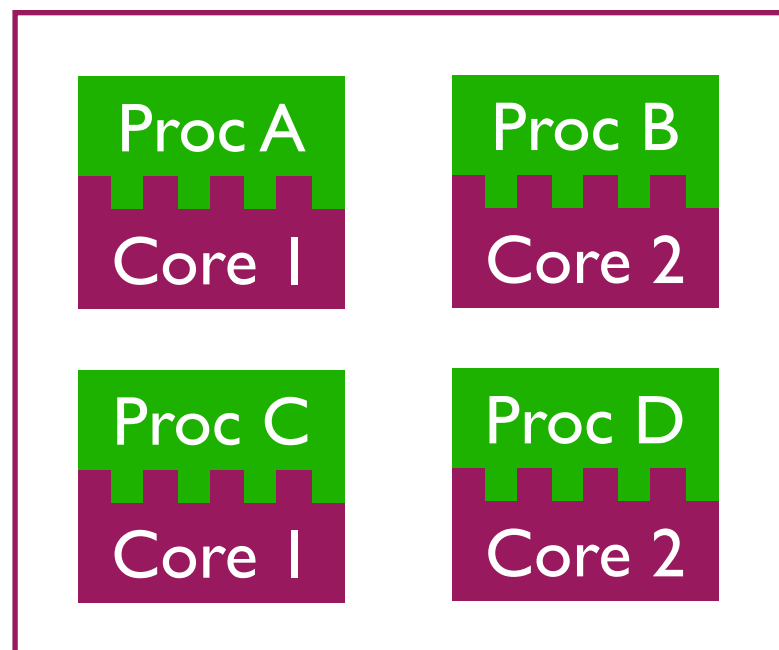
most modern CPUs actually contain multiples CPUs (called "cores") on a single chip

Later: *how can we write programs that run in parallel, going faster by using multiple cores?*

OS jobs: Allocate and Abstract Resources

[like CPU, hard drive, etc]

1 Allocation

Process B

...

Process Z

waiting

OS decides

running

Process A

CPU X

only one process can run on CPU at a time

2 Abstraction

```
f = open("file.txt")
data = f.read()
f.close()
```

convenient

Operating System

inconvenient

ignorant of files/directories
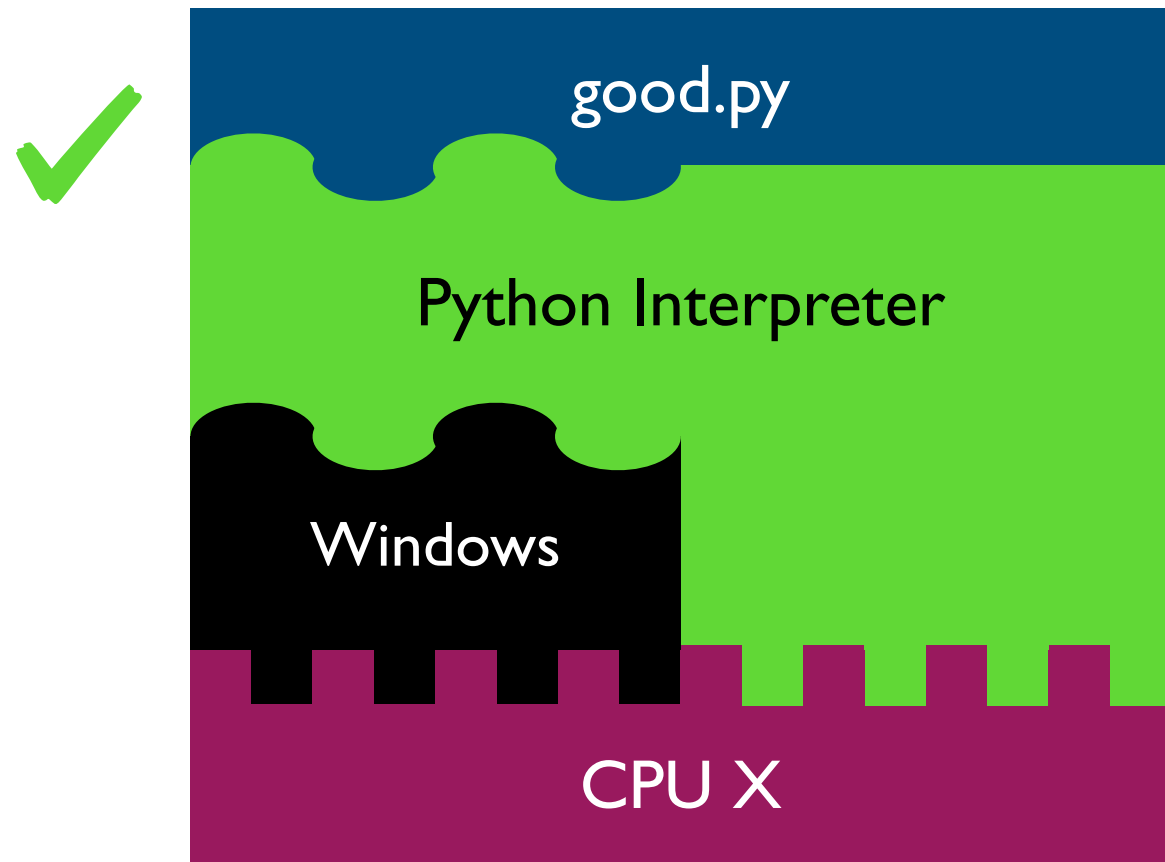
Harder to reproduce on different OS...

