An attempt was made to assign to a borrowed value.

Erroneous code example:

```
struct FancyNum {
    num: u8,
}

let mut fancy_num = FancyNum { num: 5 };
let fancy_ref = &fancy_num;
fancy_num = FancyNum { num: 6 };
// error: cannot assign to `fancy_num` because it is borrowed

println!("Num: {}, Ref: {}", fancy_num.num, fancy_ref.num);
```

Because `fancy_ref` still holds a reference to `fancy_num`, `fancy_num` can't be assigned to a new value as it would invalidate the reference.

Alternatively, we can move out of `fancy_num` into a second `fancy_num`:

```
struct FancyNum {
    num: u8,
}

let mut fancy_num = FancyNum { num: 5 };
let moved_num = fancy_num;
fancy_num = FancyNum { num: 6 };

println!("Num: {}, Moved num: {}", fancy_num.num, moved_num.num);
```

If the value has to be borrowed, try limiting the lifetime of the borrow using a scoped block:

```
struct FancyNum {
    num: u8,
}

let mut fancy_num = FancyNum { num: 5 };

{
    let fancy_ref = &fancy_num;
    println!("Ref: {}", fancy_ref.num);
}

// Works because `fancy_ref` is no longer in scope
fancy_num = FancyNum { num: 6 };
println!("Num: {}", fancy_num.num);
```

Or by moving the reference into a function:

```
struct FancyNum {
    num: u8,
}

fn print_fancy_ref(fancy_ref: &FancyNum){
    println!("Ref: {}", fancy_ref.num);
}

let mut fancy_num = FancyNum { num: 5 };

print_fancy_ref(&fancy_num);

// Works because function borrow has ended
fancy_num = FancyNum { num: 6 };
println!("Num: {}", fancy_num.num);
```