Stata users often put numeric or string values into scalars, which is as easily done as

```
. scalar answer = 42
. scalar question = "What is the answer?"
```

The main discussion of scalars is in [P] scalar. Scalars can be used interactively or in programs and are faster and more accurate than local macros for holding values. This advantage would not matter with a number like 42, but it could easily matter with a number that was very small or very large.

Scalars have one major pitfall. It is documented in [P] scalar, but users are often bitten by it, so here is another warning. If a variable and a scalar have the same name, Stata always assumes that you mean the variable, not the scalar. This naming can apply even more strongly than you first guess: recall that variable names can be abbreviated so long as the abbreviation is unambiguous (see [U] 11.2 Abbreviation rules).

Suppose that you are using the `auto` dataset:

```
. sysuse auto
(1978 Automobile Data)
. scalar p = 0.7
. display p
4099
```

What happened to 0.7? Nothing. It is still there:

```
. display scalar(p)
.7
. scalar list p
   p = .7
```

What is the 4099 result? The dataset has a variable, `price`, and no other variable names begin with `p`, so `p` is understood to mean `price`. Moreover, `display` assumes that if you specify just a variable name, you want to see its value in the first observation, namely, `price[1]`. (The full explanation for that is another story.)

What should users do to be safe?

1. Use a different name. For example, you might introduce a personal convention about uppercase and lowercase. Many Stata users use only lowercase letters (and possibly numeric digits and underscores too) within variable names. Such users
could distinguish scalars by using at least one uppercase letter in their names. You could also use a prefix such as `sc_`, as in `sc_p`. If you forget your convention or if a user of your programs does not know about this convention, Stata cannot sense the interpretation you want. Thus this method is not totally safe.

2. Use the `scalar()` pseudofunction to spell out that you want a scalar. This method is totally safe, but some people find it awkward.

3. Use a temporary name for a scalar, as in

   ```
   . tempname p
   . scalar `p' = 0.7
   ```

   Each scalar with a temporary name will be visible while the program in which it occurs is running and only in that program. Temporary names can be used interactively, too. (Your interactive session also counts for this purpose as a program.) In many ways, this method is the best solution, as it ensures that scalars can be seen only locally, which is usually better programming style.