

[320] Welcome + First Lecture

[reproducibility]

Tyler Caraza-Harter

Welcome to the first ever offering of Data Programming II!

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

CS 220

- getting results
- writing correct code
- using objects
- functions: `f(obj)`
- lists+dicts
- analyzing datasets
- plots
- tabular analysis

CS 320

- getting **reproducible** results
- writing **efficient** code
- designing **new types** of objects
- methods**: `obj.f()`
- graphs+trees
- collecting**+analyzing datasets
- animated visualizations
- simple machine learning**

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



Who am I?

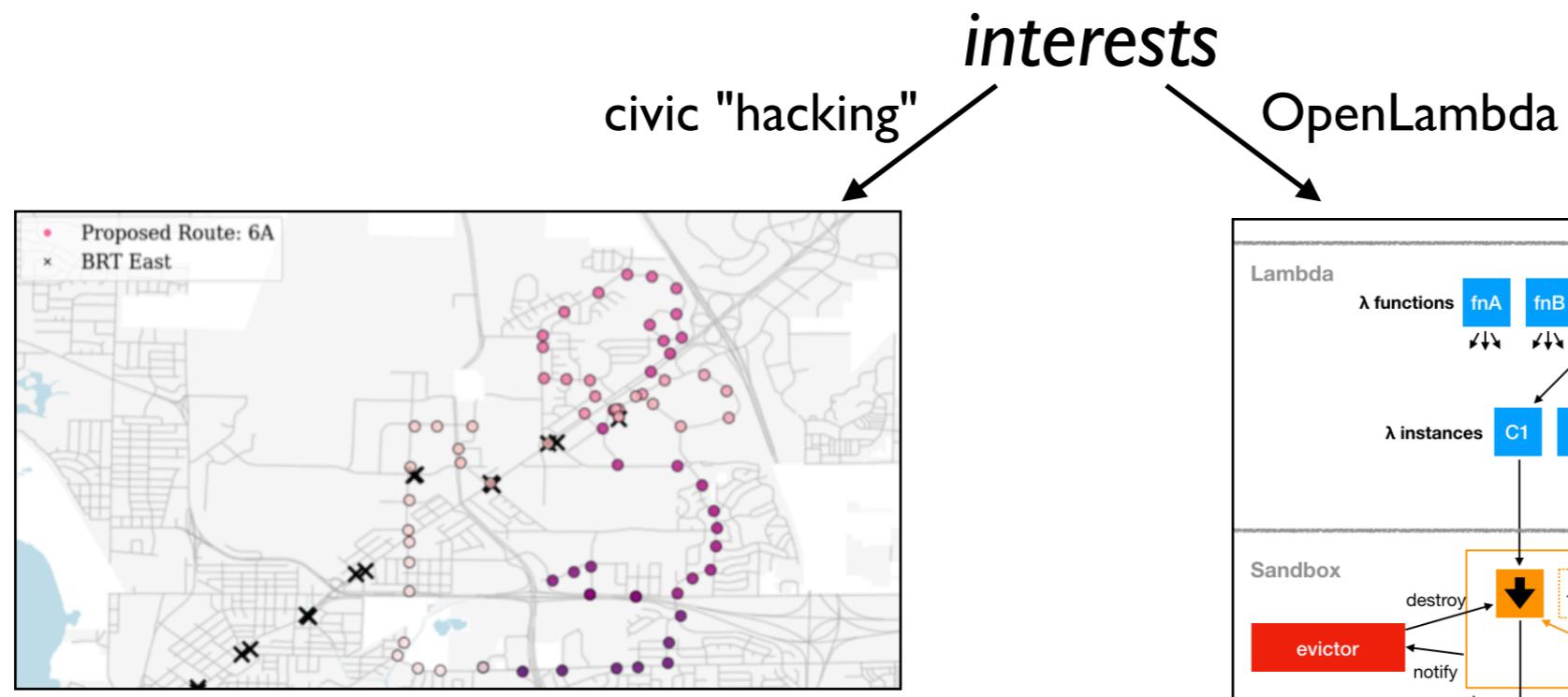
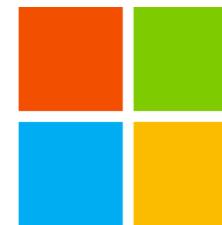
Tyler Caraza-Harter

- Long time Badger
- Email: tharter@wisc.edu
- Just call me “Tyler”



Industry experience

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



Plot by [Megan Tabbutt](#) (previous student)

More: <https://wisc-ds-projects.github.io/f19/>

Who are You?

Year in school?

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

Area of study

- Natural science, social science, engineering, statistics, data science, other?

What CS courses have people taken before?

- CS 301 (the import one here)? CS 200? CS 300? CS 354?

Who are You?

Year in school?

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

Area of study

- Natural science, social science, engineering, statistics, data science, other?

What CS courses have people taken before?

- CS 301 (the import one here)? CS 200? CS 300? CS 354?

Please fill this form: <https://forms.gle/kZYhHfhbEhDsRVW88>. Why?

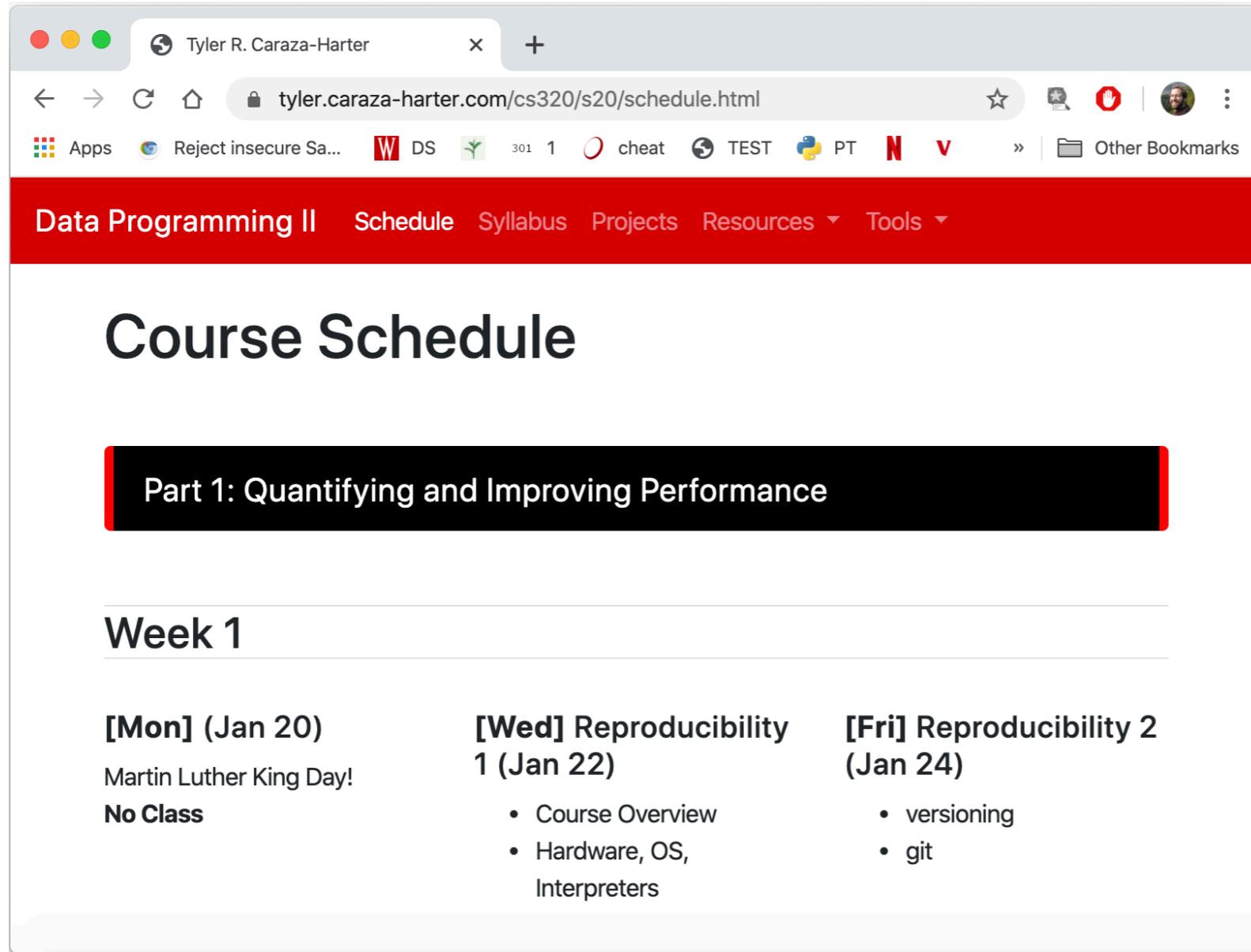
- To be helpful
- I'm more likely to round up grades of those who participate
- Help design the exam

How many questions would you like to be on each exam? *

Your answer

Course Website

It's here: <https://tyler.caraza-harter.com/cs301/fall19/schedule.html>



A screenshot of a web browser window titled "Tyler R. Caraza-Harter". The address bar shows the URL "tyler.caraza-harter.com/cs320/s20/schedule.html". The page content is for "Data Programming II". A red navigation bar at the top includes links for "Schedule", "Syllabus", "Projects", "Resources", and "Tools". Below the navigation bar, the main title is "Course Schedule". A black banner with white text reads "Part 1: Quantifying and Improving Performance". The "Week 1" section is detailed below:

[Mon] (Jan 20)	[Wed] Reproducibility 1 (Jan 22)	[Fri] Reproducibility 2 (Jan 24)
Martin Luther King Day! No Class	<ul style="list-style-type: none">• Course Overview• Hardware, OS, Interpreters	<ul style="list-style-type: none">• versioning• git

read syllabus carefully
and checkout other content

I'll use **Canvas** for two things only:

- simple grade summaries (not feedback or exam answers)
- general announcements

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 and our 3 TAs)

**no Shelf, sadly, because it's CS 320's first offering
(hopefully some of you will become Shelf mentors)**

Forms

- <https://tyler.caraza-harter.com/cs320/s20/surveys.html>
- Who are you? **Feedback Form.** Exam Conflicts. Thank you!

