LISPing on Purpose – Parts 1 & 2

Craig P. Black

Instructor of CAD Management, CAD Drafting, and Mechanical Design courses at Fox Valley Technical College

Twitter: CraigPBlack Facebook: Craig Black Email: black@fvtc.edu

CM1755-L & CM1757-L
LISPing on Purpose – Parts 1 & 2

Craig P. Black

Fox Valley Technical College – Appleton, WI, USA

Teach technical drafting and Autodesk software related courses for the Mechanical Design Technology Associate of Applied Science Degree, CAD Management Certificate, and Mechanical CAD Drafting Technical Diploma programs

Previous industry experience in mechanical, architectural, civil, and electrical disciplines

Co-author, along with Terence Shumaker, David A. Madsen, David P. Madsen, Jeffrey Laurich, and J.C. Malitzke of AutoCAD And Its Applications – Advanced, published by Goodheart-Wilcox

Former member and committee chair of the Autodesk Training Center Executive Committee, a liaison between Autodesk and Authorized Training Centers, focusing on educational issues

Lab Assistants

Your resource for a successful session!

LISPing on Purpose: Learning Objectives (page 1)

At the end of this class, you will be able to:

- Explain common AutoLISP programming terms
- Identify the common data types encountered within Autodesk® AutoCAD® software and AutoLISP
- Use the Visual LISP Editor to write and format simple LISP routines
- Load, use, and make available for future use those programs that will automate workflow

(There will be a short quiz at the end of the session!)

Answers to “Quiz” at the end of class...

1. Extremely Relevant
2. Just Right
3. 10, 10, 10, 10, 10
4. 10
5. Yes, just right
6. Yes

Comments: Invite this guy back next year! Best Instructor Ever at AU!

(See? This stuff isn’t that hard!)

Concerns and Expectations

- Cell phones off or on vibrate
- Try to hold questions until the end
- Stretching break at end of class
- Comfortable working with AutoCAD 2010 or later

LISPing on Purpose – Part 1

Terminology (page 4)

- **List** – anything contained within a set of parenthesis
- **Atom** – the individual items within a list
- **Function** – AutoLISP's commands, typically the first element within a list
- **Argument** – the elements that follow, or are "passed" to the function (arguments will be of specific data types)
- **Expression** – a list that involves a function, and possibly, arguments
- **Data Types** – see the next 4 terms
  - **Integers** – whole numbers, no decimal point
  - **Reals** – floating point numbers, always has a decimal point
  - **Strings** – words, always contained in a set of quotation marks: "Like This"
  - **Lists** – since AutoLISP stores point coordinates, or X, Y, and Z values within a set of parenthesis, or more specifically as a point list (see the first term!) – a list can be considered a data type

Prep Work for Practicing (page 5)

- Toggle off Dynamic Input
- Expand Command Prompt Window

Basic Syntax (page 5)

(functionname argument1 argument2 ...)

Math Functions (pages 6 and 7)

(+ *num num* …)

(- *num num* …)

(* *num num* …)

(/ *num num* …)

Rules for numbers:

- **Decimal numbers MUST have a number in front of the decimal**
- **An integer will be returned if all arguments are integers**

Storing Data (page 6)

(setq sym value)

(setq A 1)	returns 1
(setq B 2.5)	returns 2.5
(setq C “AutoLISP”)	returns “AutoLISP”

Getting Specific Data (page 8)

(getint *str*)

(getreal *str*)

(getpoint *pt* *str*)

(getstring *flag* *str*)

Nesting Expressions (page 8)

```
(setq ROWS (getint “Enter number of rows: “))

(setq RAD (getreal “Enter circle radius: “))

(setq CPT (getpoint “Enter circle center point: “))

(setq FRSTNM (getstring “Enter your first name: “))

(setq FULLNM (getstring T “Enter your full name: “))
```

Using Stored Data – Method 1 (page 9)

Command: **(setq X 1.25)**
1.25
Command: **!X**
1.25
Command: **offset**
Current settings: Erase source=No Layer=Source OFFSETGAPTYPE=0
Specify offset distance or [Through/Erase/Layer] <1.0000>: **!X**
1.25
Select object to offset or [Exit/Undo] <Exit>: **…**

Using Stored Data – Method 2 (page 10)

(**command** arguments…)

Command: **(command “circle” CPT RAD)**

Putting It All Together

```
(setq PT1 (getpoint "\nEnter first point: "))

(setq PT2 (getpoint "\nEnter next point: "))

(setq PT3 (getpoint "\nEnter next point: "))

(setq PT4 (getpoint "\nEnter next point: "))

(command "line" PT1 PT2 PT3 PT4 "c")
```

LISPing on Purpose – Part 2

Defining a Function (page 11)

(**defun** sym list expressions…)