bad.py

Python Interpreter

Windows

CPU X
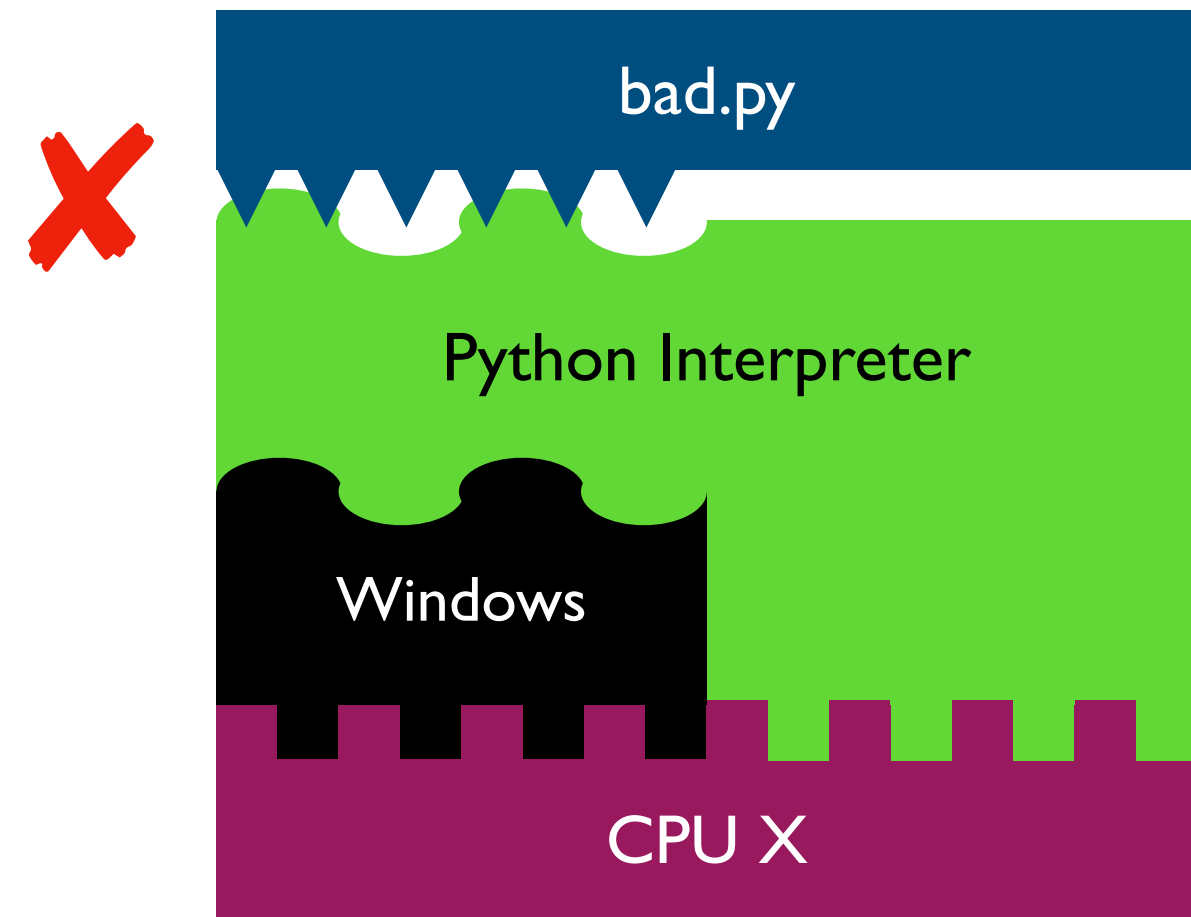
```
f = open("/data/file.txt")
...
```