**extra important for a
first-time class offering!**

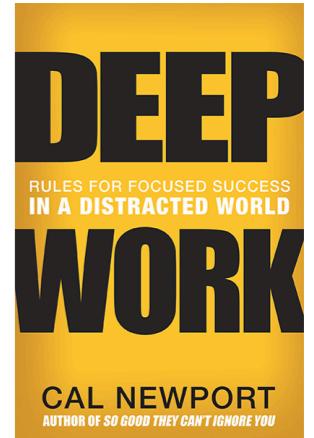
Email

- me: tharter@wisc.edu
- your TA: <https://tyler.caraza-harter.com/cs320/s20/status.html>

Course Etiquette

Meetings

1. office hours are walk-in (no need to reserve). I'll help multiple students at once, so don't wait outside
2. email me to schedule individual meetings
3. above options are preferred over drop-ins at other times



Email

4. let us know your NetID (if not from netid@wisc.edu)
5. CC your project partner (when project related)
6. Don't start new email thread if topic is the same
7. Unless urgent, please give me 48 hours to respond before following up (I'll try to be faster usually)
8. Use your judgement about whether to email me or TA first

Graded Work

2 Exams - 20% each

- multiple choice (how many questions? you decide!)
- one notesheet allowed (printed or handwritten)
- midterm tentatively scheduled for March 11 (evening)

6 Projects - 10% each

- **format:** notebook, module, or program
- with one partner or alone
- still a test .py, but more depends on TA evaluation (more plots)
- **ask for specific feedback**
(giving constructive criticism is a priority in CS 320)
- suggestion: write code (and certainly test) on [Linux](#)

all points get added up and a curve set at the end
(see syllabus for "worst case" scenario)

Academic Misconduct

In Fall 2019, I made the following misconduct reports:

- **23** 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

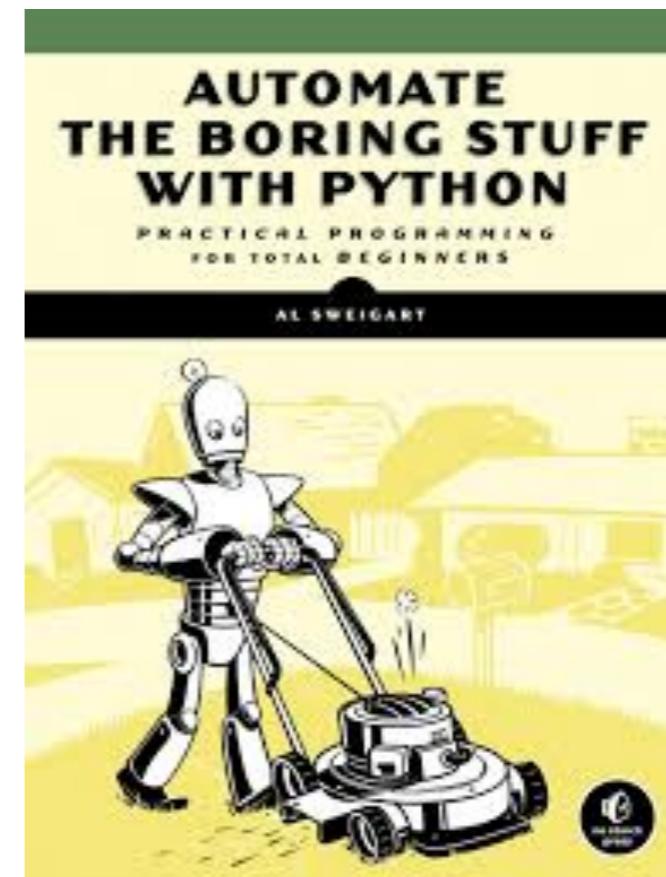
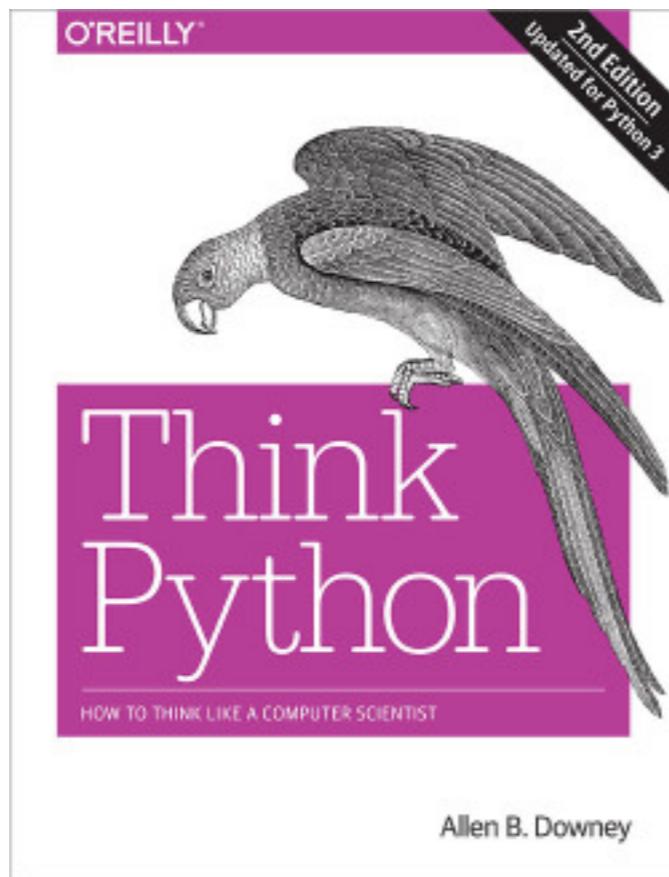
Please talk to me if you're feeling overwhelmed with 320 or your semester in general!

read syllabus to make sure you know what is and isn't OK.
(it might be less obvious than you think)

is it OK? 

- using a function found on StackOverflow (with citation)
- sitting next to your friend during an exam

Reading: same as 301 and some others...



I'll post links to other online articles and my own notes

Lecture+Lab

THREE lectures per week (50 minutes)

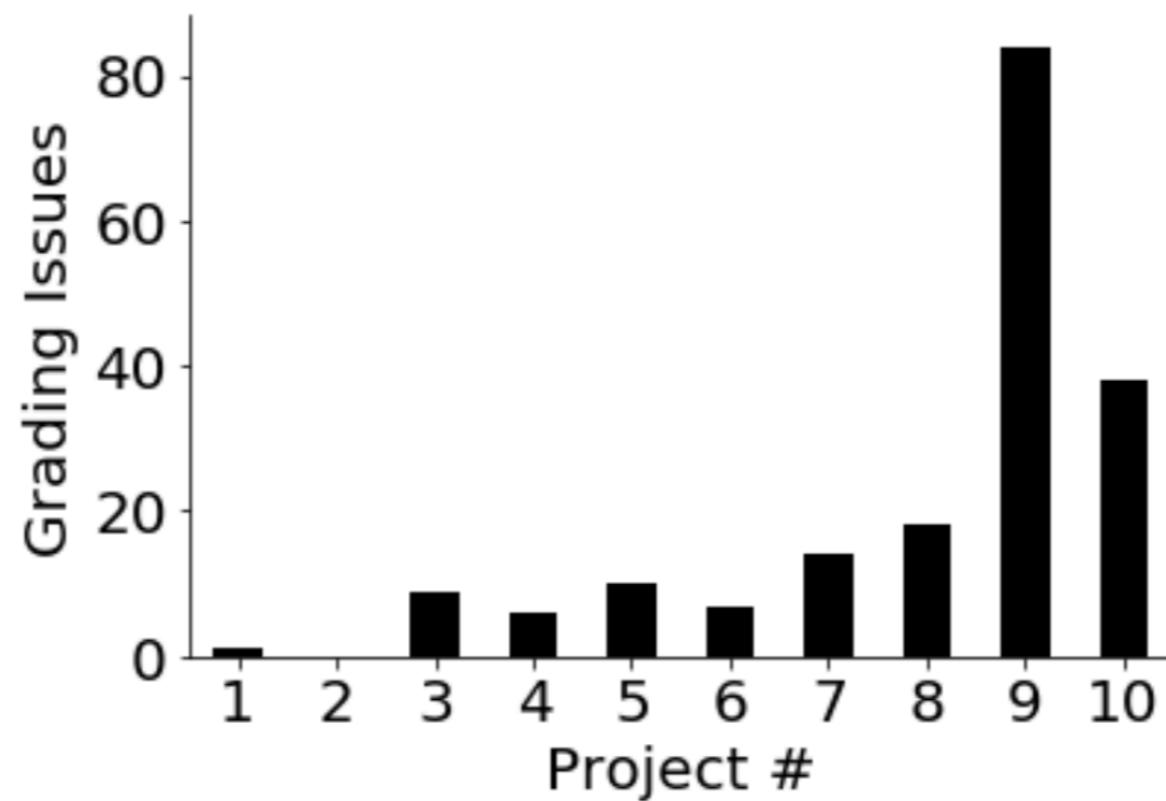
- feel free to bring your laptop
- **rule:** don't do anything more interesting than my lecture

ONE lab per week (75 minutes)

- complete lab document (at home prior), or there in person
- bring your laptop (or use lab machines as backup)
- one TA (not your email TA) will be there to answer questions
- **rule:** don't ask for project help until you've done the labs

Today's Lecture: Reproducibility

Reproducibility (Fall 19 Grading for CS 301)



why was project 9 so problematic?

[All](#)[News](#)[Images](#)[Books](#)[Videos](#)[More](#)[Settings](#)[Tools](#)

About 44,700,000 results (0.64 seconds)

Dictionary

Search for a word 



re·pro·duc·i·bil·i·ty

/rēprə'd(y)oosə'bilədē/

noun

noun: reproducibility

the ability to be reproduced or copied.

"the reproducibility of reconstructive surgery techniques"

- the extent to which consistent results are obtained when an experiment is repeated.

