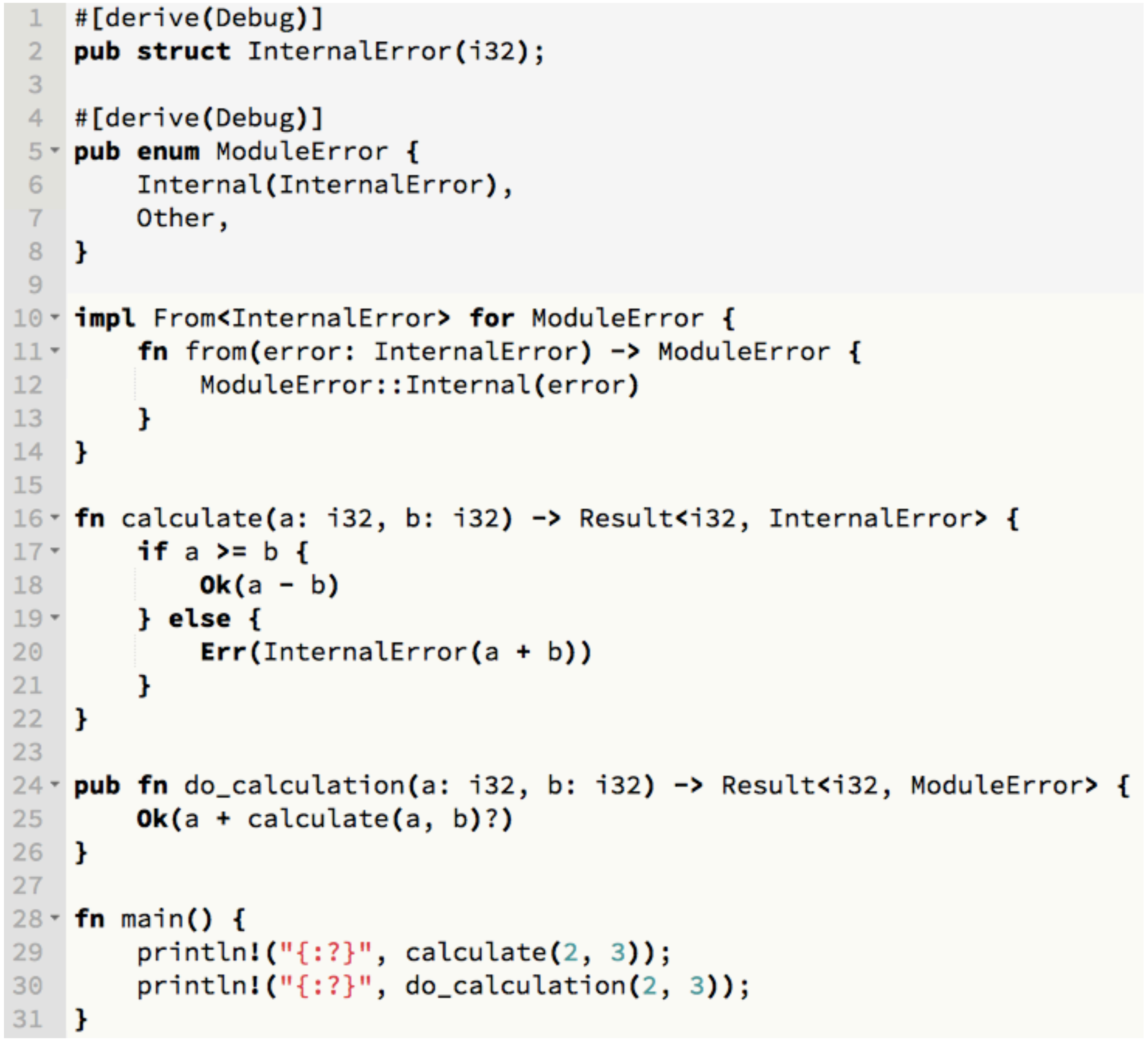
Error handling with From

- Used internally by `try!` and `?` to convert returned error to error type in returned `Result`
- Used to force trait implementers user to provide conversion for custom error types



```rust
#[derive(Debug)]
pub struct InternalError(i32);

#[derive(Debug)]
pub enum ModuleError {
    Internal(InternalError),
    Other,
}

impl From<InternalError> for ModuleError {
    fn from(error: InternalError) -> ModuleError {
        ModuleError::Internal(error)
    }
}

fn calculate(a: i32, b: i32) -> Result<i32, InternalError> {
    if a >= b {
        Ok(a - b)
    } else {
        Err(InternalError(a + b))
    }
}

pub fn do_calculation(a: i32, b: i32) -> Result<i32, ModuleError> {
    Ok(a + calculate(a, b)?)
}

fn main() {
    println!("{:?}", calculate(2, 3));
    println!("{:?}", do_calculation(2, 3));
}
```

```
Err(InternalError(5))
Err(Internal(InternalError(5)))
```