You tried to use a type which doesn't implement some trait in a place which expected that trait.

Erroneous code example:

```
// here we declare the Foo trait with a bar method
trait Foo {
    fn bar(&self);
}

// we now declare a function which takes an object implementing the Foo trait
fn some_func<T: Foo>(foo: T) {
    foo.bar();
}

fn main() {
    // we now call the method with the i32 type, which doesn't implement
    // the Foo trait
    some_func(5i32); // error: the trait bound `i32 : Foo` is not satisfied
}
```

In order to fix this error, verify that the type you're using does implement the trait. Example:

```
trait Foo {
    fn bar(&self);
}

// we implement the trait on the i32 type
impl Foo for i32 {
    fn bar(&self) {}
}

fn some_func<T: Foo>(foo: T) {
    foo.bar(); // we can now use this method since i32 implements the
               // Foo trait
}

fn main() {
    some_func(5i32); // ok!
}
```

Or in a generic context, an erroneous code example would look like:

```
fn some_func<T>(foo: T) {
    println!("{:?}", foo); // error: the trait `core::fmt::Debug` is not
                           //        implemented for the type `T`
}
```

```
fn main() {
    // We now call the method with the i32 type,
    // which *does* implement the Debug trait.
    some_func(5i32);
}
```

Note that the error here is in the definition of the generic function. Although we only call it with a parameter that does implement `Debug`, the compiler still rejects the function. It must work with all possible input types. In order to make this example compile, we need to restrict the generic type we're accepting:

```
use std::fmt;

// Restrict the input type to types that implement Debug.
fn some_func<T: fmt::Debug>(foo: T) {
    println!("{:?}", foo);
}

fn main() {
    // Calling the method is still fine, as i32 implements Debug.
    some_func(5i32);

    // This would fail to compile now:
    // struct WithoutDebug;
    // some_func(WithoutDebug);
}
```

Rust only looks at the signature of the called function, as such it must already specify all requirements that will be used for every type parameter.