An unary operator was used on a type which doesn't implement it.

Erroneous code example:

```
enum Question {
    Yes,
    No,
}

!Question::Yes; // error: cannot apply unary operator `!` to type `Question`
```

In this case, `Question` would need to implement the `std::ops::Not` trait in order to be able to use `!` on it. Let's implement it:

```
use std::ops::Not;

enum Question {
    Yes,
    No,
}

// We implement the `Not` trait on the enum.
impl Not for Question {
    type Output = bool;

    fn not(self) -> bool {
        match self {
            Question::Yes => false, // If the `Answer` is `Yes`, then it
                                    // returns false.
            Question::No => true, // And here we do the opposite.
        }
    }
}

assert_eq!(!Question::Yes, false);
assert_eq!(!Question::No, true);
```