Only traits defined in the current crate can be implemented for arbitrary types.

Erroneous code example:

```
impl Drop for u32 {}
```

This error indicates a violation of one of Rust's orphan rules for trait implementations. The rule prohibits any implementation of a foreign trait (a trait defined in another crate) where

- the type that is implementing the trait is foreign
- all of the parameters being passed to the trait (if there are any) are also foreign.

To avoid this kind of error, ensure that at least one local type is referenced by the `impl`:

```
pub struct Foo; // you define your type in your crate

impl Drop for Foo { // and you can implement the trait on it!
    // code of trait implementation here
#   fn drop(&mut self) { }
}

impl From<Foo> for i32 { // or you use a type from your crate as
                         // a type parameter
    fn from(i: Foo) -> i32 {
        0
    }
}
```

Alternatively, define a trait locally and implement that instead:

```
trait Bar {
    fn get(&self) -> usize;
}

impl Bar for u32 {
    fn get(&self) -> usize { 0 }
}
```

For information on the design of the orphan rules, see RFC 1023.