The `#[rustc_on_unimplemented]` attribute lets you specify a custom error message for when a particular trait isn't implemented on a type placed in a position that needs that trait. For example, when the following code is compiled:

```
#![feature(rustc_attrs)]

#[rustc_on_unimplemented = "error on `{Self}` with params `<{A},{B}>`"] // error
trait BadAnnotation<A> {}
```

There will be an error about `bool` not implementing `Index<u8>`, followed by a note saying "the type `bool` cannot be indexed by `u8`".

As you can see, you can specify type parameters in curly braces for substitution with the actual types (using the regular format string syntax) in a given situation. Furthermore, `{Self}` will substitute to the type (in this case, `bool`) that we tried to use.

This error appears when the curly braces contain an identifier which doesn't match with any of the type parameters or the string `Self`. This might happen if you misspelled a type parameter, or if you intended to use literal curly braces. If it is the latter, escape the curly braces with a second curly brace of the same type; e.g., a literal `{` is `{{`.