The `Self` keyword was used outside an impl, trait, or type definition.

Erroneous code example:

```
<Self>::foo; // error: use of `Self` outside of an impl, trait, or type
             // definition
```

The `Self` keyword represents the current type, which explains why it can only be used inside an impl, trait, or type definition. It gives access to the associated items of a type:

```
trait Foo {
    type Bar;
}

trait Baz : Foo {
    fn bar() -> Self::Bar; // like this
}
```

However, be careful when two types have a common associated type:

```
trait Foo {
    type Bar;
}

trait Foo2 {
    type Bar;
}

trait Baz : Foo + Foo2 {
    fn bar() -> Self::Bar;
    // error: ambiguous associated type `Bar` in bounds of `Self`
}
```

This problem can be solved by specifying from which trait we want to use the `Bar` type:

```
trait Foo {
    type Bar;
}

trait Foo2 {
    type Bar;
}

trait Baz : Foo + Foo2 {
    fn bar() -> <Self as Foo>::Bar; // ok!
}
```