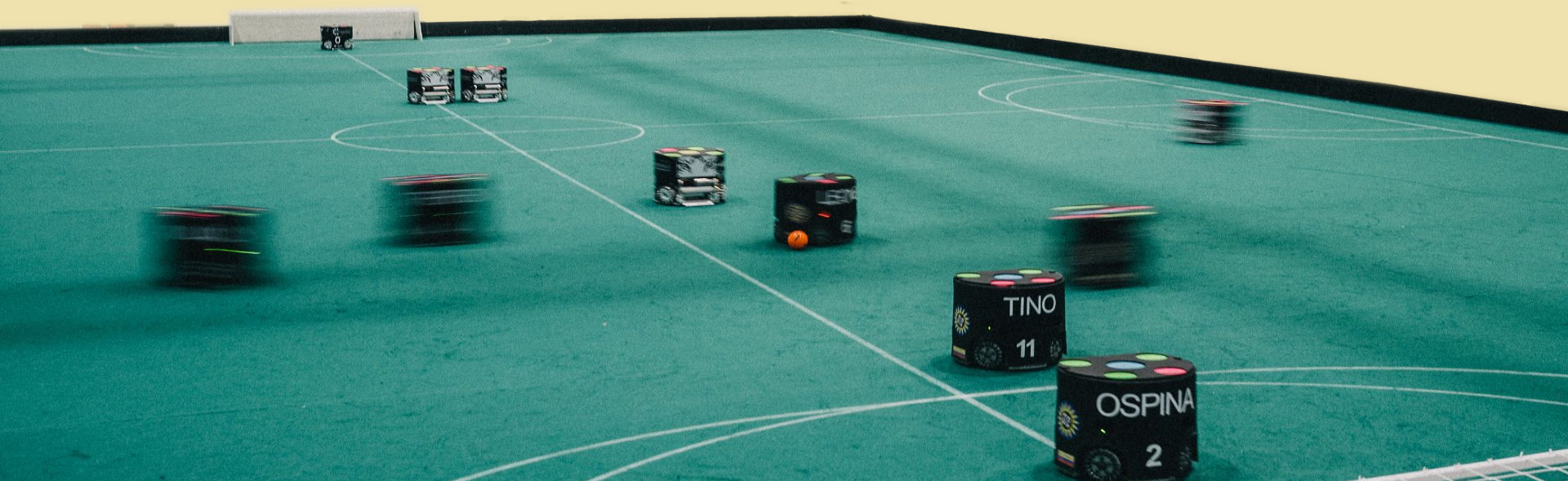
# RUST WORKSHOP

## by Delft Mercurians

Presented by Bálint Magyar

# The Delft Mercurians

As TU Delft's **first** RoboCup team, we build **cutting-edge robots** to play football against other robots around the world.



# The Delft Mercurians

We use **Rust** to create reliable and fast software for our robots.

# About Me

## Bálint Magyar

- Software Team Lead at Delft Mercurians
- 3+ years as a professional developer
- Started using Rust 2 years ago
- Work at Fusion Engineering, where we exclusively use Rust

/in/mablin7

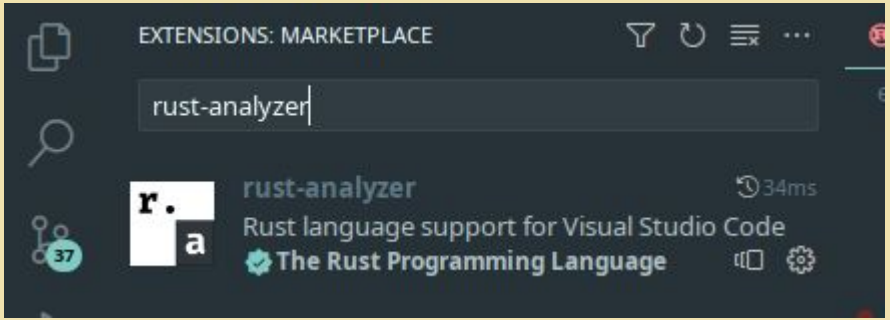
# Project Layout

```
📦 rust-workshop
├─ .vscode
├─ src
│  └─ main.rs
├─ examples
│  ├─ 00-hello-world.rs
│  └─ ...
├─ Cargo.toml
└─ Cargo.lock
```