"the experiments were conducted numerous times to test the reproducibility of the results"

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

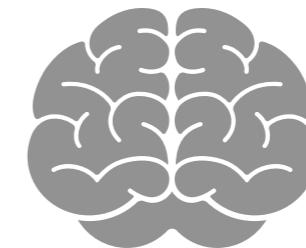
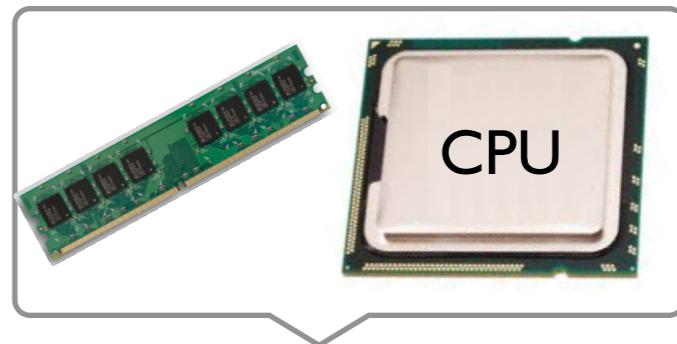
Hardware

2

Operating System

3

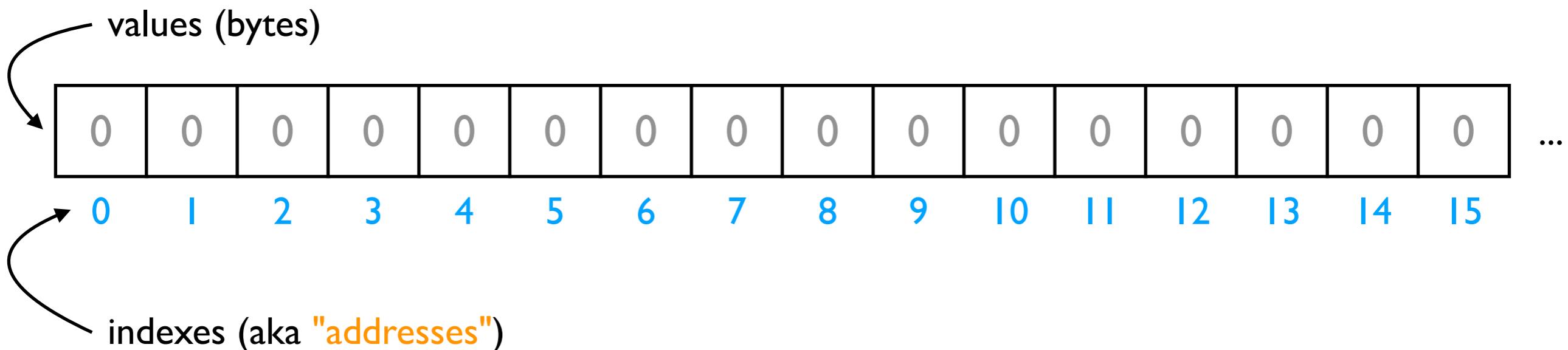
Dependencies ← next lecture



Hardware: Mental Model of Process Memory

Imagine...

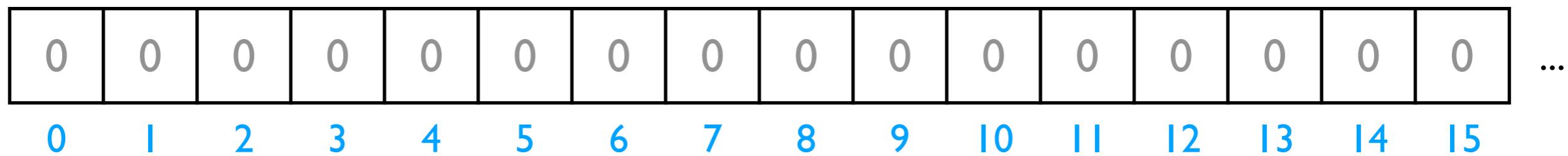
- one huge list, **per each running program process**
- every entry in the list is an integer between 0 and 255 (aka a "byte")



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

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

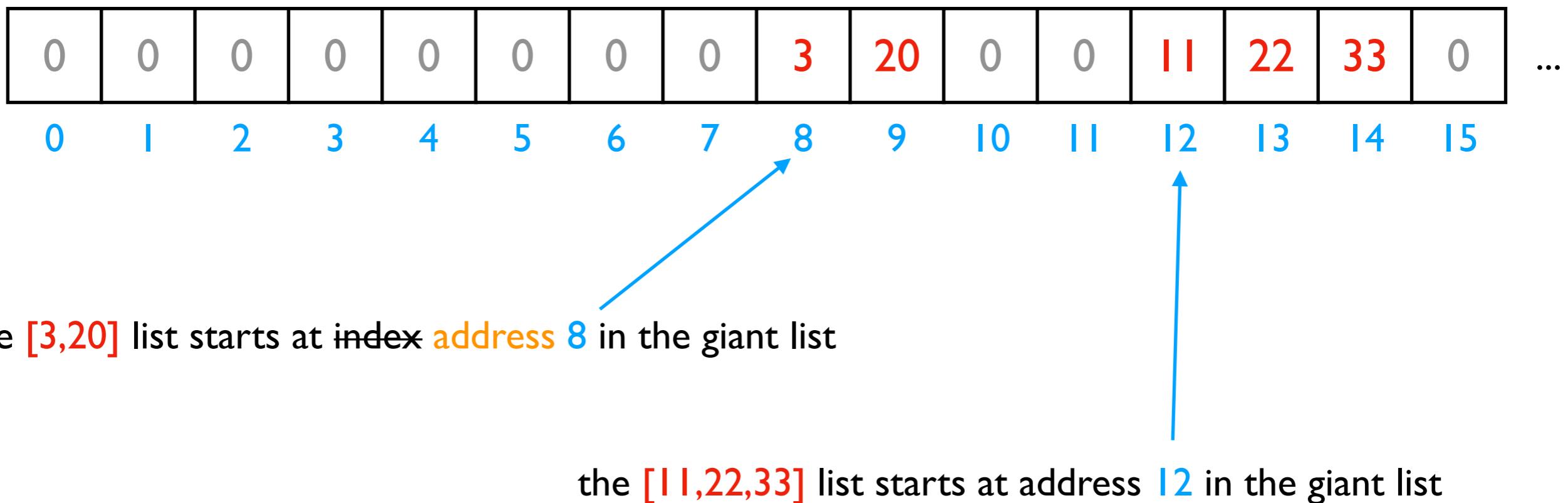
data



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



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

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

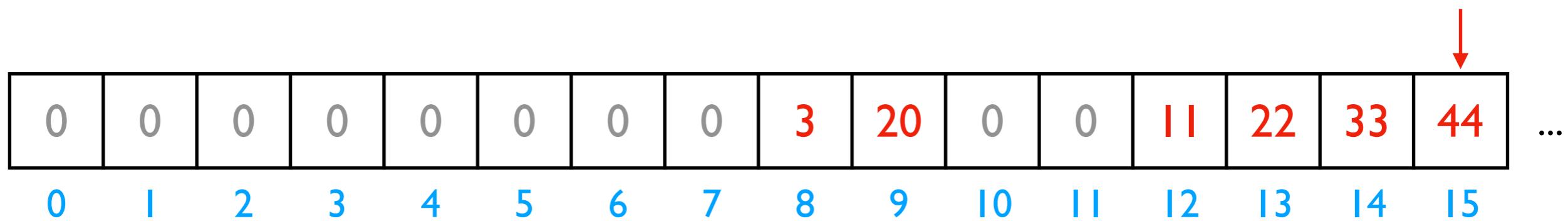
0	0	0	0	0	0	0	0	3	20	0	0	11	22	33	0	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

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

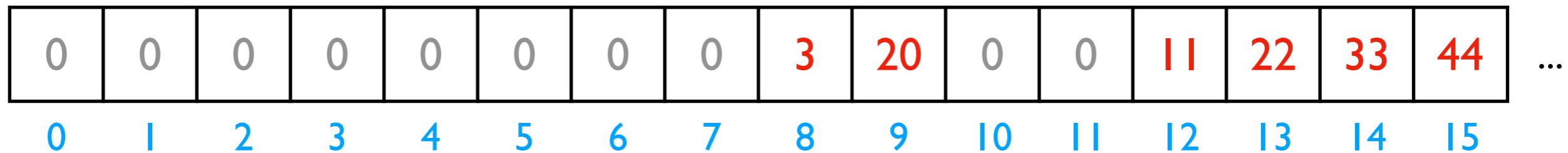


implications for performance...

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

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

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



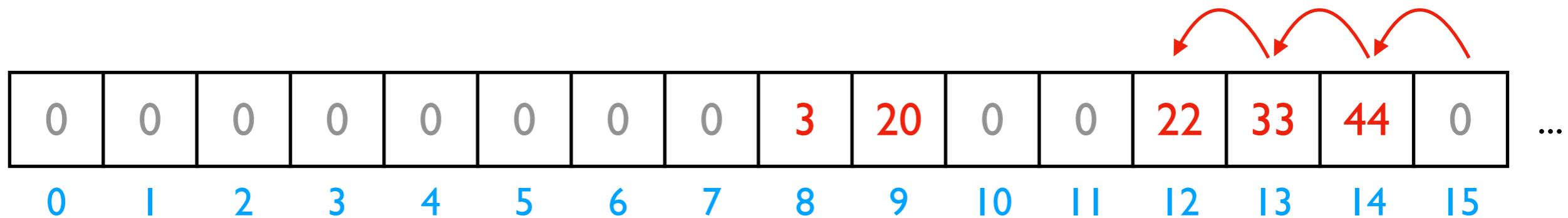
implications for performance...

```
# 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



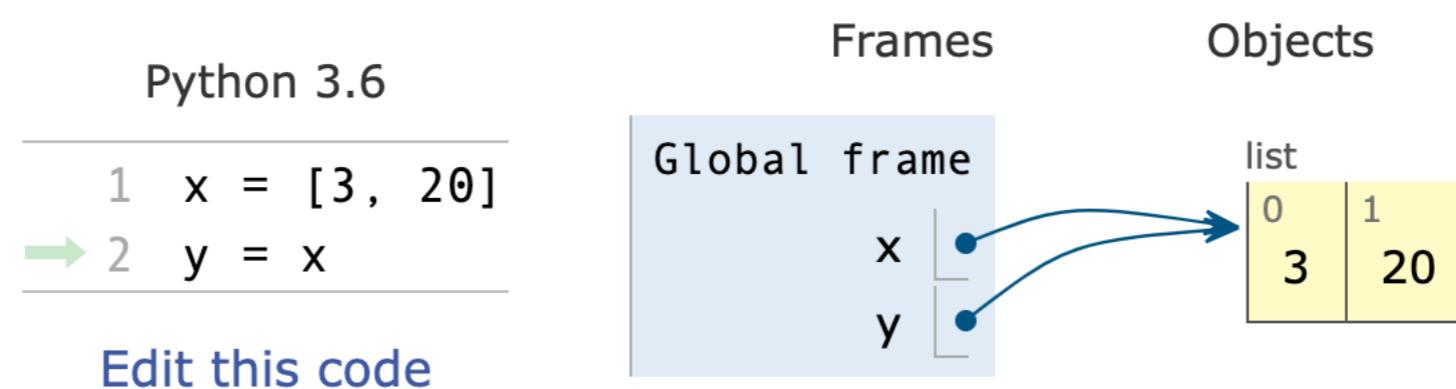
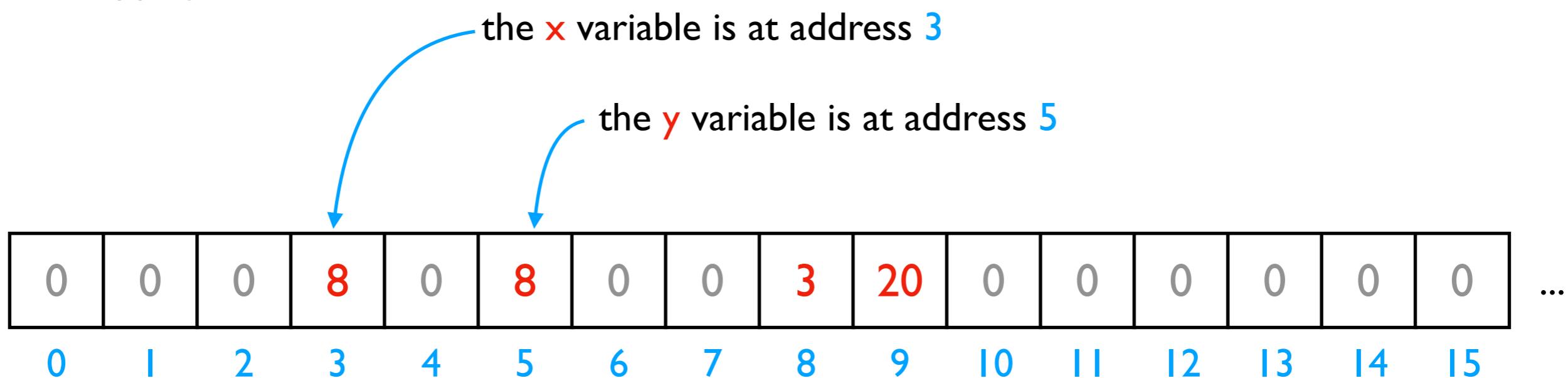
We'll think more rigorously about
performance in CS 320 (big-O notation)

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

