A borrowed variable was used by a closure.

Erroneous code example:

```
fn you_know_nothing(jon_snow: &mut i32) {
    let nights_watch = &jon_snow;
    let starks = || {
        *jon_snow = 3; // error: closure requires unique access to `jon_snow`
                       //        but it is already borrowed
    };
    println!("{}", nights_watch);
}
```

In here, `jon_snow` is already borrowed by the `nights_watch` reference, so it cannot be borrowed by the `starks` closure at the same time. To fix this issue, you can create the closure after the borrow has ended:

```
fn you_know_nothing(jon_snow: &mut i32) {
    let nights_watch = &jon_snow;
    println!("{}", nights_watch);
    let starks = || {
        *jon_snow = 3;
    };
}
```

Or, if the type implements the `Clone` trait, you can clone it between closures:

```
fn you_know_nothing(jon_snow: &mut i32) {
    let mut jon_copy = jon_snow.clone();
    let starks = || {
        *jon_snow = 3;
    };
    println!("{}", jon_copy);
}
```