AUTODESK® UNIVERSITY

SD468505 Which Editor to Use for AutoLISP Development: Notepad to Visual Studio Code

Lee Ambrosius

Principal Learning Experience Designer | @leeambrosius (Twitter)

Who’s this Session For

Those that want to learn how to:

- **Utilize a specialized AutoLISP editor**
- **Improve AutoLISP workflows used to**
  - Edit
  - Debug
  - Manage

What you should already know:

- **AutoCAD 2021 (or AutoCAD 2016 and later)**
- **AutoLISP programming**

About the Speaker

My name is Lee Ambrosius:

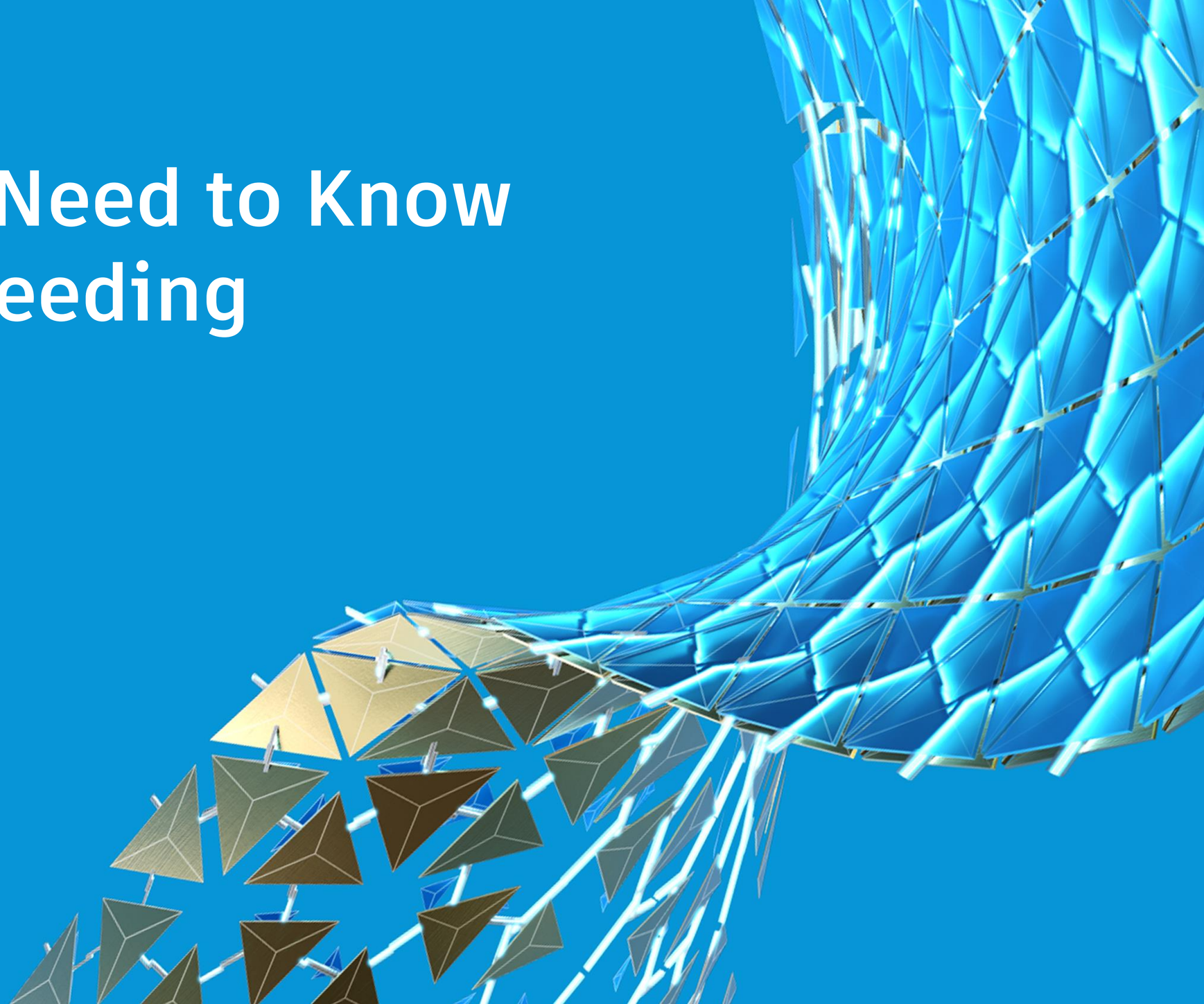
- **Principal Learning Experience Designer at Autodesk, Inc.**
  - Technical writer and data analyst
  - Customization, Developer, and CAD Administration documentation
- **Over 20+ years of AutoCAD customization and programming experience**
- **Authored AutoCAD Customization Platform book series published by Wiley & Sons**

My job in a nutshell:

- **Document the past and present AutoCAD releases for the future**

Yeah, running 48.6 miles in 4 Days is Dopey

Things You Need to Know Before Proceeding



What You Need to Get Started

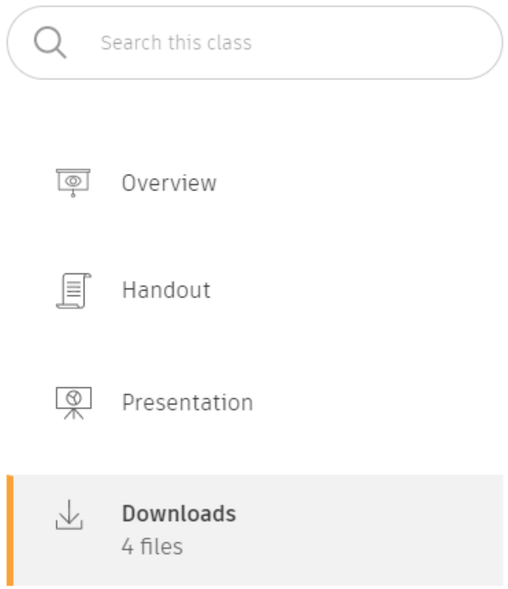
For this session, you will need/want:

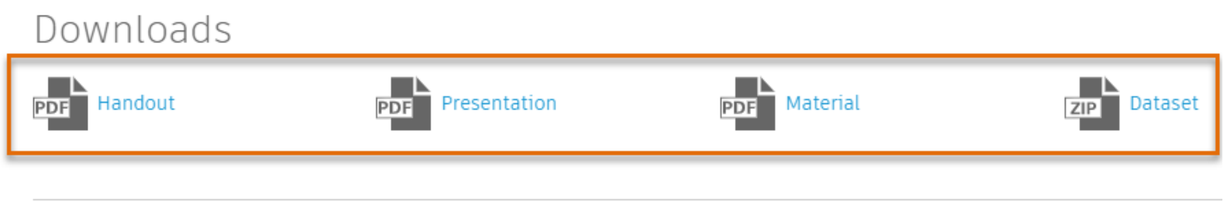
- **AutoCAD 2021 (or AutoCAD 2016 and later)**
- **Notepad**
- **Visual LISP Integrated Development Environment (VLIDE)**
- **Visual Studio (VS) Code and the AutoCAD AutoLISP Extension**
- **Materials for this session from the AU website**
  - Handout
  - Additional Materials
    - Extract to *C:\Datasets\SD468505*
- **Setup the *AutoLISP Files* folder under the *Documents* folder**
  - See *Things You Should Know* in the handout