```
# slow  
L2.pop( 0 )
```

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

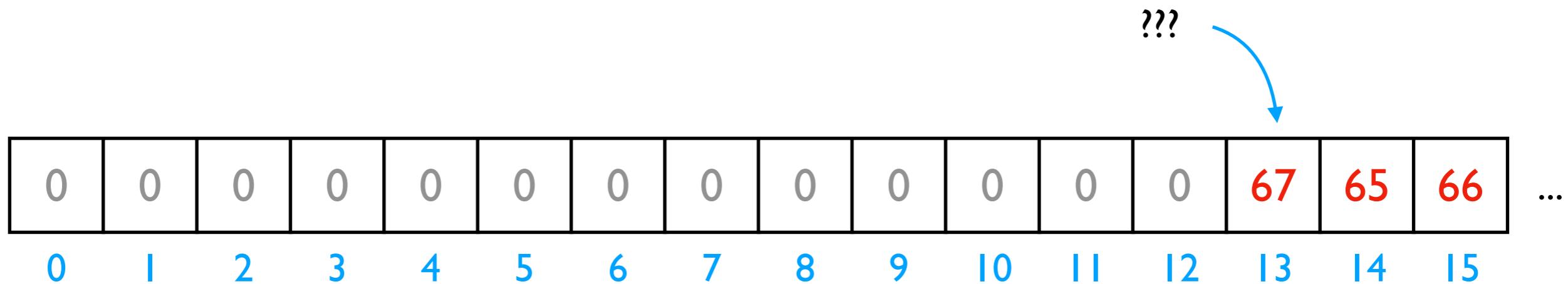
- multiple lists
- **variables and other references**
- strings
- code



PythonTutor's visualization

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

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



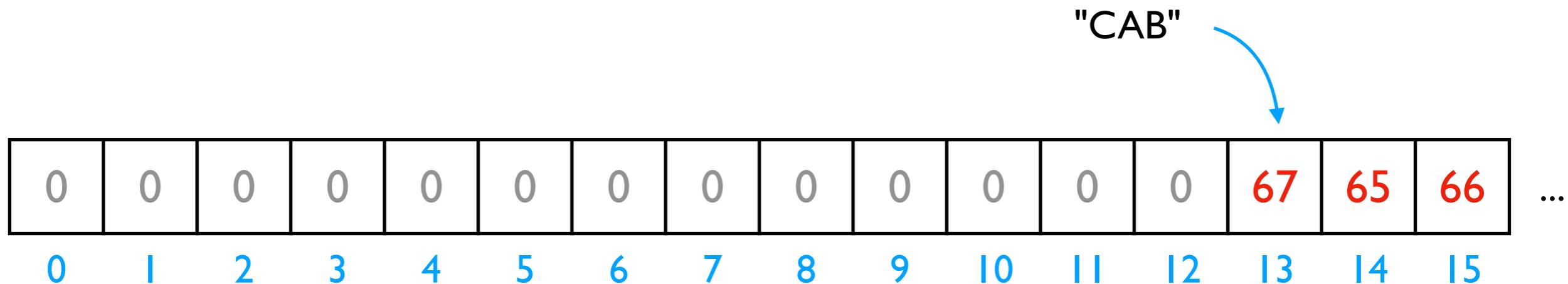
code	letter
65	A
66	B
67	C
68	D

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



encoding:	code	letter
	65	A
	66	B
	67	C
	68	D


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

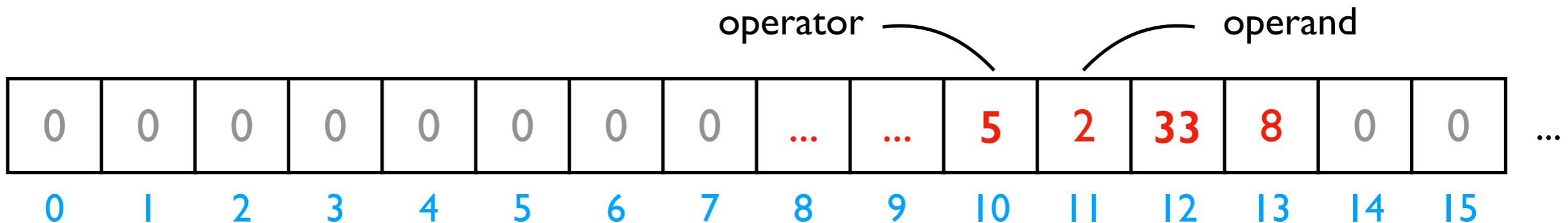
How can we use one giant list to handle the following?

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

```
while ???:
```

```
i += 2
```

```
# what line next?
```

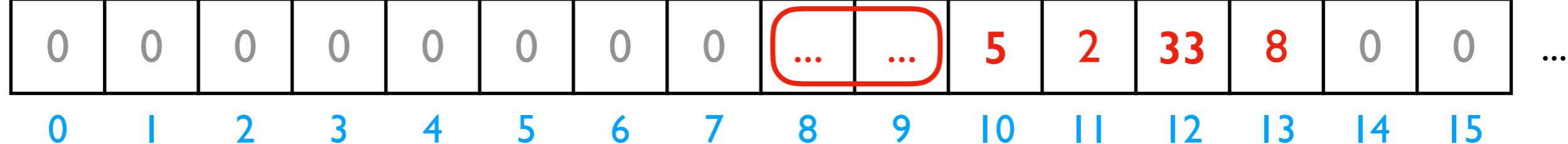
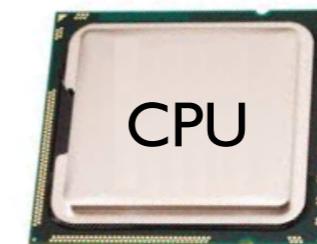


code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:

- keep track of what instruction we're on
- understand instruction codes
- much more



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