Open the Visual LISP Editor...

```
(defun c:BOX1 ()
   (setq PT1 (getpoint "\nEnter first point: "))
   (setq PT2 (getpoint "\nEnter next point: "))
   (setq PT3 (getpoint "\nEnter next point: "))
   (setq PT4 (getpoint "\nEnter next point: "))
   (command "line" PT1 PT2 PT3 PT4 "c")
)
```

Save, load, activate AutoCAD, run the program

(prompt), (initget), (princ), error trapping (page 14)

```
(defun C:BOX2 () ; Change the version number
  (prompt "\nYou are about to draw a four sided polygon...") ; Add this line
  (initget 1) ; Add this line
  (setq PT1 (getpoint "\nEnter first point: "))
  (initget 1) ; Add this line
  (setq PT2 (getpoint "\nEnter next point: "))
  (initget 1) ; Add this line
  (setq PT3 (getpoint "\nEnter next point: "))
  (initget 1) ; Add this line
  (setq PT4 (getpoint "\nEnter next point: "))
  (command "line" PT1 PT2 PT3 PT4 "c")
  (princ) ; Add this line
)
```

(getvar), (setvar) (page 15)

```
(defun C:BOX3 () ; Change the version number
  (prompt "\nThis program will draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho")) ; Add this line
  (setvar "cmdecho" 0) ; Add this line
  (initget 1)
  (setq PT1 (getpoint "\nEnter first point: "))
  (initget 1)
  (setq PT2 (getpoint "\nEnter next point: "))
  (initget 1)
  (setq PT3 (getpoint "\nEnter next point: "))
  (initget 1)
  (setq PT4 (getpoint "\nEnter next point: "))
  (command "line" PT1 PT2 PT3 PT4 "c")
  (setvar "cmdecho" OCMDECHO) ; Add this line
  (princ)
)
```

(getstring) (page 16)

```
(defun C:BOX4 () ; Change the version number
  (prompt "\nYou are about to draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho"))
  (setvar "cmdecho" 0)
  (setq OCLR (getvar "cecolor")) ; Add this line
  (setq CLR (getstring "\nEnter RED, YELLOW, or BLUE: ")) ; Add this line
  (command "color" CLR) ; Add this line
  (initget 1)
  (setq PT1 (getpoint "\nEnter first point: "))
  (initget 1)
  (setq PT2 (getpoint "\nEnter next point: "))
  (initget 1)
  (setq PT3 (getpoint "\nEnter next point: "))
  (initget 1)
  (setq PT4 (getpoint "\nEnter next point: "))
  (command "line" PT1 PT2 PT3 PT4 "c")
  (setvar "cecolor" OCLR) ; Add this line
  (setvar "cmdecho" OCMDECHO)
  (princ)
)
```

(getcorner), (car), (cadr), (list) (page 17)

```
(defun C:BOX5 () ; Change the version number
  (prompt "\nThis program will draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho"))
  (setvar "cmdecho" 0)
  (setq OCLR (getvar "cecolor"))
  (initget 1 "Red Yellow Blue")
  (setq CLR (getkword "\nEnter Red, Yellow, or Blue: "))
  (command "color" CLR)
  (initget 1)
 ; Change the prompt below
  (setq PT1 (getpoint "\nEnter first corner: "))
  (initget 1)
 ; Change "point" in the next line to "corner" and add PT1
  (setq PT2 (getcorner PT1 "\nEnter opposite corner: "))
 ; Delete these 4 lines of (initget)s and (getpoints)
 ; (initget 1)
 ; (setq PT3 (getpoint "\nEnter next point: "))
 ; (initget 1)
 ; (setq PT4 (getpoint "\nEnter next point: "))
 ; And replace them with the folowing 6 lines
  (setq X1 (car PT1)) ; Add this line
  (setq Y1 (cadr PT1)) ; Add this line
  (setq X2 (car PT2)) ; Add this line
  (setq Y2 (cadr PT2)) ; Add this line
  (setq PT3 (list X1 Y2)) ; Add this line
  (setq PT4 (list X2 Y1)) ; Add this line
 ; Change the order of the point symbols in the next line
  (command "line" PT1 PT4 PT2 PT3 "c")
  (setvar "cecolor" OCLR)
  (setvar "cmdecho" OCMDECHO)
  (princ)
)
```

