A value was used after it was mutably borrowed.

Erroneous code example:

```
fn main() {
    let mut value = 3;
    // Create a mutable borrow of `value`.
    let borrow = &mut value;
    let _sum = value + 1; // error: cannot use `value` because
                          //        it was mutably borrowed
    println!("{}", borrow);
}
```

In this example, `value` is mutably borrowed by `borrow` and cannot be used to calculate `sum`. This is not possible because this would violate Rust's mutability rules.

You can fix this error by finishing using the borrow before the next use of the value:

```
fn main() {
    let mut value = 3;
    let borrow = &mut value;
    println!("{}", borrow);
    // The block has ended and with it the borrow.
    // You can now use `value` again.
    let _sum = value + 1;
}
```

Or by cloning `value` before borrowing it:

```
fn main() {
    let mut value = 3;
    // We clone `value`, creating a copy.
    let value_cloned = value.clone();
    // The mutable borrow is a reference to `value` and
    // not to `value_cloned`...
    let borrow = &mut value;
    // ... which means we can still use `value_cloned`,
    let _sum = value_cloned + 1;
    // even though the borrow only ends here.
    println!("{}", borrow);
}
```

For more information on Rust's ownership system, take a look at the References & Borrowing section of the Book.