while — Looping

Syntax

```
while exp {
    stata_commands
}
```

Braces must be specified with `while`, and
1. the open brace must appear on the same line as `while`;
2. nothing may follow the open brace, except, of course, comments; the first command to be executed must appear on a new line;
3. the close brace must appear on a line by itself.

Description

`while` evaluates `exp` and, if it is true (nonzero), executes the `stata_commands` enclosed in the braces. It then repeats the process until `exp` evaluates to false (zero). `while`s may be nested within `while`s. If the `exp` refers to any variables, their values in the first observation are used unless explicit subscripts are specified; see [U] 13.7 Explicit subscripting.

Also see [P] foreach and [P] forvalues for alternatives to `while`.

Remarks and examples

`while` may be used interactively, but it is most often used in programs. See [U] 18 Programming Stata for a description of programs.

The `stata_commands` enclosed in the braces may be executed once, many times, or not at all. For instance,

```
program demo
    local i = i
    while `i'>0 {
        display "i is now `i'"
        local i = `i' - 1
    }
    display "done"
end
. demo 2
i is now 2
i is now 1
done
. demo 0
done
```
The above example is a bit contrived in that the best way to count down to one would be

```stata
program demo
    forvalues i = 1(-1)1 {
        display "i is now \"i\""
    }
    display "done"
end
```

while is used mostly in parsing contexts

```stata
program ...
    ...
    gettoken tok 0 : 0
    while "\"tok\"" != ""
        ...
        gettoken tok 0 : 0
    }
    ...
end
```

or in mathematical contexts where we are iterating

```stata
program ...
    ...
    scalar \texttt{curval} =.
    scalar \texttt{lastval} =.
    while abs(\texttt{lastval} - \texttt{curval}) > \texttt{epsilon} {
        scalar \texttt{lastval} = \texttt{curval}
        scalar \texttt{curval} = ...
    }
    ...
end
```

or in any context in which loop termination is based on calculation (whether it be numeric or string).

You can also create endless loops by using while,

```stata
program ...
    ...
    while 1 {
        ...
    }
end
```

which is not really an endless loop if the code reads

```stata
program ...
    ...
    while 1 {
        if (...) exit
        ...
    }
    // this line is never reached
end
```

Should you make a mistake and really create an endless loop, you can stop program execution by pressing the Break key.
Also see

[P] continue — Break out of loops
[P] foreach — Loop over items
[P] forvalues — Loop over consecutive values
[P] if — if programming command
[U] 13 Functions and expressions
[U] 18 Programming Stata