

For/Switch/While/Try

UC Berkeley

Fall 2004, E77

<http://jagger.me.berkeley.edu/~pack/e77>

Copyright 2005, Andy Packard. This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

for, end (page 163-174)


Execute collection of statements a fixed number of times.

```
for x=expression
```

```
    statements
```

```
end
```

Execution continues with any
statements beyond the end



The **expression** is evaluated once before the loop starts. The value is called the *controlvalue*. The **statements** are executed one time for each column in the *controlvalue*. In the code above, **x** is the *loopvariable*.

Before each “execution pass,” the *loopvariable* (**x**) is assigned to the corresponding column of *controlvalue*


1st column on first pass,

2nd column on second pass, and so on

for loops, most common use(!)

The most common value for expression is a row vector of integers, starting at **1**, and increasing to a limit **n**

```
for x=1:n
    statements
end
```



The *controlvalue* is simply the row vector

[**1** **2** **3** **4** ... **n**]

Hence, the **statements** are executed **n** times.

- The first time through, the value of **x** is set equal to **1**;
- the **k**'th time through, the value of **x** is set equal to **k**.

for loops, most common use

The expression can be created before the loop itself, so

```
for x=1:n  
    statements  
end
```

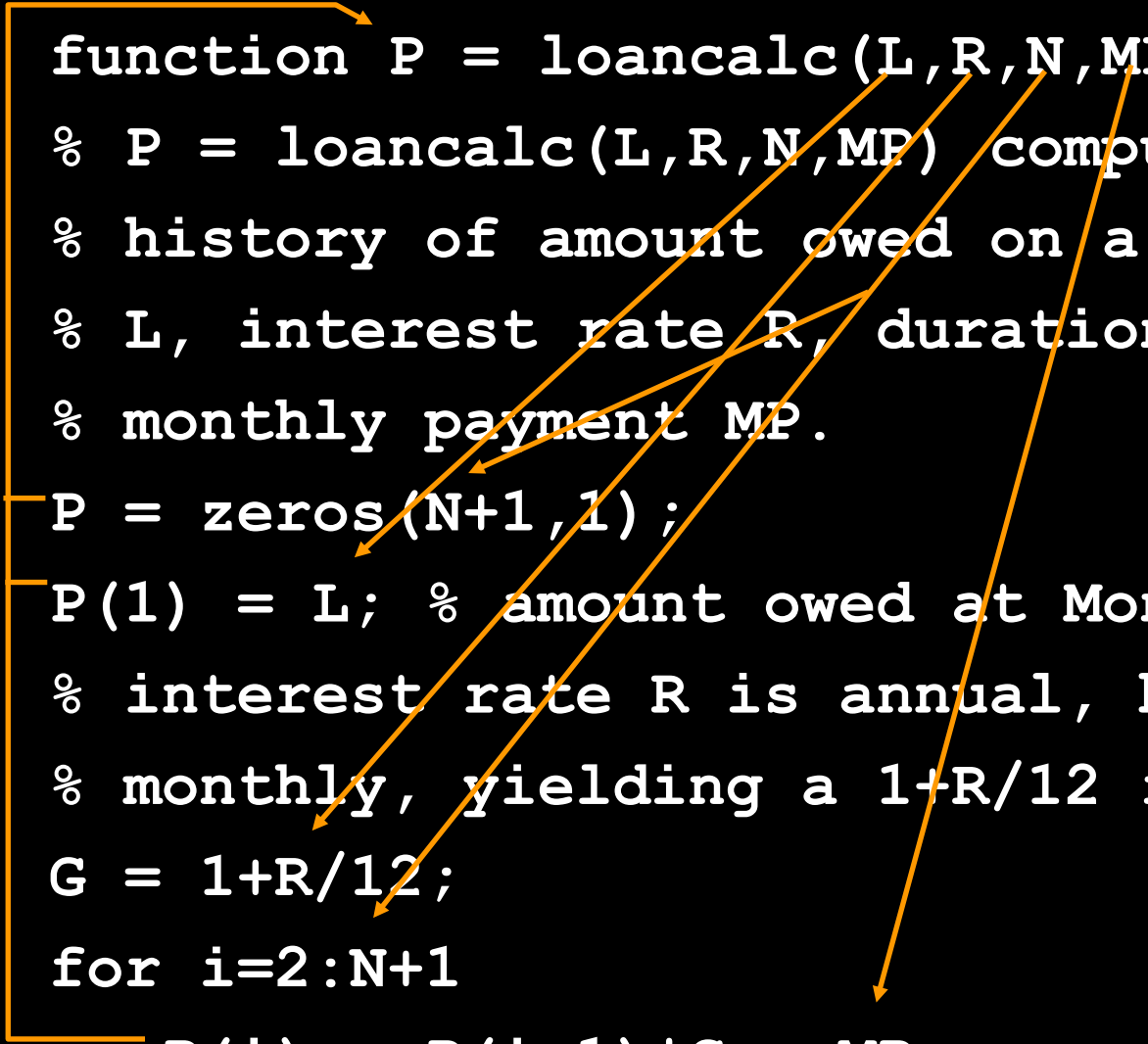
```
xValues = 1:n;  
for x=xValues  
    statements  
end
```

is the same as

A diagram illustrating the equivalence of two MATLAB for loop syntaxes. On the left, a code snippet shows a for loop with a range expression 'x=1:n' and 'statements' inside, followed by 'end'. On the right, a code snippet shows a variable 'xValues' assigned to '1:n', followed by a for loop with 'x=xValues', 'statements' inside, and 'end'. Two white arrows originate from the 'end' of the first snippet and the 'end' of the second snippet, both pointing towards the central text 'is the same as'.

