Why Does My Computer Do That? Intro to Coding with Python– Intro to Python

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

Plan for Today

- Intro to Python programming language
- Intro to pair programming
- Intro to repl.it

Recap

What do you remember from Friday's class?

Recap: the good news

- “High level” programming languages like Python mean we don’t have to write in “low level” binary

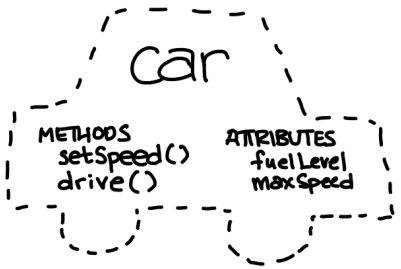
- Instead, we write statements like:

```
print(“hello”)
```

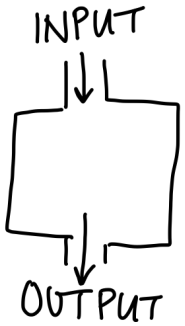
multi-paradigm
interpreted language
with dynamic typing
and automatic memory management

python™

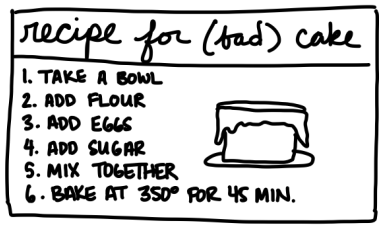
multi-paradigm
interpreted language
with dynamic typing
and automatic memory management



object-oriented



functional



imperative

declarative

multi-paradigm
interpreted language
with dynamic typing
and automatic memory management

Quick digression

	COMPILER	INTERPRETER
What it takes in:		
What it returns:		
Relative speed:		
Memory usage:		
Work is done:		
Reports errors:		
Example language:		

multi-paradigm
interpreted language
with dynamic typing
and automatic memory management

more about this
a bit later

Core Concepts to Get Us Started

Pair Programming

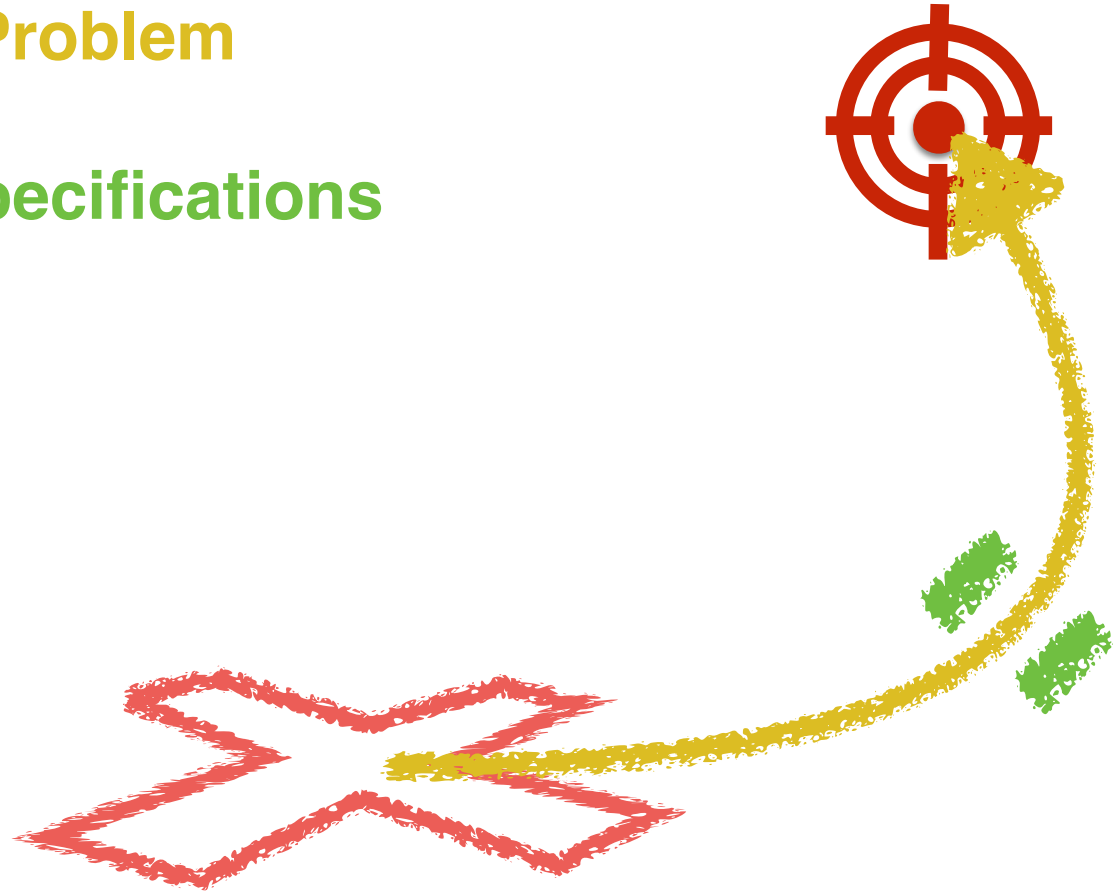
The programming process

The programming process (idealized)

- Analyze the **Problem**

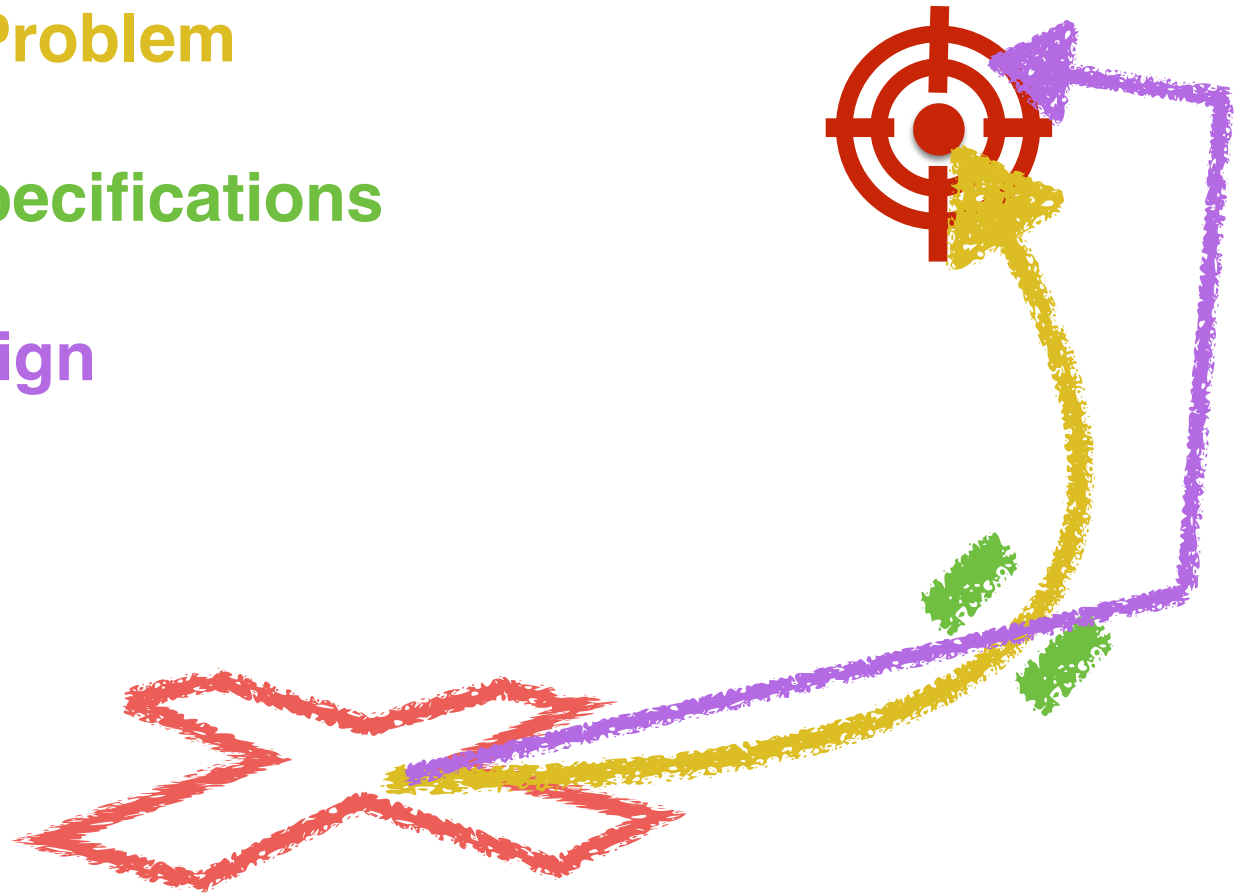
The programming process (idealized)

