FAB323093-L

Fabrication Scripting - 201 (Advanced) Lab

Darren Young
Hermanson Company – Seattle Washington / Portland, Oregon

Learning Objectives

- Learn about creating reusable functions
- Learn about processing a directory folder (like all the ITMs in your library)
- Learn how to read and write to files
- Learn about fixing content and drawing issues

Description

Managing your MEP fabrication content is critical for successful prefabrication. Fabrication COD Scripting can significantly help with that management. During this class, we'll teach advanced fabrication scripting for Fabrication CADmep software, Fabrication ESTmep software, and Fabrication CAMduct software using *.COD scripts. Prior knowledge of COD scripting is highly recommended, but not required.

Speaker(s)

Based in Washington state, Darren Young has been a veteran Autodesk University speaker for well over a decade. His unique ability to leverage multiple everyday technologies in interesting ways to solve complicated and laborious tasks has been valued by users around the world. Down to earth and approachable, he's always willing to help his peers anytime of the year even outside of Autodesk University. Darren's background includes a wide variety of disciplines such as Construction, Engineering, Manufacturing, LEAN, Information Technology, Computer Programming, Author and Technical Editor. His lectures and labs are not just a training opportunity for others but a venue which connects him personally with users helping him learn as well.

Assistants(s)

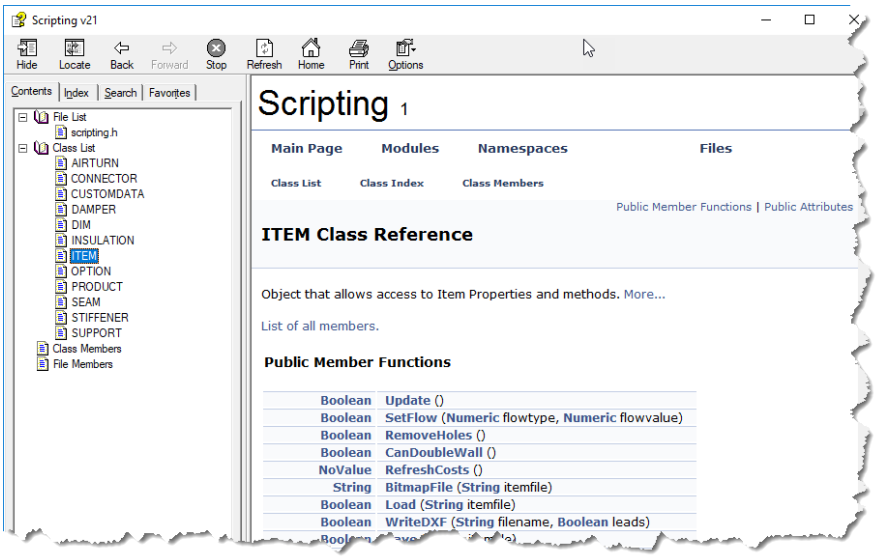
Ralph Schoch – Revit, Technology and Internal Support Manager – Victaulic / Lehighton, PA
David Ronson – Product Manager – eVolve Mechanical – Austin, TX
Troy Vansanten – Fabrication Database Manager – Hermanson Company – Seattle, WA

Resources

Help file installed locally…

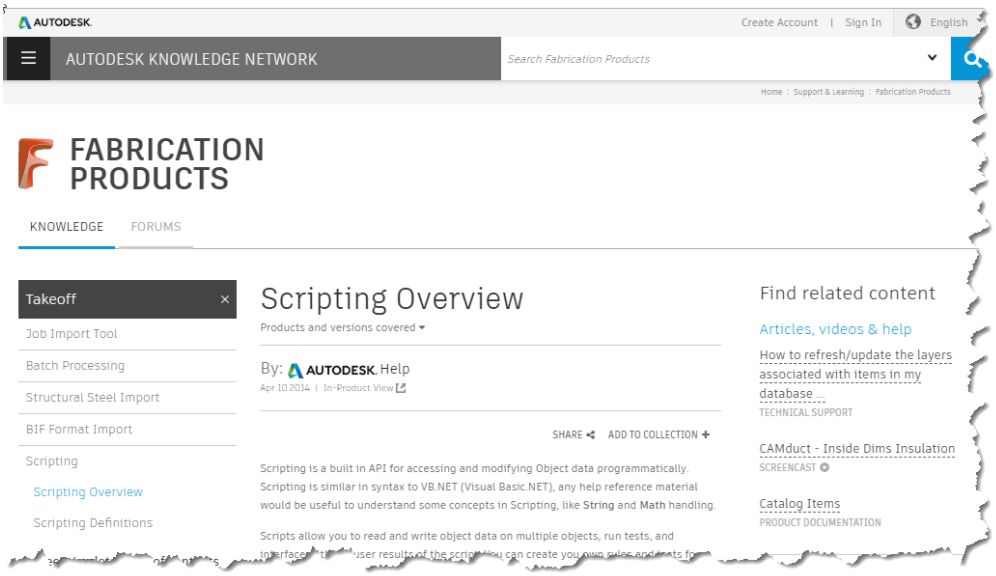
C:\Program Files\Autodesk\Fabrication *<year><product>*\en-US\Scripting.chm**