Setting Up for this Session

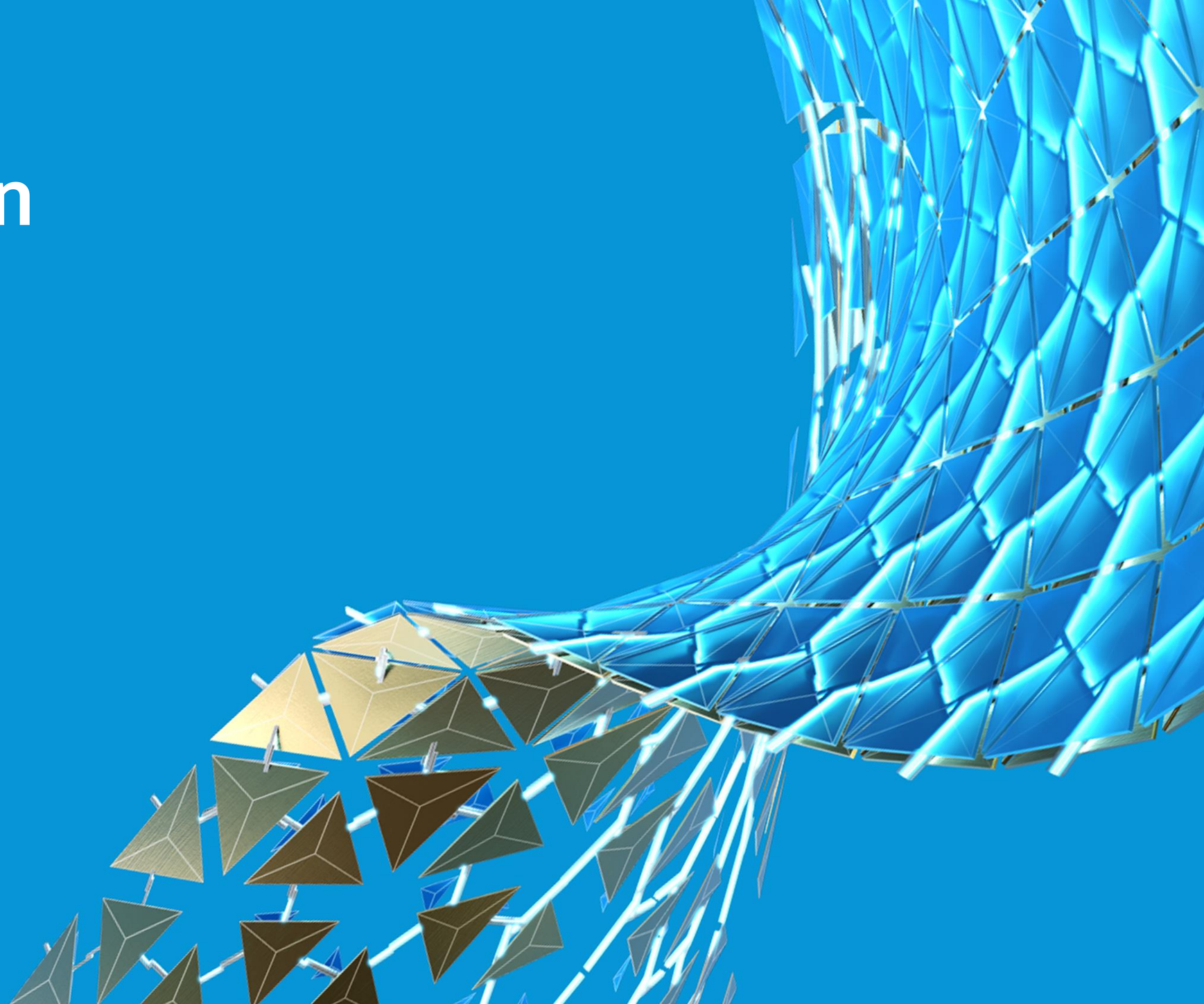
Materials for this session can be obtained by:

1. Going to the Autodesk University website and search on this session's ID of **SD468505**.
2. In the search results, click the entry for this session.
3. On the session page, click Downloads and then download
   a. Handout
   b. Material





Introduction



What You Will Learn Today

At the end of this session, you will have learned to:

- Work with LSP files in
  - Visual LISP
  - VS Code
- Format and debug LSP files
- Manage LSP files with AutoLISP projects
- Choose which specialized editor is right for you

Introduction

AutoLISP has been around since 1986

Many editors have been used to develop LSP programs

- Edlin
- Notepad
- TextEdit
- LispPad by Tony Tanzillo
- LispLink by CAE-Link Corporation
- Vital LISP by WSSW

Introduction

Vital LISP acquired around 1997

- Renamed to Visual LISP and shipped with AutoCAD 2000

With AutoCAD 2021, Visual LISP was deprecated

Support for VS Code was introduced with AutoCAD 2021

- Cross-platform
- Unicode support

Introduction

During this session, I will be talking these editors:

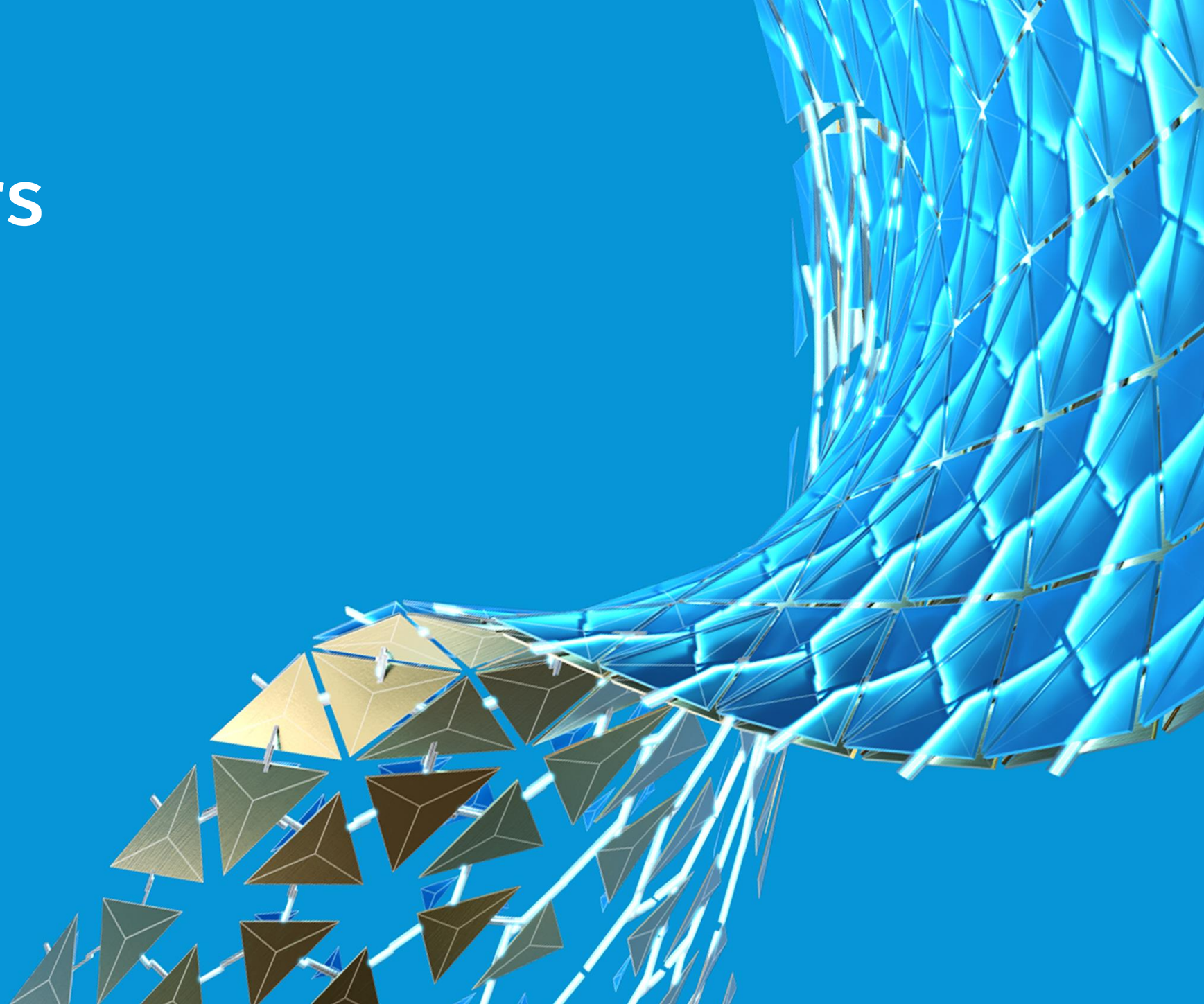
Editor	Supported Releases
Notepad	All releases of AutoCAD on Windows
TextEdit	All releases of AutoCAD on Mac OS
Visual LISP	AutoCAD 2000 and later on Windows
VS Code with AutoLISP Extension	AutoCAD 2021 on Windows or Mac OS

Related Commands and System Variables

VLISP command – Opens the default AutoLISP development environment from inside AutoCAD on Windows

LISPSYS system variable – Controls which AutoLISP environment is used to execute and edit LSP files (AutoCAD 2021)

Basic Editors



Basic Editors

Notepad on Windows

TextEdit on Mac OS

Both allow you to

- Edit ASCII and Unicode file formats
- Find/replace text strings
- Move and duplicate text with the Clipboard
- Control font size in the editor window

Format and Syntax Checking

Loading an AutoLISP file might produce the error:

- ; error: malformed list on input

Format checking can be done at the AutoCAD Command prompt

```
(defun c:MISSINGPAREN ()
  (prompt "\nMissing Paren"
)

(defun c:MISSINGPAREN ()
(_>   (prompt "\nMissing Paren"
((_> )
(_>
```

```
(defun c:MISSINGQUOTE ()
  (prompt "\nMissing Paren)
)

(defun c:MISSINGQUOTE ()
(_>   (prompt "\nMissing Paren)
(("_> )
(("_>
```

Format and Syntax Checking

```
(defun c:INVALIDDOTTEDPAIR1 ()
  (setq pair ("DOOR" . 36))
)

(defun c:INVALIDDOTTEDPAIR1 ()
(_>   (setq pair ("DOOR" . 36))
(_> )
; error: bad argument type: consp 36
```

Debugging

Debugging must be done as part of the program

- Output variable values during execution
  - `princ` function
- Output messages, Here I am!... Past ABC statement
  - `prompt` function
- Trace the execution of a custom function
  - `trace` function
  - `untrace` function

Even if you use a specialized editor, these functions can be helpful

- “Going on a Bug Hunt: Debugging and Handling Errors in AutoLISP” from AU 2013

Basic Editors: Notepad and TextEdit

Workflow of editing and loading files:

1. Create or open a LSP file.
2. Add/edit AutoLISP statements in the LSP file.
3. Save the LSP file.
4. Load and test the LSP file in AutoCAD.

