[320] Welcome + First Lecture (reproducibility)

Tyler Caraza-Harter

Welcome to Data Programming II!

TODO

Today's Topics

Introductions

Course overview

- Topics
- Lecture
- Lab
- Readings
- Class communication
- Grades
- Projects
- Exams

Computer hardware basics

Website

Today's Lecture:

Reproducibility

Reproducibility

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

/ˌrēprəˌd(y)o͞osəˈbilədē/

noun

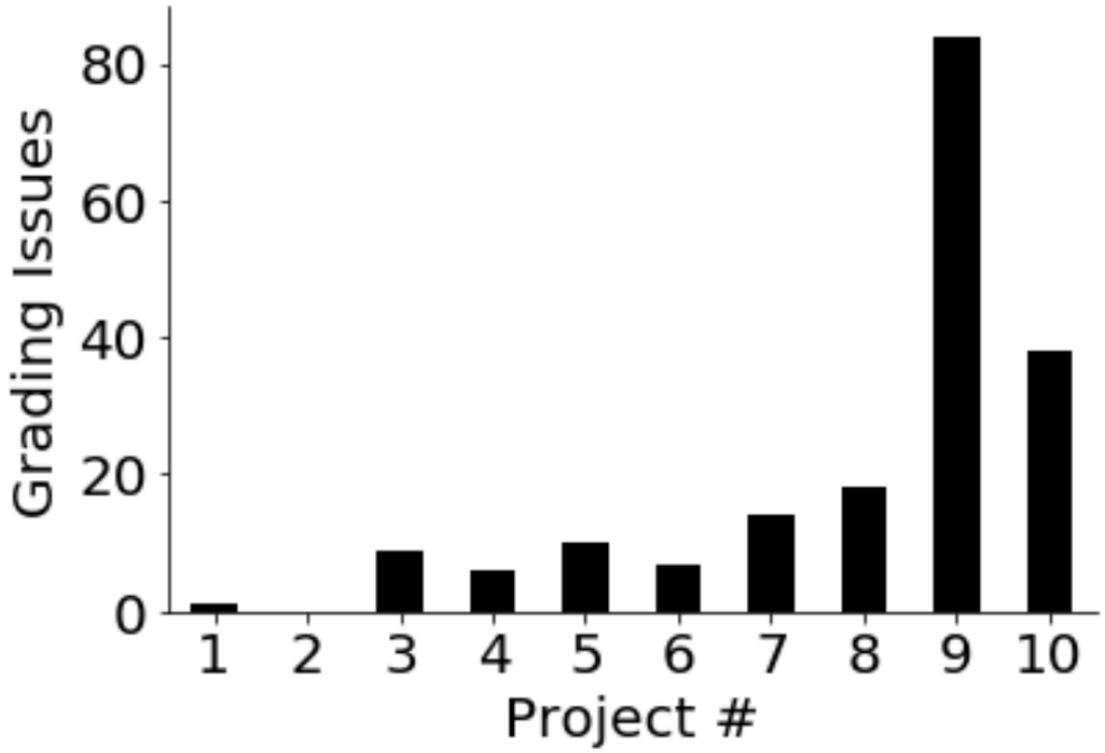
noun: **reproducibility**

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"

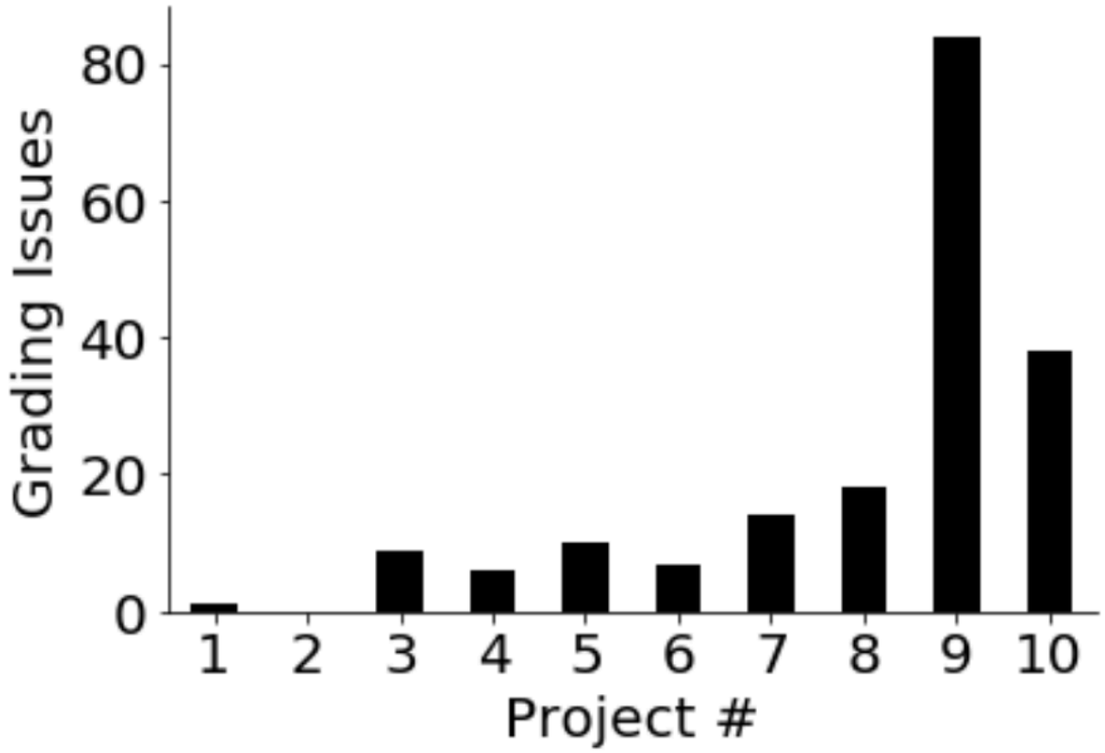
Discuss: *how might we define "reproducibility" for a data scientist?*