(repeat), new bit code for (initget), looping

(page 18)

```
(defun C:BOX6 (); Change the version number
  (prompt "\nYou are about to draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho"))
  (setvar "cmdecho" 0)
  (setq OCLR (getvar "cecolor"))
  (initget 1 "Red Yellow Blue")
  (setq CLR (getkword "\nEnter [Red/Yellow/Blue]: "))
  (command "color" CLR)
  (initget 7); Add this line
  (setq TIMES (getint "\nEnter number of boxes: ")); Add this line
  (repeat TIMES; Add this line
    (initget 1)
    (setq PT1 (getpoint "\nEnter first point: "))
    (initget 1)
    (setq PT2 (getcorner PT1 "\nEnter opposite corner: "))
    (setq X1 (car PT1))
    (setq Y1 (cadr PT1))
    (setq X2 (car PT2))
    (setq Y2 (cadr PT2))
    (setq PT3 (list X1 Y2))
    (setq PT4 (list X2 Y1))
    (command "line" PT1 PT4 PT2 PT3 "c")
  ); Add this closing parenthesis for the (repeat) function
  (setvar "cecolor" OCLR)
  (setvar "cmdecho" OCMDECHO)
  (princ)
)
```

(foreach), more looping (page 19)

```
(defun C:BOX7 () ; Change the version number
  (prompt "\nYou are about to draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho"))
  (setvar "cmdecho" 0)
  (setq OCLR (getvar "cecolor"))
  (initget 1 "Red Yellow Blue")
  (setq CLR (getkword "\nEnter [Red/Yellow/Blue]: "))
  (command "color" CLR)
  (initget 7)
  (setq TIMES (getint "\nEnter number of boxes: "))
  (repeat TIMES
    (initget 1)
    (setq PT1 (getpoint "\nEnter first point: "))
    (initget 1)
    (setq PT2 (getcorner PT1 "\nEnter opposite corner: "))
    (setq X1 (car PT1))
    (setq Y1 (cadr PT1))
    (setq X2 (car PT2))
    (setq Y2 (cadr PT2))
    (setq PT3 (list X1 Y2))
    (setq PT4 (list X2 Y1))
    (command "line" PT1 PT4 PT2 PT3 "c")
    (foreach XXX (list PT1 PT2 PT3 PT4) ; Add this line
      (command "circle" XXX 1) ; Add this line
      (command "circle" XXX 0.5) ; Add this line
    ) ; Add this line
  )
  (setvar "cecolor" OCLR)
  (setvar "cmdecho" OCMDECHO)
  (princ)
)
```

(if), (getreal), (getint) (page 20)

```
(defun C:BOX8 () ; Change the version number
  (prompt "\nYou are about to draw a four sided polygon...")
  (setq OCMDECHO (getvar "cmdecho"))
  (setvar "cmdecho" 0)
  (setq OCLR (getvar "cecolor"))
  (initget 1 "Red Yellow Blue")
  (setq CLR (getkword "\nEnter [Red/Yellow/Blue]: "))
  (command "color" CLR)
  (initget 7) ; Add this line
  (setq RAD (getreal "\nEnter corner circle radius: ")) ; Add this line
  (initget 6) ; Change the 7 to a 6
  (setq TIMES (getint "\nEnter number of boxes <1>: ")) ; Add "<1>" to prompt
  (if TIMES
    (setq TIMES TIMES)
    (setq TIMES 1)
  ) ; Add this line
  (repeat TIMES
    (initget 1)
    (setq PT1 (getpoint "\nEnter first point: "))
    (initget 1)
    (setq PT2 (getcorner PT1 "\nEnter opposite corner: "))
    (setq X1 (car PT1))
    (setq Y1 (cadr PT1))
    (setq X2 (car PT2))
    (setq Y2 (cadr PT2))
    (setq PT3 (list X1 Y2))
    (setq PT4 (list X2 Y1))
    (command "line" PT1 PT4 PT2 PT3 "c")
    (foreach XXX (list PT1 PT2 PT3 PT4)
      (command "circle" XXX RAD) ; Change 1 to RAD
      (command "circle" XXX (/ RAD 2)) ; Change 0.5 to (/ RAD 2)
    )
  )
  (setvar "cecolor" OCLR)
  (setvar "cmdecho" OCMDECHO)
  (princ)
)
```

Functions you have learned about:

(+ *num num* …)
(- *num num* …)
(* *num num* …)
(/ num num …)
(setq sym value)
(getint *str*)
(getreal *str*)
(getpoint *pt str*)
(getstring *flag str*)
(command str …)
(defun sym list (functions…))
(load str)
(getvar str)

(prompt str)
(initget int *str*)
(setvar str value)
(princ *value*)
(getkword *str*)
(getcorner *str*)
(car *list*)
(cadr *list*)
(list atoms)
(foreach sym list (functions))
(repeat int (functions))
(if list (function) *(function)*)

And That Concludes Our Training Session

- I appreciate your attention over the last 2.5 hours
- I will be in the hall, happy to answer all questions
- thank you, Thank You, THANK YOU!!
- Do not forget to fill out and turn in your evaluations
- black@fvtc.edu

AUTODESK®