<year>	*2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020*
<product>	*CADmep, ESTmep, CAMduct*



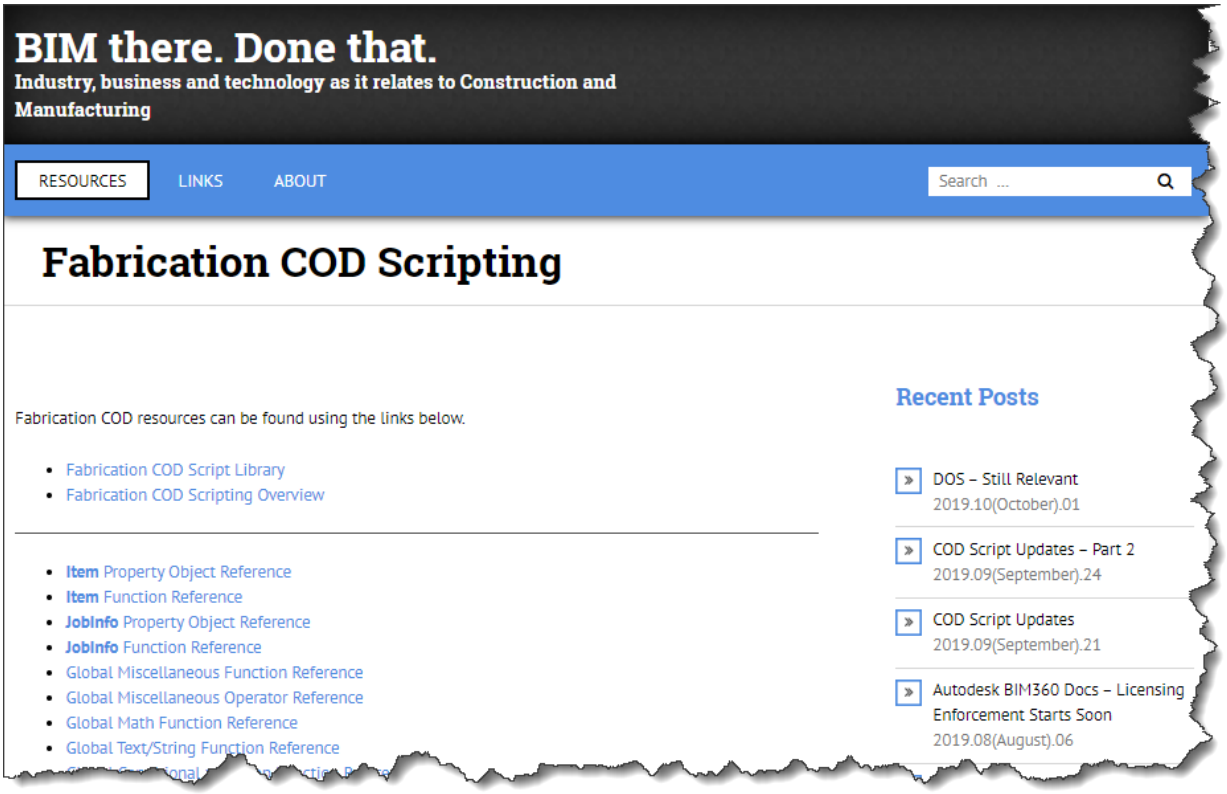
Online Tutorial & Documentation…

https://tinyurl.com/2020-COD-Scripting



My site…

http://www.darrenjyoung.com/resources/autodesk-fabrication/fabrication-cod-scripting/



Scripting in a NutShell….

