An argument lifetime was elided in an async function.

Erroneous code example:

When a struct or a type is bound/declared with a lifetime it is important for the Rust compiler to know, on usage, the lifespan of the type. When the lifetime is not explicitly mentioned and the Rust Compiler cannot determine the lifetime of your type, the following error occurs.

```
use futures::executor::block_on;
struct Content<'a> {
    title: &'a str,
    body: &'a str,
}
async fn create(content: Content) { // error: implicit elided
                                    // lifetime not allowed here
    println!("title: {}", content.title);
    println!("body: {}", content.body);
}
let content = Content { title: "Rust", body: "is great!" };
let future = create(content);
block_on(future);
```

Specify desired lifetime of parameter `content` or indicate the anonymous lifetime like `content: Content<'_>`. The anonymous lifetime tells the Rust compiler that `content` is only needed until create function is done with it's execution.

The `implicit elision` meaning the omission of suggested lifetime that is `pub async fn create<'a>(content: Content<'a>) {}` is not allowed here as lifetime of the `content` can differ from current context:

```
ignore (needs futures dependency) async fn create(content: Content<'_>)
{ // ok!     println!("title: {}", content.title);     println!("body:
{}", content.body); }
```

Know more about lifetime elision in this chapter and a chapter on lifetimes can be found here.