Remember: No format checking or debugging tools are available

Basic Editors: Notepad and TextEdit

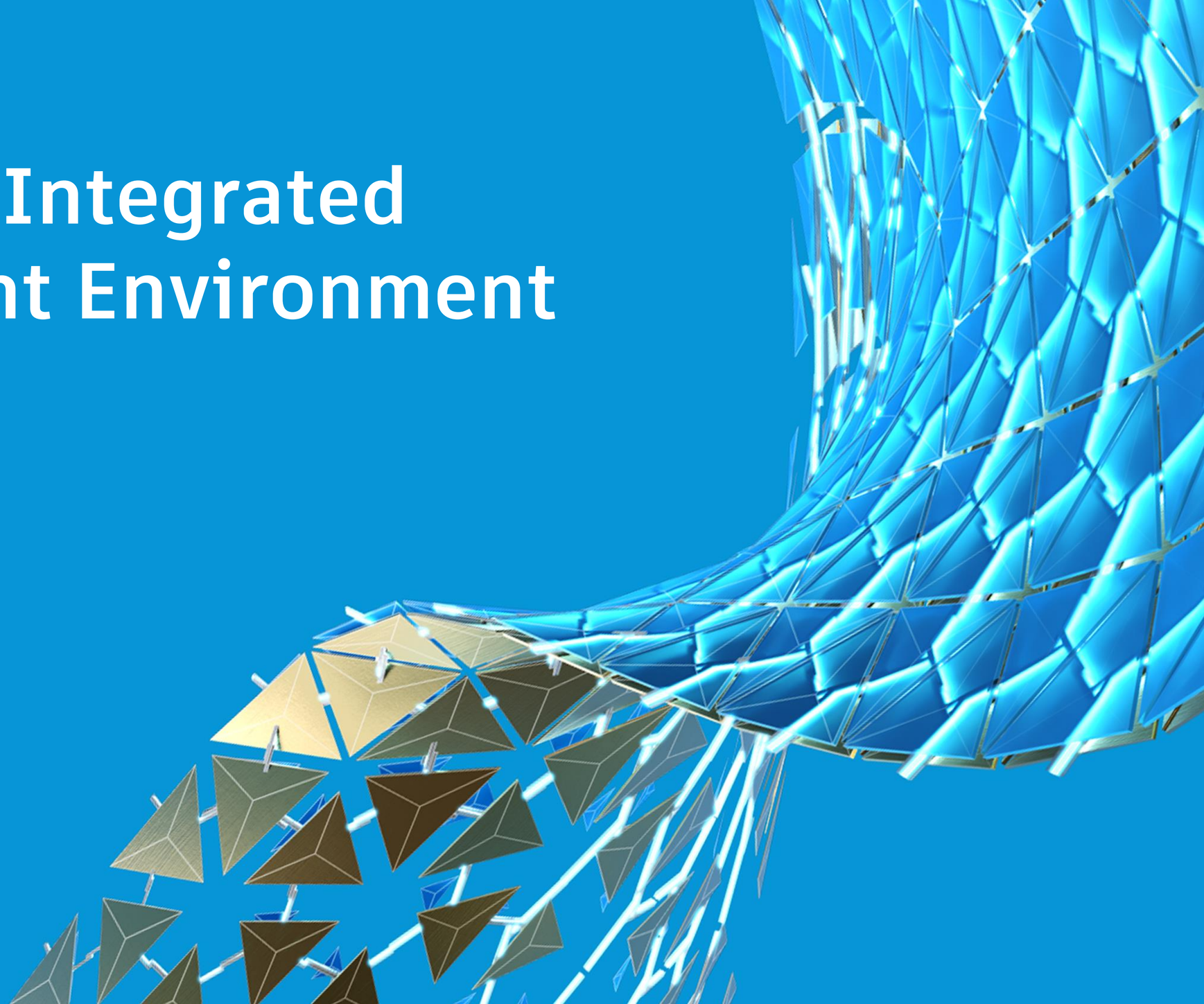
Workflow of editing and loading files:

1. Create or open a LSP file.
2. Add/edit AutoLISP statements in the LSP file.
3. Save the LSP file.
4. Load and test the LSP file in AutoCAD.

Remember: No format checking or debugging tools are available

DEMO

Visual LISP Integrated Development Environment



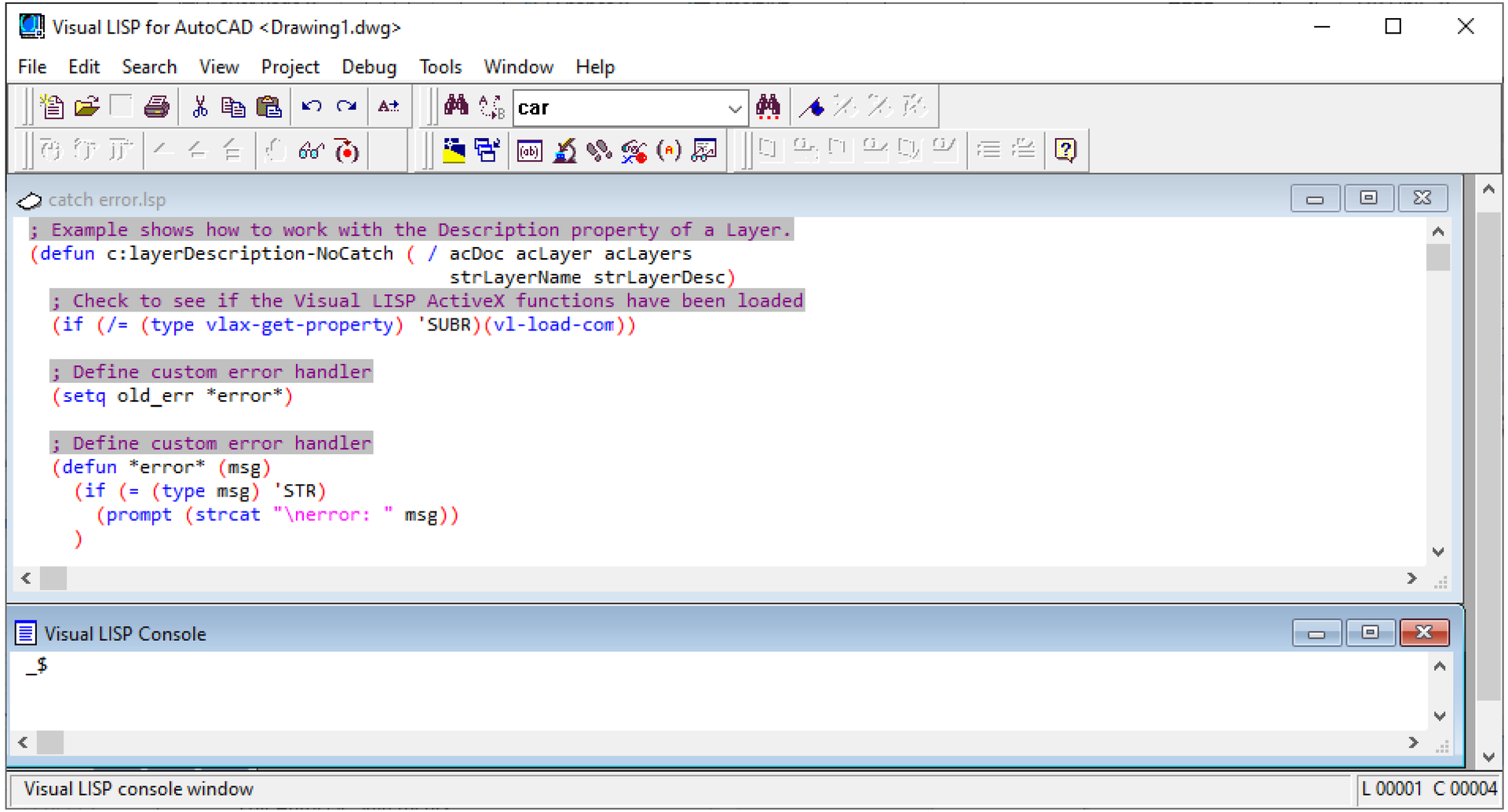
Visual LISP Integrated Development Environment

Allows you to

- Create and edit LSP files
- View AutoLISP statements with syntax highlighting
- Format and check AutoLISP statements
- Load and debug AutoLISP statements
- Manage LSP files with AutoLISP projects
- Compile and protect LSP files

Not available in AutoCAD for Mac OS

Visual LISP Integrated Development Environment



Launch Visual LISP

Launch the Visual LISP Editor

- Manage tab > Applications panel > Visual LISP Editor (VLISP command)

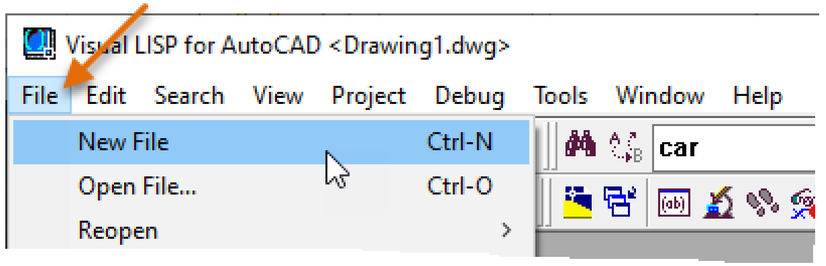
LISPSYS system variable must be set to 0