Reproducibility (Fall 19 Grading for CS 301)



why was project 9 so problematic?

Reproducibility (Fall 19 Grading for CS 301)



why was project 9 so problematic?

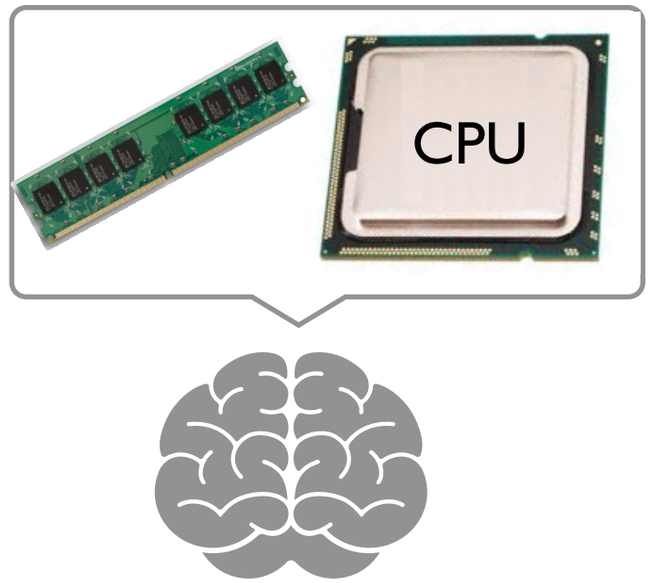
Windows+UNIX:

- / *vs* \
- os.listdir order

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

Things to match:

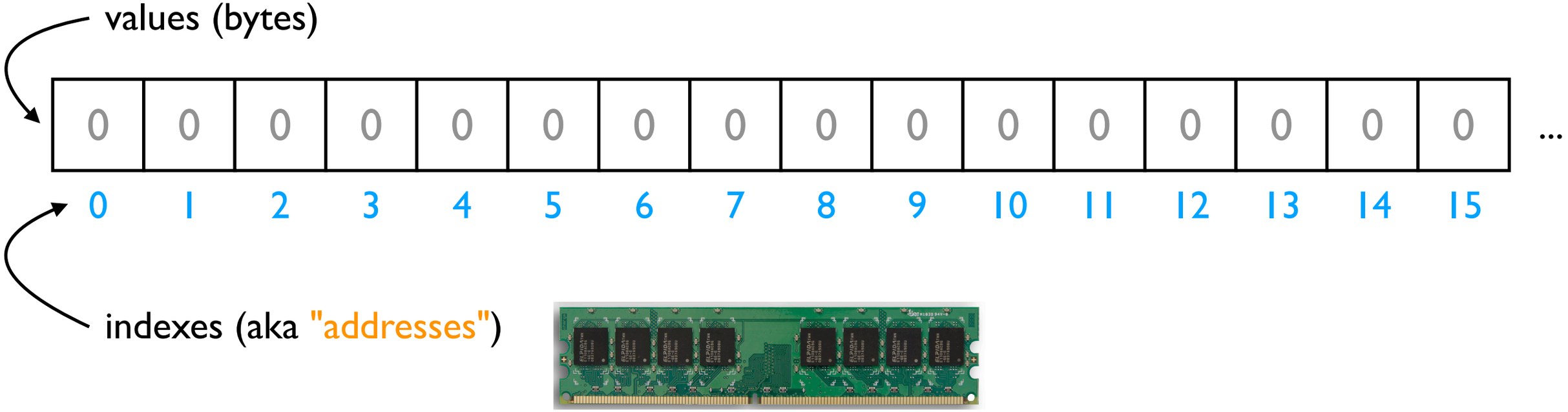
1. Hardware
2. Operating System
3. Dependencies ← next lecture



Hardware: Mental Model of Memory

Imagine...

