A pattern binding is using the same name as one of the variants of a type.

Erroneous code example:

```
# #![deny(warnings)]
enum Method {
    GET,
    POST,
}

fn is_empty(s: Method) -> bool {
    match s {
        GET => true,
        _ => false
    }
}

fn main() {}
```

Enum variants are qualified by default. For example, given this type:

```
enum Method {
    GET,
    POST,
}
```

You would match it using:

```
enum Method {
    GET,
    POST,
}

let m = Method::GET;

match m {
    Method::GET => {},
    Method::POST => {},
}
```

If you don't qualify the names, the code will bind new variables named "GET" and "POST" instead. This behavior is likely not what you want, so `rustc` warns when that happens.

Qualified names are good practice, and most code works well with them. But if you prefer them unqualified, you can import the variants into scope:

```
use Method::*;
enum Method { GET, POST }
# fn main() {}
```

If you want others to be able to import variants from your module directly, use `pub use`:

```
pub use Method::*;
pub enum Method { GET, POST }
# fn main() {}
```