# Project Layout

```
📦 rust-workshop
├─ .vscode
├─ src
│  └─ main.rs
├─ examples
│  ├─ 00-hello-world.rs
│  └─ ...
├─ Cargo.toml
└─ Cargo.lock
```

# VSCode and rust-analyzer




- ➔ Provides autocomplete
- ➔ Highlights errors
- ➔ And more!

# VSCode and rust-analyzer

inlay hints

```
let (tx: Sender<RobotCmd>, rx: Receiver<RobotCmd>) = std::sync::mpsc::channel();
let port_name: String = port_name.to_owned();
std::thread::spawn(move || -> ! {
    let mut port: Box<dyn SerialPort> = serialport::new(path: port_name, baud_rate: 115200) SerialPortBuilder
        .timeout(Duration::from_millis(10)) SerialPortBuilder
        .open() Result<Box<dyn SerialPort>, …>
        .expect(msg: "Failed to open port");
```

They help you keep track of the types, but they can be overwhelming when you are just starting out.

# VSCode and rust-analyzer

Open the settings (ctrl/cmd + shift + P) and search for “inlay”

**Rust-analyzer › Inlay Hints › Parameter Hints: Enable**

Whether to show function parameter name inlay hints at the call site.

**Rust-analyzer › Inlay Hints › Reborrow Hints: Enable**

Whether to show inlay hints for compiler inserted reborrows. This setting is deprecated in favor of #rust-analyzer.inlayHints.expressionAdjustmentHints.enable#.

