SKILL Cheat Sheet

Note: Output of command is shown with => Example
linecount = "abc" , output will be "abc"

Variables

Variables are used to store values.
No need to declare variables in SKILL before using
Assign values to variables using the equals sign (=)
Variables are untyped, which means that the same variable name can store any data

```
linecount = 4 => 4
linecount="abc" > "abc"
```

The ***type*** function returns the data type of the variable's current value

```
type(linecount)=>string
```

Accessing a value of variable by just using name of variable or printf function

```
Linecount => "abc"
printf(linecount) => "abc" t
println(linecount)=> "abc"
```

DataTypes

SKILL supports following data types like any programming language

Integer, Floating point, Strings, Boolean (nil, t)

```
bike = "Davison"
count = 1
round = .004
isTrue = nil
```

Boolean values are represented as ***nil*** and ***t***

```
found = t
notFound = nil
```

Arrays

An array represents aggregate data objects in SKILL. Unlike simple data types, you must explicitly create arrays before using them so the necessary storage can be allocated.

```
declare( week[7] ) => array[7]:9780700
week => array[7]:9780700
type( week ) => array
days = '(monday tuesday wednesday
thursday friday saturday sunday)
for( day 0 length(week)-1
week[day] = nth(day days))
```

HashMap (association table)

An association table is a generalized array, a collection of key/value pairs

```
myTable = makeTable("atable1" 0) =>
table:atable1
tablep(myTable) => t
myTable[1] = "blue" => "blue"
myTable["two"] = '(r e d) => (r e d)
myTable["three"] = 'green => green
length(myTable) => 3
```

List

A SKILL list is an ordered collection of SKILL data objects

The elements of a list can be of any data type, including variables and other lists.

A list can contain any number of objects (or be empty)

```
numbers = '( 2 3 )
newList = list( a b 3 ) =>("a" "b" 3);if
a="a" and b = "b"
car( numbers ) => 2
result = cons( 1 numbers ) => ( 1 2 3 )
nth( 1 numbers ) => 2

length( numbers ) => 3
```

Comments

SKILL permits two different styles of comments

the semicolon (;) indicates that the rest of the input line is a comment.

block-oriented, where comments are delimited by /* and */

```
x = 1; comment following a statement
; comment line 1
/* This is a block of (C style) comments
comment line 2
comment line 3 etc.
*/
```

Operators

Like any other programming language SKILL supports Relational Operators, Logical Operators, control structures and iteration functions

Relational Operators <, <=, >, >=, ==, !=

Logical Operators !, &&, |

Branching: if ,when, unless,.case

Iteration: for, foreach

Example of Operators

Case, foreach

```
rectCount = lineCount = polygonCount = 0
shapeTypeList = '( "rect" "polygon" "rect"
"line" )
foreach( shapeType shapeTypeList
    case( shapeType
      ( "rect" ++rectCount )
      ( "line" ++lineCount )
      ( "polygon" ++polygonCount )
      ( t ++miscCount )
    ) ;case
 ) ; foreach
```

Conditional : if, else if

```
if( shapeType == "rect" then
  println( "Shape is a rectangle" )
  ++rectCount
else
  println( "Shape is not a rectangle" )
)
```

File Read and Write

SKILL has functions to read and write files

```
inPort = infile( "readfile.txt" )
when( inPort
  while( gets( nextLine inPort )
      println( nextLine ) )
 close( inPort ) )
 ;Write
 myPort = outfile( "/tmp/myFile" )
 for( i 1 3
    println( list( "Number:" i) myPort ) )
 close( myPort )
```

Functions

Skill supports functions or procedures for modularity
Its return value is the symbol with the name of the function

```
procedure( ComputeBBoxHeight( )
  bBox = list( 100:150 250:400)
  ll = car( bBox )
  ur = cadr( bBox )
  lly = yCoord( ll )
  ury = yCoord( ur )
  ury - lly
) ; procedure
bBoxHeight = ComputeBBoxHeight()
```