The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

Harder to reproduce on different OS...

good.py

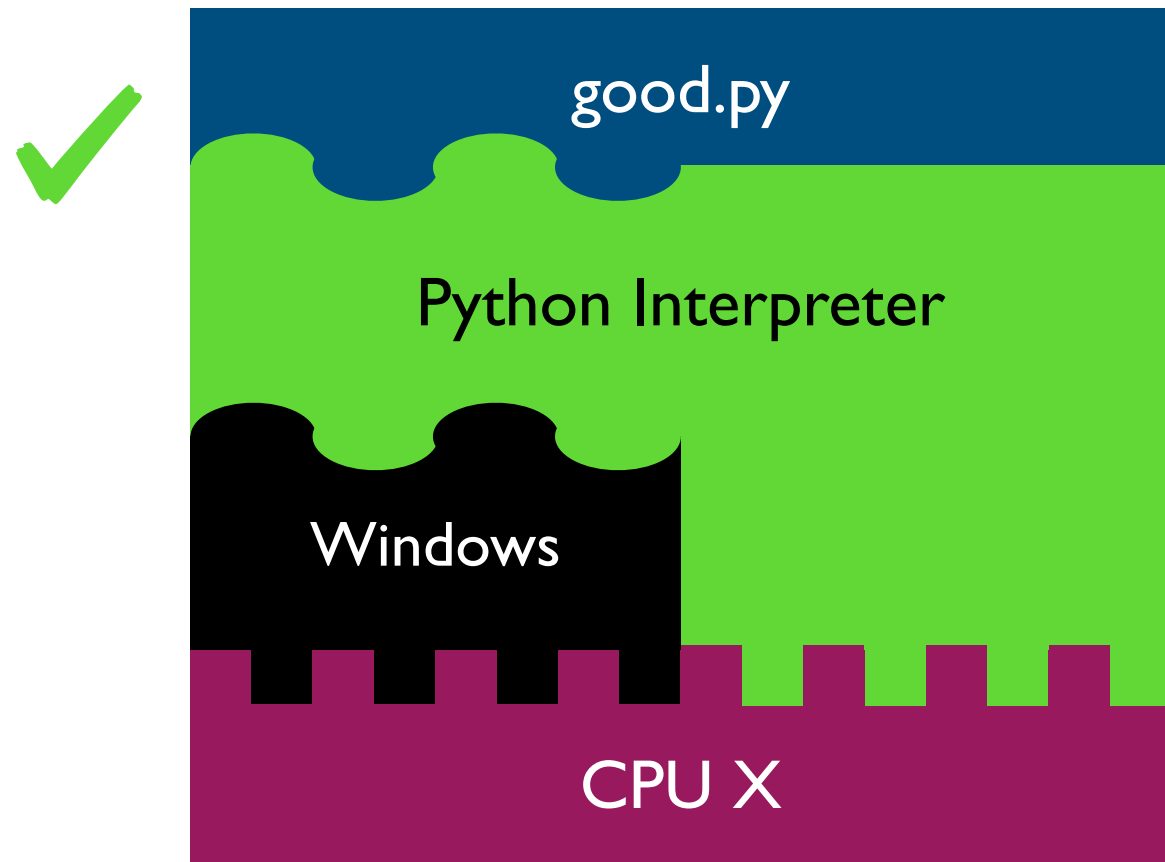Python Interpreter

Windows

CPU X

```
f = open("c:\data\file.txt")
...
```

The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

Harder to reproduce on different OS...