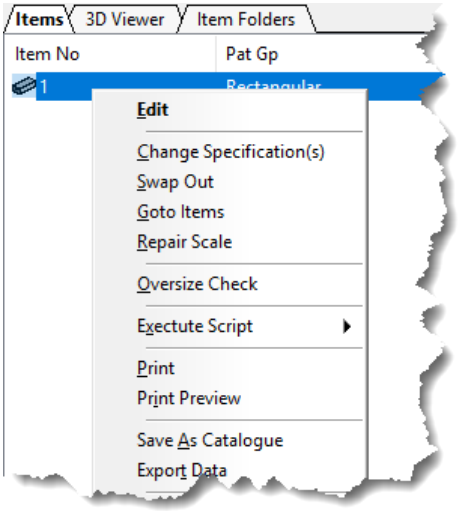
- Written in ASCII Text (Windows Notepad, etc.)
- Can use Script Editor in Fabrication but can be buggy when running scripts.
- File extension for scripts = *.COD
- Text/Strings surrounded by double quotes
- Double quotes from Word/Email (“ ”) are not the same as Notepad (" ")
- Use ASCII Character codes to mimic Enter, Tab, Quotes within text strings
- Variables must be declared with **DIM**
- Variable names must be begin w/alpha character, no spaces odd characters
- Functions/Properties/Variables are NOT CaSe SeNsItIvE
- Scripts Read/Write properties, they don’t automate drawing.
- ITEM is default object for items in drawing
- JOB is default object for job properties
- Unless using advanced processing tasks or functions (not covered in this 101 session), scripts are run on each selected object. E.g. 2 selected objects runs the script twice, once for each object.

Calling Scripts in CADmep…

- Type “EXECUTESCIPT” at the command line
- Use (executescript “script name”) function from AutoLISP

Calling Scripts in ESTmep/CAMduct…

- Use Script Editor to Load & Run – **File -> Open Script**
- Highlight Items from Takeoff – **Right-Click -> Execute Script**



Advanced Scripting in a NutShell….

- Scripts can **INCLUDE** other COD script files
- Scripts can contain reusable **FUNCTION**s
- Script Functions can take additional arguments passed to them
- Script Functions can return values to the function that called them
- Script Functions can be called, passed values and/or return values even when Included from an external script file
- COD scripts can build **ARRAY** objects for easier processing of large datasets
- Use ASCII Carriage Return and Tab codes to format data
- Scripts can read and write files
- Both Binary and Text files can be read/written
- Scripts can scan for Files and Folders to be processed
- File and Folder scanning can match against wildcard/mask patterns
- Scripts by default execute once for each item selected
- Scripts can use a **TASK SELECTION** to execute once regardless of number of selected items *(you are then responsible to process them in a loop in your own code)*
- Processing of large datasets can produce progress bars using a **TASK**
- All scripts using **TASK**s must be declared at the beginning of the file with a **REQUIRES** statement.
- Use the **EXEC** function to call other programs or DOS batch files to do more advanced, non-fabrication related functions.
- COD Scripts can be called from AutoLISP using the **(executescript)** function to further enhance functionality.
- Use Custom Data or Note fields to pass data to Lisp from the a COD script.
- Use **List Setup** to access fabrication data via Lisp (entget) DXF codes.

ITEM Object Properties	
Read	**item.airturns**
Read/Write	**item.airturn[**<*index*>**].value**
Read/Write	**item.airturn[**<*index*>**].locked**
Read/Write	**item.alias**
Read/Write	**item.alternate**
Read/Write	**item.bitmap**
Read/Write	**item.boughtout**
Read/Write	***** item.box**
Read/Write	**item.buttonalias**
Read/Write	**item.buttoncode**
Read/Write	**item.cadblock**
Read	***** item.catalogue**
Read/Write	**item.cid**
Read/Write	**item.comment**
Read	**item.connectors**
Read/Write	**item.connector[**<*index*>**].alt**
Read	**item.connector[**<*index*>**].group**
Read/Write	**item.connector[**<*index*>**].locked**
Read	**item.connector[**<*index*>**].type**
Read/Write	**item.connector[**<*index*>**].value**
Read/Write	**item.costbylength**
Read/Write	**item.costtype**
Read	**item.customdata[**<*name*>/<*index*>**].id**
Read/Write	**item.customdata[**<*name*>/<*index*>**].value**
Read/Write	**item.cuttype**
Read	**item.dampers**
Read/Write	**item.damper[**<*index*>**].locked**
Read/Write	**item.damper[**<*index*>**].value**
Read/Write	**item.databaseid**
Read	**item.dblock**
Read	**item.dblock.history**
Read	**item.dblock.history.changed**
Read	**item.dblock.history.***history*
Read	**item.dblock.history.info**
Read	**item.dblock.history.owner**
Read	**item.dblock.history.version**
Read	**item.dblock.owner**
Read	**item.dblock.version**
Read	**item.decoiler.beading**
Read	**item.decoiler.coilwidth**
Read	**item.decoiler.smalllength**
Read	**item.decoiler.stdlength**
Read	**item.decoiler.stdqty**
Read/Write	**item.description**
Read	**item.dims**

