Welcome to this Rust tutorial!

Systems Programming Before Rust…

Photo credit: Giorgio Monteforti


Hack without fear!

Hello, world!

Hello, world!

```
fn main() {
  println!(“Hello, world!”);
}
```

Exercise: **hello world**

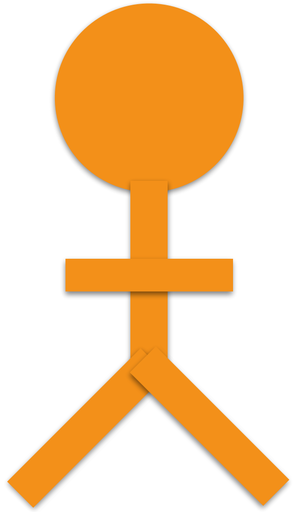
http://rust-tutorials.com/exercises/

Cheat sheet:

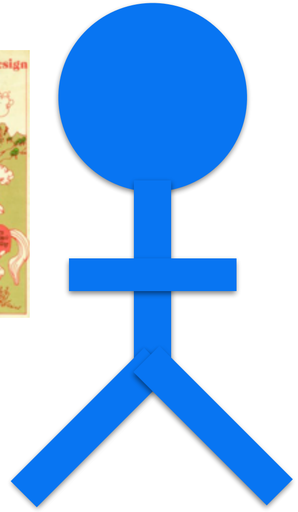
```
let name = “foo”;          // make a variable
println!(“{}”, name);      // format string
```


Principles of Compiler Design
Alfred V. Aho
Jeffrey D. Ullman

Ownership



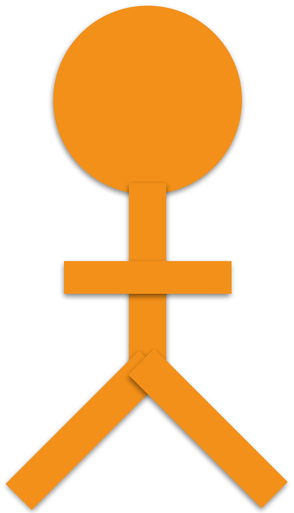

Principles of Compiler Design
Alfred V. Aho
Jeffrey D. Ullman



Ownership

```
fn main() {
    let name = format!("…");
    helper(name);
    helper(name);
}
```

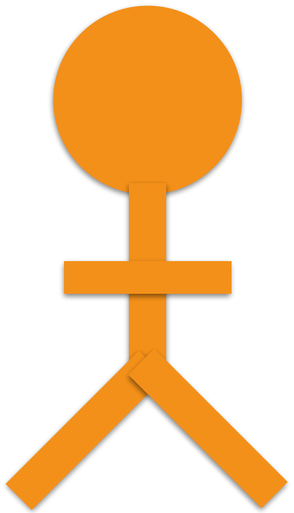
```
fn helper(name: String) {
    println!(..);
}
```

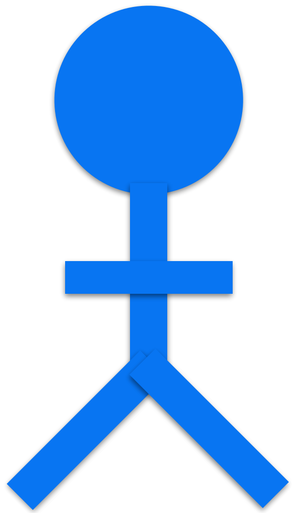


Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```



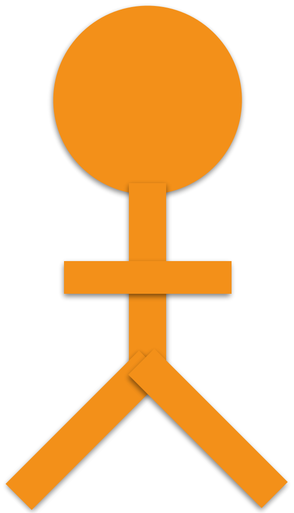


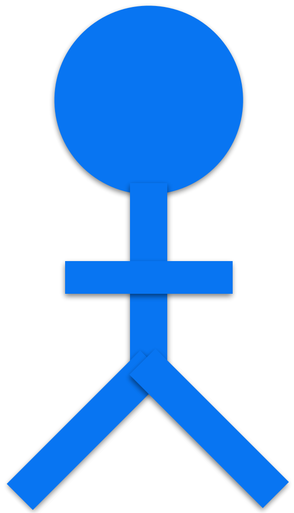
Ownership

```
fn main() {
    let name = format!("…");
    helper(name);
    helper(name);
}
```

```
fn helper(name: String) {
    println!(..);
}
```