Create and Open LSP Files

Create and open LSP files

- File menu > New File
- File menu > Open
- File menu > Save as (specify file format)

File menu > Reopen to open a previous file

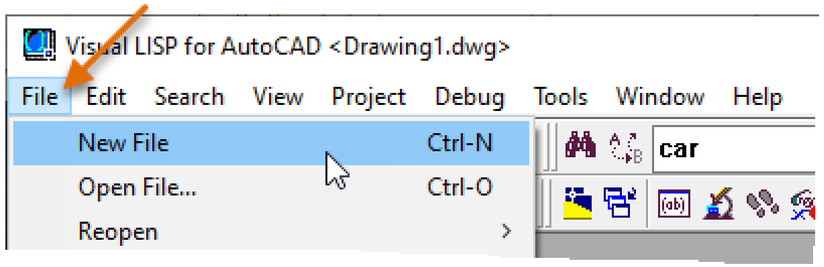


Create and Open LSP Files

Create and open LSP files

- File menu > New File
- File menu > Open
- File menu > Save as (specify file format)

File menu > Reopen to open a previous file



DEMO

Edit LSP Files

You can use these editing features with a LSP file

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting

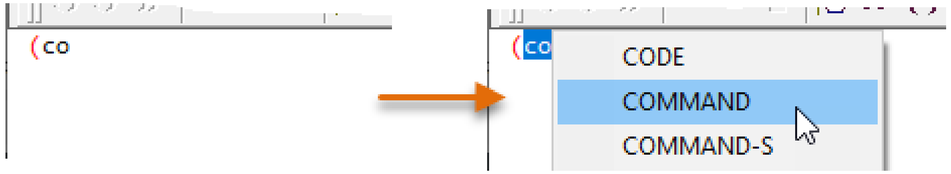
```
(foreach dir dirs
  (if (and (/= dir ".") (/= dir ".."))
    (progn
      (setq indent "\n)

      (repeat level
        (setq indent (strcat indent "  "))
      )

      (prompt (strcat indent dir))
      (recurse_folders (strcat path dir) (1+ level))
    )
  )
)
```

```
; Recurse folders
; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
(defun recurse_folders (path level / indent dir dirs)
  (setq dirs (vl-directory-files path nil -1))

  (foreach dir dirs
    (if (and (/= dir ".") (/= dir ".."))
      (progn
        (setq indent "\n")

        (repeat level
          (setq indent (strcat indent "  "))
        )

        (prompt (strcat indent dir))
        (recurse_folders (strcat path dir) (1+ level))
      )
    )
  )
  (princ)
)
```

Edit LSP Files

You can use these editing features with a LSP file

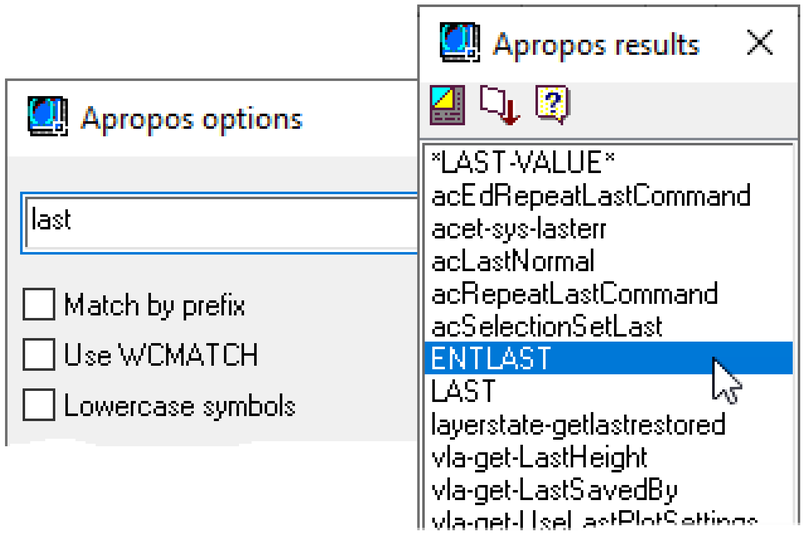
- **Syntax highlighting**
- **Complete word by match or Apropos**
  - Typing in partial function name and pressing Shift+Ctrl+Spacebar



Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Complete word by match or Apropos
- List available functions with the Apropos window



Edit LSP Files

You can use these editing features with a LSP file

- **Syntax highlighting**
- **Complete word by match or Apropos**
- **List available functions with the Apropos window**
- **Toggle statements as comments**
  - Mark selected lines as comments
    - Edit menu > Extra Commands > Comment Block
    - Comment Block on the Tools toolbar
  - Unmark selected lines as comments
    - Edit menu > Extra Commands > Uncomment Block
    - Uncomment Block on the Tools toolbar

```
;;; Program created by Lee Ambrosius
;;; Last updated on: 8/22/2020
```

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Complete word by match or Apropos
- List available functions with the Apropos window
- Toggle statements as comments
- Select elements and statements between parentheses
  - Selecting and matching open parenthesis
    - Edit menu > Parentheses Matching > Match Backwards
    - Edit menu > Parentheses Matching > Select Backwards
  - Selecting and matching closing parenthesis
    - Edit menu > Parentheses Matching > Match Forwards
    - Edit menu > Parentheses Matching > Select Forwards

```
; Recurse folders
; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
(defun recurse_folders (path level / indent dir dirs)
  (setq dirs (vl-directory-files path nil -1))

  (foreach dir dirs
    (if (and (/= dir ".") (/= dir ".."))
      (progn
        (setq indent "\n")

        (repeat level
          (setq indent (strcat indent "  "))
        )

        (prompt (strcat indent dir))
        (recurse_folders (strcat path dir) (1+ level))
      )
    )
  )
  (princ)
)
```

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Complete word by match or Apropos
- List available functions with the Apropos window
- Toggle statements as comments
- Select elements and statements between parentheses
- Save and insert code statements
  - Selected statements can be saved to a LSP file
  - LSP files can be inserted into a LSP file

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Complete word by match or Apropos
- List available functions with the Apropos window
- Toggle statements as comments
- Select elements and statements between parentheses
- Save and insert code statements

DEMO

Format LSP Statements

Selected statements

- Tools menu > Format Code in Selection
- Format Selection on the Tools toolbar

All statements

- Tools menu > Format Code in Editor
- Format Edit Window on the Tools toolbar

```
(defun c:NINSERT ( / lastEnt)
(setq lastEnt (entlast))
(initdia)
(command "INSERT")
(while (not (zerop (getvar "CMDACTIVE")))
(command PAUSE)
)
(if (eq lastEnt (entlast))
(alert "No block was inserted.")
(alert "Block was inserted")
)
(princ)
)
```

```
(defun c:NINSERT (/ lastEnt)
  (setq lastEnt (entlast))
  (initdia)
  (command "INSERT")
  (while (not (zerop (getvar "CMDACTIVE")))
    (command PAUSE)
  )
  (if (eq lastEnt (entlast))
    (alert "No block was inserted.")
    (alert "Block was inserted")
  )
  (princ)
)
```

Check LSP Statements

Selected statements

- Tools menu > Check Selection
- Check Selection on the Tools toolbar

All statements

- Tools menu > Check Text in Editor
- Check Edit Window on the Tools toolbar

```
(defun c:NINSERT (/ lastEnt)
  (setq lastEnt (entlast))
  (initdia)
  (command "INSERT)
  (while (not (zerop (getvar "CMDACTIVE")))
    (command PAUSE)
  )
  (if (eq lastEnt (entlast))
    (alert "No block was inserted.")
```

Missing quotation mark

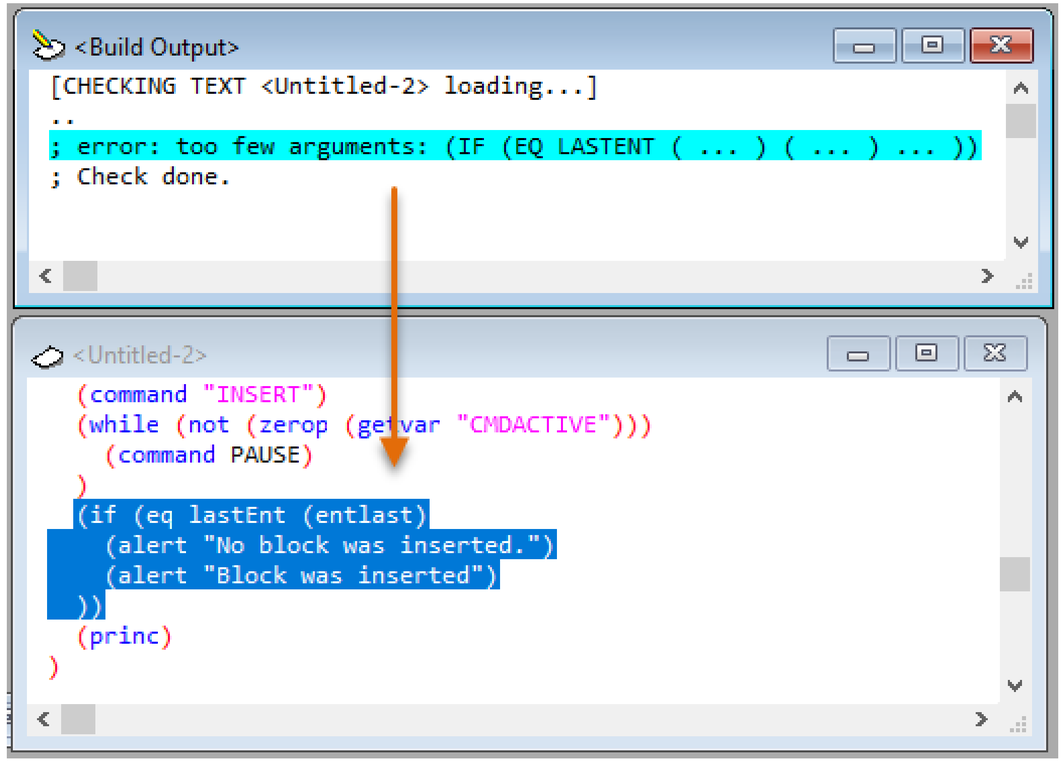
```
(defun c:NINSERT (/ lastEnt)
  (setq lastEnt (entlast))
  (initdia)
  (command "INSERT")
  (while (not (zerop (getvar "CMDACTIVE")))
    (command PAUSE)
  )
  (if (eq lastEnt (entlast)
    (alert "No block was inserted.")
    (alert "Block was inserted")
  )
  (princ)
)
```