Read	**item.dim[**<*name*>/<*index*>**].annotation**
Read/Write	**item.dim[**<*name*>/<*index*>**].locked**
Read	**item.dim[**<*name*>/<*index*>**].name**
Read	**item.dim[**<*name*>/<*index*>**].numvalue**
Read	**item.dim[**<*name*>/<*index*>**].status**
Read/Write	**item.dim[**<*name*>/<*index*>**].value**
Read/Write	**item.dimside**
Read/Write	**item.dimsidelock**
Read/Write	**item.doublewall**
Read/Write	**item.drawing**
Read/Write	**item.dwlock**
Read/Write	***** item.etag**
Read/Write	**item.extraetime**
Read/Write	**item.extraetimerate**
Read/Write	**item.extraetimeunits**
Read/Write	**item.extraftime**
Read/Write	**item.extraftimerate**
Read/Write	**item.extraftimeunits**
Read/Write	**item.fabtable**
Read/Write	**item.fabtablelock**
Read/Write	**item.facing**
Read/Write	***** item.facinglock**
Read/Write	**item.filename**
Read/Write	**item.fixrelative**
Read/Write	**item.gauge**
Read/Write	**item.gaugelock**
Read	***** item.guid**
Read	***** item.guid64**
Read	***** item.handle**
Read	**item.hasproduct**
Read/Write	**item.insspec**
Read/Write	**item.installtable**
Read/Write	**item.installtablelock**
Read/Write	**item.insulation**
Read/Write	**item.insulation.facing**
Read/Write	**item.insulation.gauge**
Read/Write	**item.insulation.material**
Read/Write	**item.insulation.materiallock**
Read/Write	**item.insulation.status**
Read/Write	**item.insulation.statuslock**
Read/Write	**item.ispeclock**
Read	**item.library**
Read/Write	***** item.lifespan**
Read	**item.links**
Read/Write	**item.link[**<*name*>/<*index*>**].name**
Read/Write	**item.link[**<*name*>/<*index*>**].param**

Read/Write	**item.link[**<*name*>/<*index*>**].target**
Read	**item.manyoldstatus**
Read	**item.matabrv**
Read/Write	**item.material**
Read/Write	**item.nestpriority**
Read/Write	**item.notes**
Read/Write	**item.number**
Read	**item.oldstatus[**<*name*>/<*index*>**]**
Read	**item.oldstatus[**<*name*>/<*index*>**].datetime**
Read	**item.oldstatus[**<*name*>/<*index*>**].id**
Read/Write	**item.oldstatus[**<*name*>/<*index*>**].userid**
Read	**item.oldstatus[**<*name*>/<*index*>**].value**
Read/Write	***** item.operatingcost**
Read	**item.options**
Read/Write	**item.option[**<*name*>/<*index*>**].locked**
Read	**item.option[**<*name*>/<*index*>**].name**
Read	**item.option[**<*name*>/<*index*>**].status**
Read/Write	**item.option[**<*name*>/<*index*>**].value**
Read/Write	**item.order**
Read/Write	**item.pallet**
Read	***** item.partscut**
Read	***** item.partcut{*<index>*]**
Read/Write	**item.path**
Read	**### item.patno**
Read/Write	**item.pricelist**
Read/Write	**item.pricetablelock**
Read	**item.product.entries**
Read/Write	**item.product.entry[**<*index*>**].alias**
Read/Write	**item.product.entry[**<*index*>**].area**
Read/Write	**item.product.entry[**<*index*>**].boughtout**
Read/Write	**item.product.entry[**<*index*>**].cadblock**
Read/Write	**item.product.entry[**<*index*>**].customdata[**<*name*>/<*index*>**]**
Read/Write	**item.product.entry[**<*index*>**].databaseid**
Read/Write	**item.product.entry[**<*index*>**].dim[**<*index*>**]**
Read/Write	***** item.product.entry[*<index>*].flowmax**
Read/Write	***** item.product.entry[*<index>*].flowmin**
Read/Write	**item.product.entry[**<*index*>**].model**
Read/Write	**item.product.entry[**<*index*>**].name**
Read/Write	**item.product.entry[**<*index*>**].option[**<*index*>**]**
Read/Write	**item.product.entry[**<*index*>**].order**
Read/Write	***** item.product.entry[*<index>*].skey**
Read/Write	**item.product.entry[**<*index*>**].weight**
Read	**item.product.hasalias**
Read	**item.product.hasarea**
Read	**item.product.hasboughtout**
Read	**item.product.hascadblock**

