A `#[repr(..)]` attribute was placed on an unsupported item.

Examples of erroneous code:

```
#[repr(C)]
type Foo = u8;

#[repr(packed)]
enum Foo {Bar, Baz}

#[repr(u8)]
struct Foo {bar: bool, baz: bool}

#[repr(C)]
impl Foo {
    // ...
}
```

- The `#[repr(C)]` attribute can only be placed on structs and enums.
- The `#[repr(packed)]` and `#[repr(simd)]` attributes only work on structs.
- The `#[repr(u8)]`, `#[repr(i16)]`, etc attributes only work on enums.

These attributes do not work on typedefs, since typedefs are just aliases.

Representations like `#[repr(u8)]`, `#[repr(i64)]` are for selecting the discriminant size for enums with no data fields on any of the variants, e.g. `enum Color {Red, Blue, Green}`, effectively setting the size of the enum to the size of the provided type. Such an enum can be cast to a value of the same type as well. In short, `#[repr(u8)]` makes the enum behave like an integer with a constrained set of allowed values.

Only field-less enums can be cast to numerical primitives, so this attribute will not apply to structs.

`#[repr(packed)]` reduces padding to make the struct size smaller. The representation of enums isn't strictly defined in Rust, and this attribute won't work on enums.

`#[repr(simd)]` will give a struct consisting of a homogeneous series of machine types (i.e., `u8`, `i32`, etc) a representation that permits vectorization via SIMD. This doesn't make much sense for enums since they don't consist of a single list of data.