

## Syntax

```
var = value;
var = cond ? value_if_true : value_if_false;
var = function (x) x + x;
module name(...) { ... }
name();
function name(...) = ...
name();
include <...scad>
use <...scad>
```

## Constants

```
undef undefined value
PI mathematical constant  $\pi$  (~3.14159)
```

## Operators

```
n + m Addition
n - m Subtraction
n * m Multiplication
n / m Division
n % m Modulo
n ^ m Exponentiation
n < m Less Than
n <= m Less or Equal
b == c Equal
b != c Not Equal
n >= m Greater or Equal
n > m Greater Than
b && c Logical And
b || c Logical Or
!b Negation
```

## Special variables

```
$fa minimum angle
$fs minimum size
$fn number of fragments
$t animation step
$vp viewport rotation angles in degrees
$vp viewport translation
$vpd viewport camera distance
$vpf viewport camera field of view
$children number of module children
$preview true in F5 preview, false for F6
```

## Modifier Characters

```
* disable
! show only
# highlight / debug
% transparent / background
```

## 2D

```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font,
      halign, valign, spacing,
      direction, language, script)
import("...ext", convexity)
projection(cut)
```

## 3D

```
sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height], center)
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, faces, convexity)
import("...ext", convexity)
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(angle,convexity)
surface(file = "...ext",center,convexity)
```

## Transformations

```
translate([x,y,z])
rotate([x,y,z])
rotate(a, [x,y,z])
scale([x,y,z])
resize([x,y,z],auto,convexity)
mirror([x,y,z])
multmatrix(m)
color("colorname",alpha)
color("#hexvalue")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski(convexity)
```

## Lists

```
list = [...,...,...]; create a list
var = list[2]; index a list (from 0)
var = list.z; dot notation indexing (x/y/z)
```

## Boolean operations

```
union()
difference()
intersection()
```

## List Comprehensions

```
Generate [ for (i = range|list) i ]
Generate [ for (init;condition;next) i ]
Flatten [ each i ]
Conditions [ for (i = ...) if (condition(i)) i ]
Conditions [ for (i = ...) if (condition(i)) x else y ]
Assignments [ for (i = ...) let (assignments) a ]
```

## Flow Control

```
for (i = [start:end]) { ... }
for (i = [start:step:end]) { ... }
for (i = [...,...,...]) { ... }
for (i = ..., j = ..., ...) { ... }
intersection for(i = [start:end]) { ... }
intersection for(i = [start:step:end]) { ... }
intersection for(i = [...,...,...]) { ... }
if (...) { ... }
let (...) { ... }
```

## Type test functions

```
is undef
is bool
is num
is string
is list
is function
```

## Other

```
echo(...)
render(convexity)
children([idx])
assert(condition, message)
assign(...) { ... }
```

## Functions

```
concat
lookup
str
chr
ord
search
version
version_num
parent_module(idx)
```

## Mathematical

```
abs
sign
sin
cos
tan
acos
asin
atan
atan2
floor
round
ceil
ln
len
let
log
pow
sqrt
exp
rands
min
max
norm
cross
```