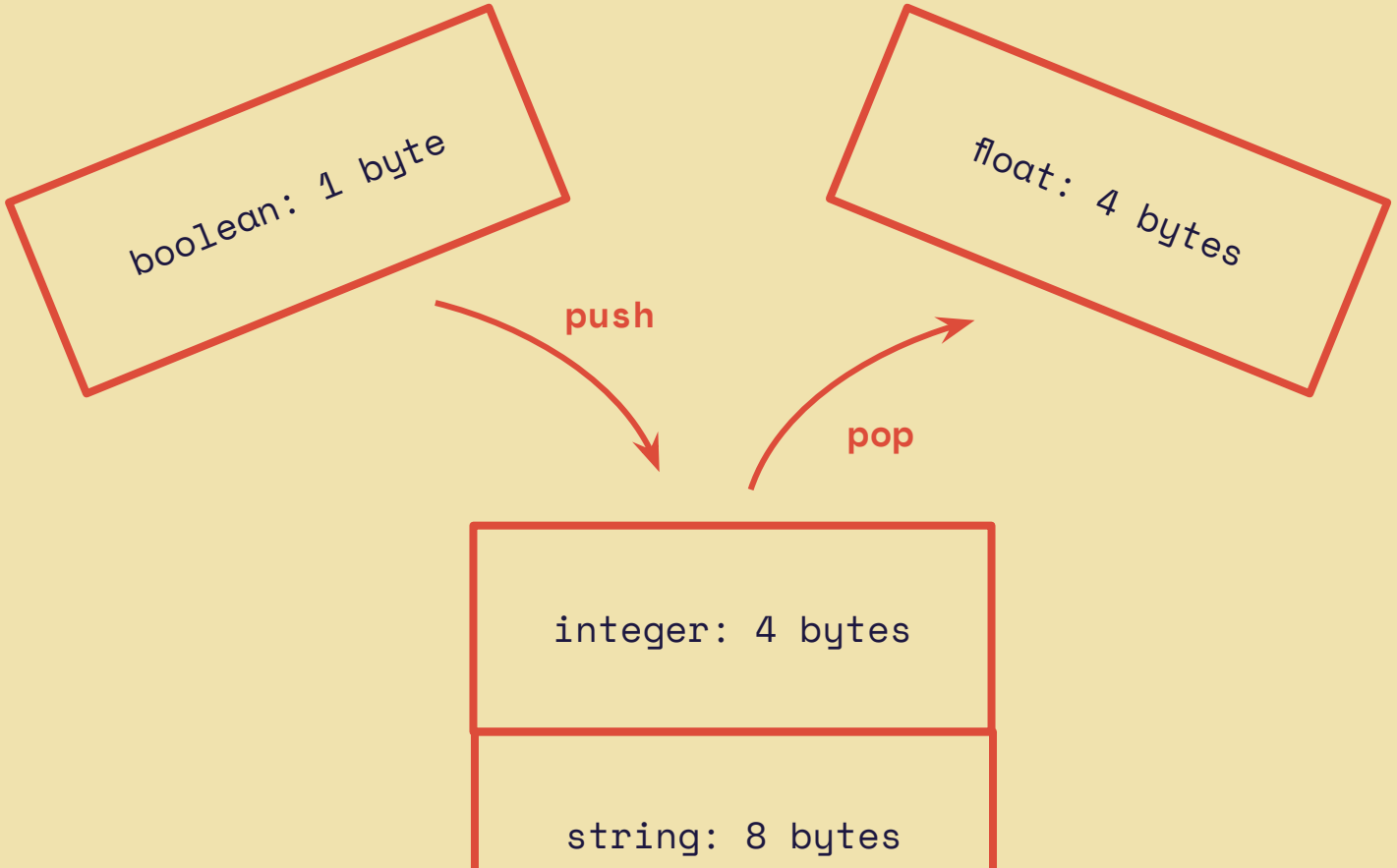
never

**Rust-analyzer › Inlay Hints › Type Hints: Enable**

Whether to show inlay type hints for variables.

