```
my %commands = (
    "happy" => \&joy,
    "sad",  => \&sullen,
    "done"  => sub { die "See ya!" },
    "mad"   => \&angry,
);

print "How are you? ";
chomp($string = <STDIN>);
if ($commands{$string}) {
    $commands{$string}->();
} else {
    print "No such command: $string\n";
}
```

23.1.23 How can I catch accesses to undefined variables, functions, or methods?

The AUTOLOAD method, discussed in Autoloading in *perlsub* and AUTOLOAD: Proxy Methods in *perltoot*, lets you capture calls to undefined functions and methods.

When it comes to undefined variables that would trigger a warning under `use warnings`, you can promote the warning to an error.

```
use warnings FATAL => qw(uninitialized);
```

23.1.24 Why can't a method included in this same file be found?

Some possible reasons: your inheritance is getting confused, you've misspelled the method name, or the object is of the wrong type. Check out *perltoot* for details about any of the above cases. You may also use `print ref($object)` to find out the class `$object` was blessed into.

Another possible reason for problems is because you've used the indirect object syntax (eg, `find Guru "Samy"`) on a class name before Perl has seen that such a package exists. It's wisest to make sure your packages are all defined before you start using them, which will be taken care of if you use the `use` statement instead of `require`. If not, make sure to use arrow notation (eg., `Guru->find("Samy")`) instead. Object notation is explained in *perlobj*.

Make sure to read about creating modules in *perlmod* and the perils of indirect objects in Method Invocation in *perlobj*.

23.1.25 How can I find out my current package?

If you're just a random program, you can do this to find out what the currently compiled package is:

```
my $packname = __PACKAGE__;
```

But, if you're a method and you want to print an error message that includes the kind of object you were called on (which is not necessarily the same as the one in which you were compiled):

```
sub amethod {
    my $self  = shift;
    my $class = ref($self) || $self;
    warn "called me from a $class object";
}
```