Take ownership of a String

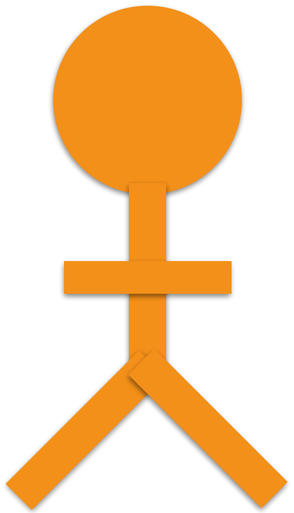


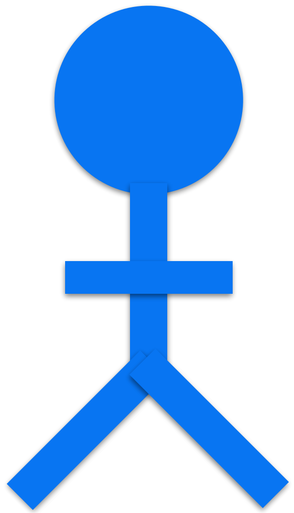


Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

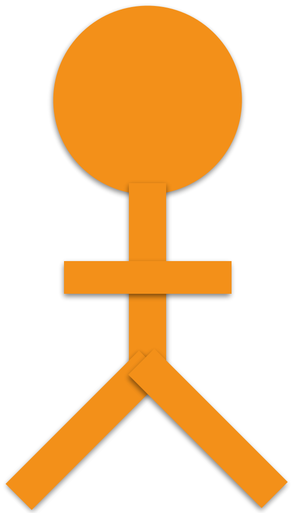


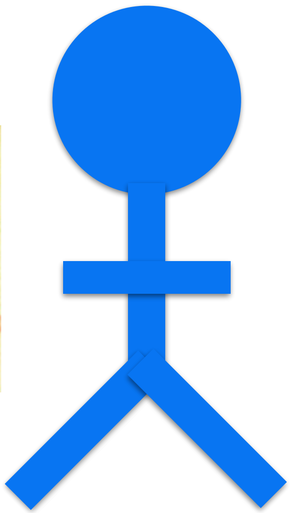


Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

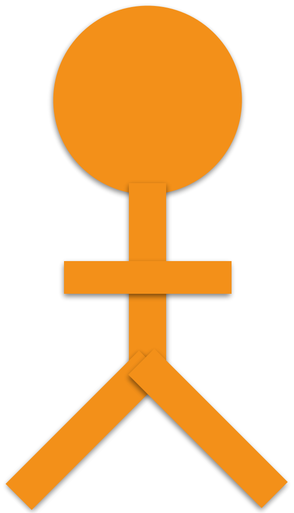


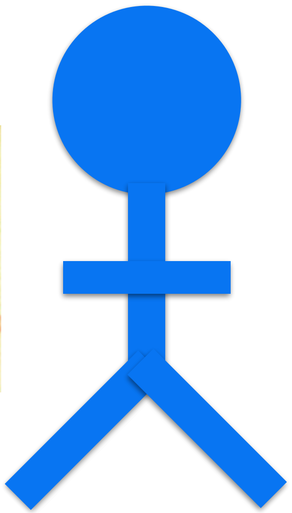


Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

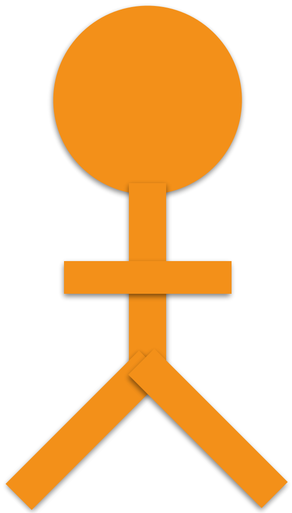




Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

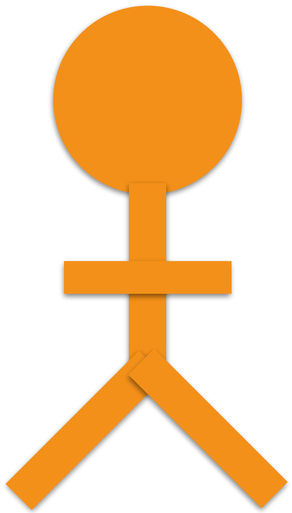
```
fn helper(name: String) {
  println!(..);
}
```



Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

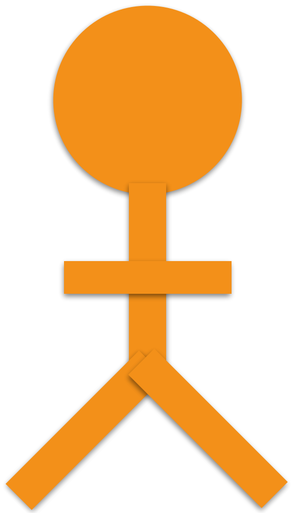
```
fn helper(name: String) {
  println!(..);
}
```



Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

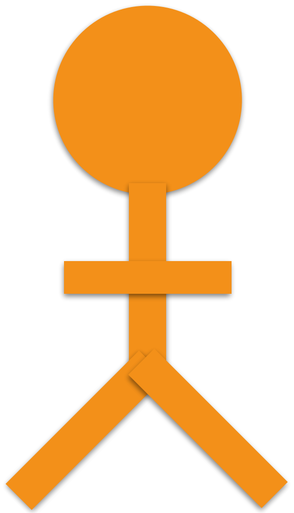


Ownership

```
fn main() {
  let name = format!("…");
  helper(name);
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

Error: use of moved value: `name`



Ownership

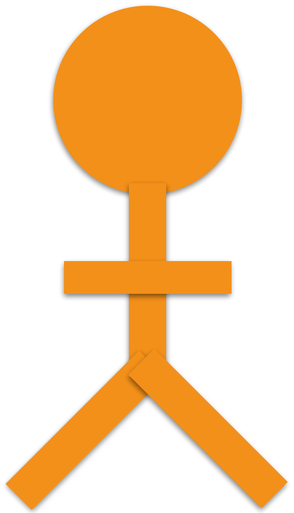
```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