Read	**item.product.hascustomdata**
Read	**item.product.hascustomdatas**
Read	**item.product.hasdatabaseid**
Read	**item.product.hasdims**
Read	***** item.product.hasflow**
Read	**item.product.hasoptions**
Read	**item.product.hasorder**
Read	**item.product.hasrevision**
Read	***** item.product.hasskey**
Read	**item.product.hasweight**
Read/Write	**item.qty**
Read/Write	**item.scale**
Read/Write	***** item.sealant.locked**
Read/Write	***** item.sealant.value**
Read	**item.seams**
Read/Write	**item.seam[***<index>***].alt**
Read/Write	**item.seam[***<index>***].locked**
Read/Write	**item.seam[***<index>***].value**
Read/Write	**item.section**
Read/Write	**item.service**
Read/Write	**item.servicetype**
Read/Write	**item.skinconnector[***<index>***].alt**
Read	**item.skinconnector[***<index>***].group**
Read/Write	**item.skinconnector[***<index>***].locked**
Read	**item.skinconnector[***<index>***].type**
Read/Write	**item.skinconnector[***<index>***].value**
Read	**item.skindecoiler.beading**
Read	**item.skindecoiler.coilwidth**
Read	**item.skindecoiler.smalllength**
Read	**item.skindecoiler.stdlength**
Read	**item.skindecoiler.stdqty**
Read	**item.skindecoiler.stdlength**
Read/Write	**item.skingauge**
Read/Write	**item.skinmaterial**
Read/Write	**item.skinmateriallock**
Read/Write	**item.skinseam[***<index>***].alt**
Read/Write	**item.skinseam[***<index>***].locked**
Read/Write	**item.skinseam[***<index>***].value**
Read/Write	**item.skinside**
Read/Write	**item.specification**
Read/Write	**item.speclock**
Read	**item.splitters**
Read/Write	**item.splitter[***<index>***].value**
Read/Write	**item.splitter[***<index>***].locked**
Read/Write	**item.spool**
Read/Write	**item.spoolcolour**

Read/Write	**item.status**
Read	**item.stiffeners**
Read/Write	**item.stiffener[***<index>***].locked**
Read/Write	**item.stiffener[***<index>***].qty**
Read/Write	**item.stiffener[***<index>***].spacing**
Read/Write	**item.stiffener[***<index>***].value**
Write	**item.structuretype**
Read	**item.subtems**
<varies>	**item.subtem[***<index>***].***<Same Properties as Item Objects>*
Read/Write	**item.support.locked**
Read/Write	**item.support.qty**
Read/Write	**item.support.spacing**
Read/Write	**item.support.value**
Read	**item.type**
Read/Write	**item.weight**
Read/Write	**item.weightlock**
Read/Write	**item.wiregauge**
Read/Write	**item.zone**

*****(undocumented)**

###(new – 2019.1.0 and later only)

ITEM Functions		
addcustomdata	**item.addcustomdata("Room number")**	**n/a**
	item.addcustomdata(52)	
bitmapfile	**item.bitmapfile("./Charlotte/pipe.itm")**	**image file**
candoublewall	**item.candoublewall()**	**True/False**
canrotary	***** item.canrotary()**	**True/False**
dblock.can	**item.dblock.can(lock_user)**	**True/False**
	item.dblock.can(lock_owner)	
dblock.setowner	**item.dblock.setowner("*<owner>*","*<reason>*")**	**True/False**
dblock.setversion	**item.dblock.setversion("*<owner>*","*<reason>*")**	**True/False**
endlocation	**item.endlocation(*<index>*)**	**"x y z"**
	item.endlocation(*<index>*, "xyz")	**"x y z"**
	item.endlocation(*<index>*, "x")	**X**
	item.endlocation(*<index>*, "y")	**Y**
	item.endlocation(*<index>*, "z")	**Z**
	item.endlocation(*<index>*, "btm")	**-(OD/2)**
	item.endlocation(*<index>*, "top")	**+(OD/2)**
level	**item.level("Floor")**	**Level**
	item.level("Soffit")	**Level**
load	**item.load("./Charlotte/pipe.itm")**	**True/False**
refreshcosts	**item.refreshcosts()**	**n/a**
removeholes	**item.removeholes()**	**n/a**
save	**item.save("./Charlotte/pipe.itm")**	**True/False**
setflow	**item.setflow(0,)** = Not Set	**True/False**
	item.setflow(1,250) = Supply	
	item.setflow(2,250) = Return	
	item.setflow(3,) = None	
update	**item.update()**	**True/False**
writedxf	**item.writedxf("./mydxfs/elbow", True)**	**True/False**
	item.writedxf("./mydxfs/elbow", False)	