- one huge list, per ~~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

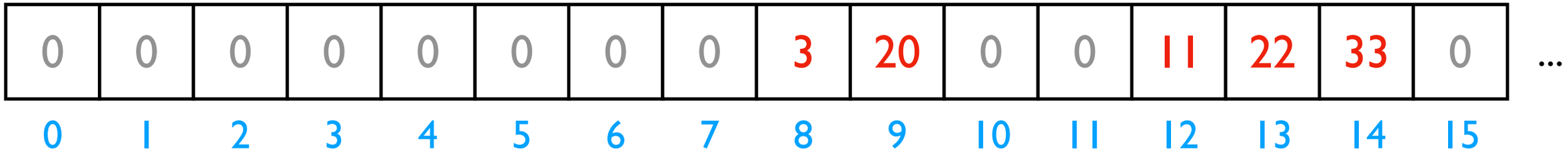
data

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

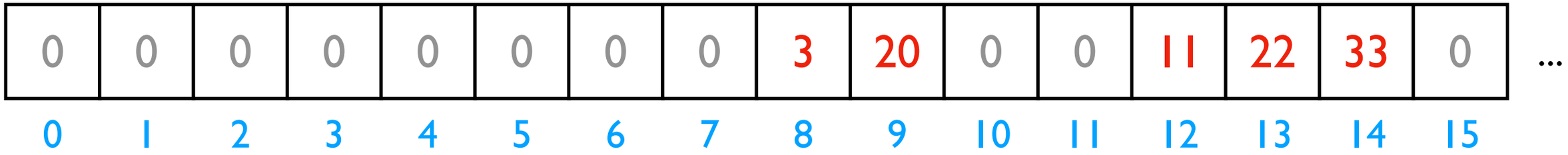


the [3,20] list starts at ~~index~~ address 8 in the giant list

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

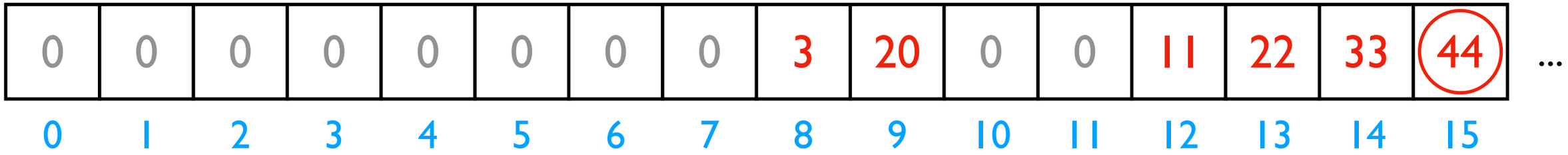


implications for performance...

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

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

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

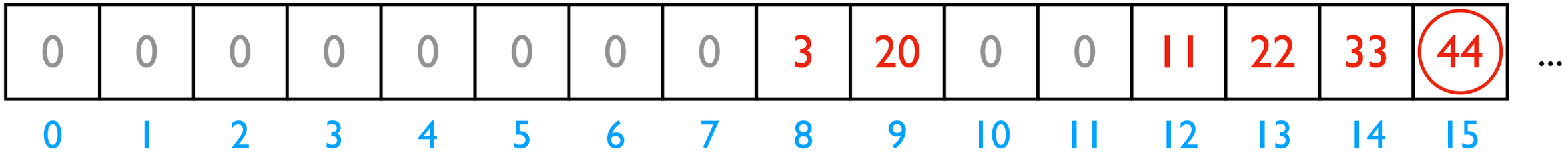


implications for performance...