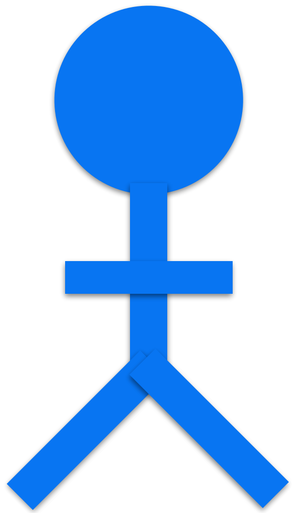
```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```



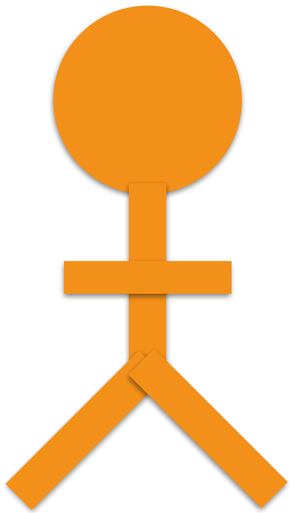


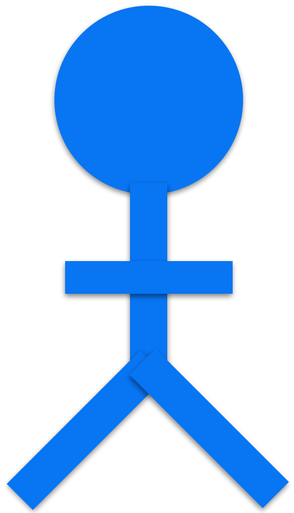
“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

Take **reference** to Vector

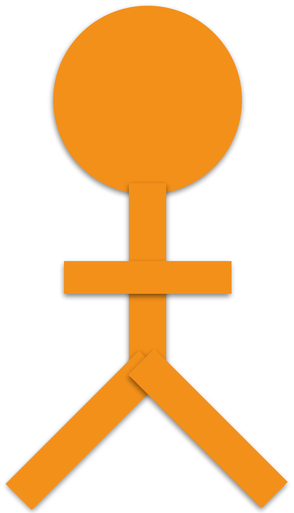


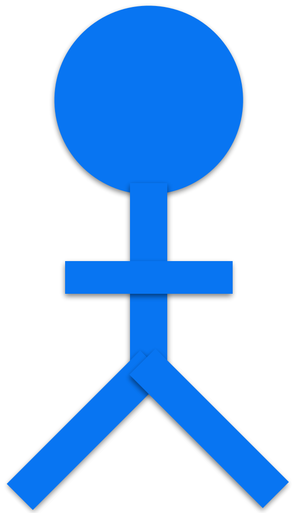


“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

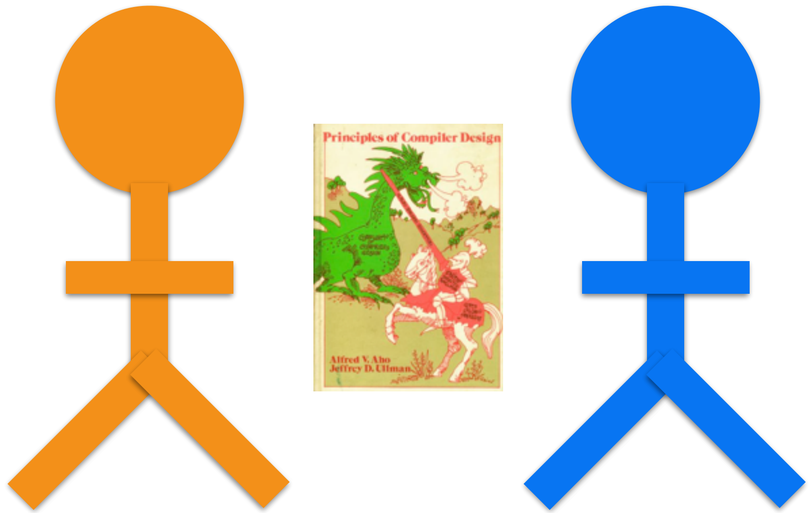




“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```



“Ownership” in Java

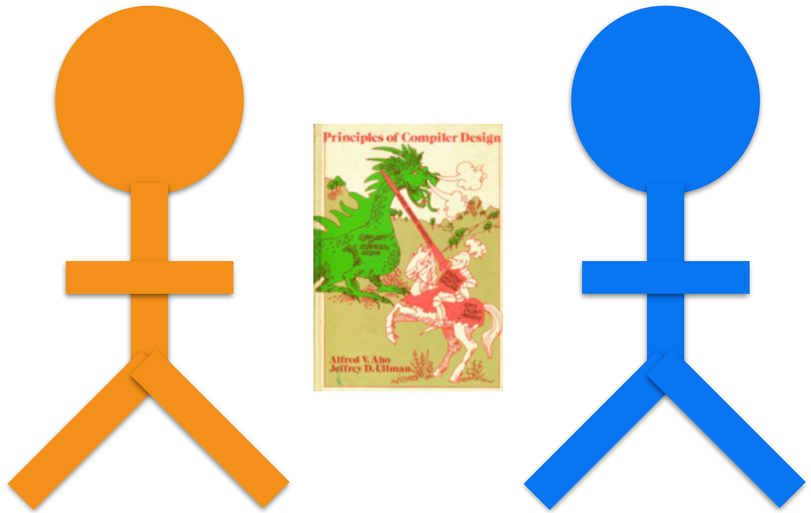
```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