*****(undocumented)**

JOB Object Properties	
Read/Write	**job.colour**
Read	**job.customdata[***<name>/<index>***].id**
Read/Write	**job.customdata[***<name>/<index>***].value**
Read	**job.date**
Read/Write	**job.field1**
Read/Write	**job.field2**
Read	**job.items**
Read/Write	**job.item[***<index>***].…** *(see ITEM above)*
Read	**job.name**
Read/Write	**job.notes**
Read	**job.project**
Read/Write	**job.reference**
Read	**job.statuses**
Read	**job.status.[***<name>/<index>***].active**
Read	**job.status.[***<name>/<index>***].lastactivated**
Read	**job.status.[***<name>/<index>***].name**

JOB Object Functions		
setstatus	**job.setstatus(***<index>***, True)**	**True/False**

Global Misc Functions		
debug	**debug("My Alert Box")**	
dim	**dim** *<my variable>*	
	dim *<my array>* **as array**	
	dim *<my array>* **as array(3)**	
	dim *<my object>* **as itemstruct**	
exec	**exec("***<prog>***",***<flags>***,"***<params>***","***<work dir>***")** *<prog>* = Program to execute *<flags>* = exec_default = exec_wait = exec_show_normal = exec_show_max = exec_show_min *<params>* = Parameters passed to program *<work dir>* = Working directory for program	
	exec("Notepad.exe",exec_wait+exec_show_min,,"")	
include	**include "C:\\My Script.cod"**	
inputbox	**inputbox("Title", "Prompt", "Default")**	*<string>*
object	***object*** *<my file>* = new file("*<file>*",*<mode>*)	
	object *<my file>* *as* file("*<file>*",*<mode>*)	
query	**query("Chose Yes or No")**	**True/False**
rem	**Rem This is a Note**	
requires	**requires** *task*	
	requires *task.selection*	

Global Misc Operators		
+	**debug 1 + 1**	**2**
	debug("This is" + " " + "a test")	
-	**debug 5 - 3**	**2**
*	**debug 2 * 2**	**4**
/	**debug 4 / 2**	**2**
and	**debug 1 = 1 and 2 = 2**	
not	**debug Not True**	
or	**debug Test = 1 or Test = 2**	
( )	**(2 + 2) * 5**	**20**

Global Math Functions		
acos	**acos(3, 4)**	**41.4096221**
asin	**asin(3, 4)**	**48.5903779**
atan	**atan(3, 4)**	**36.8698976**
cos	**cos(45)**	**0.7071068**
exp	**exp(2)**	**10**
log	**log(100)**	**2**
number	**number("5")**	**5**
	number("5.0")	**5**
	number("5.5")	**5.5**
pow	**pow(5, 2)**	**25**
	pow(2, 3)	**8**
round	**round(5.49)**	**5**
	round(5.50)	**6**
rounddown	**rounddown(5.9)**	**5**
roundup	**roundup(5.1)**	**6**
sign	**sign(-3.4)**	**-1**
	sign(3.4)	**1**
	sign(0.000001)	**0**
sin	**sin(45)**	**0.7071068**
sqrt	**sqrt(4)**	**2**
sqr	**sqr(2)**	**4**
tan	**tan(45)**	**1**

Global Variables		
pi	3.1415	
eo_never	2	Unknown purpose / use
False	0	
True	1	
null	0	Number meaning no value. Same as zero
void		Special variable used when wanting to specify a default parameter
time_secs	1	Second Time Units
time_mins	2	Minute Time Units
time_hours	3	Hours Time Units
for_output	1	File access mode for writing files
for_input	2	File access mode for reading files
istext	4	File access mode for text files
isunicode	12	File access mode for Unicode text files
file_end	-1	Used by ***file.position*** to indicate file end
file_start	0	Used by ***file.position*** to indicate file start
lock_user	1	Used by **item.dblock.can()** to check access
Lock_owner	2	
exec_default	0	Used by **EXEC** function for default execution
exec_wait	1	Used by **EXEC** function to wait for program to complete before continuing
exec_show_normal	0	Used by **EXEC** function for normal window
exec_show_min	16	Used by **EXEC** function for minimized window
exec_show_max	32	Used by **EXEC** function for maximized window
mappath_backup	*<path>*	Path to **BACKUP** folder
mappath_blocks	*<path>*	Path to **BLOCKS** folder
mappath_cnc	*<path>*	Path to **CNC** folder
mappath_database	*<path>*	Path to **DATABASE** folder
mappath_dxf	*<path>*	Path to **DXF** folder
mappath_filter	*<path>*	Path to **FILTER** folder
mappath_home	*<path>*	Path to **MAP.ini** folder
mappath_images	*<path>*	Path to **IMAGES** folder
mappath_install	*<path>*	Path to **MAP.ini** folder
mappath_items	*<path>*	Path to **ITEMS** folder