- Analyze the **Problem**
- Determine **Specifications**



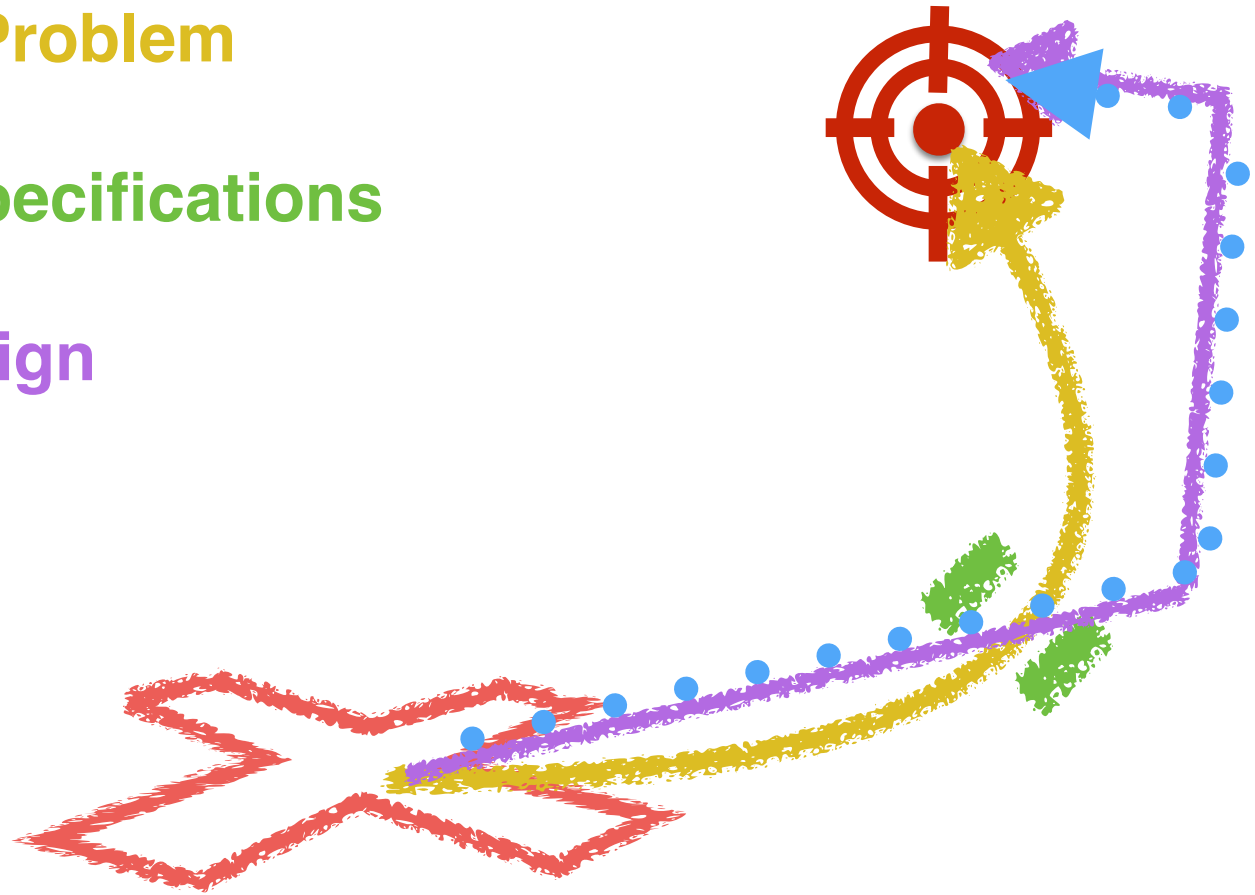
The programming process (idealized)

- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**



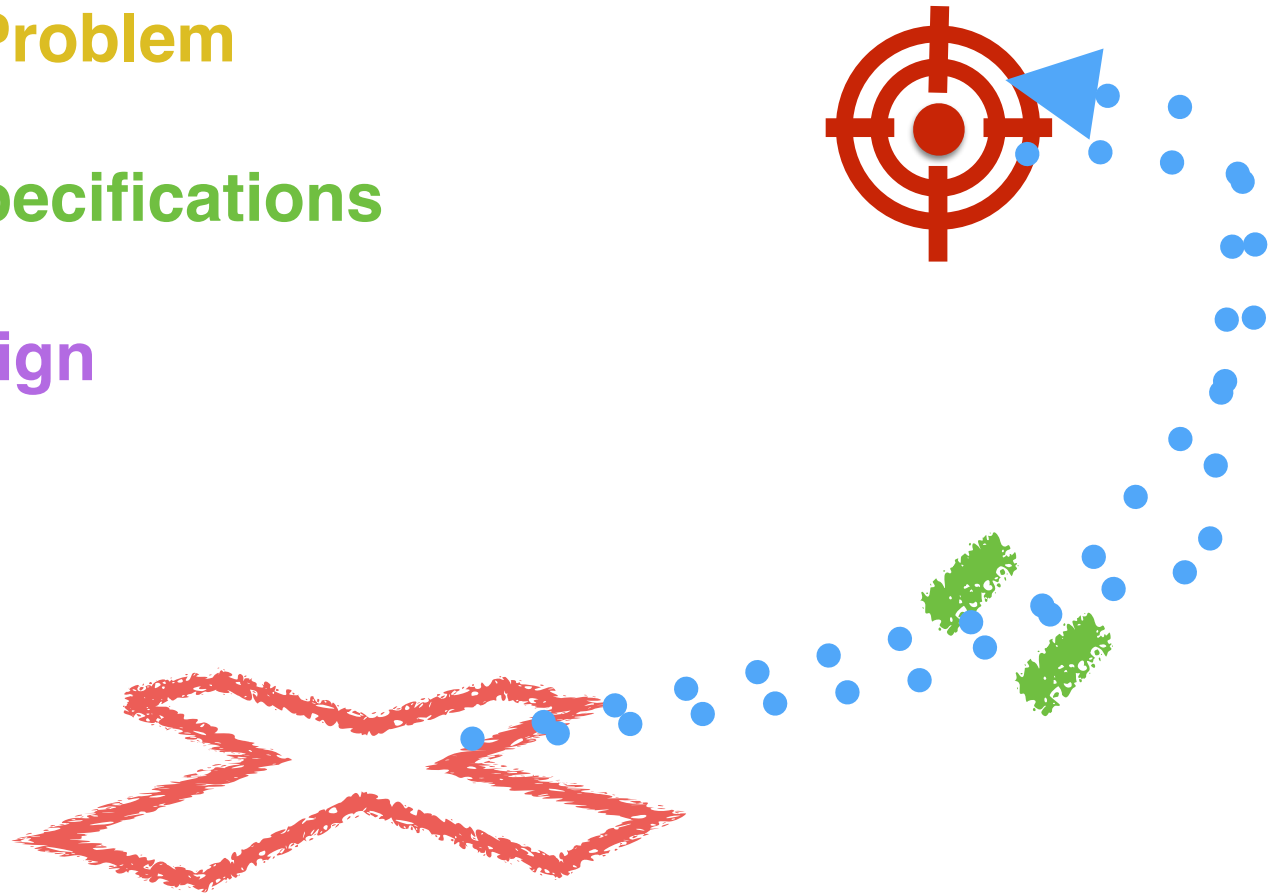
The programming process (idealized)

- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**
- **Implement**



The programming process (idealized)

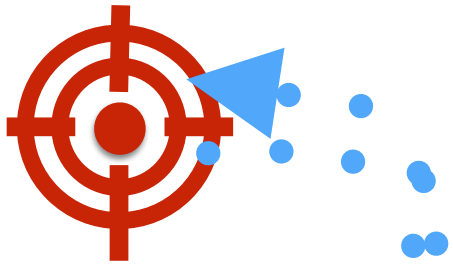
- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**
- **Implement**
- Test & Debug

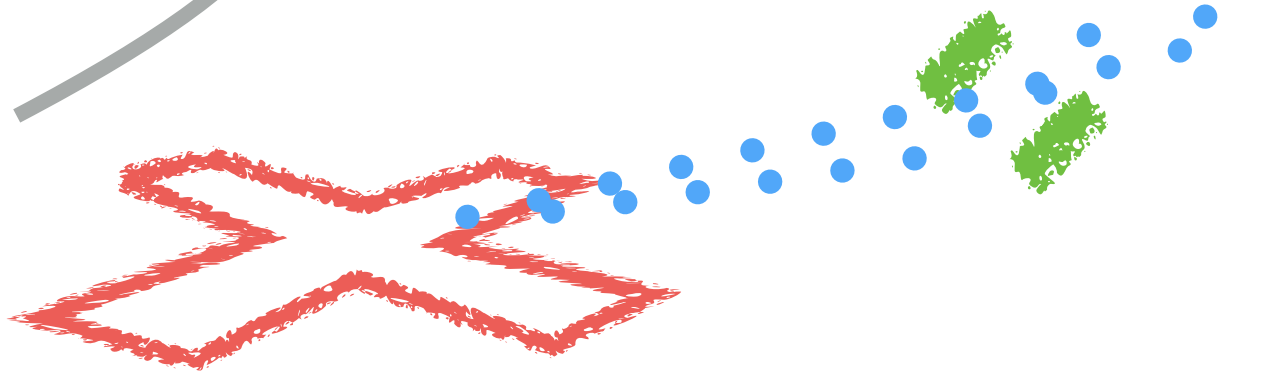


The programming process (more realistic)

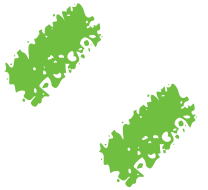
- Analyze the **Problem**
- Determine **Specifications**
- *Refine the* ~~Create a~~ **Design**
- **Implement**
- Test & Debug