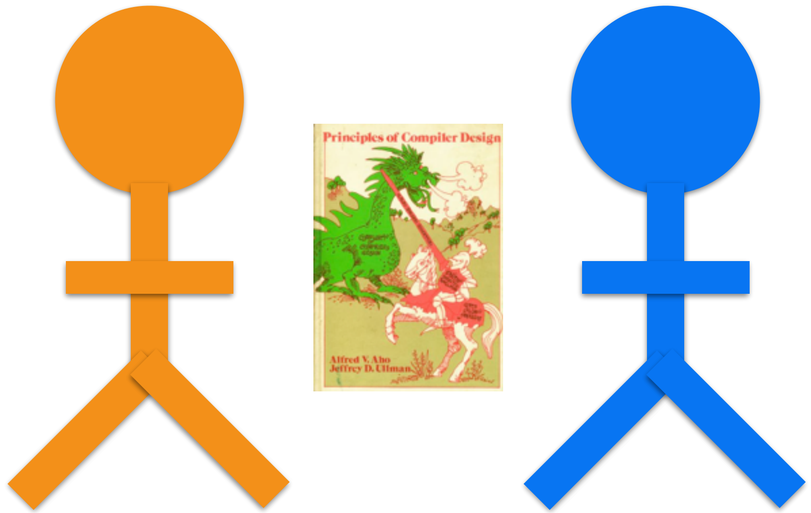
```
void helper(Vector name) {
  …
}
```



“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```



“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
```

```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  …
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  new Thread(…);
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  new Thread(…);
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  new Thread(…);
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

```
void helper(Vector name) {
  new Thread(…);
}
```

“Ownership” in Java

```
void main() {
  Vector name = …;
  helper(name);
  helper(name);
}
```

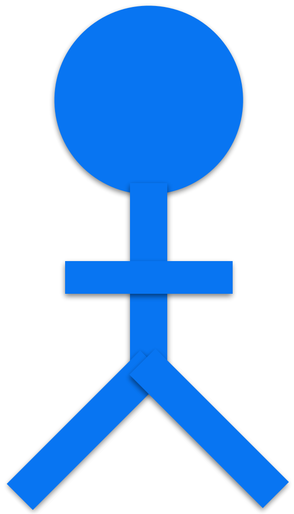
```
void helper(Vector name) {
   new Thread(…);
}
```

“Ownership” in Java

Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

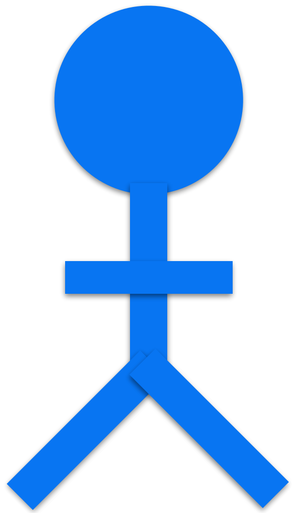
```
fn helper(name: String) {
  println!(..);
}
```



Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

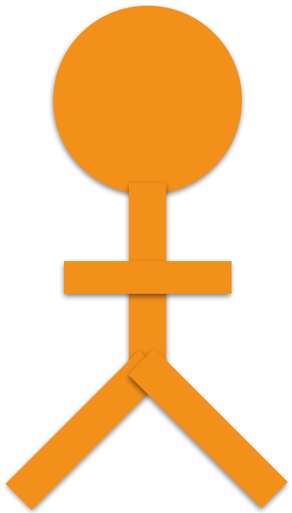


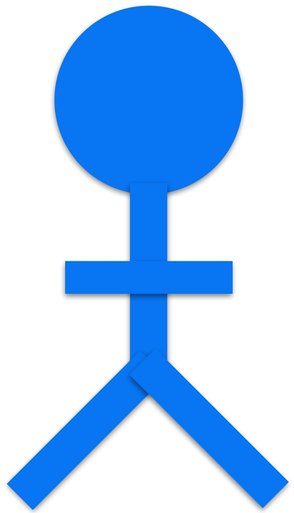
Clone

```
fn main() {
    let name = format!("…");
    helper(name.clone());
    helper(name);
}
```

