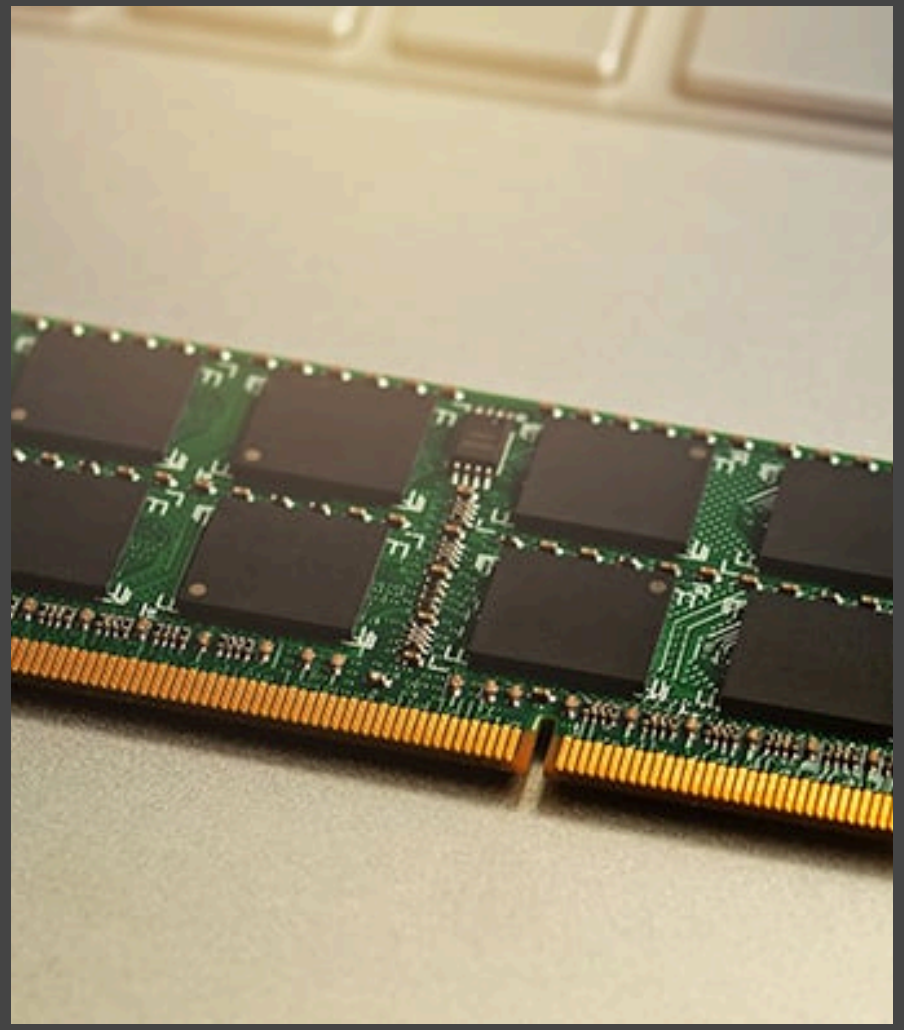
Ownership

Learn to Code with Rust

Memory

- **Ownership** is a set of rules that the compiler checks for to ensure the program will be free of memory errors.
- Memory refers to the area of your computer that is responsible for storing the information your programs use.
- It’s ideal to free memory when it is no longer in use.
- Programming languages implement different strategies for memory management.



Manual Memory Management

- In languages like **C** and **C++**, the programmer is responsible for allocating (requesting memory) and deallocating it (giving it back to the computer).
- Unfortunately, human beings make mistakes.
  - Forgetting to deallocate memory that has been allocated
  - Trying to deallocate memory that has already been deallocated


Social Networks
Facebook
Instagram
Twitter
Google+
Pinterest
Tumblr
LinkedIn
WhatsApp
Messenger

Automatic Garbage Collection

- Languages like **Java**, **Python**, **Ruby**, and **Go** implement a tool called the garbage collector.
- The garbage collector looks for data that is no longer in use and deallocates it. It "automates" the cleanup process.
- The garbage collector itself occupies memory and can run at disadvantageous times.

Ownership

- Rust introduces a new paradigm: **ownership**.
- **Ownership** is set of rules on how Rust manages your computer's memory.
- The Rust compiler does not compile the program if an ownership rule is violated.
- Best of all worlds: the speed of a language like C but with less room for error.

What is ownership?

- The **owner** is who/what is responsible for cleaning up a piece of data when it is no longer in use.
- Every value in a Rust program has one **owner.**
- The owner can change over the course of the program, but there is only 1 owner for a value at a time.
- The **owner** is usually a name.
  - A **variable** can be an owner.
  - A **parameter** can be an owner.
- Ownership also extends to composite types that own their elements.
  - A tuple and array own their values.