Missing closing parenthesis

Check LSP Statements

Statement errors output to the Build Output window

Double-click an error to navigate to it



Load LSP Statements

Selected statements

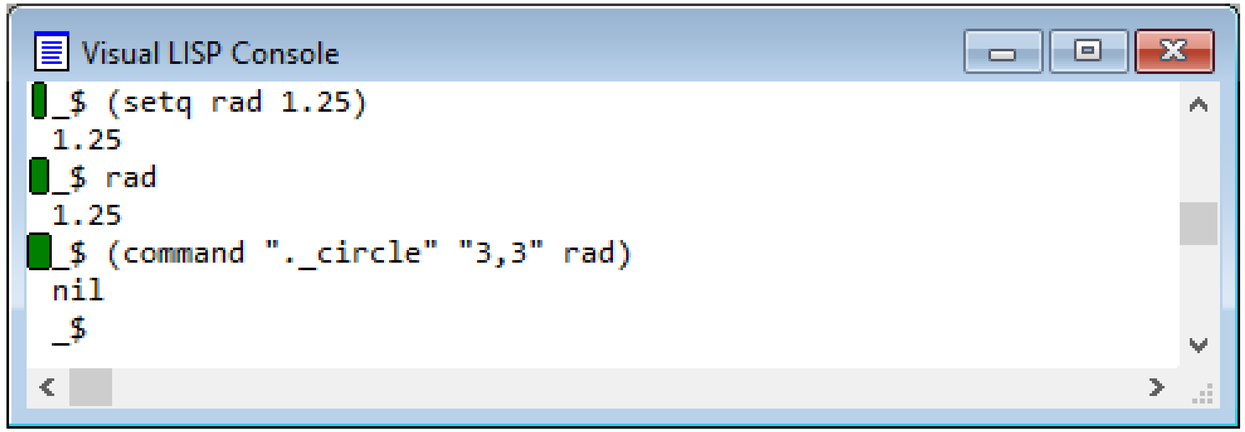
- Tools menu > Load Selection
- Load Selection on the Tools toolbar

All statements

- Tools menu > Load Text in Editor
- Load Active Edit Window on the Tools toolbar

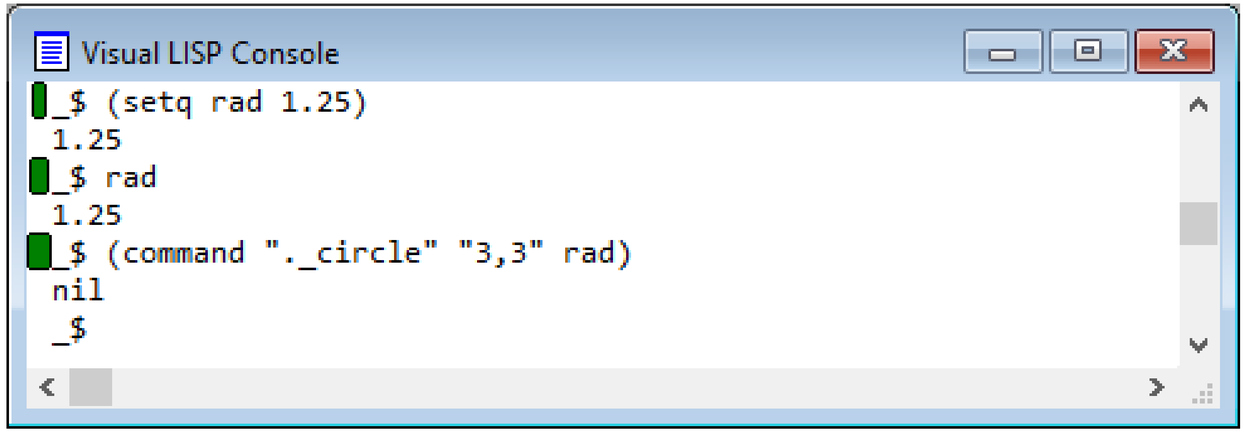
Load LSP Statements

Statements can also be entered in the Visual LISP Console window



Load LSP Statements

Statements can also be entered in the Visual LISP Console window



DEMO

Debug LSP Files

You can use these debugging features on a LSP file

Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints

```
(defun main ( / )
  (setq dir "LOCALROOTPREFIX"
        msg (strcat "\nListing directories in " dir)
  )
  (prompt msg)
  (recurse_folders (getvar "LOCALROOTPREFIX") 0)
)
```

```
(defun main ( / )
  (
  )
  (
  (
)
```

Cut
Copy
Paste
Find...
Go to Last Edited
Toggle Breakpoint
Inspect...

```
directories in " dir)

ALROOTPREFIX") 0)
```

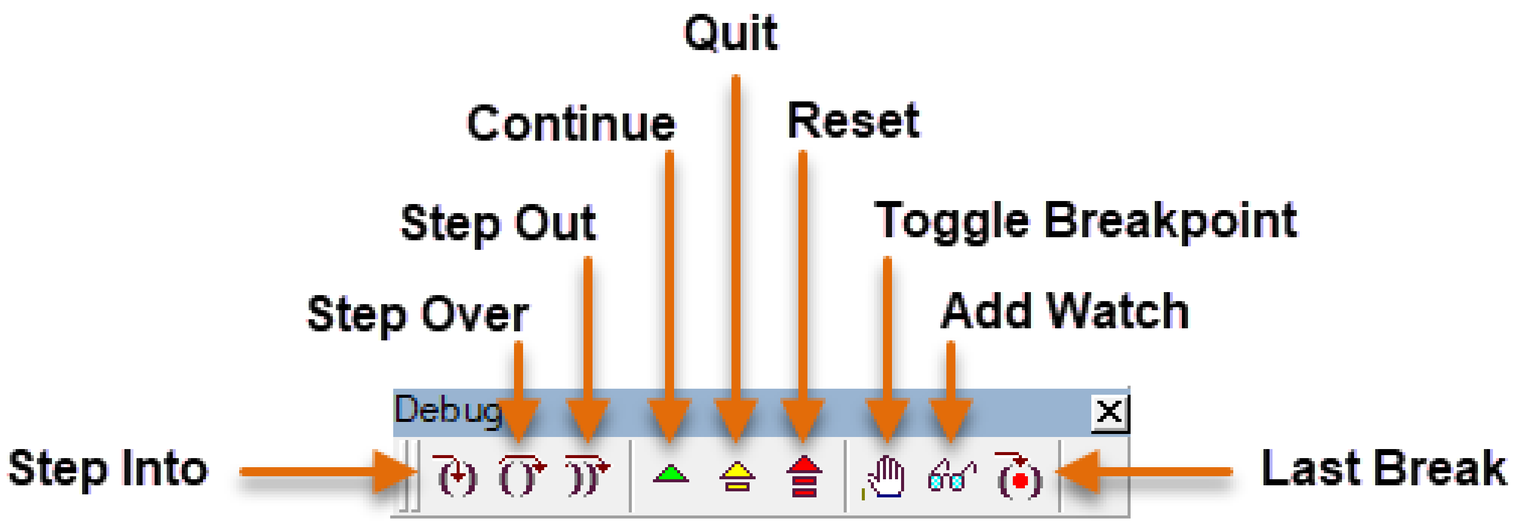
Breakpoint

```
(defun main ( / )
  (setq dir "LOCALROOTPREFIX"
        msg (strcat "\nListing directories in " dir)
  )
  (prompt msg)
  (recurse_folders (getvar "LOCALROOTPREFIX") 0)
)
```