iterate many times



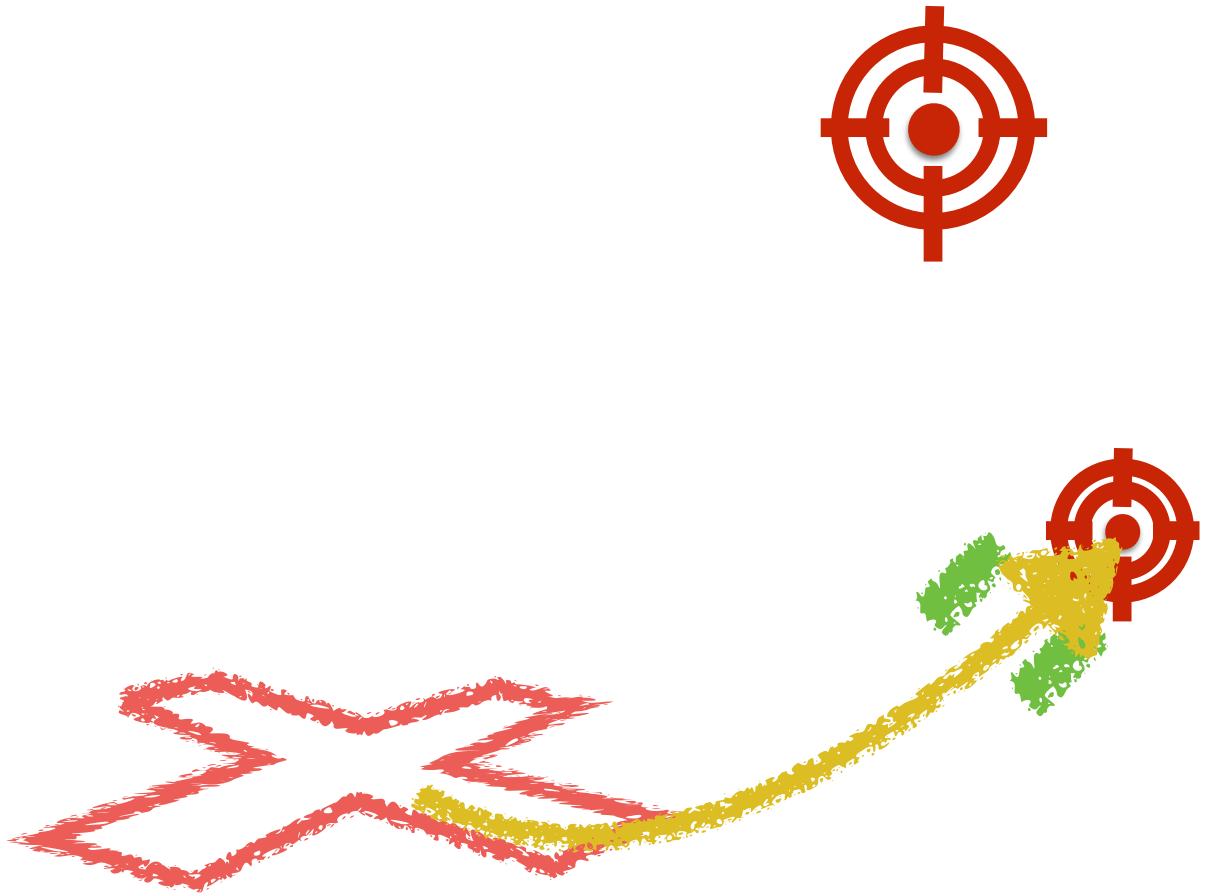


Getting started

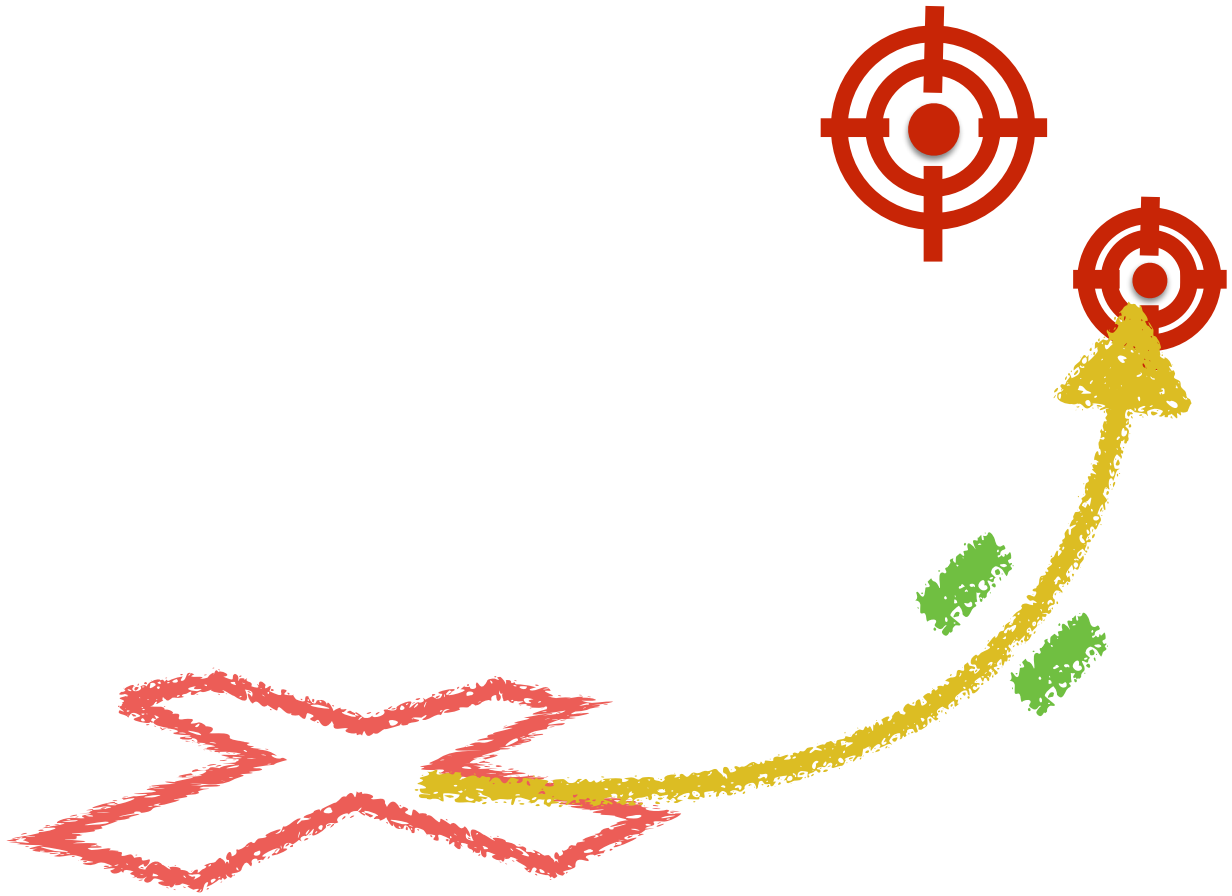


"S4": start small | slow | simple

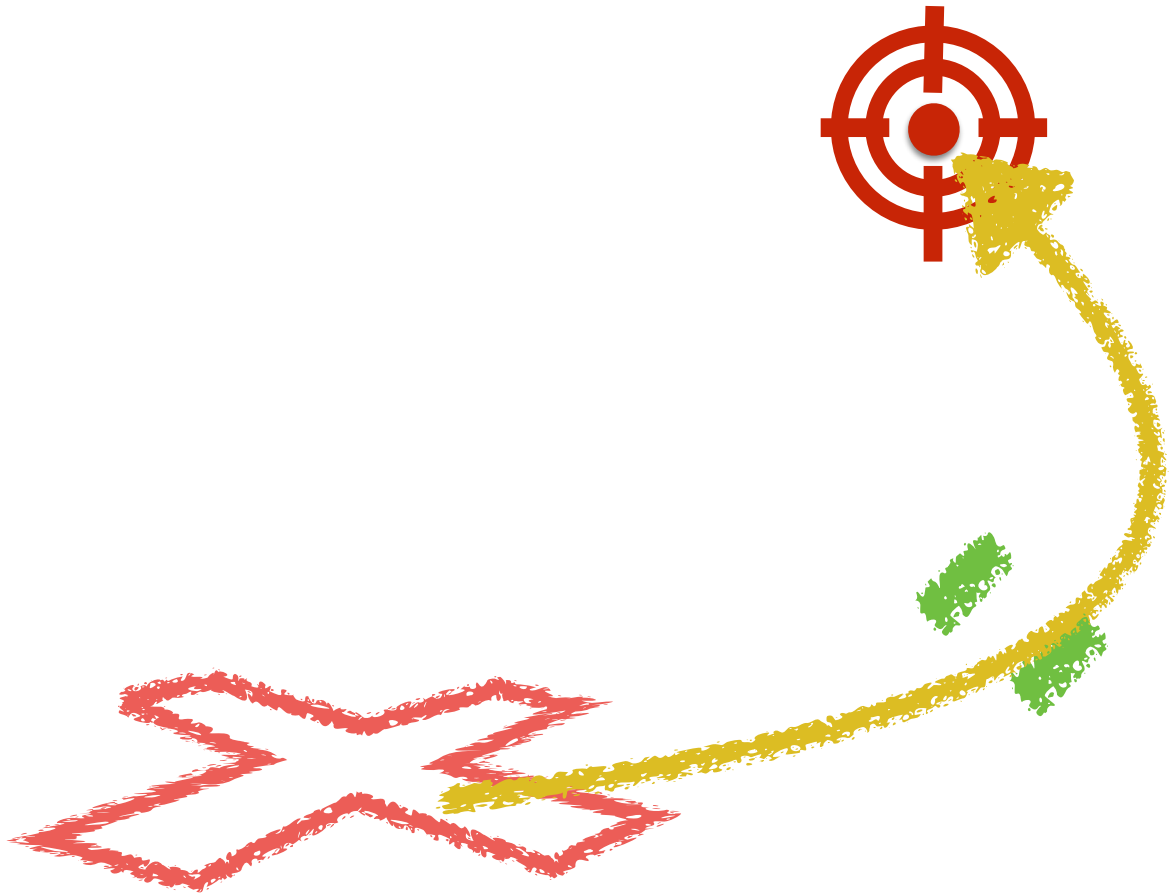
Next: address the constraints



