A binary assignment operator like `+=` or `^=` was applied to a type that doesn't support it.

Erroneous code example:

```
let mut x = 12f32; // error: binary operation `<<` cannot be applied to
                   //        type `f32`

x <<= 2;
```

To fix this error, please check that this type implements this binary operation. Example:

```
let mut x = 12u32; // the `u32` type does implement the `ShlAssign` trait

x <<= 2; // ok!
```

It is also possible to overload most operators for your own type by implementing the `[OP]Assign` traits from `std::ops`.

Another problem you might be facing is this: suppose you've overloaded the `+` operator for some type `Foo` by implementing the `std::ops::Add` trait for `Foo`, but you find that using `+=` does not work, as in this example:

```
use std::ops::Add;

struct Foo(u32);

impl Add for Foo {
    type Output = Foo;

    fn add(self, rhs: Foo) -> Foo {
        Foo(self.0 + rhs.0)
    }
}

fn main() {
    let mut x: Foo = Foo(5);
    x += Foo(7); // error, `+= cannot be applied to the type `Foo`
}
```

This is because `AddAssign` is not automatically implemented, so you need to manually implement it for your type.