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

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

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



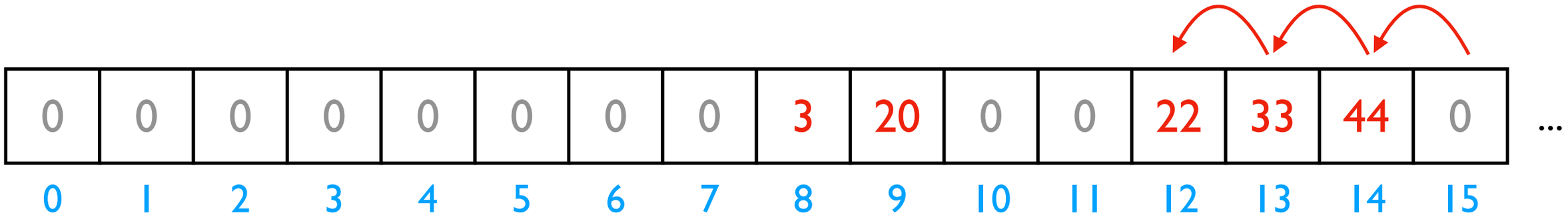
implications for performance...

```
# 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



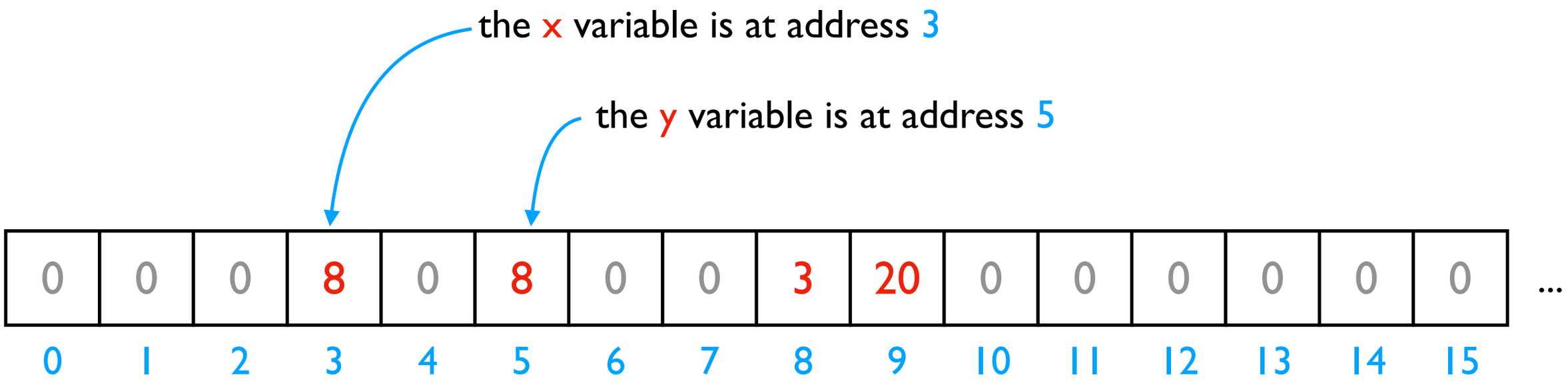
implications for performance...

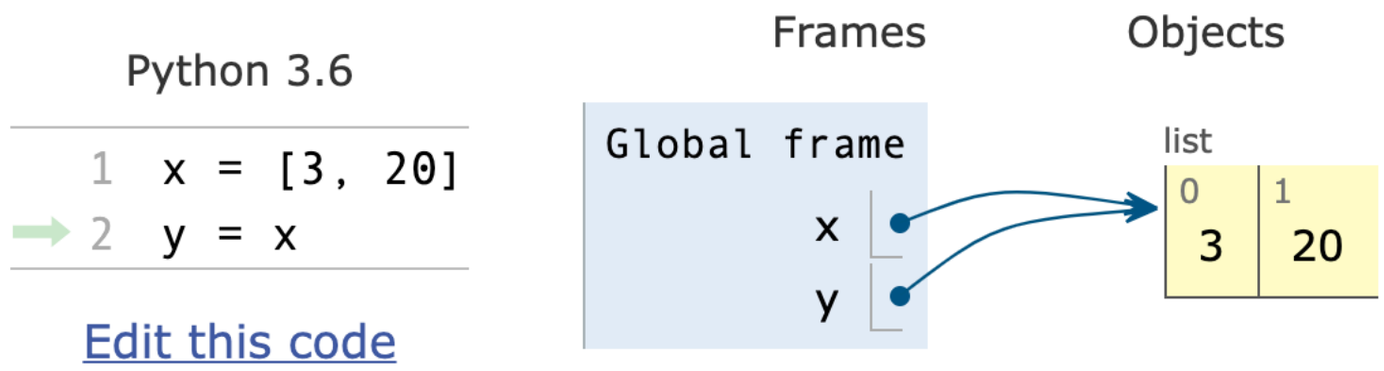
```
# fast
L2.append(44)