solution 1:
```
f = open(os.path.join("data", "file.txt"))
...
```

solution 2:
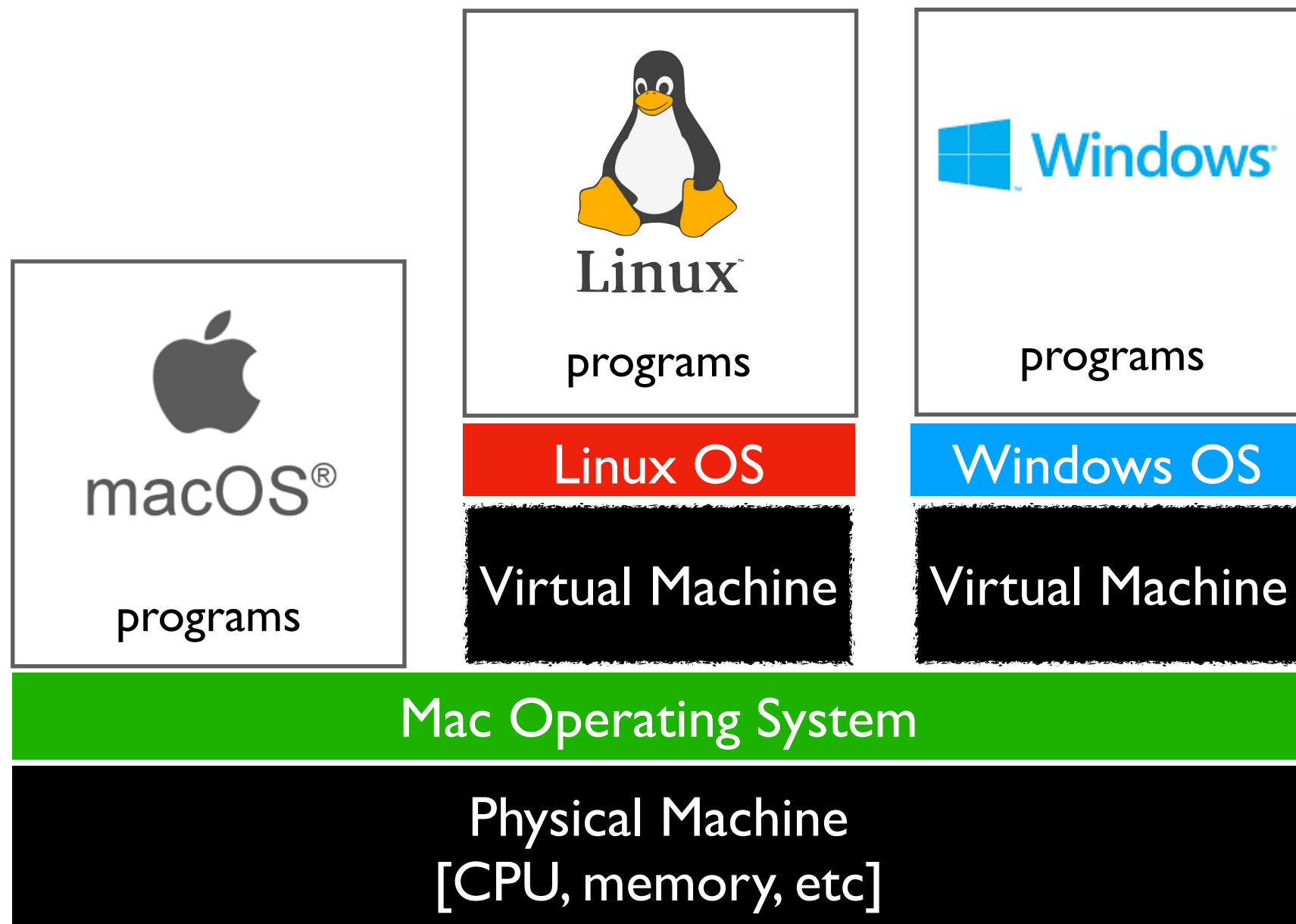tell anybody reproducing your results to use the same OS!

tradeoffs?

The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

VMs (Virtual Machines)

popular virtual machine software

programs

Linux OS

Virtual Machine

programs

Windows OS

Virtual Machine

macOS®

programs

Mac Operating System

Physical Machine
[CPU, memory, etc]