Add additional features

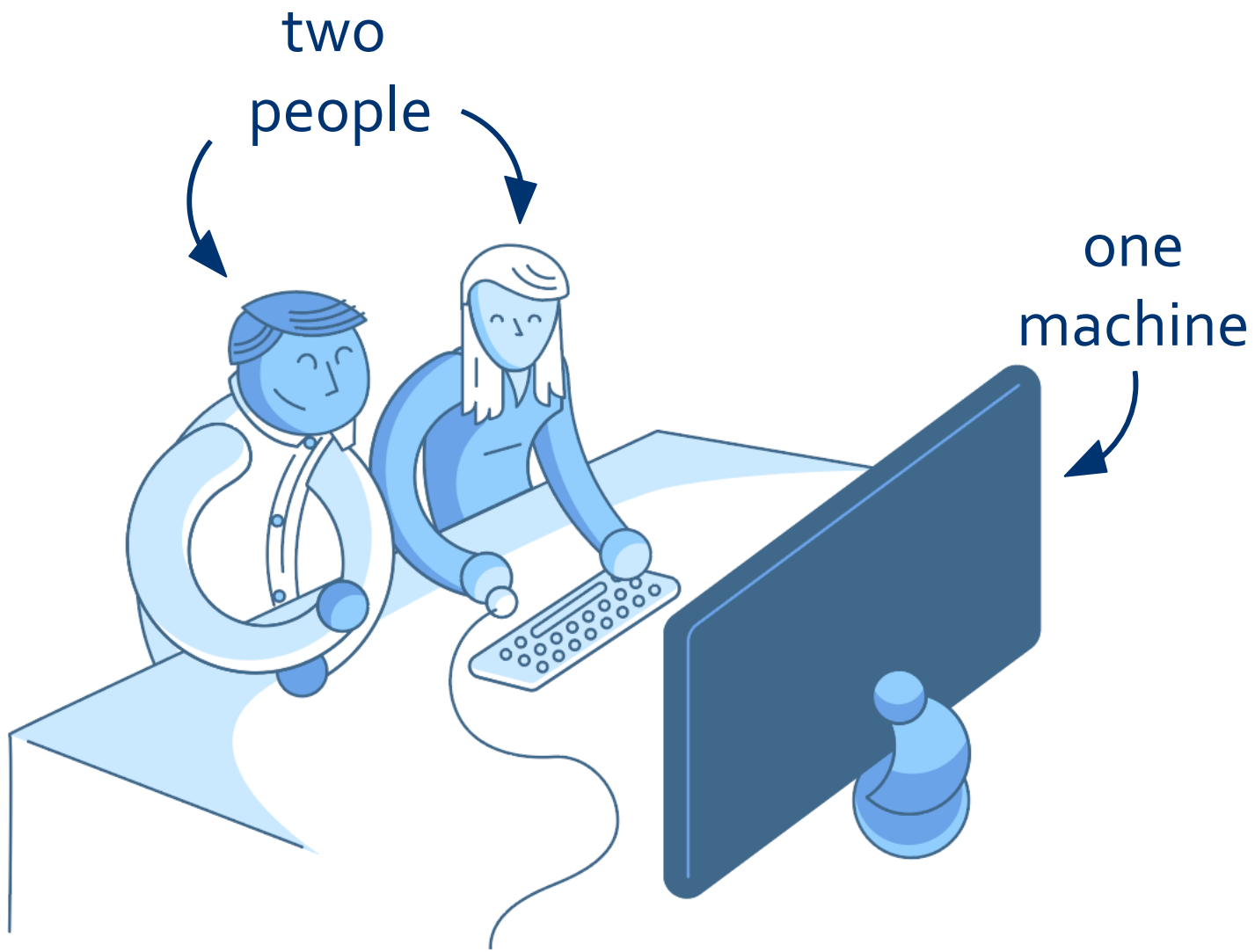


Finally: hit target

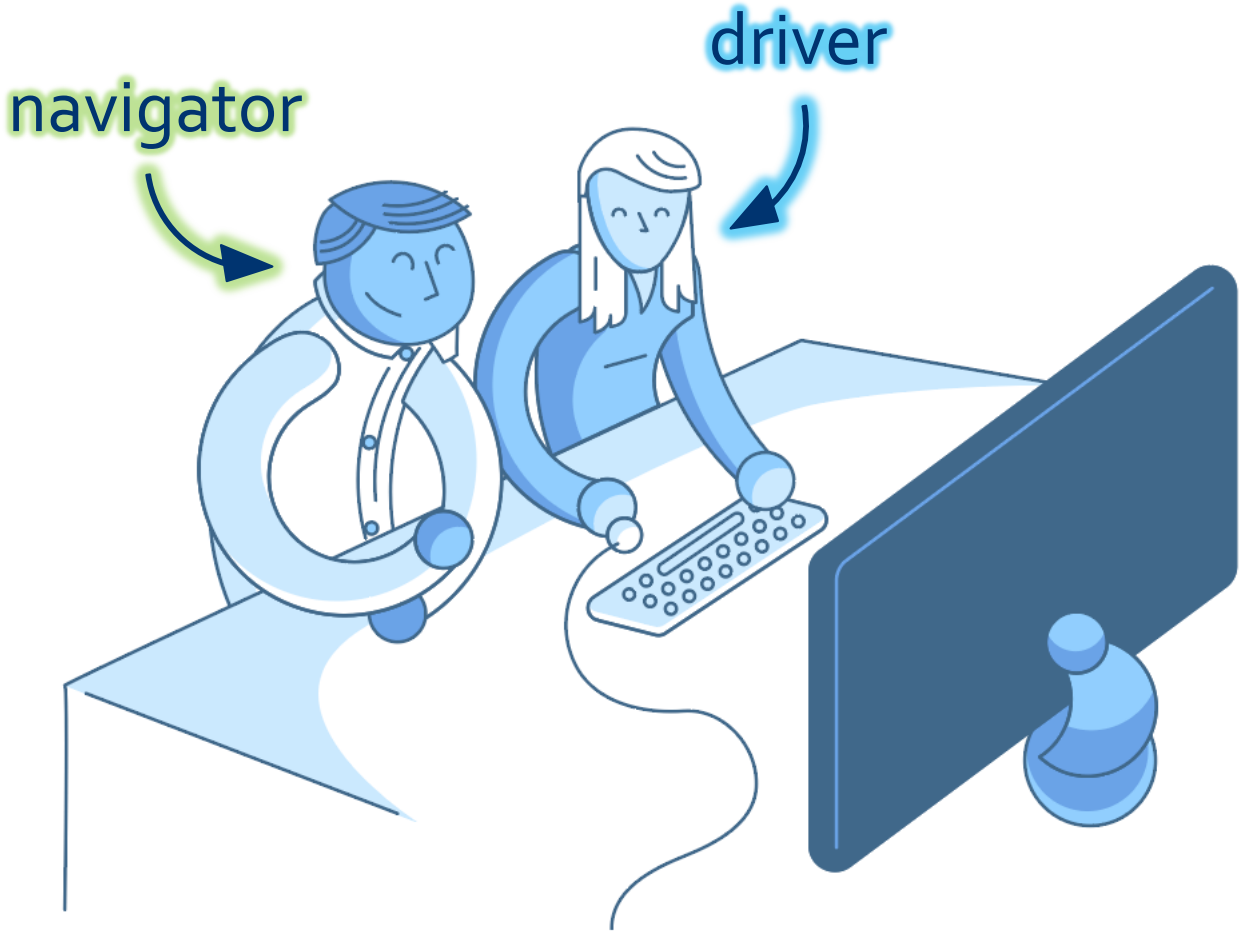


A problematic (but common) model

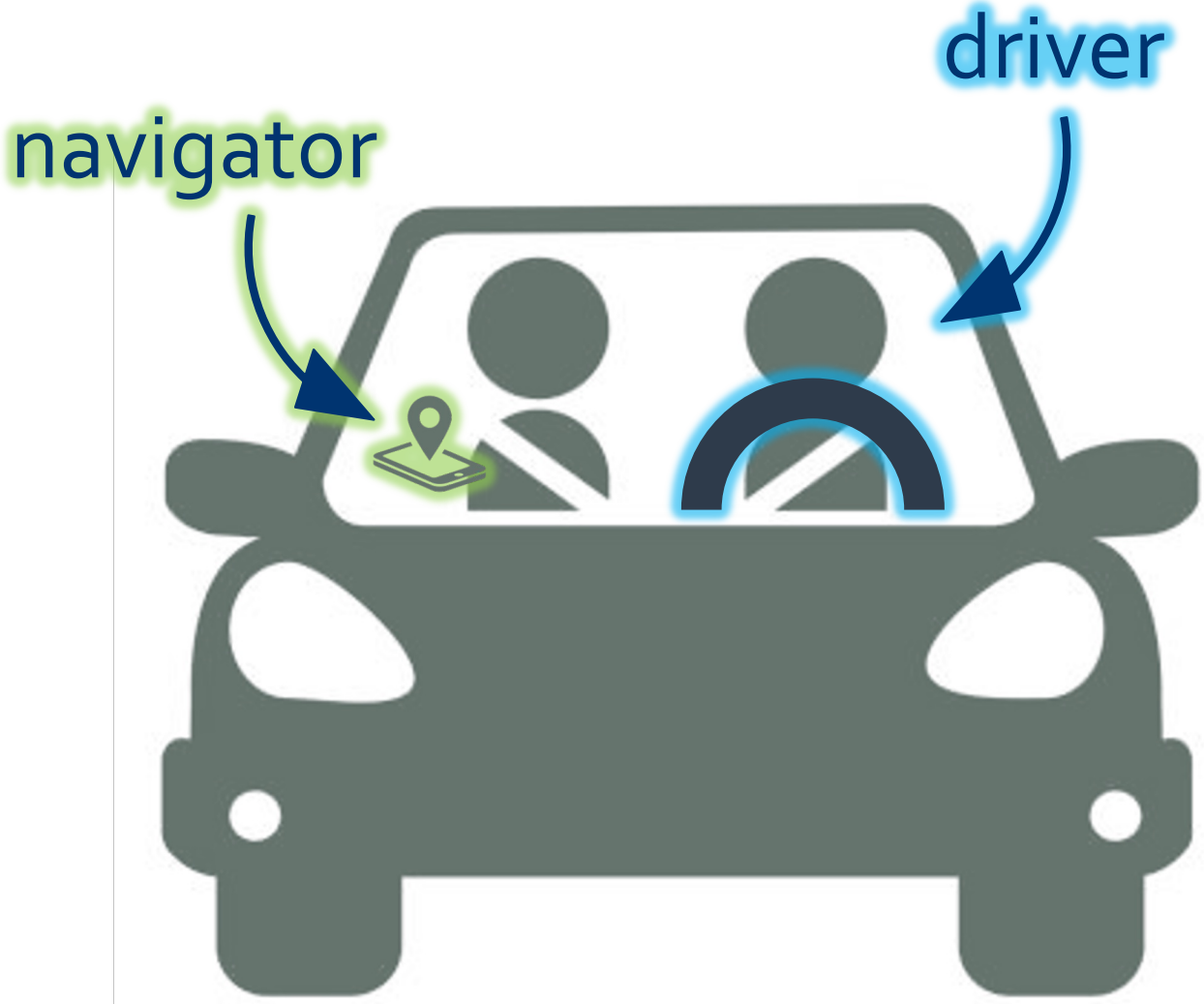
A better model: “pair programming”



