KCL: KittyCAD Language

Syntax

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

Operators

`n + m`	Addition, string concatenation
`n - m`	Subtraction
`-m`	Negative
`n * m`	Multiplication
`n / m`	Division
`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`	Logical Not

Arrays

```
myArr = [1..3]
myArr = [1..<4]
myArr = [1, 2, 3]
squared = map(myArr, 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)
```

2D

```
startSketchOn(XZ)
startSketchOn(mySolid, face = taggedFace)
startProfile(mySketch, at = [0, 0])
```

Add a line to a sketch:

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

Other:

```
circle(mySketch, center = [0, 0], diameter = 4)
subtract2d(mySketch, tool = holeShapeSketch)
mirror2d(unclosedPath, axis = Y)
```

2D to 3D

```
extrude(mySketch, length = 4, method = MERGE | NEW)
extrude(mySketch, to = endExtrudeAtThis)
sweep(mySketch, path = myPath)
loft([sketch1, sketch2])
```

Transform 3Ds

```
appearance(mySolid, color = "#00ff00")
translate(mySolid, x = 1.3, z = 2.2)
scale(mySolid, y = 0.5, z = 2)
rotate(mySolid, roll = 5deg, pitch = 5deg, yaw = 5deg)
rotate(mySolid, axis = Z, angle = 45deg)
clone(mySolid)
fillet(mySolid, radius = 5, tags = [taggedEdge])
chamfer(mySolid, radius = 5, tags = [taggedEdge])
union([mySolid, mySolid2])
intersect([solidA, solidB])
subtract(cube, tools = [myHole])
shell(cube, faces = [taggedFace], thickness = 1)
```

Edges

Tag an edge (becomes a tagged face if extruded)

Works on all lines, arcs, etc

```
line(end = [3, 4], tag = $myEdge)
```

Tag a face

```
chamfer(length = 1, tags = [a], tag = $newFace)
extrude(length = 1, tagEnd = $newFace)
```

Use a tagged edge:

```
getPreviousAdjacentEdge(myEdge)
getNextAdjacentEdge(myEdge)
getOppositeEdge(myEdge)
startSketchOn(mySolid, face = myEdge)
segLen(myEdge)
segStart(myEdge)
segEnd(myEdge)
segAng(myEdge)
getCommonEdge(faces = [myEdge, myEdge2])
```

(all extrusions have standard tags START and END)

Patterns

```
patternLinear3d(instances = 4, distance = 10, axis = X)
patternCircular3d(instances = 4, center = [0, 0, 0])
patternLinear2d(instances = 4, distance = 10, axis = X)
patternCircular2d(instances = 4, center = [0, 0])
patternTransform(instances = n, transform = someFn)
patternTransform2d(instances = n, transform = someFn)
```

Units

```
dist = 45cm - 2in
ang = 45deg - 2rad
```

Converts x into mm

```
fn f(x: number(mm)): number(in)
```

Modules

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

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