BLOCK NONCE #8: Early return

What?

When you have an if statement without an else part, flip the condition, use return to exit early, and move the main code outside the if.

Why?

This makes code easier to understand by removing nested ifs. Your code becomes a simple list of checks followed by the main logic.

How?

```rust
fn watch_movie(user: User) -> Result<(), WatchMovieError> {
    if user.is_authenticated() {
        if user.is_premium_plan() {
            user.watch_movie()
        } else {
            Err(WatchMovieError::NonPremiumUser)
        }
    } else {
        Err(WatchMovieError::NotAuthenticated)
    }
}
```

becomes

```rust
fn watch_movie(user: User) -> Result<(), WatchMovieError> {
    if !user.is_authenticated() {
        return Err(WatchMovieError::NotAuthenticated);
    }
    if !user.is_premium_plan() {
        return Err(WatchMovieError::NonPremiumUser);
    }
    user.watch_movie()
}
```

Note that Rust's ? operator is another form of early return.

Don't overuse it. Early returns can make the code complicated as it can end in the middle. Prefer using early return at the start of the function

TIP💡
1. This pattern also applies to loops by using `continue`.
2. For Rust developers, use the `let else` syntax to early return if you only handle one enum variant
3. Sometimes, you need to extract code to a function in order to use early return. Do that!

#starkcode