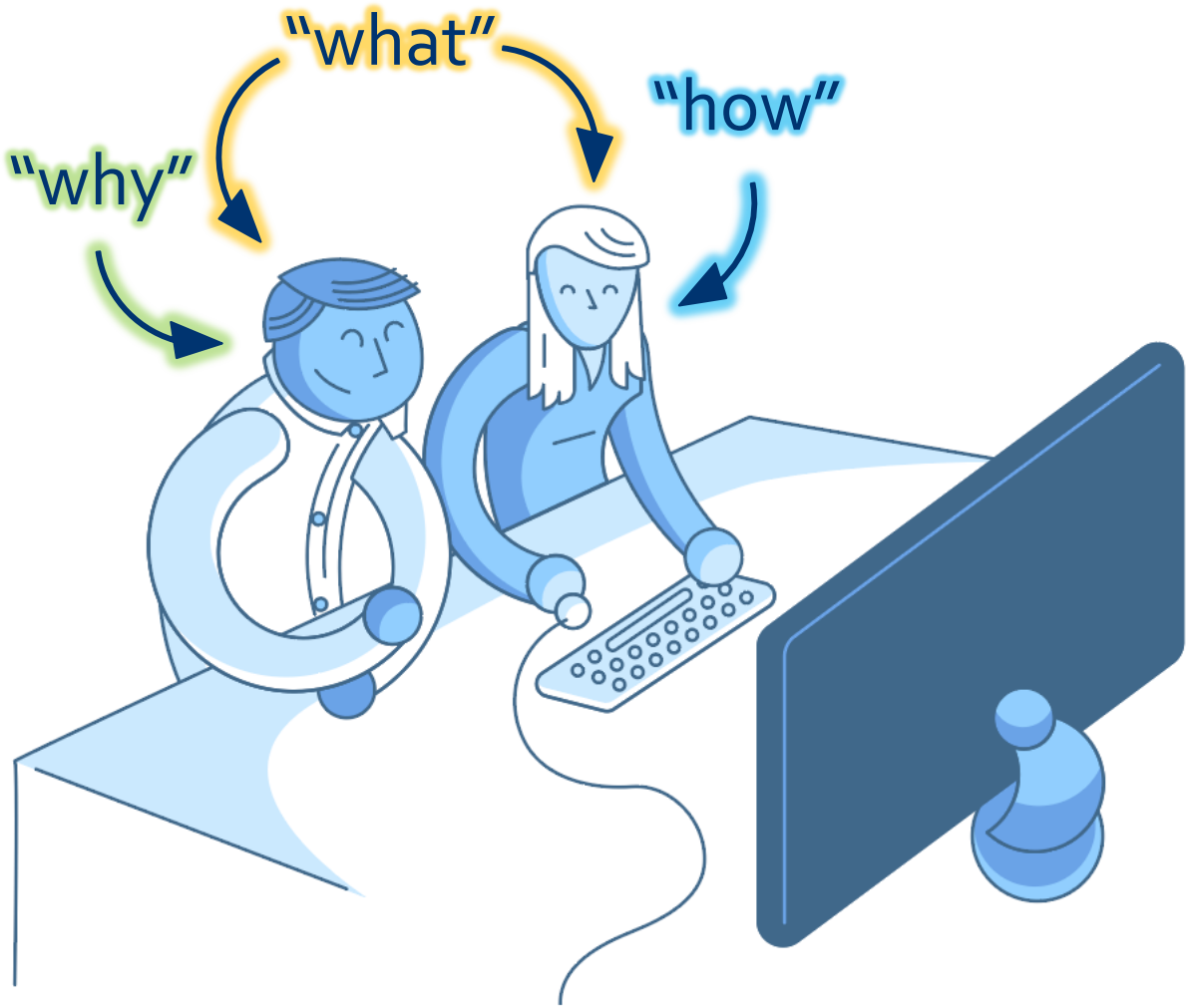
Two complimentary roles



A common analogy



Navigator vs. driver: different focus



Info & Storage

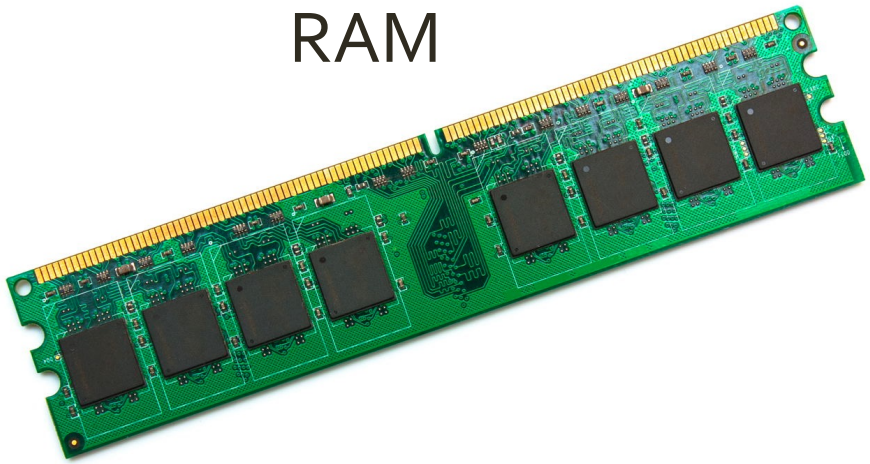
Storage

Want to store this important information for later: 3

Storage

Want to store this important information for later: 3

RAM