# slow
L2.pop(0)
```

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

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

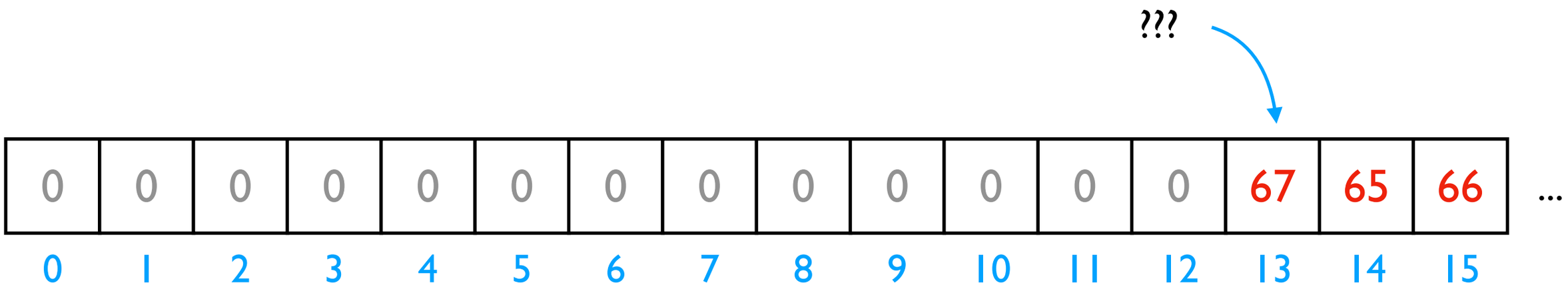




PythonTutor's visualization

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

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



encoding:

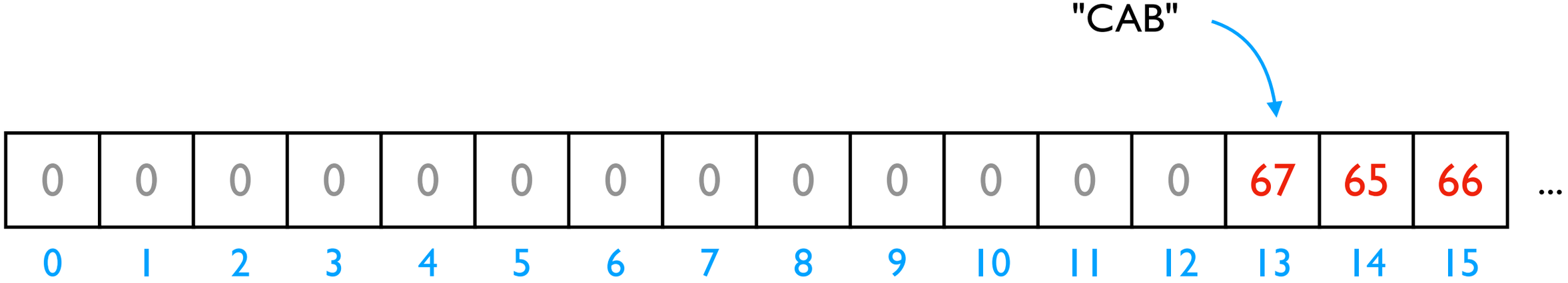
code	letter
65	A
66	B
67	C
68	D
...	...

```
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

"CAB"



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	

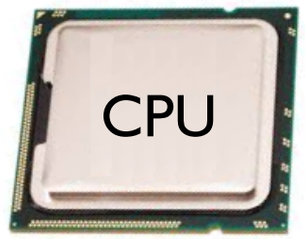
Instruction Set

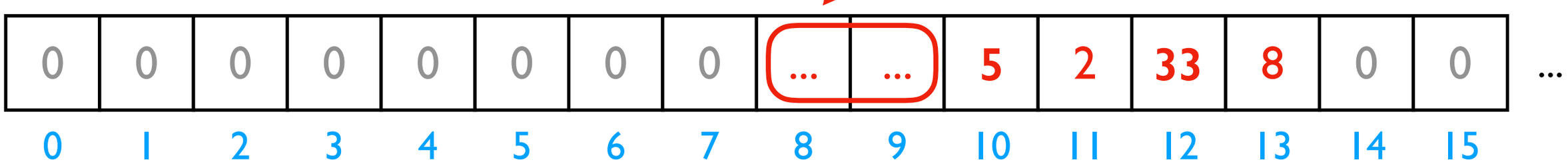
code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:

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





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

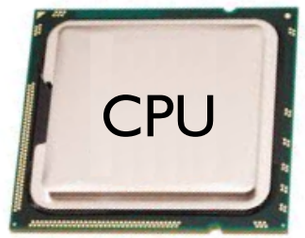
Instruction Set

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

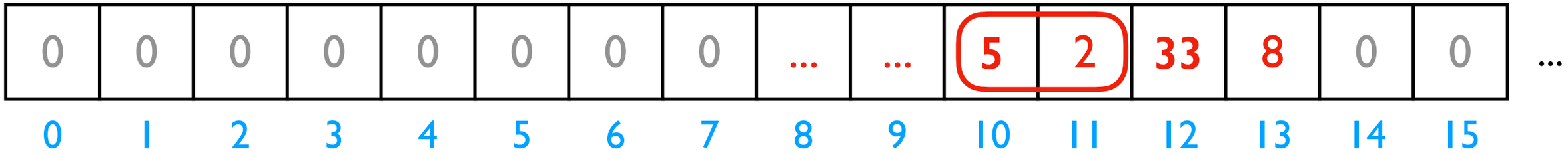
Hardware: Mental Model of CPU

CPUs interact with memory:

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



add 2 to variable



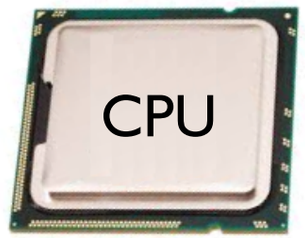
Instruction Set

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

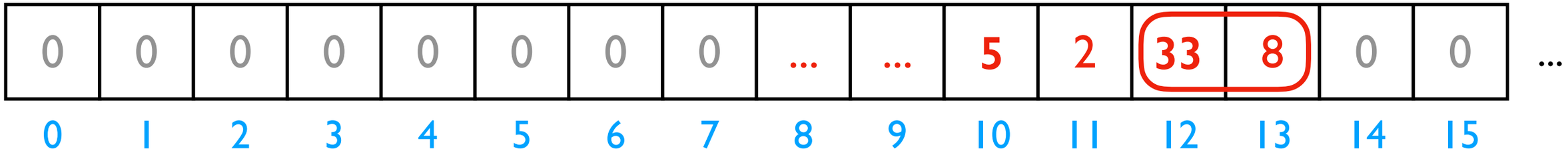
Hardware: Mental Model of CPU

CPUs interact with memory:

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



go back to top of loop



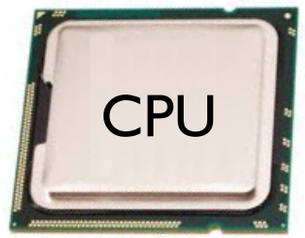
Instruction Set

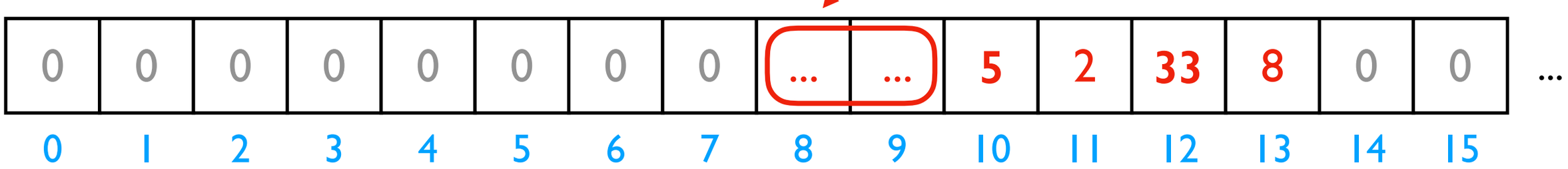
code	operation
5	ADD
8	SUB
33	JUMP
...	...

Hardware: Mental Model of CPU

CPUs interact with memory:

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



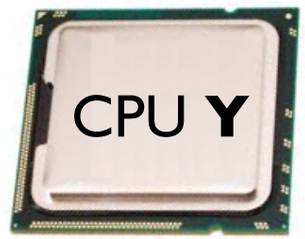


Instruction Set

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

Hardware: Mental Model of CPU

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



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

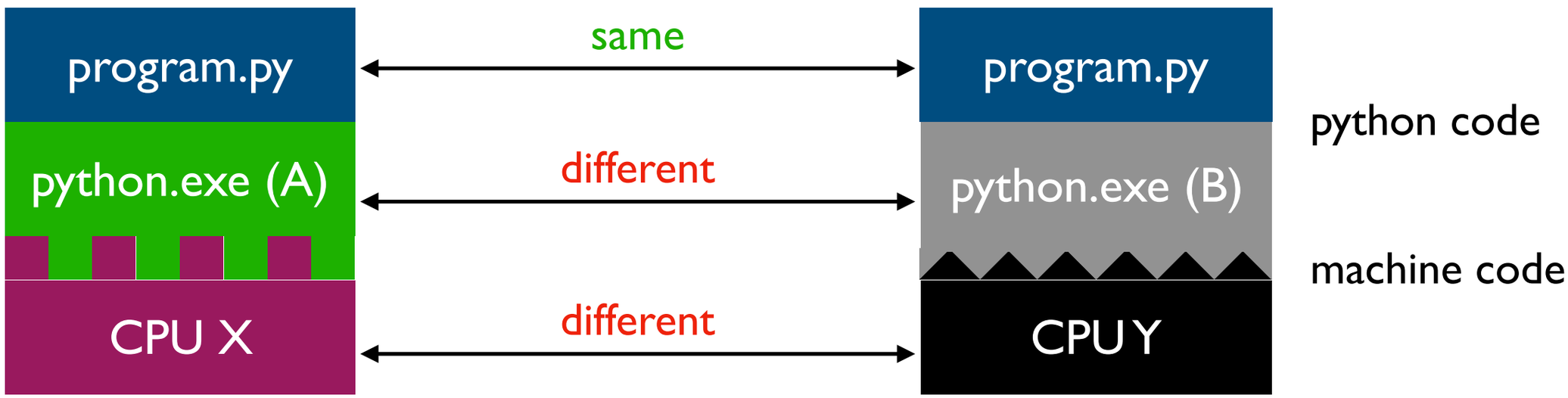
code	operation
5	SUB
8	ADD
33	undefined
...	...

A Program and CPU need to "fit"

A Program and CPU need to "fit"