Debug LSP Files

You can use these debugging features on a LSP file

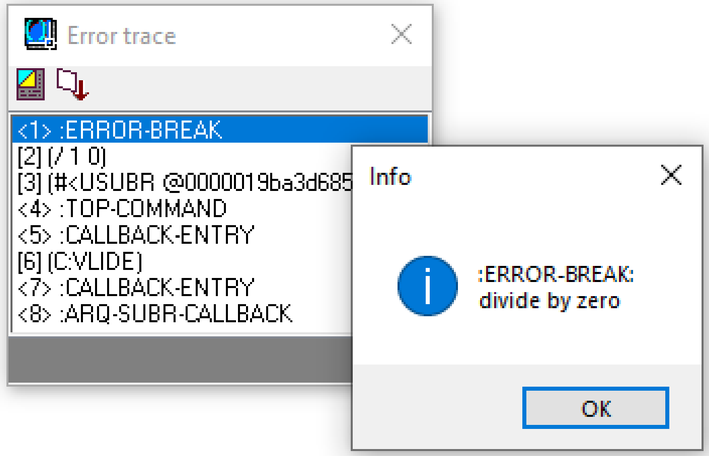
- Breakpoints
  - Debug toolbar



Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error

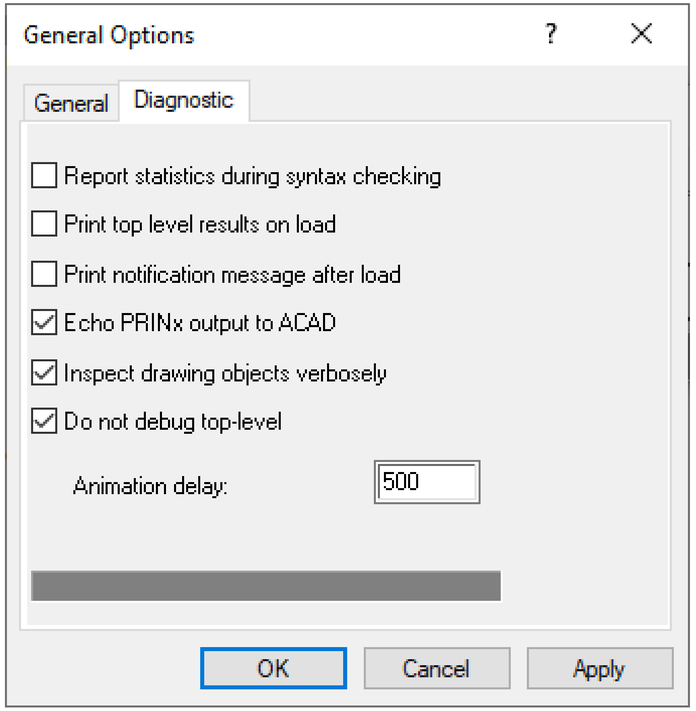


Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error
- Debug animation

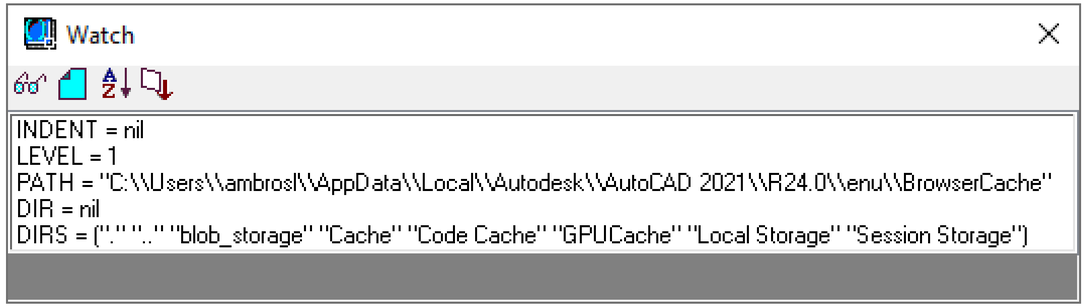
```
(defun main ( / )
  (setq dir "LOCALROOTPREFIX"
        msg (strcat "\nListing directories in " dir)
  )
  (prompt msg)
  (recurse_folders (getvar "LOCALROOTPREFIX") 0)
)
```

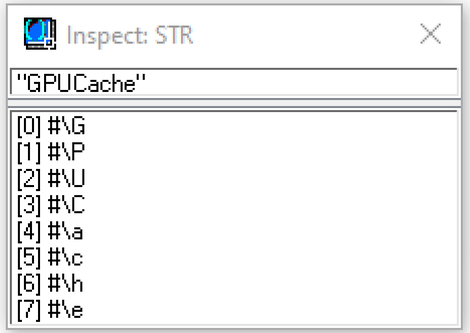


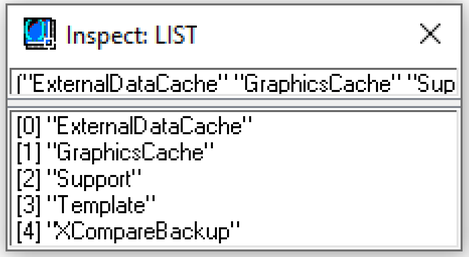
Debug LSP Files

You can use these debugging features on a LSP file

- **Breakpoints**
- **Break on Error**
- **Debug animation**
- **Watches**
  - Variables and statements can be watched for changes
  - Values in watches can be inspected



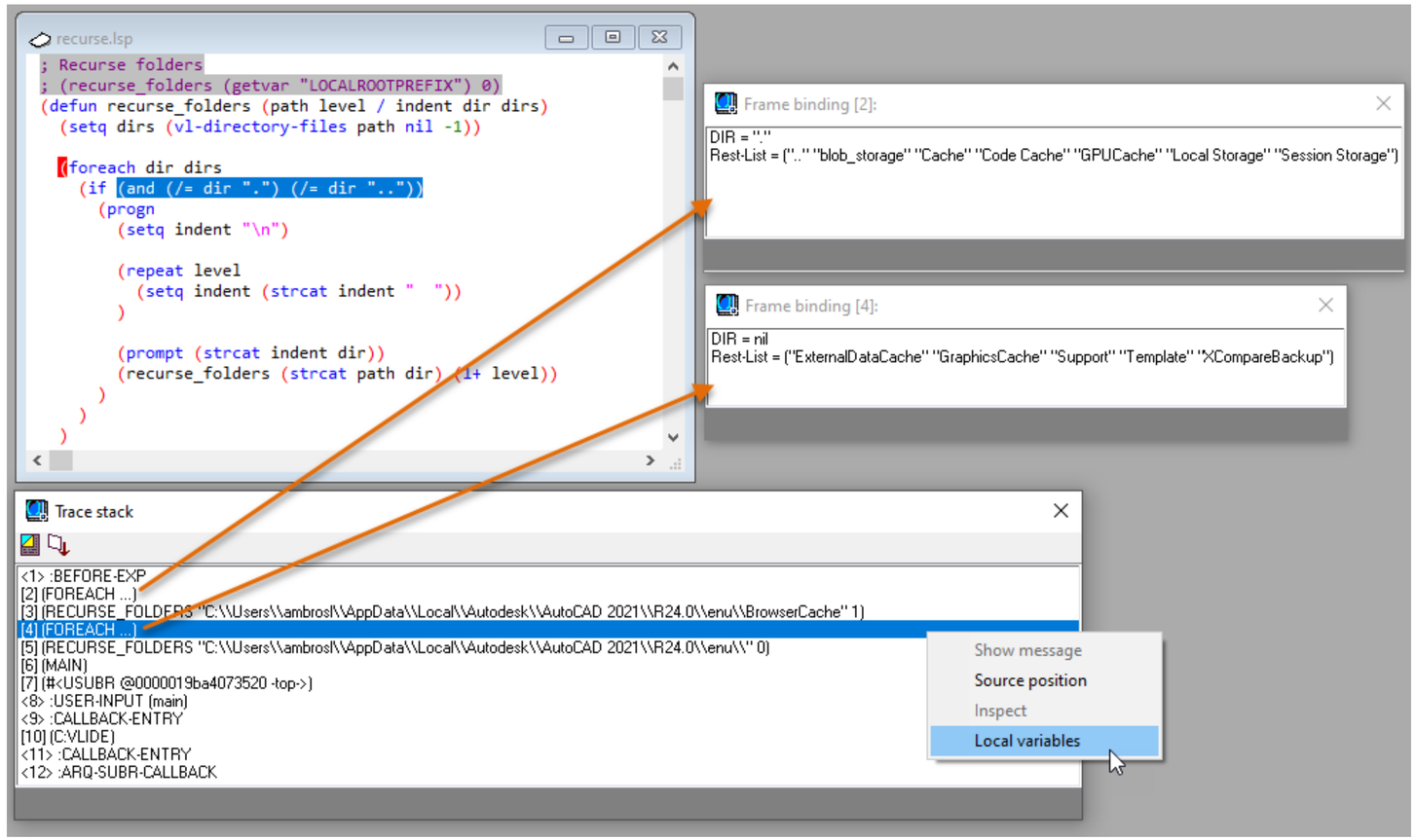




Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error
- Debug animation
- Watches
- Trace custom functions