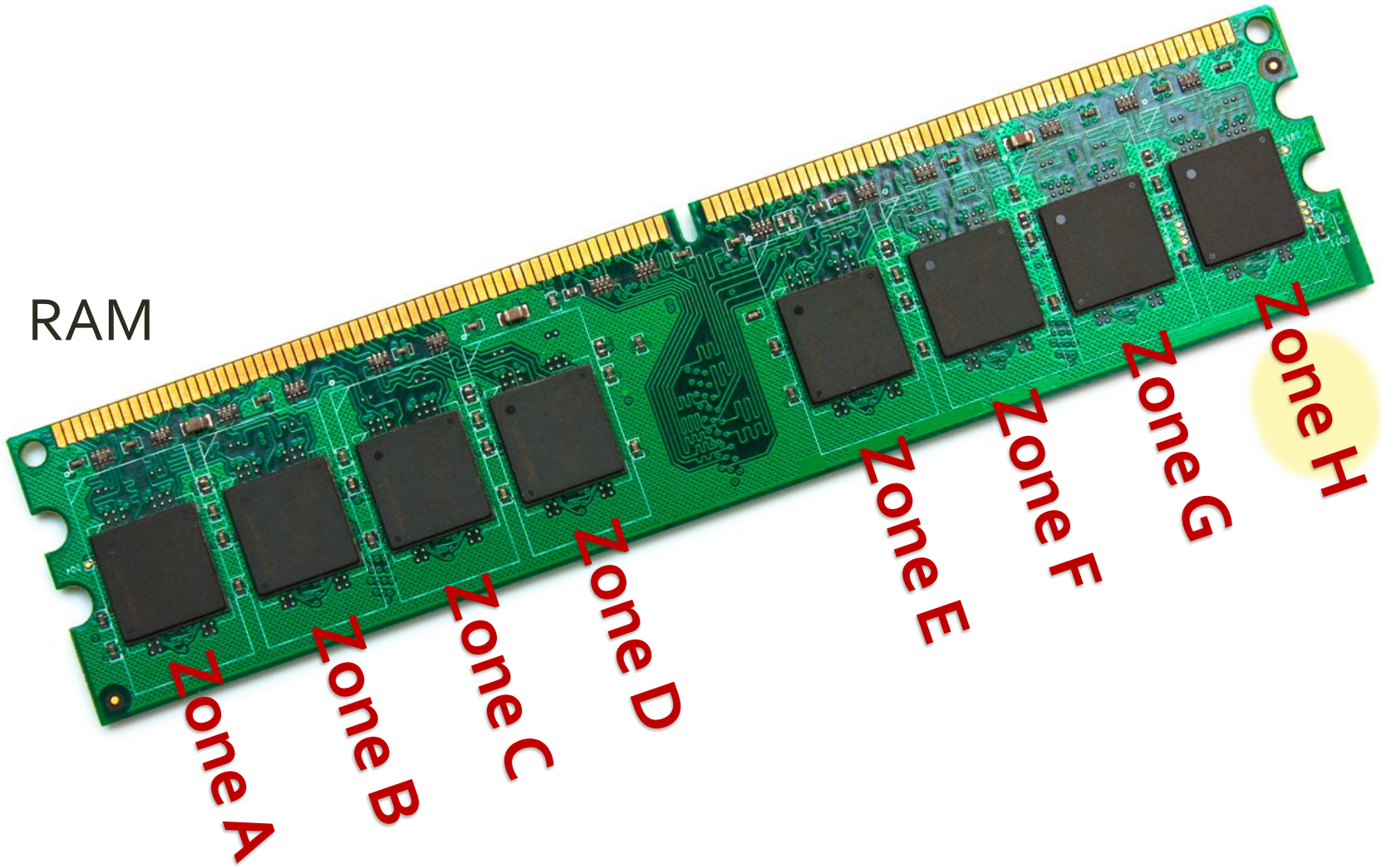


Hard Disk

Storage

Want to store this important information for later: 3

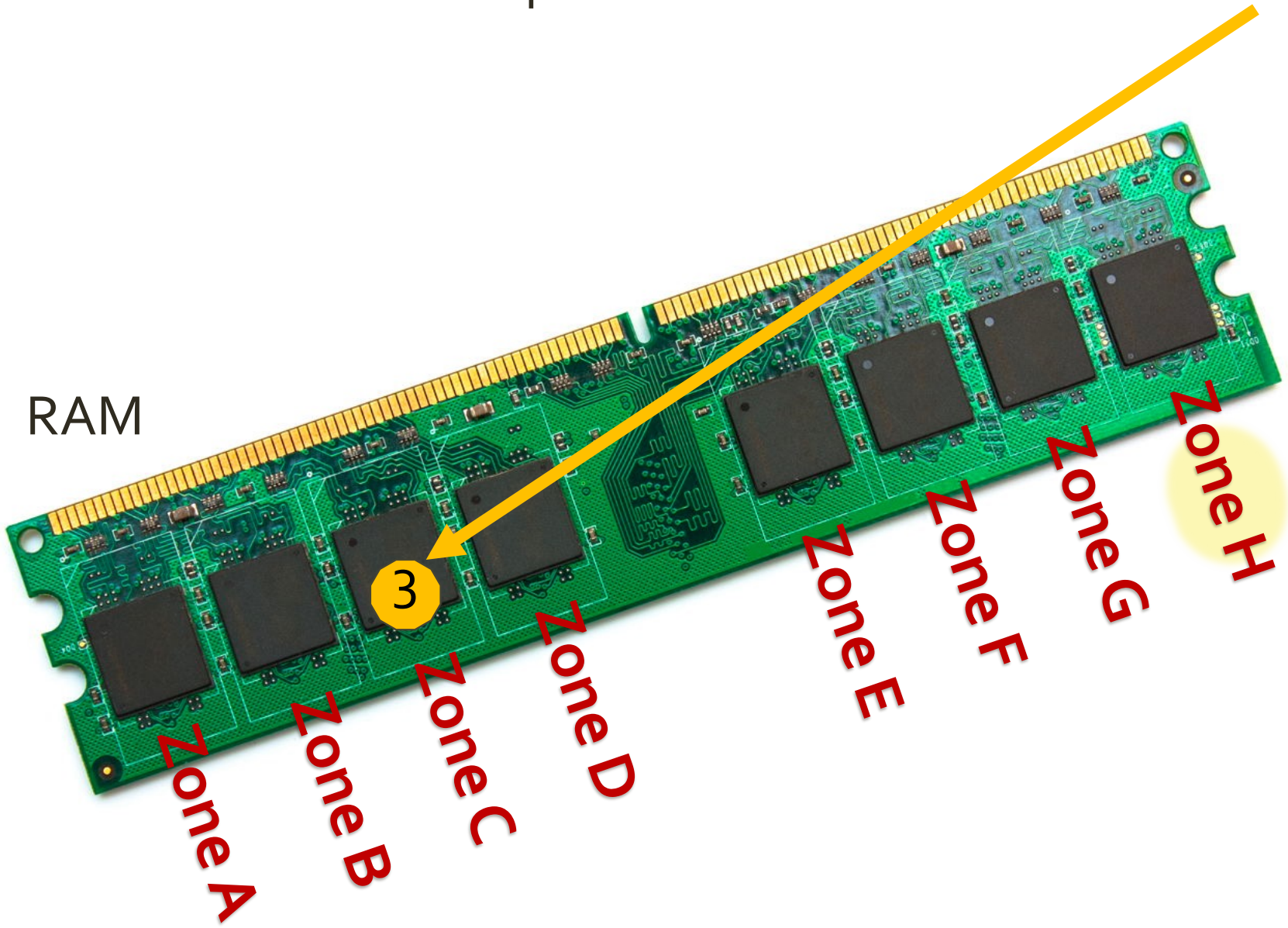
RAM



Storage

Want to store this important information for later:

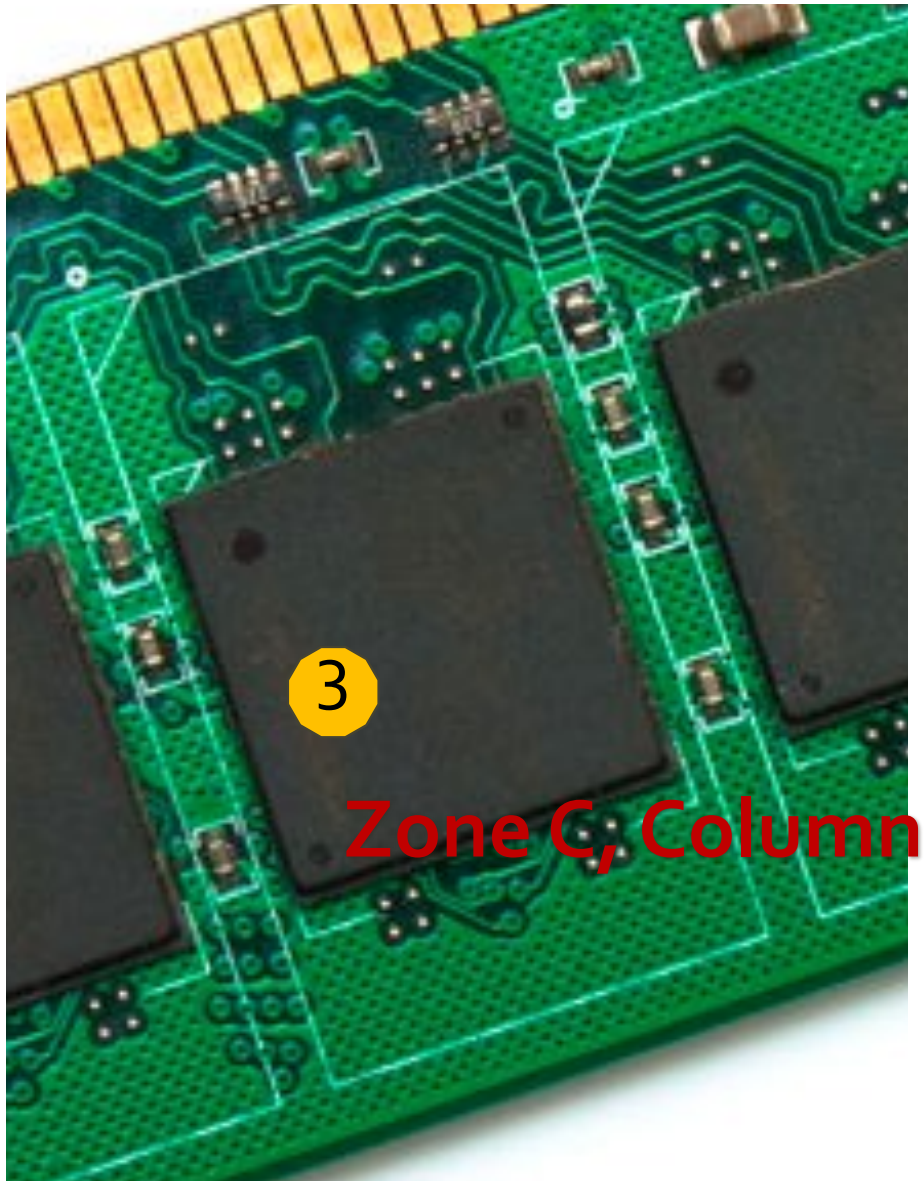
RAM



Storage

Want to store this important information for later:

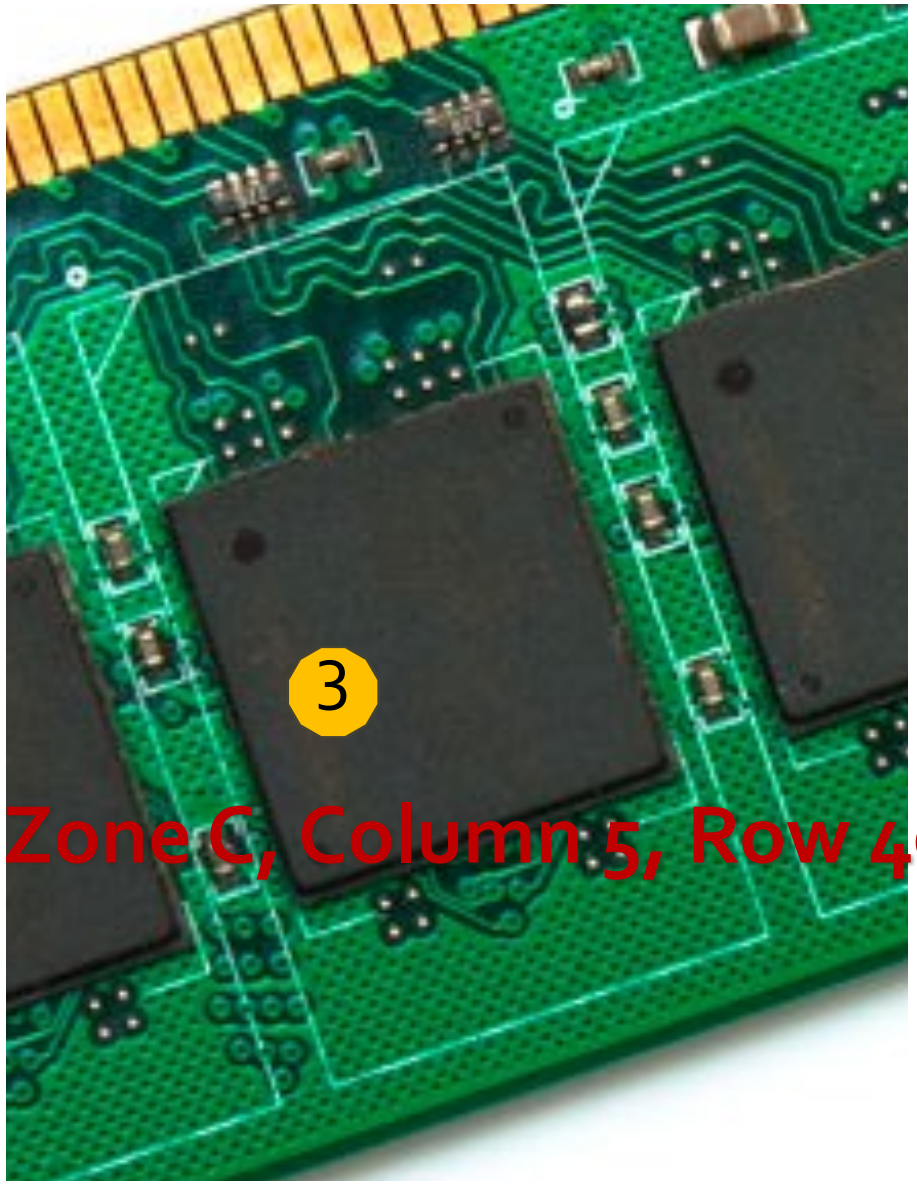
RAM



Zone C, Column 5, Row 40

Storage

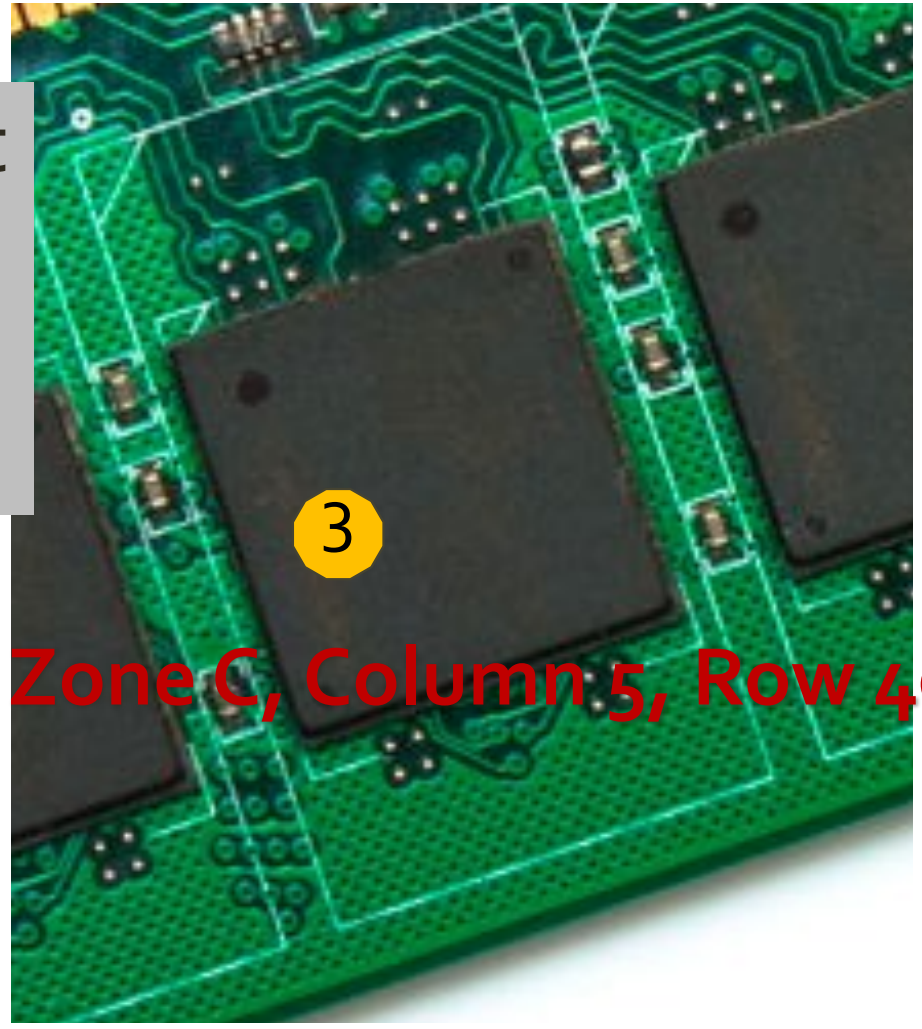
- Want the CPU to double the important information



Storage

- Want the CPU to double the important information
- We need to tell it where to look

```
double “important information” at “Zone C, Column 5, Row 40”
```

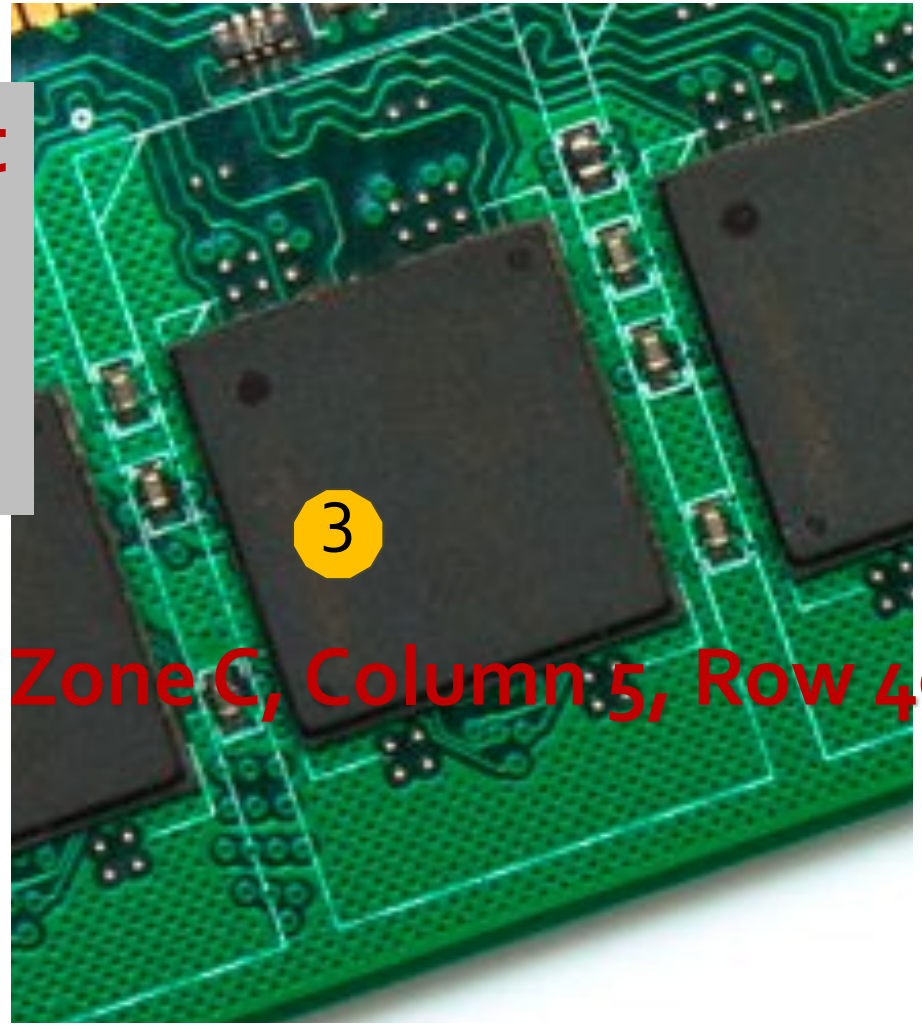


Storage

- Want the CPU to double the important information
- We need to tell it where to look

double “**important information**” at “**Zone C, Column 5, Row 40**”

- Wordy
- Refer to the same thing

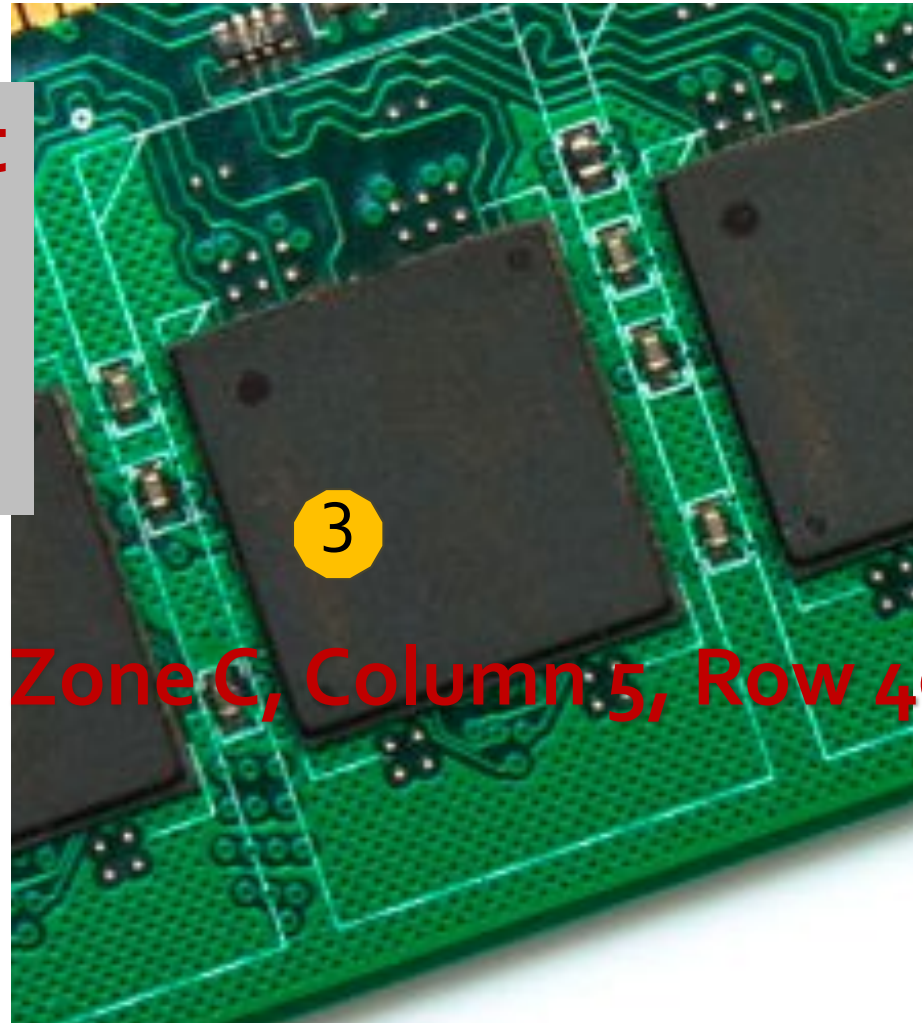


Storage

- Want the CPU to double the important information
- We need to tell it where to look

double “**important information**” at “**Zone C, Column 5, Row 40**”

- Wordy
- Refer to the same thing
- Let’s use a shorthand



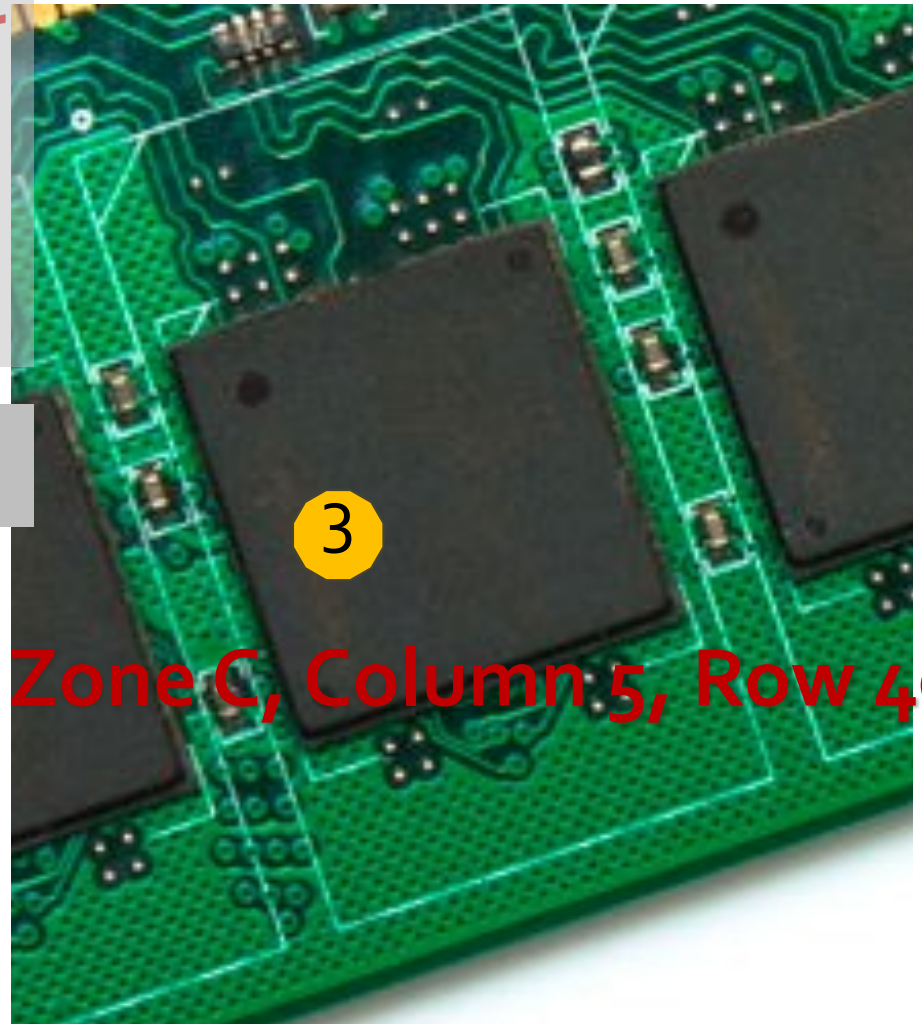
Storage

- Want the CPU to double the important information
- We need to tell it where to look

```
double “important information” at “Zone C, Column 5, Row 40”
```

```
double x
```