```
1 print("Hello, world!")
2 print("Hello, world!")
3 print("Hello, world!")
```

Instruction Set

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

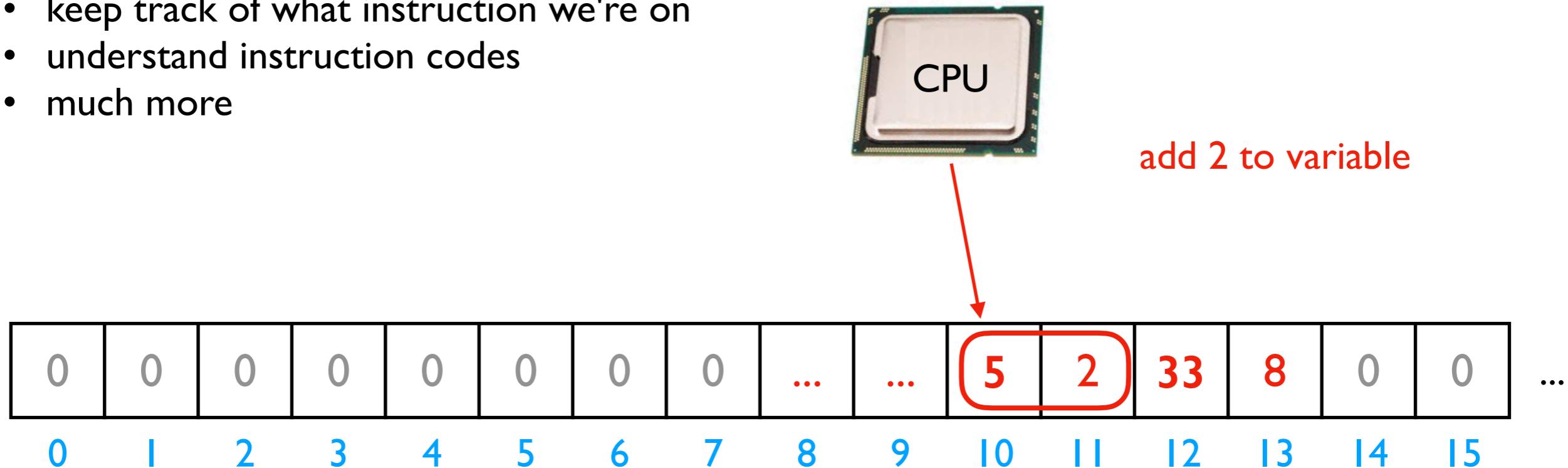
→ line that just executed

→ next line to execute

Hardware: Mental Model of CPU

CPUs interact with memory:

- keep track of what instruction we're on
- understand instruction codes
- much more

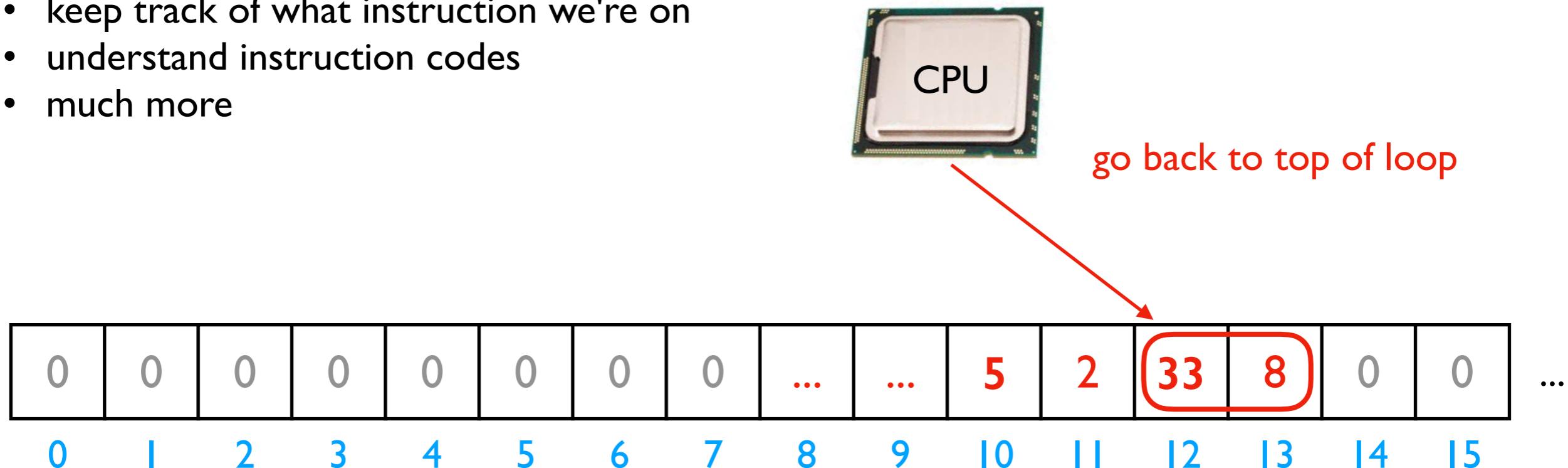


code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:

- keep track of what instruction we're on
- understand instruction codes
- much more

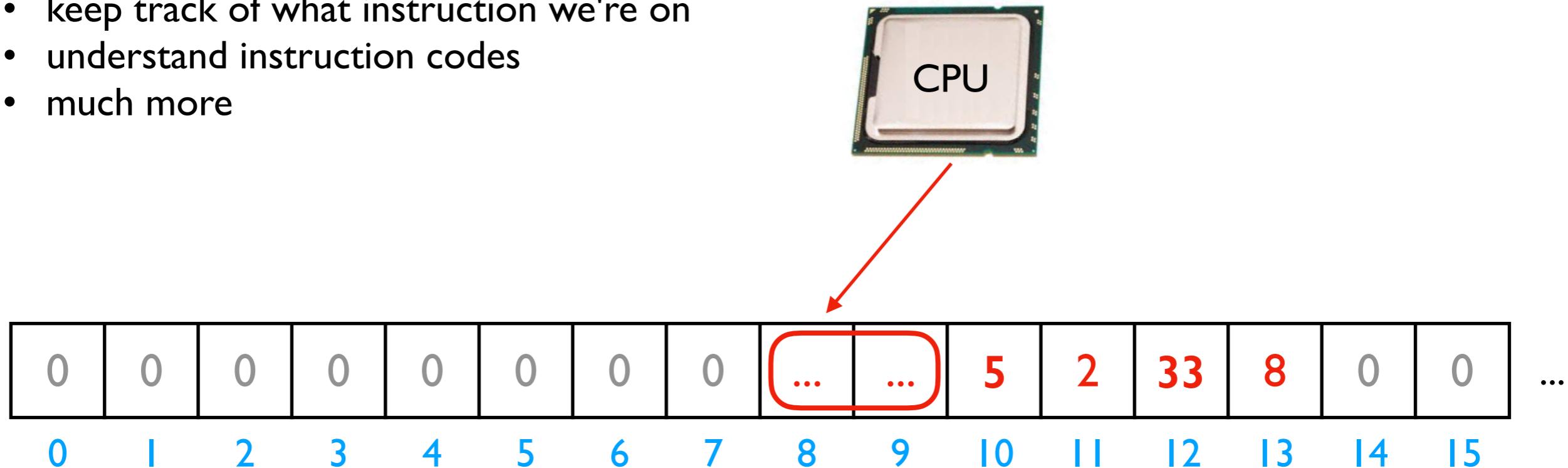


code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:

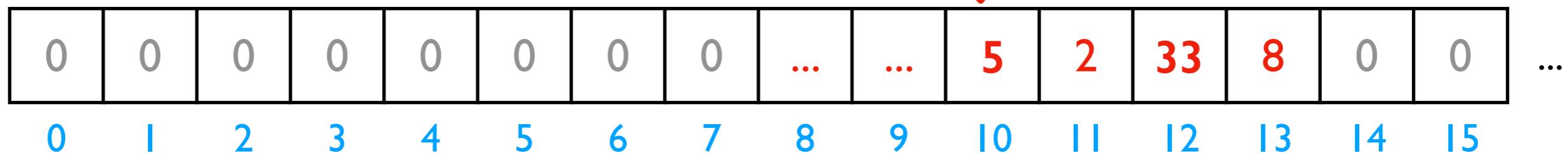
- keep track of what instruction we're on
- understand instruction codes
- much more



Instruction Set	code	operation
	5	ADD
	8	SUB
	33	JUMP

Hardware: Mental Model of CPU

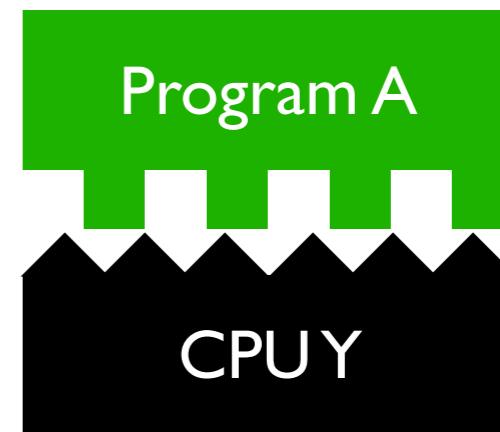
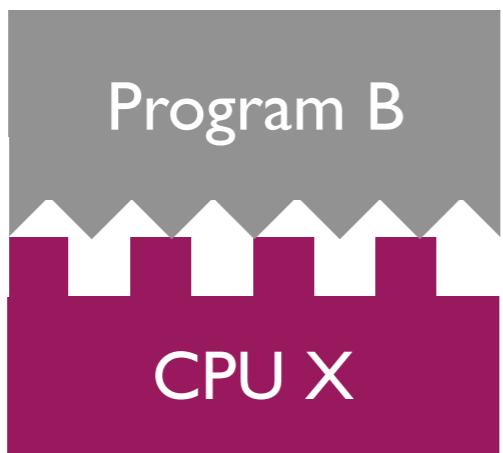
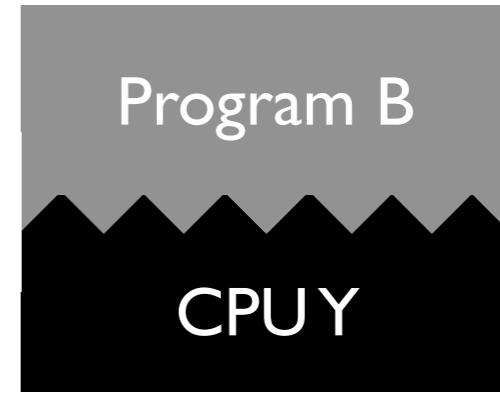
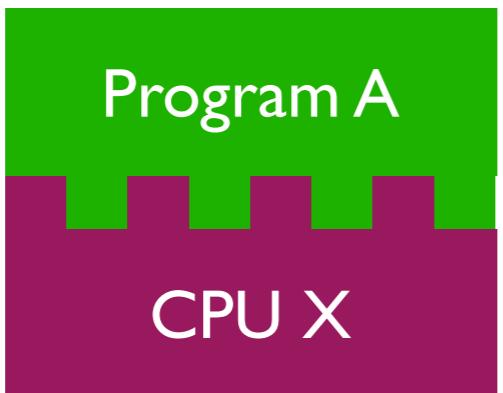
a CPU can only run programs that use instructions it understands!