With the right virtual machines created and operating systems installed, you could run programs for Mac, Linux, and Windows -- at the same time without rebooting!
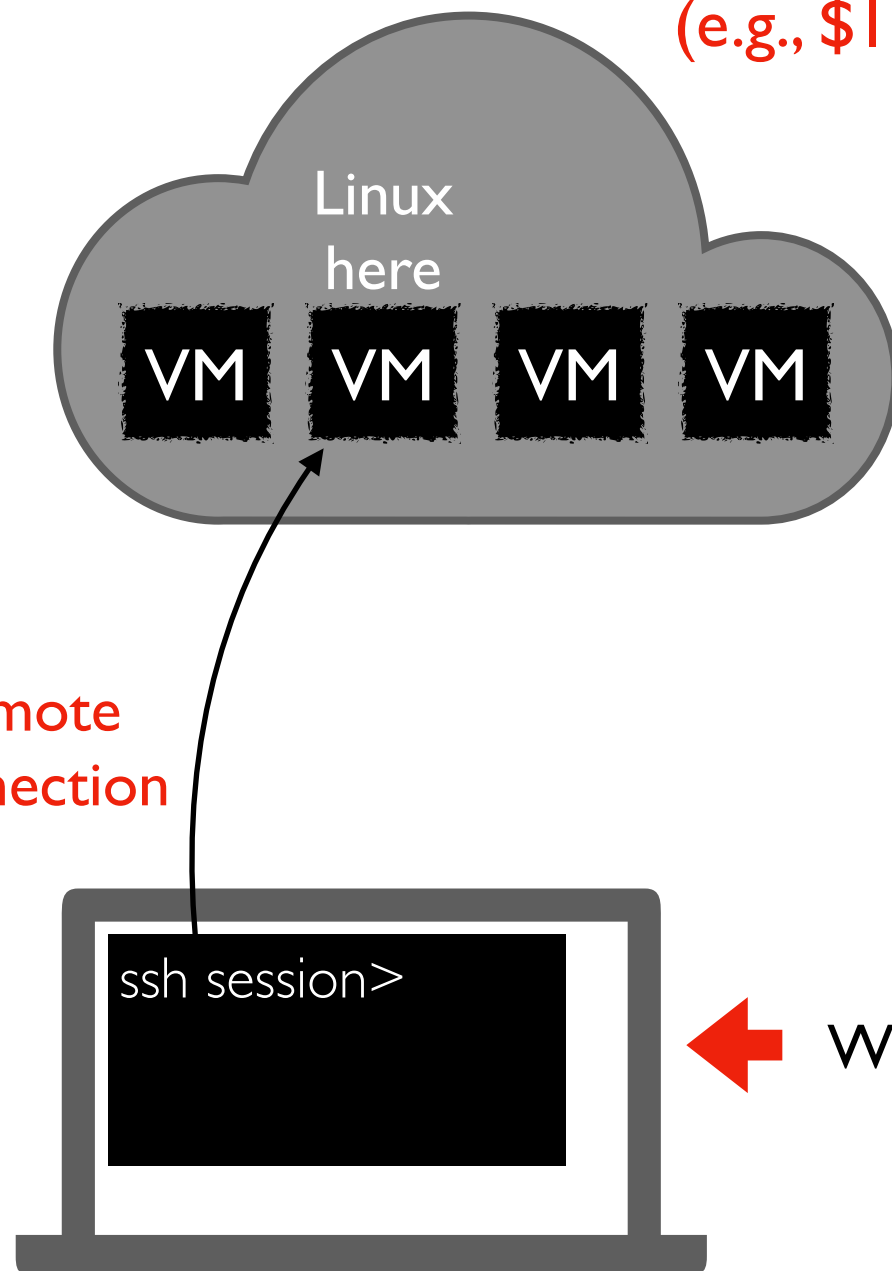
The Cloud

popular cloud providers

cloud providers let you rent VMs
in the cloud on hourly basis
(e.g., $15 / month)

Linux
here

VM VM VM VM

remote
connection

ssh session>

Windows, Mac, whatever

ssh user@best-linux.cs.wisc.edu ⬅ run in PowerShell/
bash to access CS lab

we'll use GCP virtual
machines this semester
[setup in Lab 1]

Lecture Recap: Reproducibility

Big question: *will my program run on someone else's computer?*

Things to match:

1   Hardware   ←   a program must fit the CPU; `python.exe` **will do this, so** `program.py` **won't have to**

2   Operating System   ←   we'll use Ubuntu Linux on virtual machines in the cloud

3   Dependencies   ←   next time: versioning

Recap of 15 new terms

reproducibility: others can run our analysis code and get same results

process: a running program

byte: integer between 0 and 255

process memory: a big "list" of bytes, per process, for all state

address: index in the big list

encoding: pairing of ~~letters~~ characters with numeric codes

CPU: chip that executes instructions, tracks position in code

instruction set: pairing of CPU instructions/ops with numeric codes

operating system: software that allocates+abstracts resources

resource: time on CPU, space in memory, space on SSD, etc

allocation: the giving of a resource to a process

abstraction: hiding inconvenient details with something easier to use

virtual machine: "fake" machine running on ~~real~~ physical machine
allows us to running additional operating systems

cloud: place where you can rent virtual machines and other services

ssh: secure shell -- tool that lets you remotely access another machine