Return type involving a trait did not require `'static` lifetime.

Erroneous code examples:

```
use std::fmt::Debug;

fn foo(x: &i32) -> impl Debug { // error!
    x
}

fn bar(x: &i32) -> Box<dyn Debug> { // error!
    Box::new(x)
}
```

Add `'static` requirement to fix them:

```
# use std::fmt::Debug;
fn foo(x: &'static i32) -> impl Debug + 'static { // ok!
    x
}

fn bar(x: &'static i32) -> Box<dyn Debug + 'static> { // ok!
    Box::new(x)
}
```

Both `dyn Trait` and `impl Trait` in return types have an implicit `'static` requirement, meaning that the value implementing them that is being returned has to be either a `'static` borrow or an owned value.

In order to change the requirement from `'static` to be a lifetime derived from its arguments, you can add an explicit bound, either to an anonymous lifetime `'_` or some appropriate named lifetime.

```
# use std::fmt::Debug;
fn foo(x: &i32) -> impl Debug + '_ {
    x
}
fn bar(x: &i32) -> Box<dyn Debug + '_> {
    Box::new(x)
}
```

These are equivalent to the following explicit lifetime annotations:

```
# use std::fmt::Debug;
fn foo<'a>(x: &'a i32) -> impl Debug + 'a {
    x
}
fn bar<'a>(x: &'a i32) -> Box<dyn Debug + 'a> {
    Box::new(x)
}
```