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"

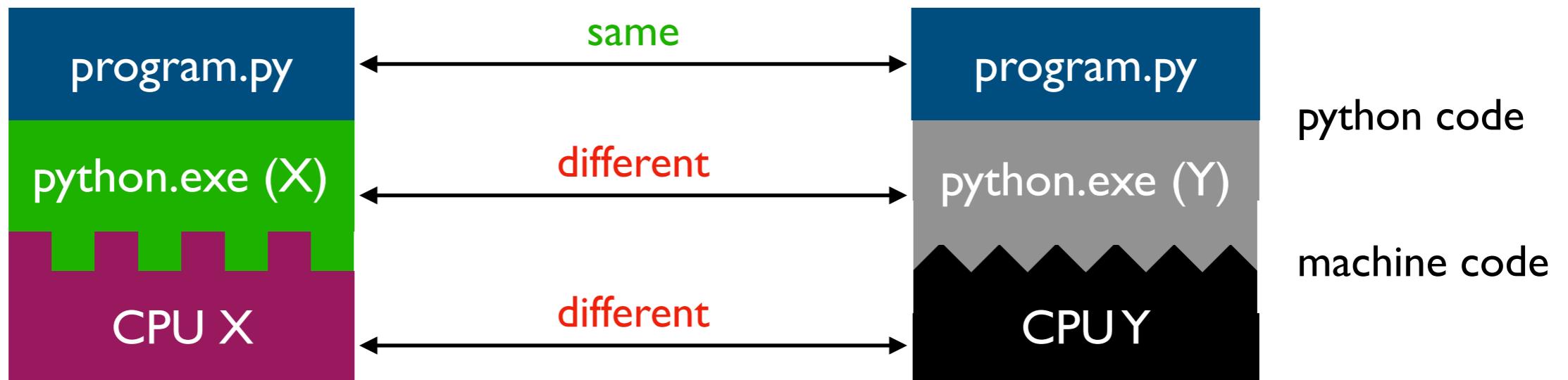


A Program and CPU need to "fit"



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

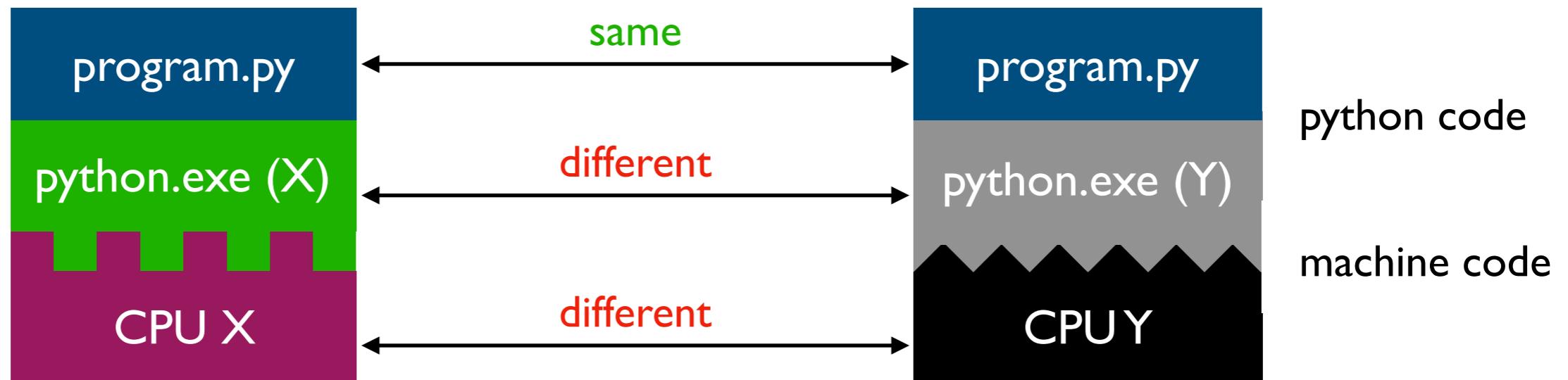
Interpreters



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:



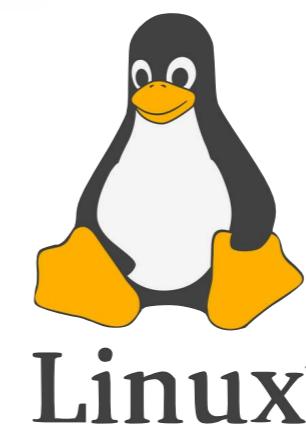
Hardware



Operating System



Dependencies ← next lecture



Linux™



ANDROID

many others...



Red Hat

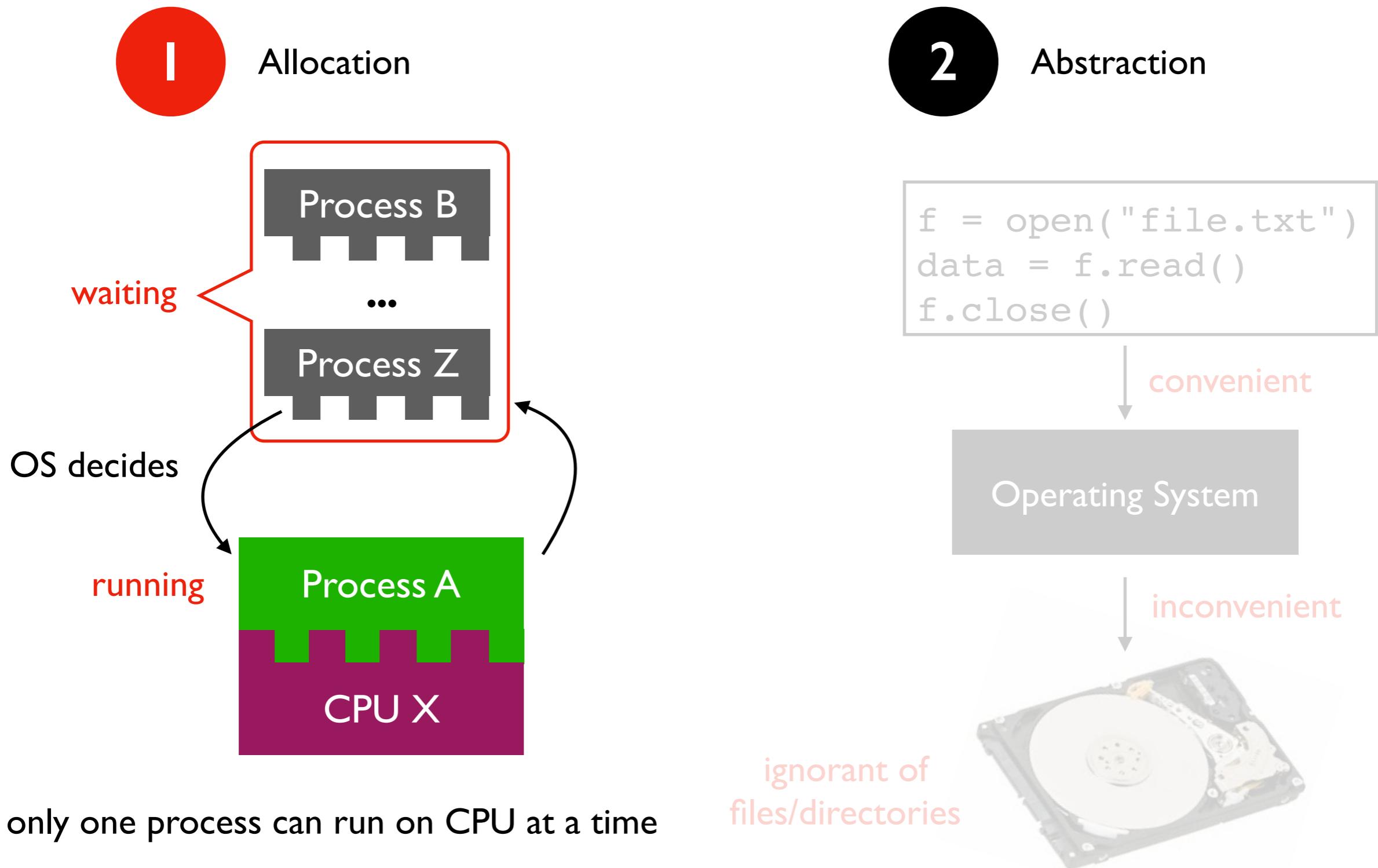


ubuntu

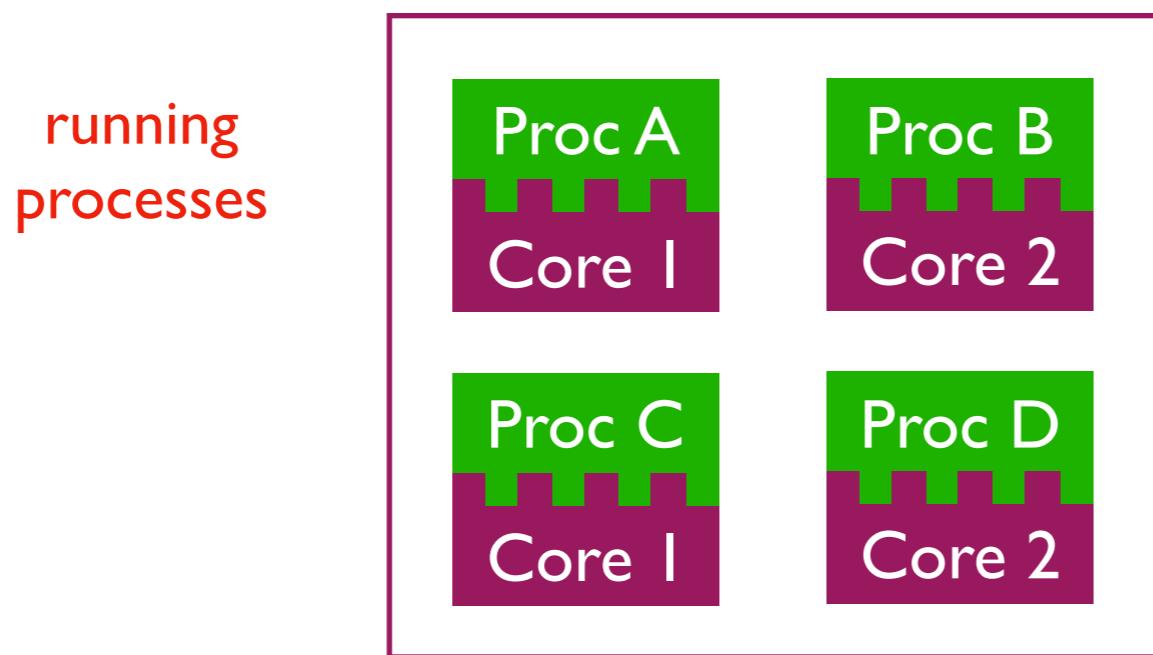
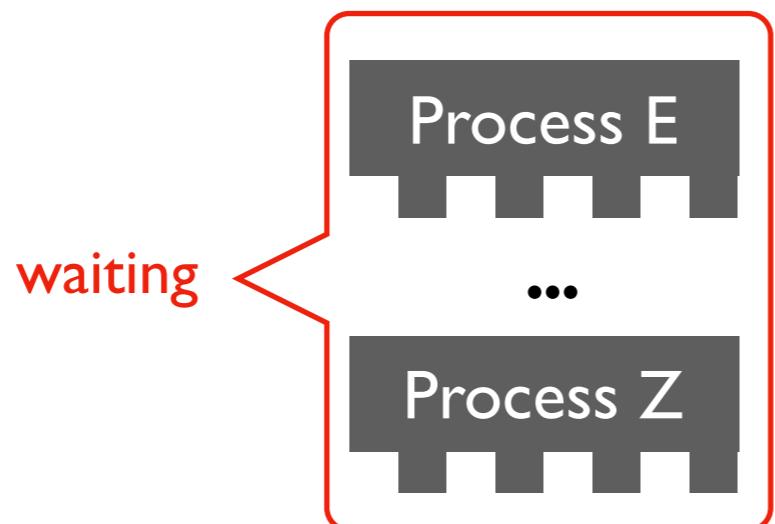
[this semester]

OS jobs: Allocate and Abstract Resources

[like CPU, hard drive, etc]



Parallelism -- more later this semester...

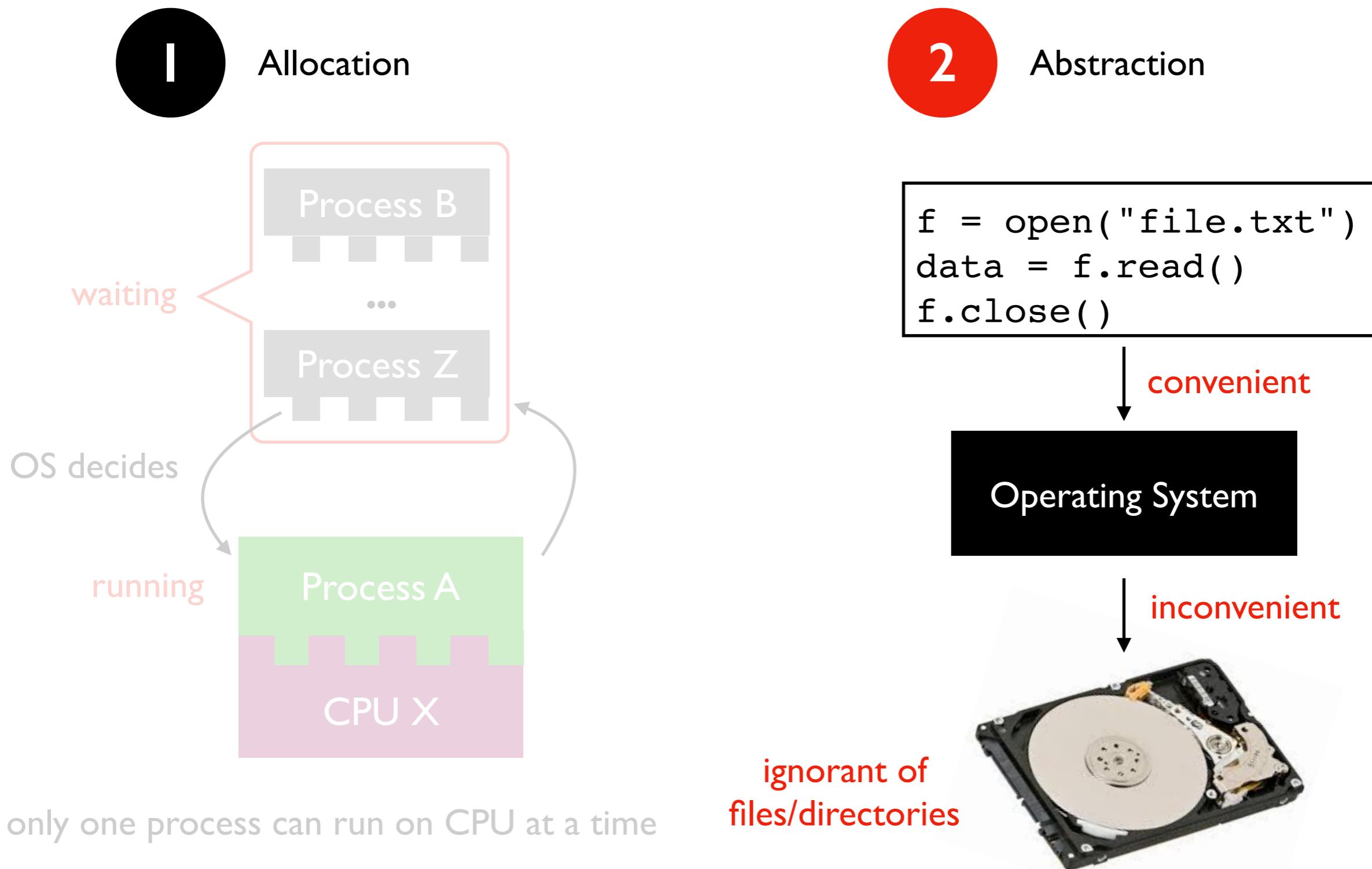


most modern CPUs actually contain multiples CPUs (called "cores") on a single chip

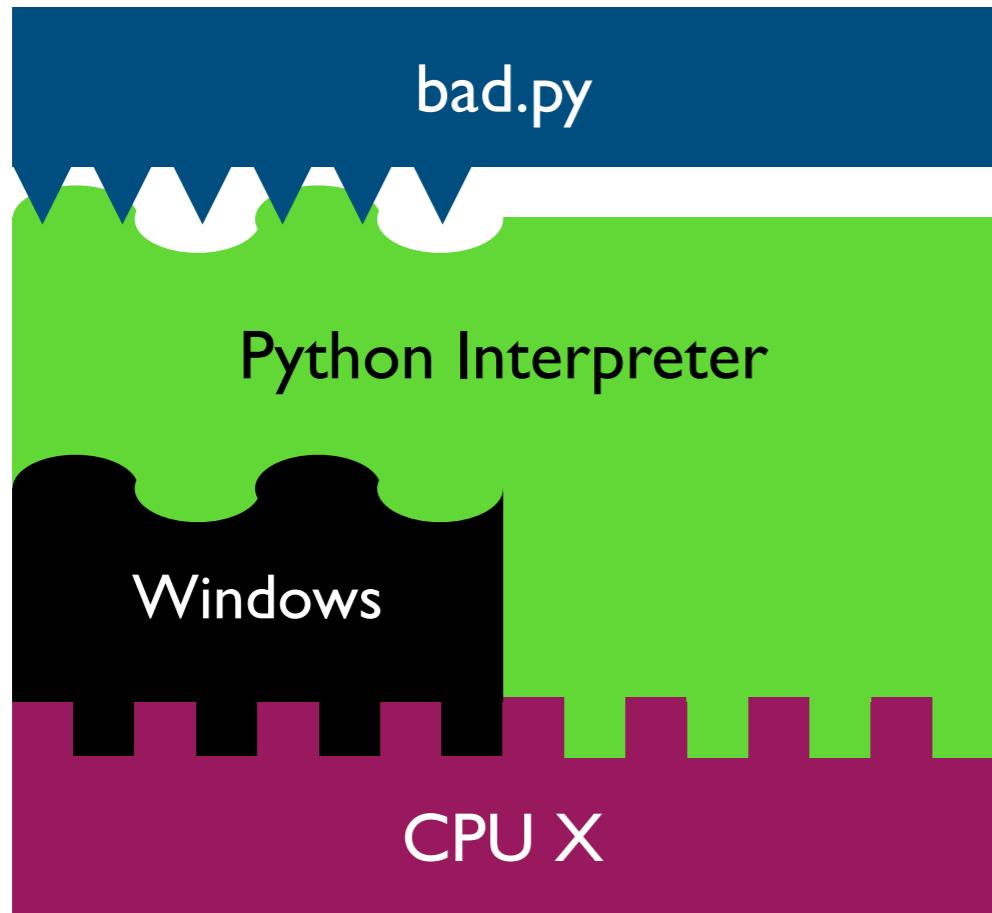
Later: how can we write programs that run in parallel, going faster by using multiple cores?

OS jobs: Allocate and Abstract Resources

[like CPU, hard drive, etc]



Harder to reproduce on different OS...

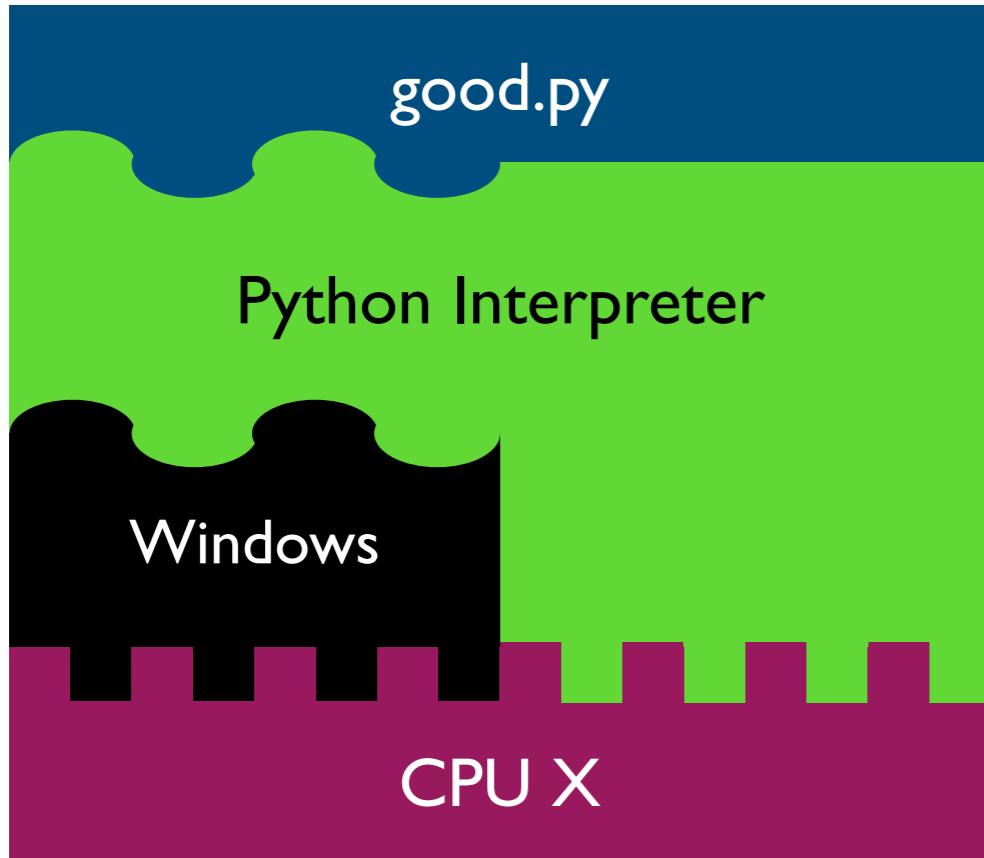


