A plugin/crate was declared but cannot be found.

Erroneous code example:

```
#![feature(plugin)]
#![plugin(cookie_monster)] // error: can't find crate for `cookie_monster`
extern crate cake_is_a_lie; // error: can't find crate for `cake_is_a_lie`
```

You need to link your code to the relevant crate in order to be able to use it (through Cargo or the `-L` option of rustc example). Plugins are crates as well, and you link to them the same way.

Common causes

- The crate is not present at all. If using Cargo, add it to `[dependencies]` in Cargo.toml.
- The crate is present, but under a different name. If using Cargo, look for `package = ` under `[dependencies]` in Cargo.toml.

Common causes for missing `std` or `core`

- You are cross-compiling for a target which doesn't have `std` prepackaged. Consider one of the following:
  - Adding a pre-compiled version of std with `rustup target add`
  - Building std from source with `cargo build -Z build-std`
  - Using `#![no_std]` at the crate root, so you won't need `std` in the first place.
- You are developing the compiler itself and haven't built libstd from source. You can usually build it with `x.py build library/std`. More information about x.py is available in the rustc-dev-guide.