# Time To Code!

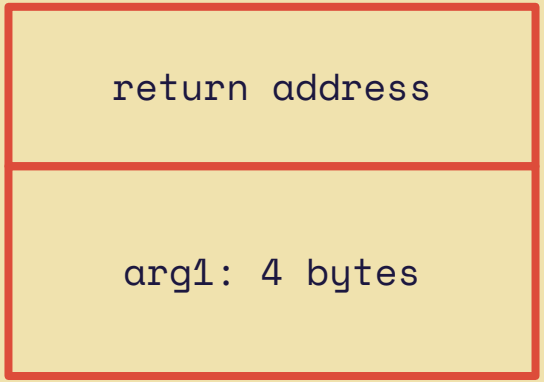
# The Stack and the Heap



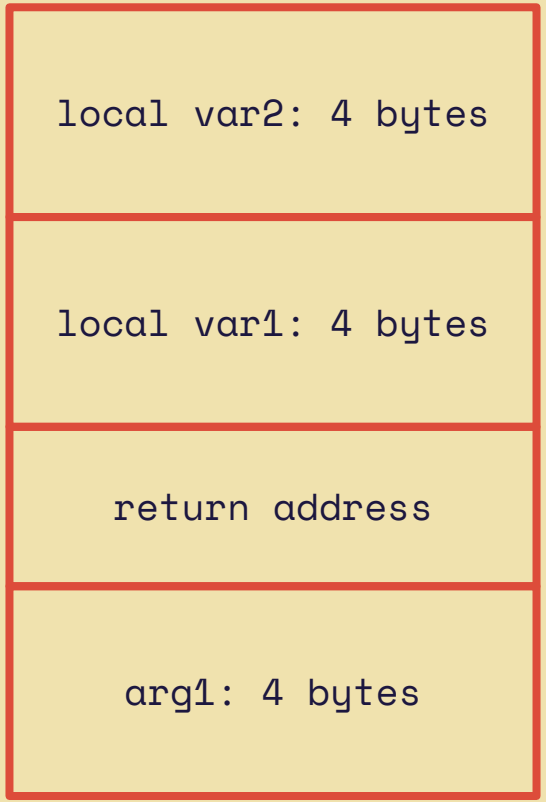
# The Stack and the Heap

arg1: 4 bytes

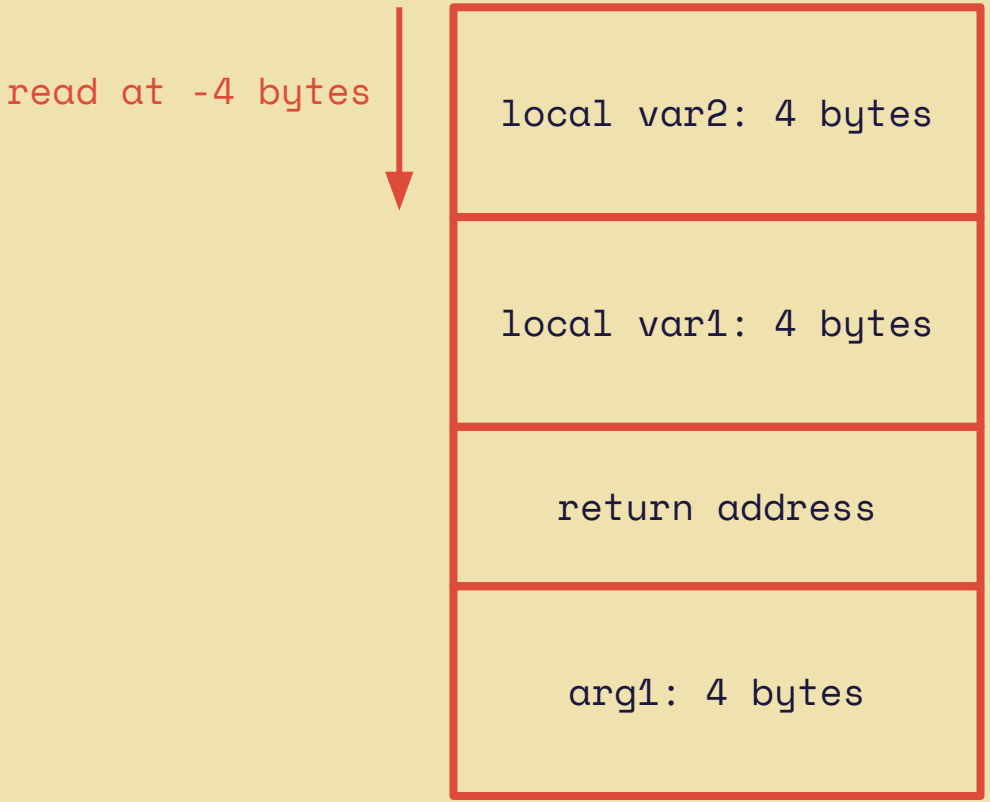
# The Stack and the Heap



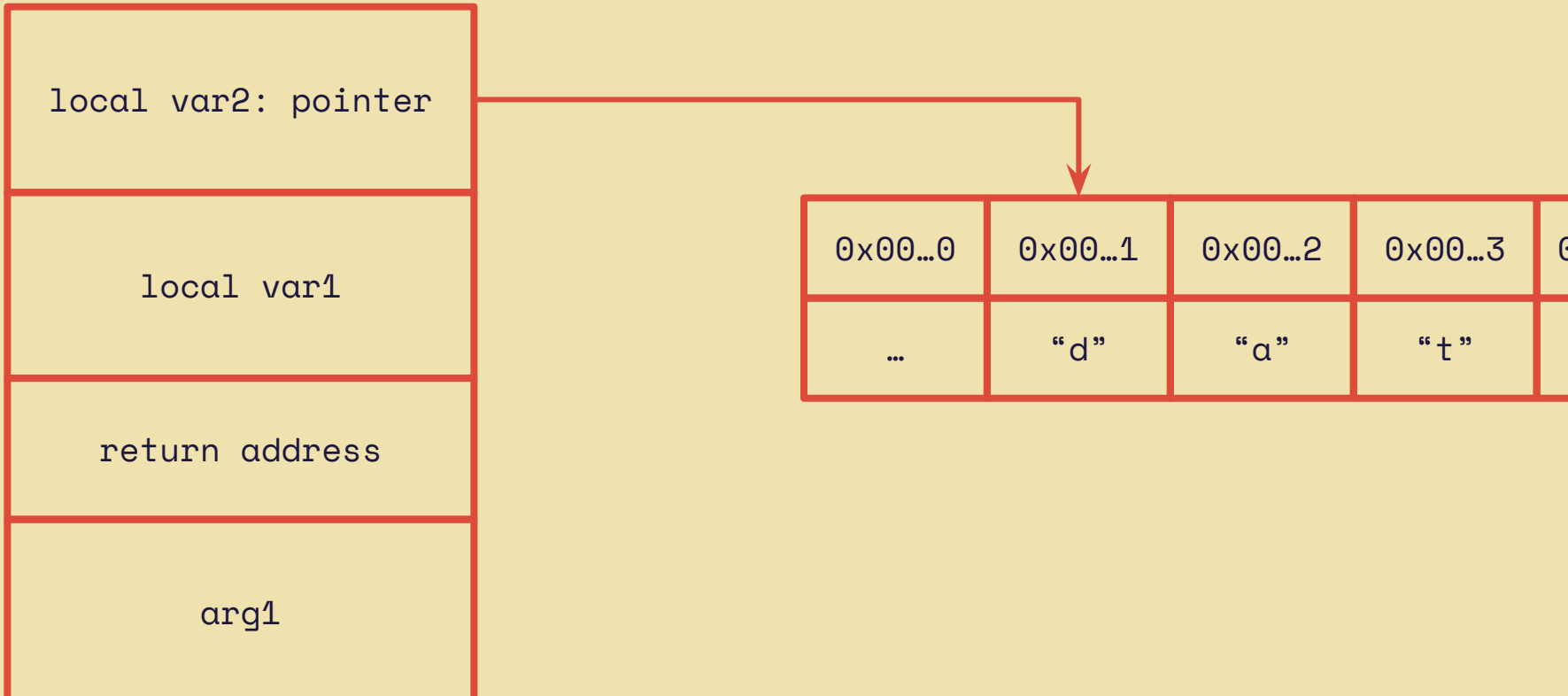
# The Stack and the Heap



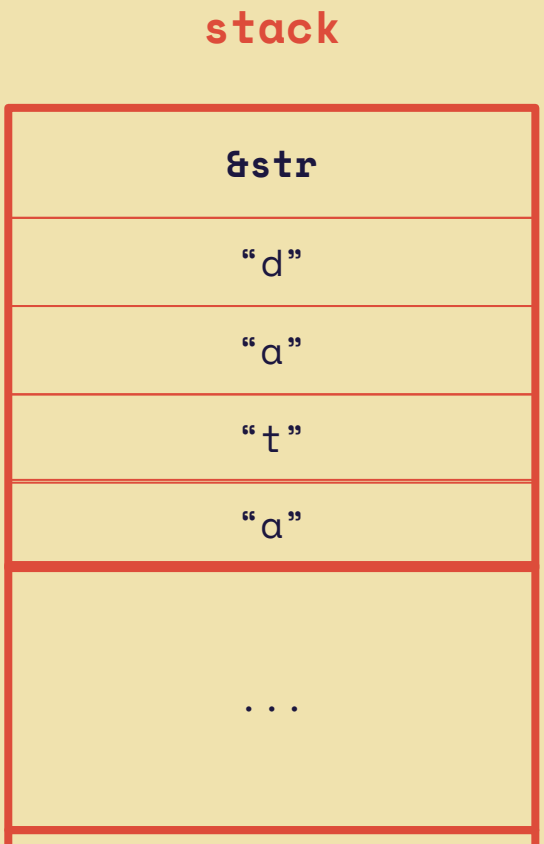