```
f = open( "/data/file.txt" )  
...
```

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...

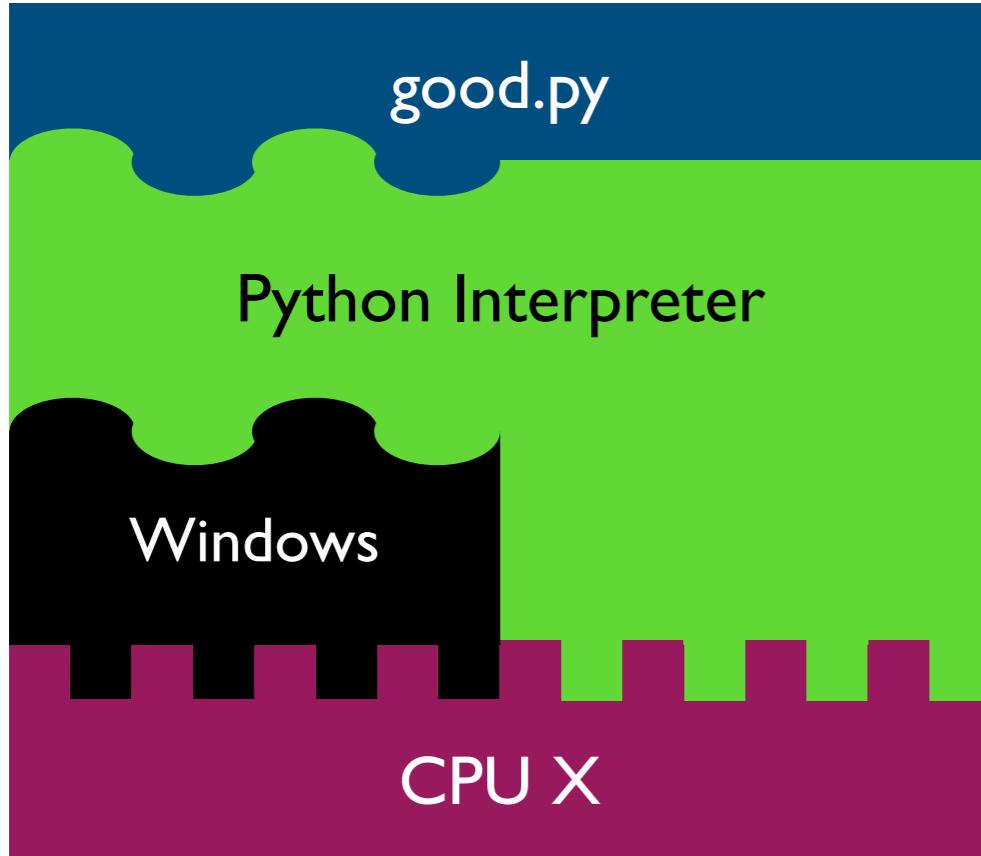