- Wordy
- Refer to the same thing
- Let’s use a shorthand



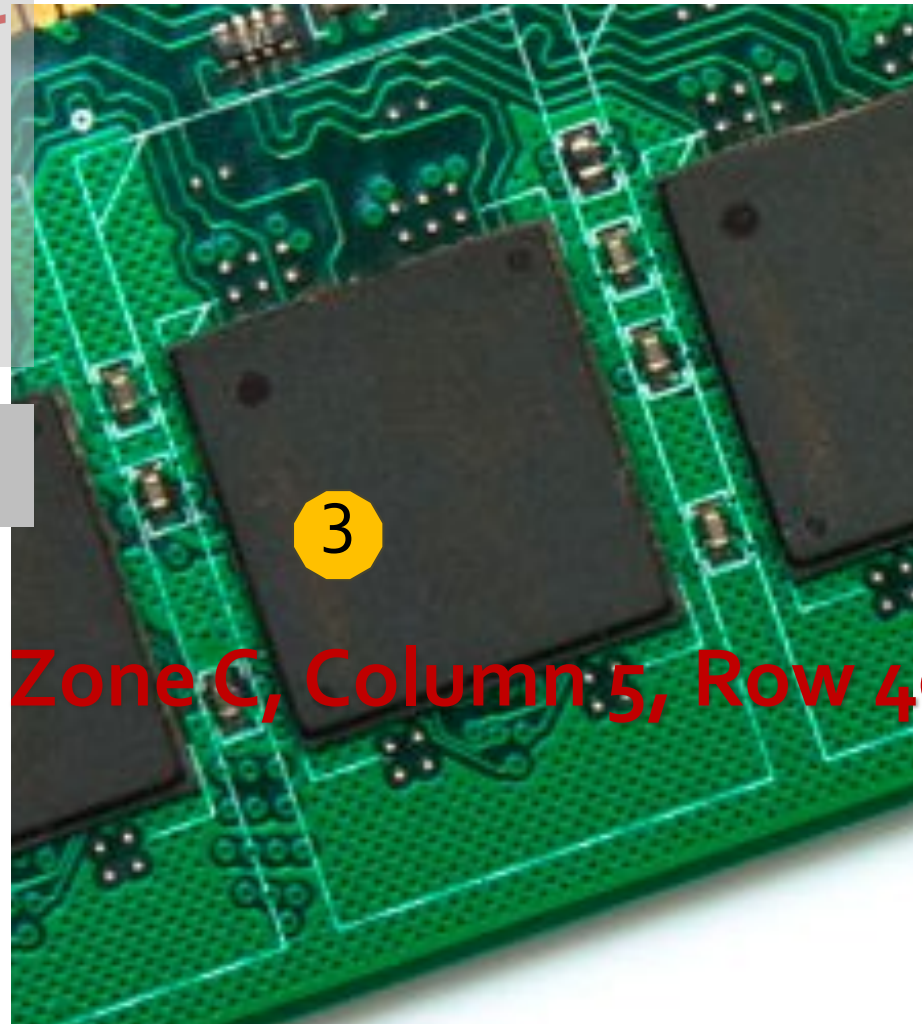
Storage

- Want the CPU to double the important information
- We need to tell it where to look

```
double "important information" at "Zone C, Column 5, Row 40"
```

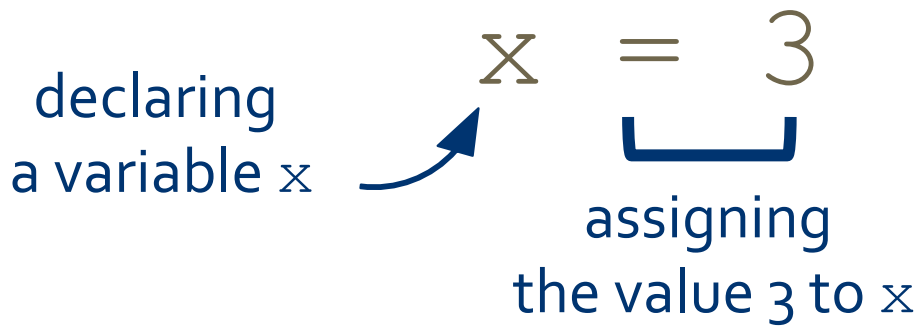
```
double x
```

- Wordy
- Refer to the same thing
- Let's use a **variable**



Core concept: `variables`

- In CS, a **`variable`** is a place to store a piece of data
- In Python, variables are:
  - **declared** by giving them a name
  - **assigned** using the equals sign
- Example:



Core concept: numeric values

- Two kinds of **numbers** in CS:
  - integers ("whole numbers")
  - floats ("decimals" or "floating point numbers")
- In Python, the kind of number is implied by whether or not the number contains a **decimal point**
- Example:

```
x = 3
x = 3.0
```

Core concept: strings

- In CS, a sequence of characters that isn't a number is called a **`string`**
- In Python, a string is declared using **quotation marks**
- Strings can contain letters, numbers, spaces, and special characters
- Example:

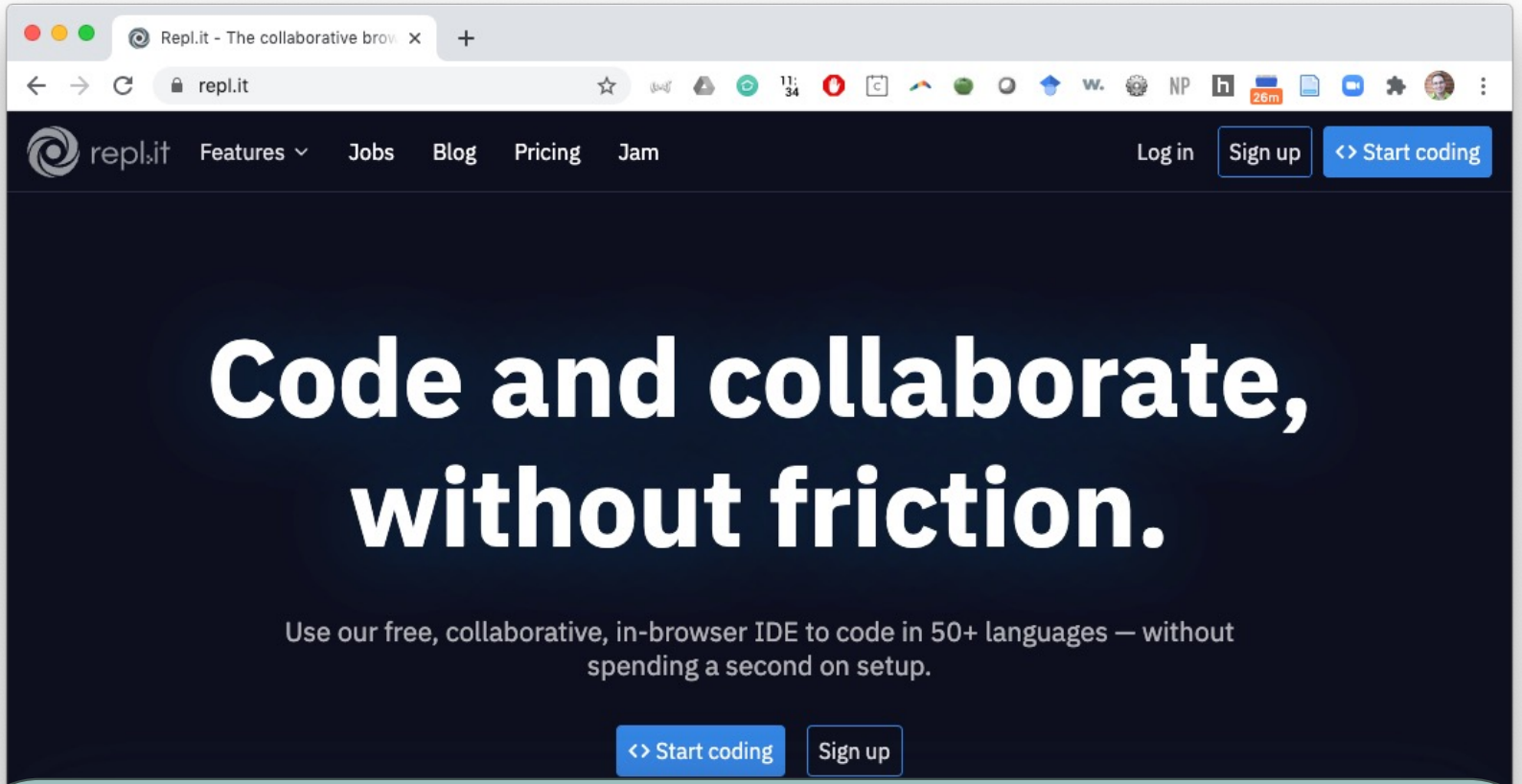
```
x = "Jordan"
x = "Stoddard G2"
```

Core concept: `print()`

- A function is a procedure / routine that takes in some input and does something with it (just like in math)
- In Python, the **`print()`** function takes in a value and outputs the value to the console
- This seems silly now, but will come in handy in lab when you write/run your first program inside a **file**

Coding Environment

repl.it



Click to accept invite to this class:
https://replit.com/teams/join/cbfzmzolnpzhbbarrspukzusuklzeihq-CAIS117-F23