Copy the String

```
fn helper(name: String) {
    println!(..);
}
```

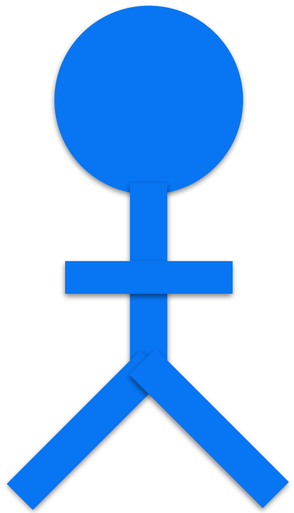




Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```



Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

Clone

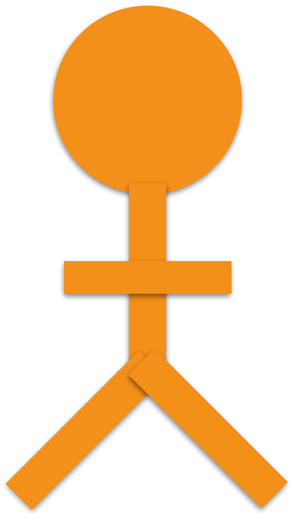
```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

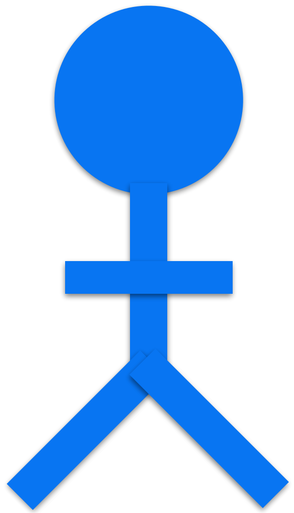
```
fn helper(name: String) {
  println!(..);
}
```

Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

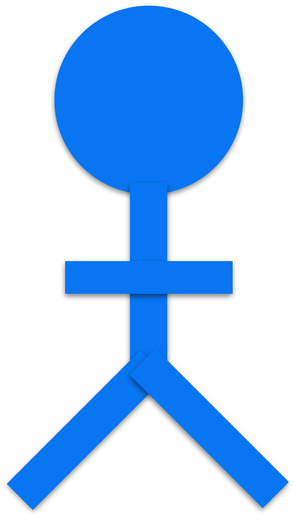




Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

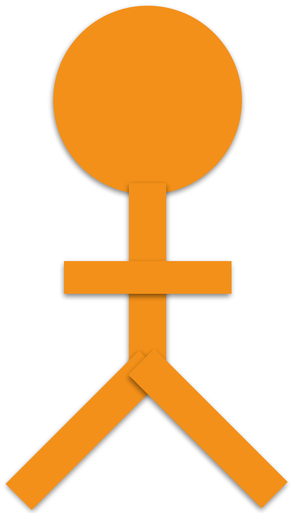
```
fn helper(name: String) {
  println!(..);
}
```

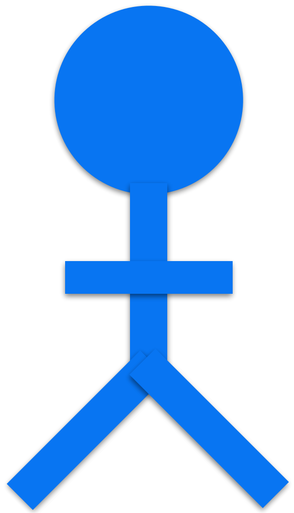


Clone

```
fn main() {
  let name = format!("…");
  helper(name.clone());
  helper(name);
}
```