```
f = open( "c:\\data\\file.txt" )  
...  
...
```

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!

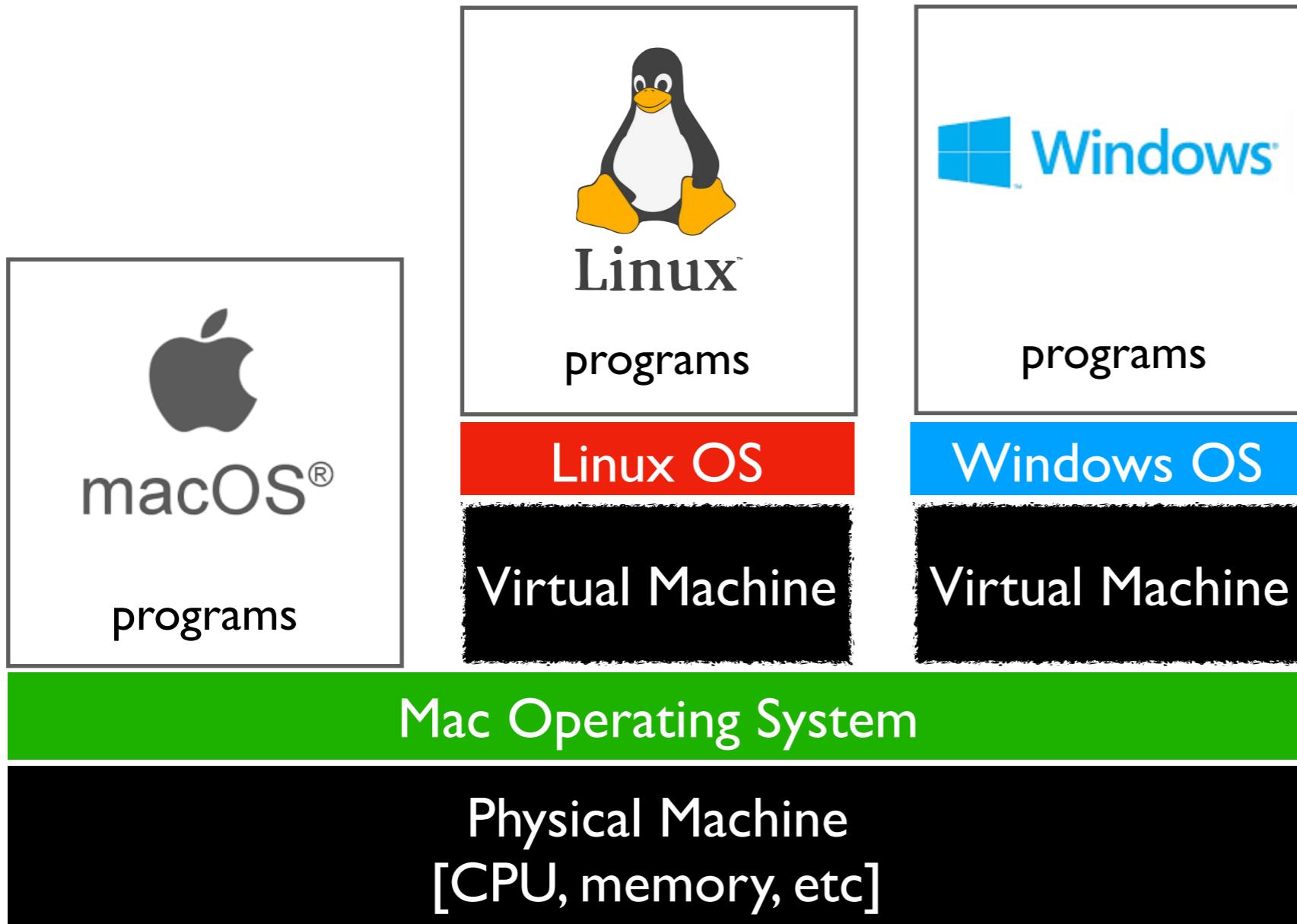
tradeoffs?

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)

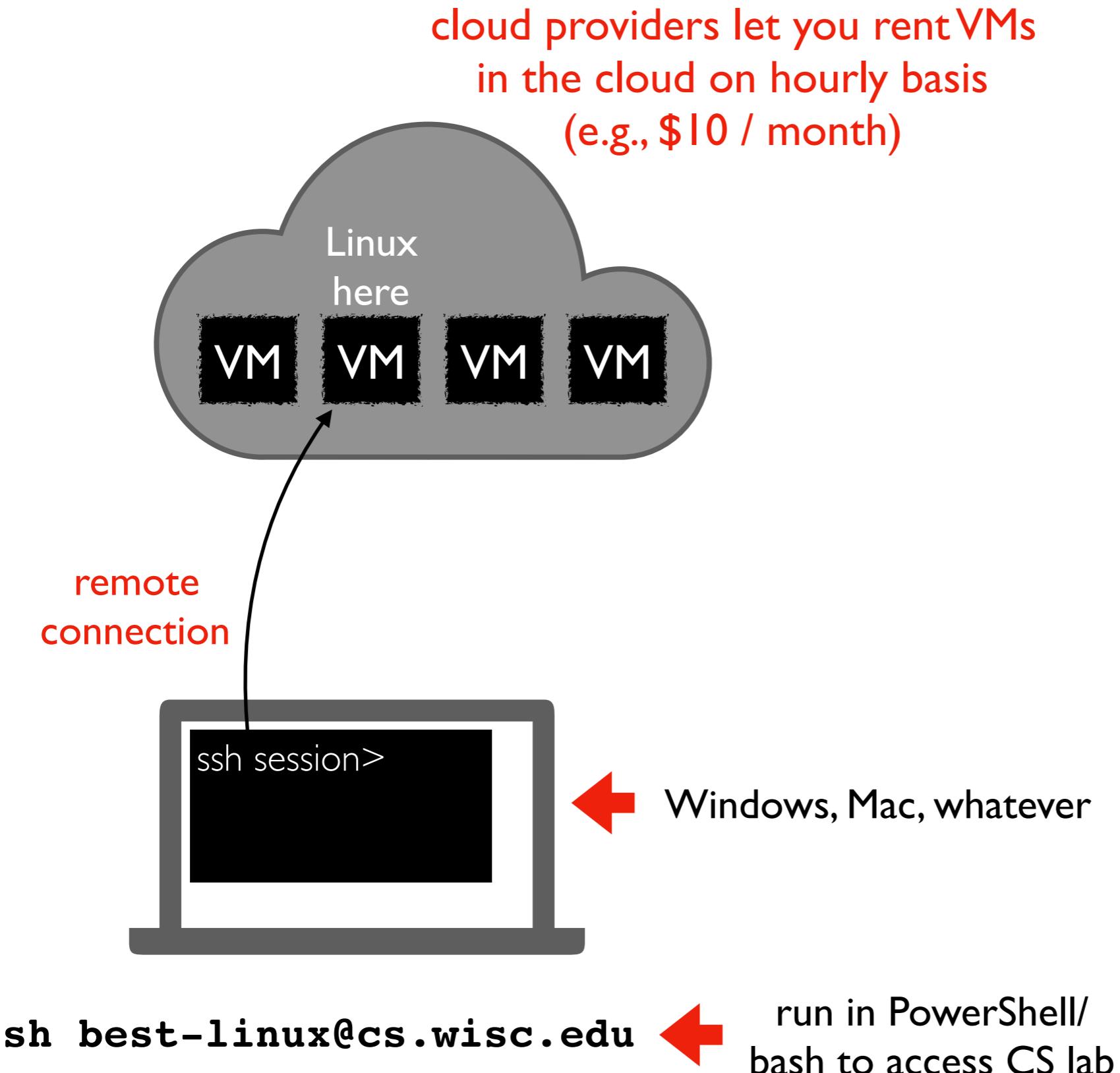
popular virtual
machine software



Microsoft
Hyper-V

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



Google Cloud Platform

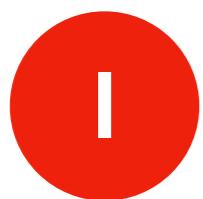


we'll use AWS virtual
machines this semester,
called "EC2" instances
[setup in Lab I]

Lecture Recap: Reproducibility

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

Things to match:



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



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



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 run 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