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

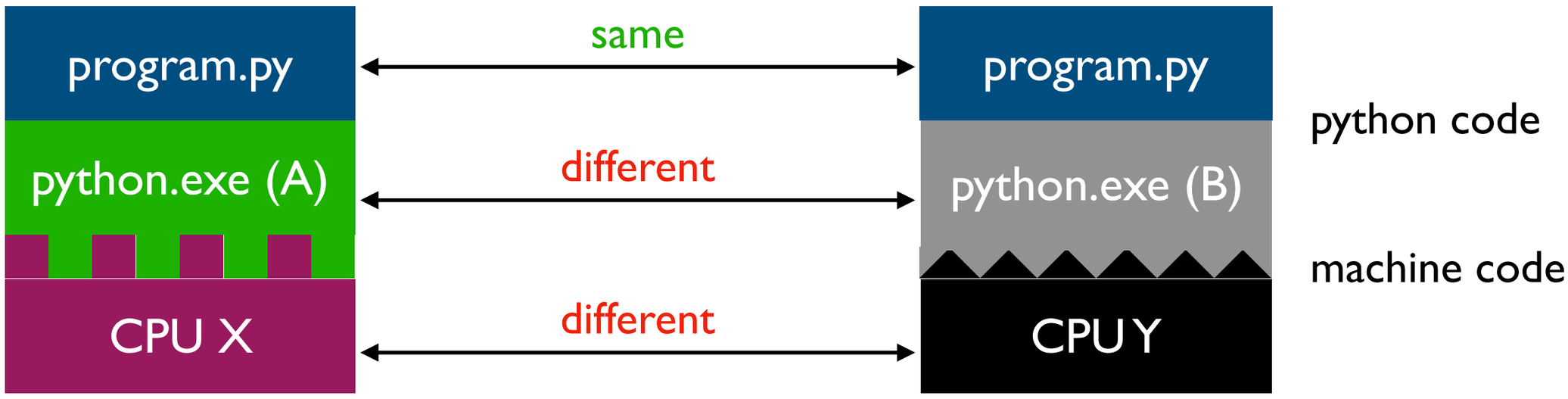
Interpreters



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



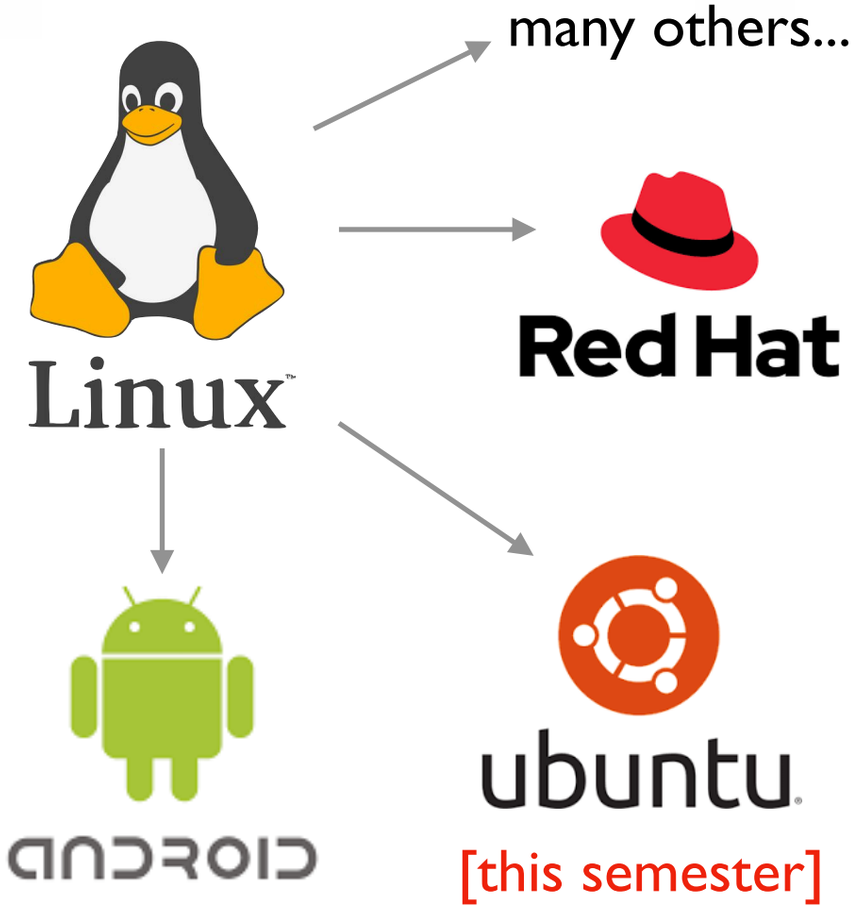
Interpreters (such as python.exe) make it easier to run the same code on different machines

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

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

Things to match:

1. Hardware
2. Operating System
3. Dependencies ← next lecture



OS jobs: Abstract and Allocate Resources

1 Abstraction

2 Allocation

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

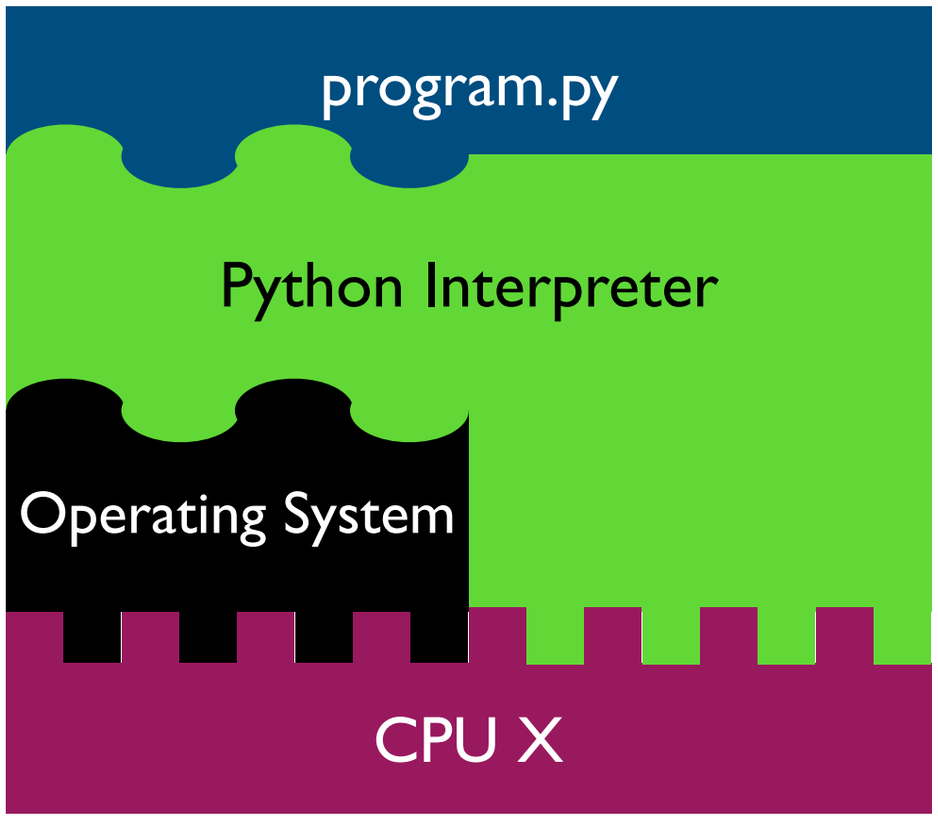
simple

Operating System

complex

ignorant of files/directories

Interpreters



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.

Terms

Memory
CPU

Operating System
Virtual Machine