KCL: KittyCAD Language

Syntax

```
x = value;
x = if cond { 1 } else { 2 }
fn function(@i, x, y?) { return x }
fn function(x: string, y?: string) {}
x |> f() |> g() Same as g(f(x))
```

Operators

`n + m`	add, string concat.
`n - m`	subtract
`-m`	negate
`n * m`	multiply
`n / m`	divide
`n < m`	less than
`n <= m`	less or equal
`b == c`	equal to
`b != c`	not equal
`n >= m`	greater or equal
`n > m`	greater than
`b & c`	logical and
`b | c`	logical or
`!b`	logical not

Units

```
dist = 45cm - 2in, ang = 45deg - 2rad
fn f(intoMm: number(mm)): number(in)
```

Arrays

```
arr = [1, 2, 3], arr = [1..3], arr = [1..<4]
squared = map(arr, fn(@x) { return x*x })
sum = reduce(
  myArr,
  initial = 0,
  f = fn(@i, accum) {return i + accum}
)
```

Math

```
sin(10deg), cos(1rad), tan(x)
asin(10deg), acos(1rad), atan(x), atan2(y, x)
polar(angle = 9deg, length = 5)
rem(7, divisor = 4)
assert(x, isEqualTo = 3)
sqrt(4)
abs(-4)
round(4.1), floor(4.1), ceil(4.1)
min([1, 2, 3]), max([1, 2, 3])
pow(5, exp = 2), log(100, base = 5)
log10(100), log2(128), ln(100)
```

Modules

```
import wheelDepth, wheelDiameter from "car.kcl"
export wheelDepth = 2cm
```

Other

```
clone(solid)
```

Profiles

```
plane = startSketchOn(XZ)
plane = startSketchOn(solid, face = taggedFace)
profile = startProfile(plane, at = [0, 0])
```

Sketching

```
line(profile, end = [x, y])
line(profile, endAbsolute = [x, y])
xLine(profile, length = 3) // Or yLine
xLine(profile, endAbsolute = 3)
tangentialArc(profile, end = [x, y])
close(profile)
```

Pre-fabricated

```
circle(profile, center = [0, 0], diameter = 4)
```

Transforms

```
translate(profile, x = 1.3, z = 2.2)
scale(profile, y = 0.5, z = 2)
rotate(profile, roll = 5deg, pitch = 5deg, yaw = 5deg)
rotate(profile, axis = Z, angle = 45deg)
```

Booleans

```
subtract2d(profile, tool = holeShapeSketch)
```

Patterns

```
patternLinear2d(profile, instances, distance = 10, axis = X)
patternCircular2d(profile, instances, center = [0, 0])
patternTransform2d(profile, instances, transform = someFn)
mirror2d(unclosedPath, axis = Y)
```

Selections

Tagging

```
line(end = [3, 4], tag = $edge)
chamfer(length = 1, tags = [a], tag = $newFace)
extrude(length = 1, tagEnd = $newFace)
   ⮡ creates implicit START and END face tags
```

Edge selecting

```
getPreviousAdjacentEdge(tag)
getNextAdjacentEdge(tag)
getOppositeEdge(tag)
startSketchOn(solid, face = tag)
segLen(tag)
segStart(tag)
segEnd(tag)
segAng(tag)
getCommonEdge(faces = [tag, tag2])
```

Profiles to Solids

```
solid = extrude(profile, length = 4, method = MERGE | NEW)
solid = ...
extrude(profile, to = endExtrudeAtThis)
sweep(profile, path = path)
loft([profile1, profile2])
```

Solids

Transforms

```
translate(solid, x = 1.3, z = 2.2)
scale(solid, y = 0.5, z = 2)
rotate(solid, roll = 5deg, pitch = 5deg, yaw = 5deg)
rotate(solid, axis = Z, angle = 45deg)
```

Booleans

```
union([solid, solid2])
subtract(cube, tools = [hole])
intersect([solidA, solidB])
```

Patterns

```
patternLinear3d(instances = 4, distance = 10, axis = X)
patternCircular3d(instances, center = [0, 0, 0])
patternTransform(instances, transform = someFn)
```

Modifiers

```
shell(cube, faces = [taggedFace], thickness = 1)
fillet(solid, radius = 5, tags = [taggedEdge])
chamfer(solid, radius = 5, tags = [taggedEdge])
appearance(solid, color = "#00ff00")
```

Example

```
xy = startSketchOn(XY)
  p = startProfile(xy, at = [0, 0])
    |> xLine(length = 1.0mm)
    |> yLine(length = 1.0mm)
    |> line(endAbsolute = [
      profileStartX(%),
      profileStartY(%)
    ])
    |> close()
  s = extrude(p, length=5cm)
```

Links: KCL book | Zoo Design Studio | Language reference | Sample models | GitHub | Treesitter | LSP

Edit me on GitHub!

Full credit to the OpenSCAD cheat sheet for inspiration and design