# The Stack and the Heap



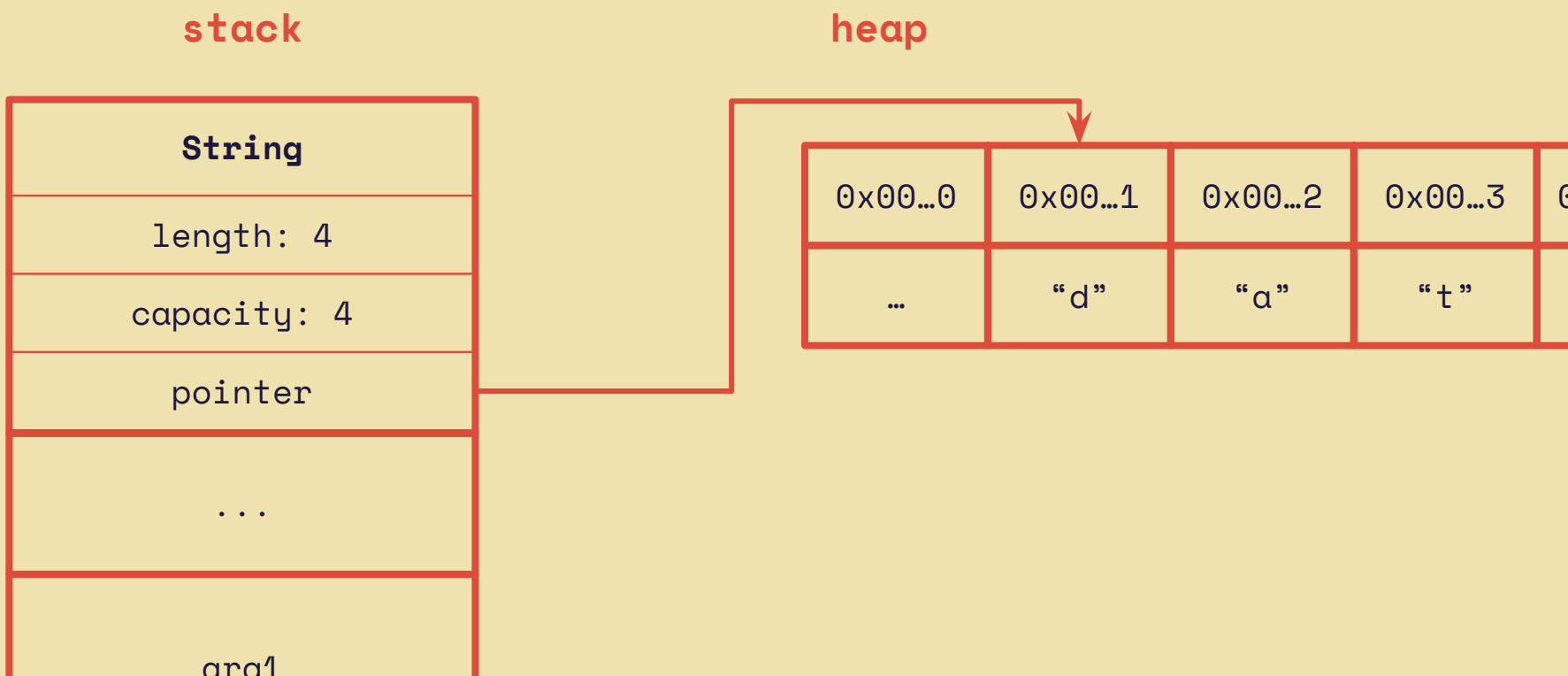
# The Stack and the Heap



# The &str type



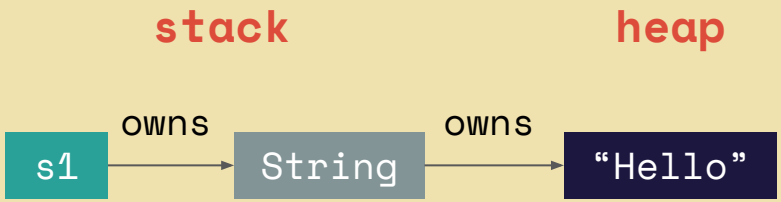
# The String Type



# Ownership

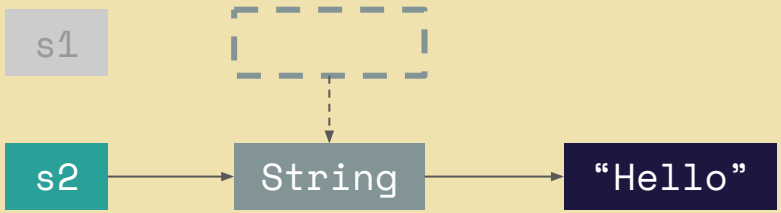
```rust
fn main() {

  let s1 = String::from("Hello");
```



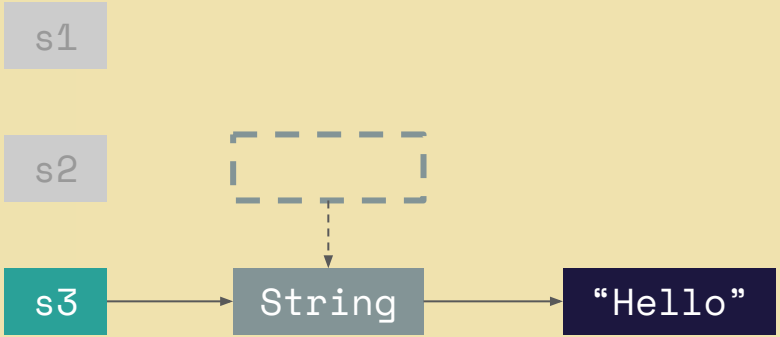
# Ownership

```rust
fn main() {
  let s1 = String::from("Hello");
  let s2 = s1;