for loop example: loancalc.m

```
function P = loancalc(L,R,N,MP)
% P = loancalc(L,R,N,MP) computes the
% history of amount owed on a loan of amount
% L, interest rate R, duration N, and fixed
% monthly payment MP.
P = zeros(N+1,1);
P(1) = L; % amount owed at Month=0
% interest rate R is annual, but applied
% monthly, yielding a 1+R/12 factor.
G = 1+R/12;
for i=2:N+1
    P(i) = P(i-1)*G - MP;
end
```



switch, case, otherwise end

switch expression

case testval1

statements1

case testval2

statements2

otherwise

statementsOther

end

Let **VAL** be value of **expression**.

Assume **VAL** is a scalar double.

Does **VAL** equal **testval1**?

Yes: Execute statements1

Jump past end.

No:

Does **VAL** equal **testval2**?

Yes: Execute statements2

Jump past end

No:

Execute statementsOther

Jump past end

Move to code beyond **end**



Any number of cases are allowed. There does not have to be an otherwise (and associated statements).

switch, case, otherwise end

switch expression

case testval1

statements1

case testval2

statements2

otherwise

statementsOther

end

VAL (the value of **expression**) need not be a *scalar double*. It can also be a *char array*.

Matlab uses **strcmp** (string compare) to check equality of **VAL** to the various **testval1**, **testval2**, etc.

```
switch, case, otherwise end
```

```
switch expression
```

```
case testval1
```

```
    statements1
```

```
case value2
```

```
    statements2
```

```
otherwise
```

```
    statementsOther
```

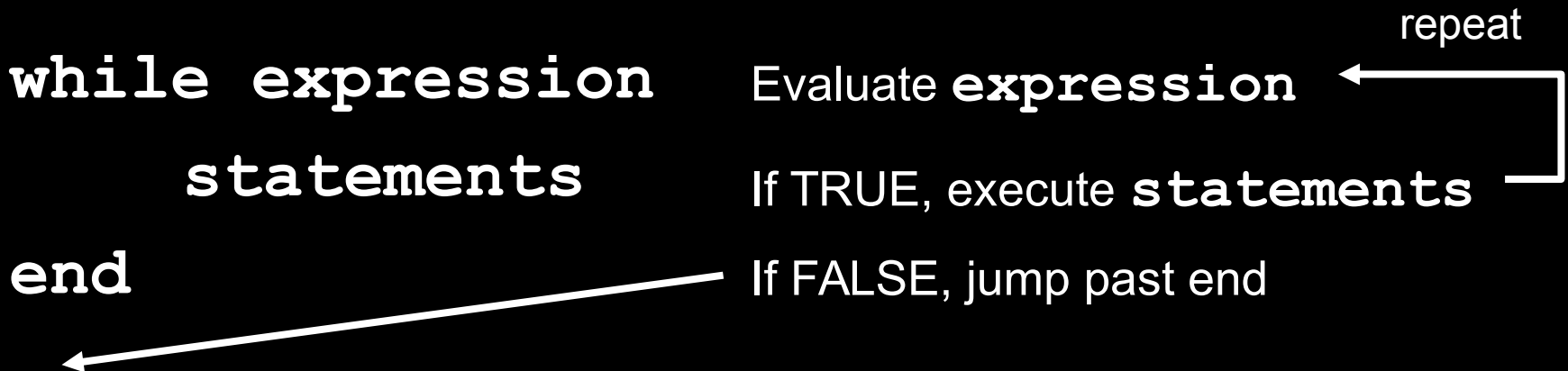
```
end
```

testval can also be cell arrays. The equality of **VAL** (value of **expression**) is tested for any of the contents of the cell array **testval**.

while

Executing commands an undetermined number of times.

```
while expression
  statements
end
```



Example using `tic/toc`

`tic` is a built-in Matlab function that starts a timer.

Every subsequent call to `toc` (also Matlab built-in) returns the elapsed time (in seconds) since the originating call to `tic`.

The code below will cause the program to “pause” for one second before proceeding.

```
tic
while toc<1
end
```

It would be clumsy to do this without a `while`-loop

try/catch

try

statements1

catch

statements2

end

Try to execute these commands

If a run-time error occurs while executing **statements1**, stop, and...

Execute these commands

If no error occurs executing **statements1**, jump beyond **end**.

Do not execute **statements2**