mappath_parts	*<path>*	Path to **PARTS** folder
mappath_project	*<path>*	Path to **PROJECT** folder
mappath_renmants	*<path>*	Path to **REMNANTS** folder
mappath_reports	*<path>*	Path to **REPORTS** folder
mappath_scripts	*<path>*	Path to **SCRIPTS** folder

Array Properties		
array	dim as **array**	*Initialize array*
array<#>	dim as **array(3)**	*Initialize 3 item array*
[<*index*>]	<*array*>**[3]**	*read 3rd value*
	<*array*>**[3] = "D"**	*set 3rd value*
add	<*array*>**.add("A", "B", "D", "E")**	*adds/append to array*
count	<*array*>**.count**	*# of items in array*
del	<*array*>**.del(3)**	*delete 3rd array item*
insert	<*array*>**.insert("C", 3)**	*insert item at position*

Array Functions		
array	dim as **array**	*Initialize array*
array<#>	dim as **array(3)**	*Initialize 3 item array*
[<*index*>]	<*array*>**[3]**	*read 3rd value*
	<*array*>**[3] = "D"**	*set 3rd value*
add	<*array*>**.add("A", "B", "D", "E")**	*adds/append to array*
count	<*array*>**.count**	*# of items in array*
del	<*array*>**.del(3)**	*delete 3rd array item*
insert	<*array*>**.insert("C", 3)**	*insert item at position*

File Object Properties		
Read	*<file>*.**EOF**	**True/False**
Read	*<file>*.**exists**	**True/False**
Read/Write	*<file>*.**filename**	
Read	*<file>*.**isopen**	**True/False**
Read	*<file>*.**isunicode**	**True/False**
Read	*<file>*.**length**	*Length in BYTEs*
Read/Write	*<file>*.**mode**	*File Open Mode Flags:* *foroutput (1)* *forinput (2)* *istext (4)* *unicodetext (12)*
Read/Write	*<file>*.**position**	*Position in BYTEs*

File Object Functions		
close	*<file>*.**close()**	**True/False**
delete	*<file>*.**delete(***<boolean>***)**	**True/False**
	<file>.**delete()**	
	<file>.**delete(True)**	
file	*object myfile =* **new file("***<file>***",***<mode>***)**	
	object myfile as **file("***<file>***",***<mode>***)**	
open	*<file>*.**open("***<file>***",***<mode>***)**	**True/False**
	<file>.**open("c:/myfile.txt",forinput)**	
readbyte	*<file>*.**readbyte()**	*<numeric>*
readchar	*<file>*.**readchar()**	*<string>*
readint	*<file>*.**readint()**	*<numeric>*
readline	*<file>*.**readline()**	*<string>*
readreal	*<file>*.**readreal()**	*<numeric>*
readstring	*<file>*.**readstring()**	*<string>*
readword	*<file>*.**readword()**	*<numeric>*
rename	*<file>*.**rename("***<string>***")**	**True/False**
	<file>.**rename("C:\MyRenamedFile.txt")**	
writebyte	*<file>*.**writebyte(***<number>***)**	**True/False**
writechar	*<file>*.**writechar("***<string>***")**	**True/False**
writeint	*<file>*.**writeint(***<number>***)**	**True/False**
writeint	*<file>*.**writeint(***<number>***)**	**True/False**
writeline	*<file>*.**writeline("***<string>***",***<boolean>***)**	*<integer>*
	<file>.**writeline("ABCdef")**	
	<file>.**writeline("ABCdef", True)**	
writereal	*<file>*.**writereal(***<number>***)**	**True/False**
writestring	*<file>*.**writereal("***<string>***")**	
writeword	*<file>*.**writeword(***<number>***)**	**True/False**

File Locator Object Properties		
Read	*<fileloc>*.**file[***<index>***]**	*<filename>*
Read	*<fileloc>*.**filecount**	*<integer>*
Read	*<fileloc>*.**folder[***<index>***]**	*<foldername>*
Read	*<fileloc>*.**foldercount**	*<integer>*
Read	*<fileloc>*.**foldercount**	*<integer>*
Read/Write	*<fileloc>*.**path**	
Read	***<fileloc>*.**scan("***<path>***","***<mask>***",<bool>,<bool>)**	*<object>*
	***<fileloc>*.**scan("C:\\", "*", True, True)**	
	***<fileloc>*.**scan("C:\\", "*.itm", True, False)**	
	***<fileloc>*.**scan("C:\\", "Pipe*", False, True)**	

