GAP programming language quick reference

GAP input

digits, uppercase and lowercase letters, *space*, *tab*, *newline*, and the special characters:

```
"     `     (     )     *     +     ,     _     #
.     /     :     ;     <     =     >     ~     &
[     \     ]     ^     -     {     }     !     ?
```

Operator and delimiter symbols

```
+     -     *     /     ^     ~     !.
=     <>    <     <=    >     >=    ![
:=    .     ..    ->    ,     ;     !{
[     ]     {     }     (     )     :
```

Keywords

```
and       do        elif     else      end       fi
for       function  if       in        local     mod
not       od        or       repeat    return    then
until     while     quit     QUIT      break     rec
continue

false     true      IsBound   Unbind    TryNextMethod
Info      Assert    SaveWorkspace       fail
```

Indentifiers

Consists of letters, digits, and underscores _, and must contain at least one letter or underscore.

Operators

Comparisons	=	<>	<	<=	>	>=	in
Arithmetic operators	+	-	*	/	mod	^	
Logical operators	not		and		or		

Function and procedure calls:

function-var`()`
function-var`(` *arg-expr*`[,` *arg-expr*`, ...] )`
function-var`(` *arg-expr*`[,` *arg-expr*`, ...][ : [` *option-expr* `[,`*option-expr*`, ....]]])`
procedure-var`();`
procedure-var`(` *arg-expr* `[,`*arg-expr*`, ...] );`

IF

```
  if bool-expr1 then
    statements1
{ elif bool-expr2 then
    statements2 }
[ else
    statements3 ]
  fi;
```

WHILE

```
while bool-expr do
  statements
od;
```

REPEAT

```
repeat
  statements
until bool-expr;
```

FOR

```
for simple-var in list-expr do
  statements
od;
```

```
for variable in iterator do
  statements
od;
```

```
for variable in object do
  statements
od;
```

Other statements to be used inside a loop

`break;` causes an immediate exit from the innermost loop enclosing it.

`continue;` causes the rest of the current iteration of the innermost loop enclosing it to be skipped.

FUNCTION

```
function( [ arg-ident {, arg-ident} ] )
[local loc-ident {, loc-ident} ; ]
statements
end
```

arg-ident **`->`** *expr* shorthand for `function ( arg-ident ) return expr; end.`

RETURN

```
return;
return expr;
```