```



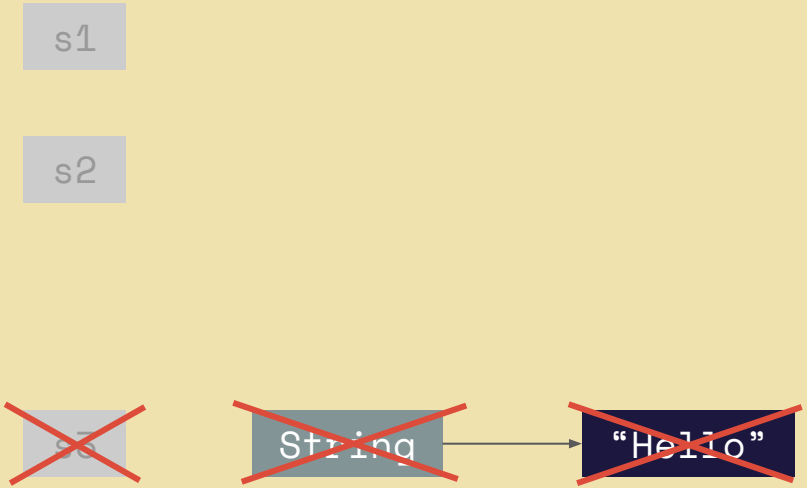
# Ownership

```rust
fn main() {
  let s1 = String::from("Hello");
  let s2 = s1;
  {
    let s3 = s2;
  }
}
```



# Ownership

```rust
fn main() {
  let s1 = String::from("Hello");
  let s2 = s1;
  {
    let s3 = s2;
  }
}
```



# The drop function

```
pub fn drop<T>(_x: T) {
  // Nothing here
}
```

# Time To Code!

# Borrowing Concept Check

```rust
let mut vec: Vec<i32> = vec![1, 2, 3];
let num: &i32 = &vec[2];
vec.push(4);
println!("3rd element is {}", num);
```

# Time To Code!

# Time To Code!

# The Delft Mercurians

As TU Delft's **first** RoboCup team, we build **cutting-edge robots** to play football against other robots around the world.

## Come with us to
## **Eindhoven in 2024**