***(undocumented)**

File Locator Object Functions	
filelocator	*dim myfl as* **filelocator**
	dim myfl as **filelocator("***<path>***","***<mask>***",<bool>,<bool>)**
	dim myfl as **filelocator("C:\\", "*", True, True)**
	dim myfl as **filelocator("C:\\", "*.itm", True, False)**
	dim myfl as **filelocator("C:\\", "Pipe*", False, True)**

Global Text/String Functions		
asc	**asc("A")**	**65**
ascii	**ascii(65)**	**"A"**
chr	**chr("Autodesk", 5)**	**"d"**
getfileext	**getfileext("c:\(temp)\test.txt")**	**".txt"**
	getfileext("c:/(temp)/test.txt")	**".txt"**
getfilename	**getfilename("c:\\(temp)\test.txt")**	**"test.txt"**
	getfilename("c:/(temp)/test.txt")	**"test.txt"**
getfilepath	**getfilepath("c:\(temp)\test.txt")**	**"c:/(temp)/ "**
	getfilepath("c:/(temp)/test.txt")	**"c:/(temp)/ "**
instr	**instr(1, "Autodesk", "D", False)**	**5**
	instr(1, "Autodesk", "D", True)	**0**
	instr(6, "Autodesk", "D", False)	**0**
left	**left("Autodesk", 4)**	**"Auto"**
len	**Len("Autodesk")**	**8**
lower	**lower("Autodesk")**	**"autodesk"**
ltrim	**ltrim(" Autodesk ")**	**"autodesk "**
mid	**mid("Autodesk", 5, 2)**	**"de"**
right	**right("Autodesk", 4)**	**"desk"**
rtrim	**rtrim(" Autodesk ")**	**" autodesk"**
substring	**substring("Autodesk", 3, 4)**	**"tode"**
trim	**trim(" Autodesk ")**	**"autodesk"**
upper	**upper("Autodesk")**	**"AUTODESK"**
wildcard	**wildcard("Autodesk", "*desk")**	**1**
	wildcard("Autodesk", "*chair")	**0**
	wildcard("Autodesk", "?ut?d*")	**1**

Task Properties		
Read	task.**aborted**	**True/False**
Read/Write	task.**message** = "My Task"	
Read/Write	task.**progress** = 1	
Read	task.**selection[***<index>***]**	*<item object>*
Read	task.**selection.count**	*<integer>*

Task Functions	
beginprogress	task.**beginprogress(***<integer>***)**
	task.**beginprogress(***5000***)**
endprogress	task.endprogress()

Function Definitions	
function	**function myfun()** Debug "Test Function" **end function**
function w/argument	**function myfun(arg)** debug arg **end function** **myfun("Test Function")**
function w/return	**function myfun()** **return** 2 * 2 **end function** dim myvar myvar = **myfun()**
function w/ret & arg	**function myfun(arg)** **return** arg * arg **end function** dim myvar myvar = myfun(2)

Conditional Test & Loop Functions	

Do - Loop	dim count = 10
	do
	  debug count
	  count = count - 1
	loop until count = 0

Do - While	dim count = 5
	do while count < 10
	  debug count
	  count = count + 1
	loop

For – Next	Dim mynum **for mynum=1 to 10** debug mynum **next mynum**
For – Next – Step	Dim mynum **for mynum=1 to 10 step 2** debug mynum **next mynum**

If	dim mynum = 1
	if mynum = 1 then
	  debug "Yes"
	endif

If - Else	dim mynum = 1
	if mynum = 1 then
	  debug "Yes"
	else
	  debug "No"
	endif

If – ElseIf - Else	dim mynum = 1
	if mynum = 1 then
	  debug "1"
	elseif mynum = 2 then
	  debug "2"
	else
	  debug "Other"
	endif

Select Case	dim mynum = 3
	select mynum
	  case 1
	  debug "Doing 1"
	  case 2, 3
	  debug "Doing 2 & 3"
	  case 4
	  debug "Doing 4"
	end select

Select Case Else	dim mynum = 3
	select mynum
	  case 1
	  debug "Doing 1"
	  case 2, 3
	  debug "Doing 2 & 3"
	  case else
	  debug "Doing something else"
	end select

While	dim loop = 1
	while loop <= 5
	  debug loop
	  loop = loop + 1
	endwhile