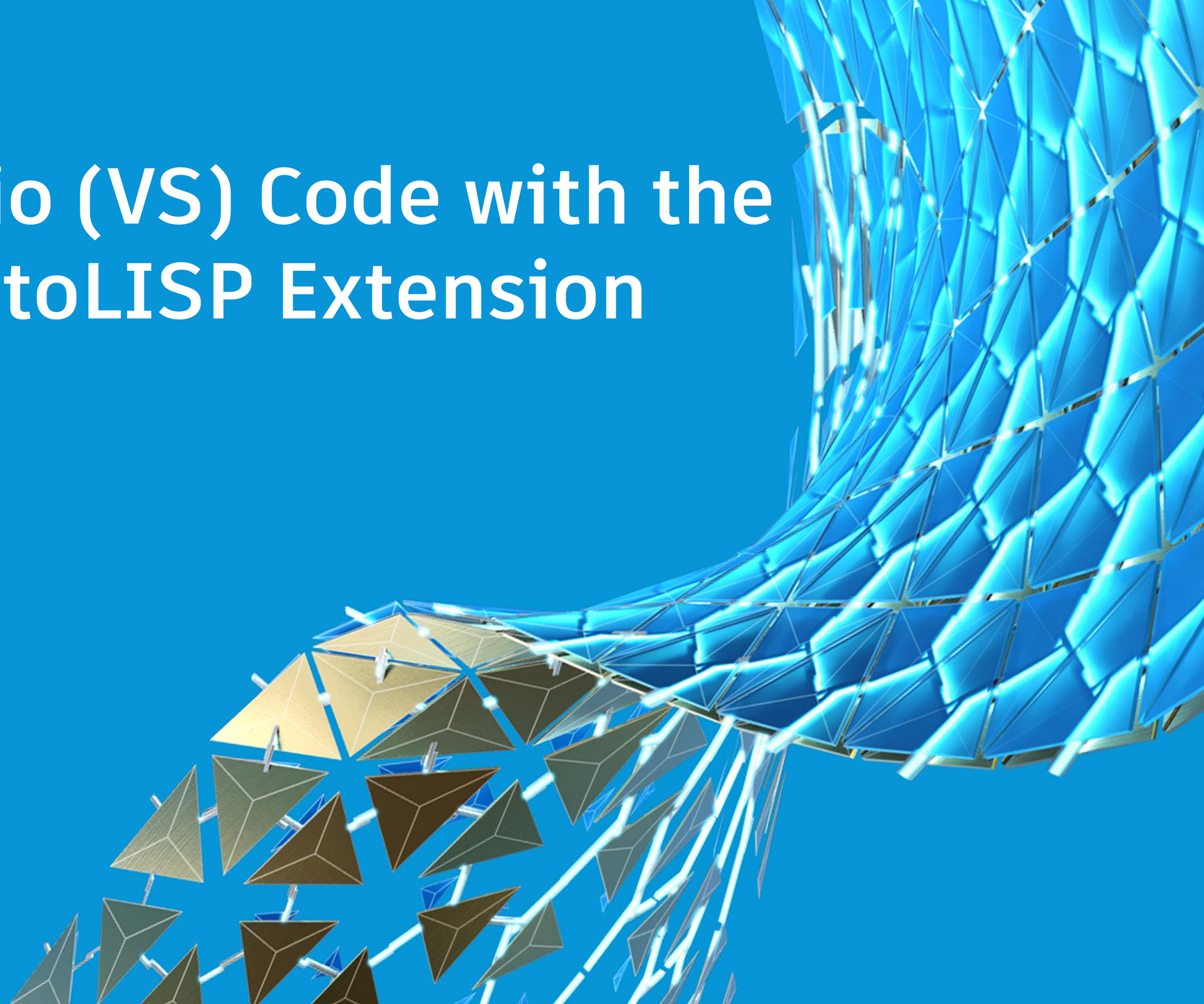
Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error
- Debug animation
- Watches
- Trace custom functions

DEMO

Visual Studio (VS) Code with the AutoCAD AutoLISP Extension



VS Code with AutoCAD AutoLISP Extension

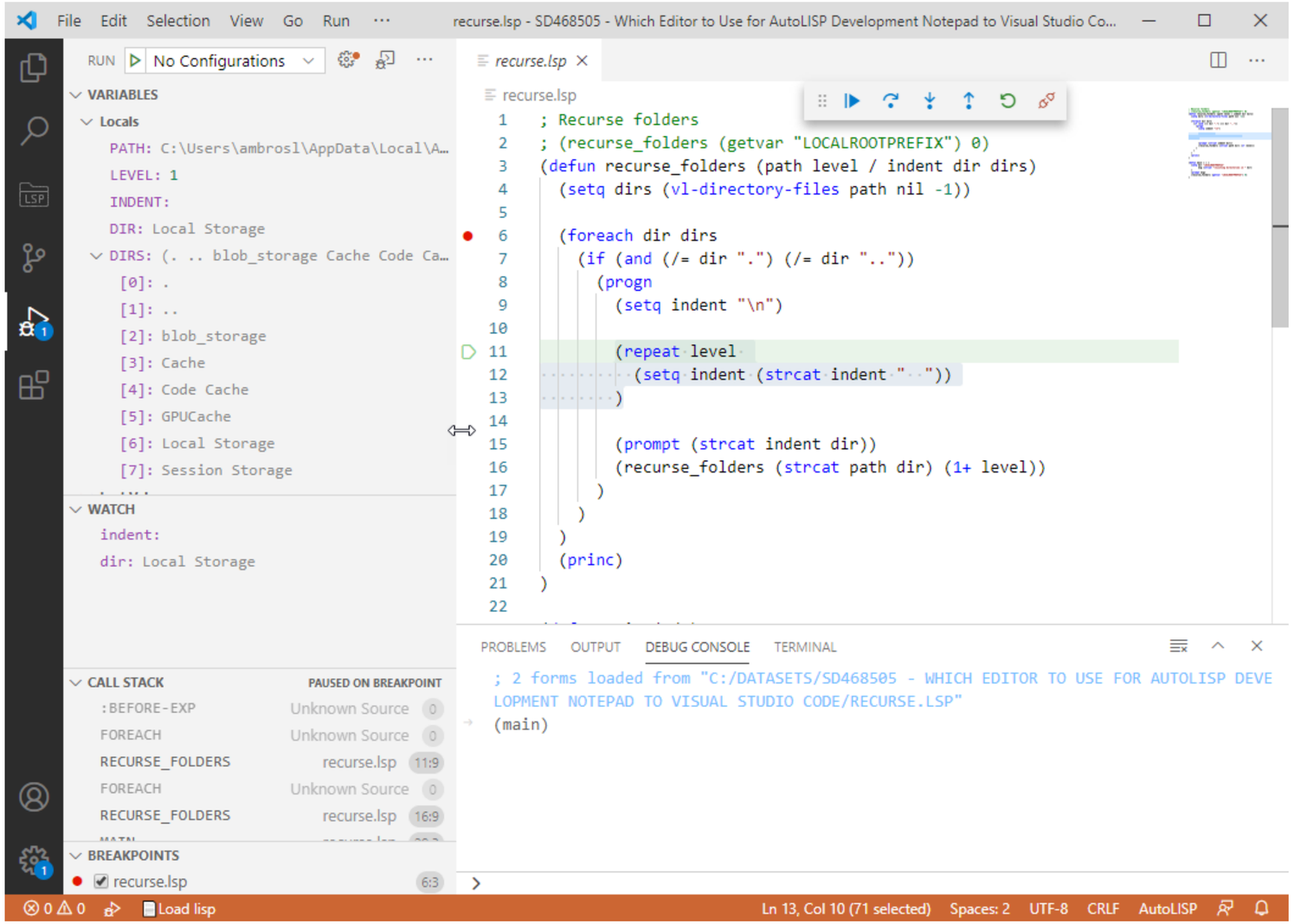
Allows you to

- Create and edit LSP files
- View AutoLISP statements with syntax highlighting
- Format AutoLISP statements
- Load and debug AutoLISP statements
- Manage LSP files with AutoLISP projects
- Compile and protect LSP files via AutoCAD command

