CS196

Variables and Mutability

CS196-128 Rust 101

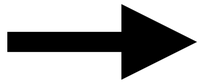
Slides by Matt Geimer (FA21)
Presented 9/1/2021

How do variables work?

```
public static void main(String[] args) {
    int x = 0;
    System.out.println("x = " + x);
}
```

How do variables work?

```java
public static void main(String[] args) {
    int x = 0;
    System.out.println("x = " + x);
}
```



```rust
fn main() {
    let x = 0;
    println!("x = {}", x);
}
```

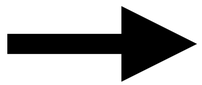
```
x = 0
```

```
x = 0
```

Extra Content: Why does println have "!" in Rust?

How do variables work?

```java
public static void main(String[] args) {
    int x = 0;
    x = x + 1;
    System.out.println("x = " + x);
}
```



```rust
fn main() {
    let x = 0;
    x = x + 1;
    println!("x = {}", x);
}
```

How do variables work?

```
fn main() {
    let x = 0;
    x = x + 1;
    println!("x = {}", x);
}
```

```
error[E0384]: cannot assign twice to immutable variable `x`
 --> src/main.rs:3:5
  |
2 |     let x = 0;
  |         -
  |         |
  |         first assignment to `x`
  |         help: make this binding mutable: `mut x`
3 |     x = x + 1;
  |     ^^^^^^^^^ cannot assign twice to immutable variable
```

What is a variable?

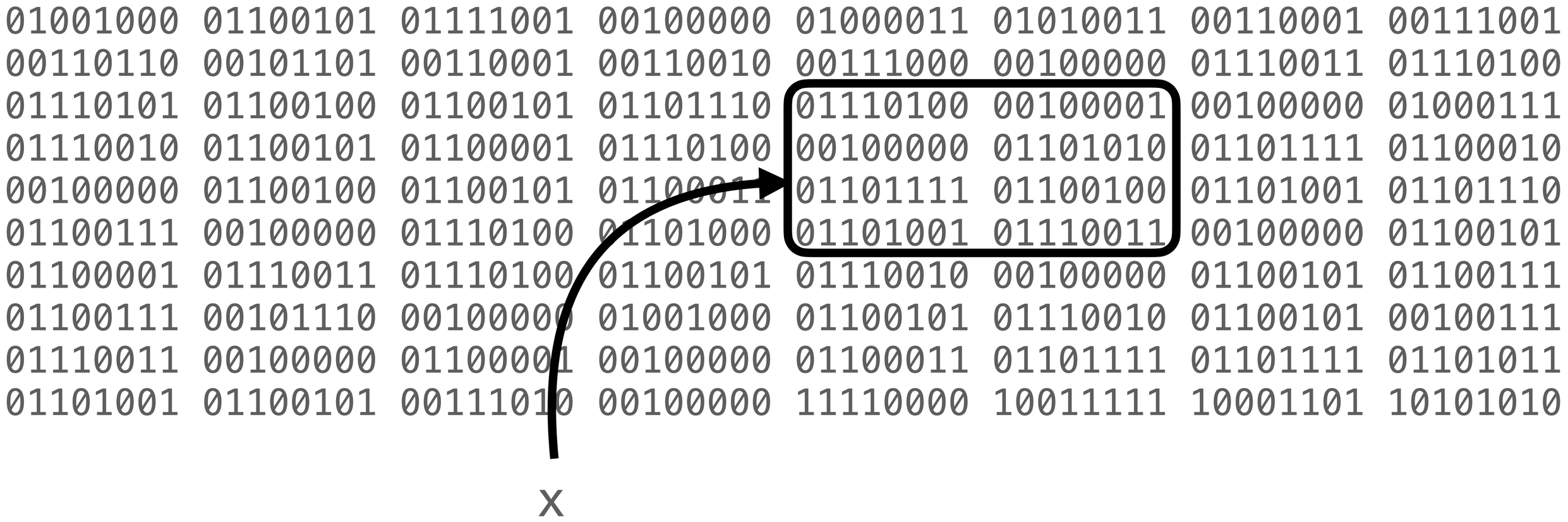
Why are we talking about variables again?

- In AP Computer Science or CS 124, how exactly variables work isn't explained

```
fn main() {
    let x = 0;
    x = x + 1;
    println!("x = {}", x);
}
```

What is a variable?

Why are we talking about variables again?



What is a variable?

Why are we talking about variables again?

- In AP Computer Science or CS 124, how exactly variables work isn't explained

```
fn main() {
    let x = 0;
    x = x + 1;
    println!("x = {}", x);
}
```

What is a variable?

Why are we talking about variables again?

- In AP Computer Science or CS 124, how exactly variables work isn't explained


```
fn main() {
    let mut x = 0;
    x = x + 1;
    println!("x = {}", x);
}
```

Mutability
An explanation

- Rust heavily favors concurrency, but only if it's done safely
- In order to encourage writing safe concurrent code, variables are immutable by default
- However, in some situations (or even many) you want mutable variables
- For these cases, we have the `mut` keyword
- `mut` is short for mutable
- Variables marked `mut` are mutable and can be modified after declaration
- The downside to this approach is that it's easier to make mistakes when writing concurrent code

Shadowing

A safer version of `mut`

- Rather than using the `mut` keyword, we can use **shadowing** to achieve similar behavior
- Shadowing is the practice of creating a new variable with the same name
  - In most other languages, this is prohibited
  - Doing this in Rust tells the compiler to ignore the previous variable
- **Danger**: Shadowing is in essence declaring a new variable. This means you can accidentally change the type of the variable

Shadowing

A safer version of `mut`

- Rather than using the `mut` keyword, we can use **shadowing** to achieve similar behavior
- Shadowing is the practice of creating a new variable with the same name
  - In most other languages, this is prohibited
  - Doing this in Rust tells the compiler to ignore the previous variable
- **Danger**: Shadowing is in essence declaring a new variable. This means you can accidentally change the type of the variable


```
fn main() {
    let x = 0;
    let x = “Matt";
    println!("x = {}", x);
}
```

```
x = Matt
```

Lesson Summary

Variables and Mutability

- Variables can be declared with the let keyword
- Variables can then be used elsewhere in the program
- Variables are immutable by default, meaning their value cannot change
- Using the `mut` keyword, variables can be made mutable
  - Beware that doing this introduces the possibility of bugs when working with concurrent code
- A safer option for similar behavior is “shadowing” immutable variables
  - Beware that doing this means you have to be careful of the types of your variable assignments

CS196

Variables and Mutability

CS196-128 Rust 101

Slides by Matt Geimer (FA21)
Presented 9/1/2021