```
fn helper(name: String) {
  println!(..);
}
```

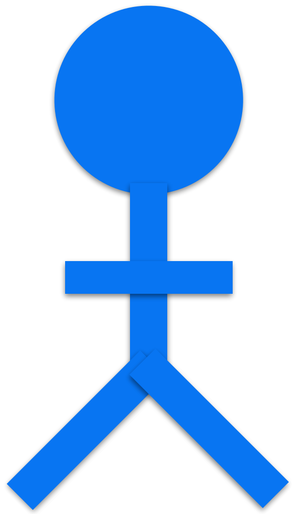




Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

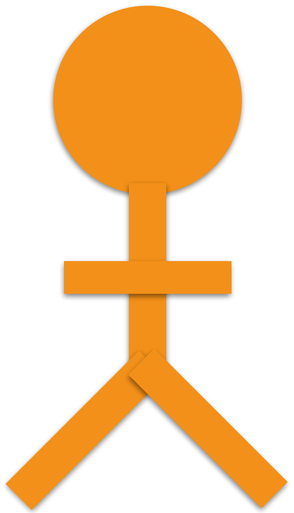


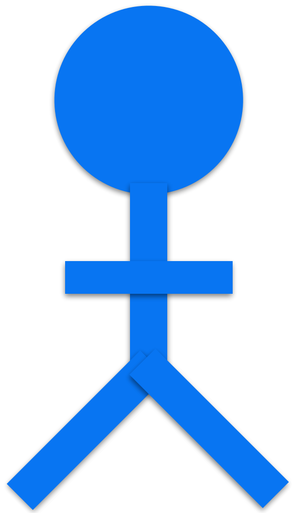
Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

i32 is a Copy type



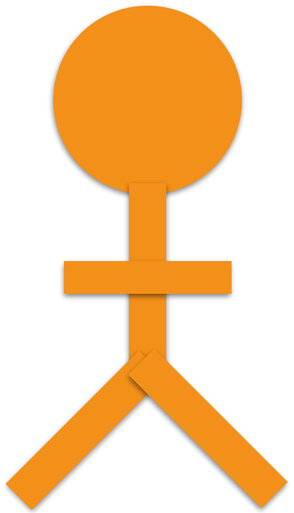


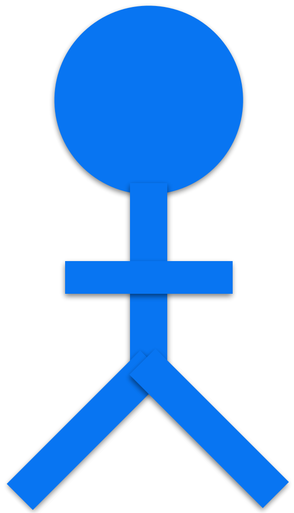
Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

i32 is a Copy type





Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

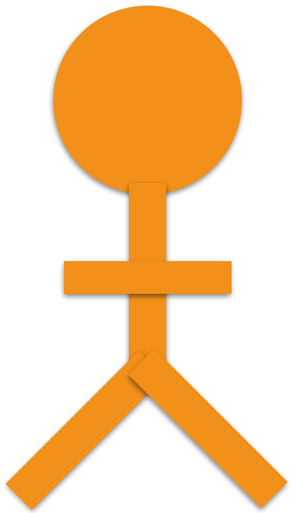
i32 is a Copy type

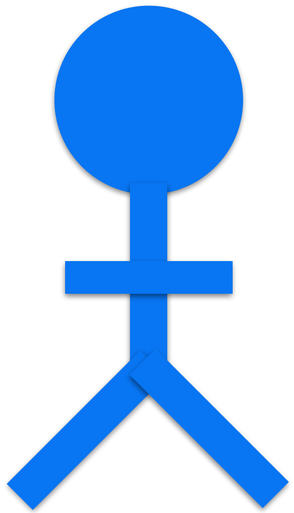
Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

i32 is a Copy type



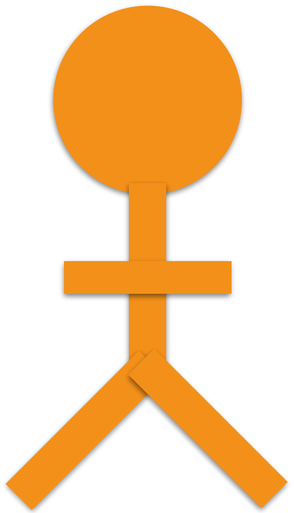


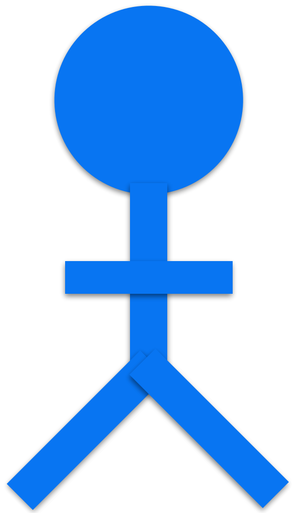
Copy (auto-Clone)

```
fn main() {
    let count = 22;
    helper(count);
    helper(count);
}
```

```
fn helper(count: i32) {
    println!(..);
}
```

i32 is a Copy type





Non-copyable: Values **move** from place to place.

Example: *money*

Clone: Run custom code to make a copy.

Example: *strings*

Copy: Type is implicitly copied when referenced.

Example: *integers or floating-point numbers*

Exercise: **ownership**

http://rust-tutorials.com/RustConf17

Cheat sheet:

```
fn helper(name: String) // takes ownership

string.clone()          // clone the string
```

http://doc.rust-lang.org/std