Cross-platform support

- Windows
- Mac OS

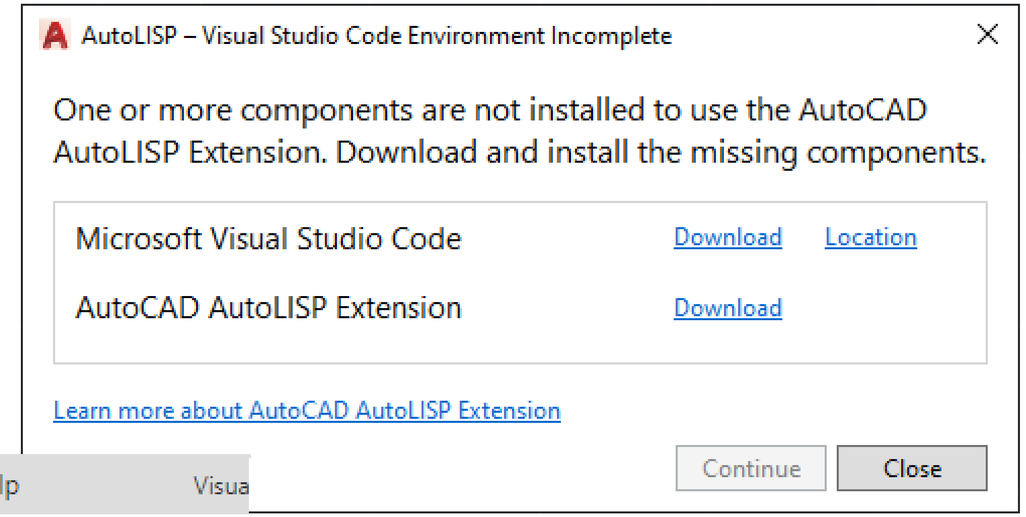
VS Code with AutoCAD AutoLISP Extension

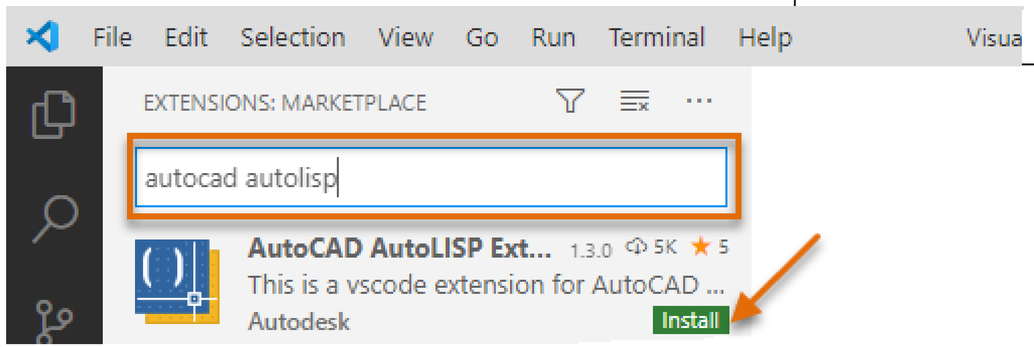


Before Using VS Code

Separately installed from AutoCAD

AutoCAD AutoLISP Extension must be installed





Launch VS Code

Launch VS Code from inside AutoCAD (Windows only)

- Manage tab > Applications panel > Visual LISP Editor (VLISP command)

Launch VS Code from outside AutoCAD

- Windows Start menu > Visual Studio Code > Visual Studio Code
- Mac OS Finder > Go > Applications > Visual Studio Code

LISPSYS system variable must be set to

- 1 – Unicode file format; compile to Unicode
- 2 – Unicode file format; compile to “legacy” ASCII (MBCS)

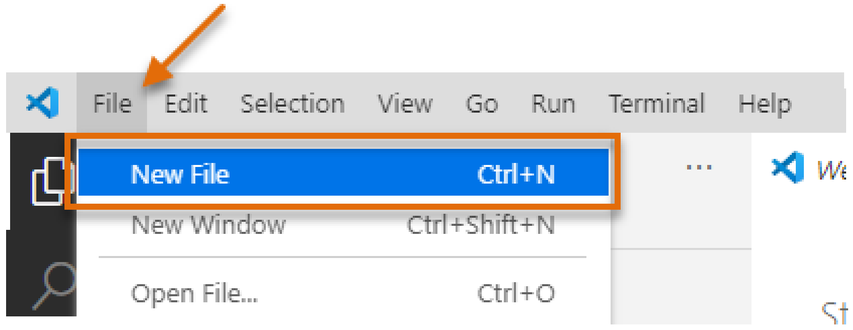
Create and Open LSP Files

Create and open LSP files

- File menu > New File
- File menu > Open File
- File menu > Save (specify .lsp extension)

File menu > Open Recent to open a previous file

File menu > Open Folder to open a folder



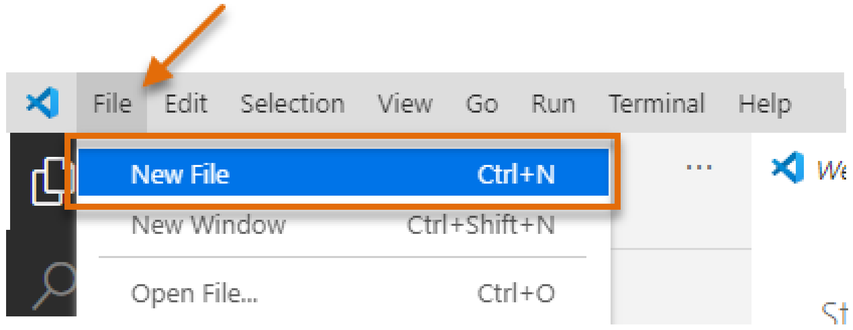
Create and Open LSP Files

Create and open LSP files

- File menu > New File
- File menu > Open File
- File menu > Save (specify .lsp extension)

File menu > Open Recent to open a previous file

File menu > Open Folder to open a folder



DEMO

Edit LSP Files

You can use these editing features with a LSP file

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting

```
6    (foreach dir dirs
7      (if (and (/= dir ".) (/= dir ".."))
8        (progn
9          (setq indent "\n")
10
11         (repeat level
12           (setq indent (strcat indent "  "))
13         )
14
15         (prompt (strcat indent dir))
16         (recurse_folders (strcat path dir) (1+ level))
17       )
18     )
19   )
```

```
1   ; Recurse folders
2   ; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
3   (defun recurse_folders (path level / indent dir dirs)
4     (setq dirs (vl-directory-files path nil -1))
5
6     (foreach dir dirs
7       (if (and (/= dir ".") (/= dir ".."))
8         (progn
9           (setq indent "\n")
10
11          (repeat level
12            (setq indent (strcat indent "  "))
13          )
14
15          (prompt (strcat indent dir))
16          (recurse_folders (strcat path dir) (1+ level))
17        )
18      )
19    )
20    (princ)
21  )
```

Edit LSP Files

You can use these editing features with a LSP file

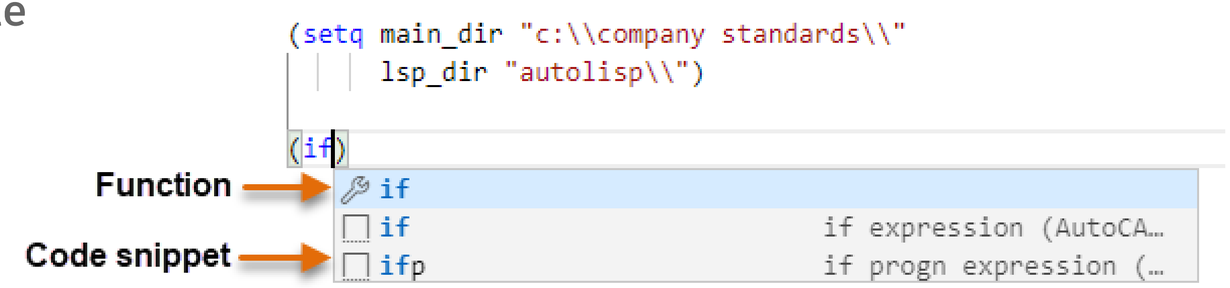
- **Syntax highlighting**
- **Smart brackets**
  - Adds a closing when an open parenthesis is typed
  - Adds a second quotation mark when one is typed

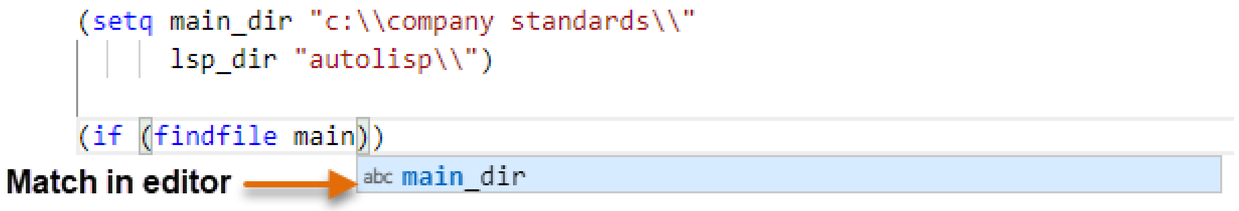
```
30
31    ()
32    ""
```

Edit LSP Files

You can use these editing features with a LSP file

- **Syntax highlighting**
- **Smart brackets**
- **Intellisense**
  - Match standard AutoLISP function names
  - Match text in the active editor window
  - Insert of code snippets





Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Smart brackets
- Intellisense
- Toggle statements as comments
  - Edit menu > Toggle Line Comment
  - Edit menu > Toggle Block Comment

```
; Recurse folders
; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
```

```
;| Program created by Lee Ambrosius
   Last updated on: 8/22/2020 |;
```

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Smart brackets
- Intellisense
- Toggle statements as comments
- Select elements and statements between parentheses
  - Select elements between parentheses
    - Click inside an AutoLISP statement
    - Click Selection menu > Expand Selection (multiple times)
  - Match parentheses
    - Click adjacent to a parenthesis

```
6   (foreach dir dirs
7     (if (and (/= dir ".") (/= dir ".."))
8       (progn
9         (setq indent "\n")
10
11        (repeat level
12          (setq indent (strcat indent "  "))
13        )
14
15        (prompt (strcat indent dir))
16        (recurse_folders (strcat path dir) (1+ level))
17      )
18    )
```

```
6   (foreach dir dirs
7     (if (and (/= dir ".") (/= dir ".."))
8       (progn
9         (setq indent "\n")
10
11        (repeat level
12          (setq indent (strcat indent "  "))
13        )
14
15        (prompt (strcat indent dir))
16        (recurse_folders (strcat path dir) (1+ level))
17      )
18    )
19  )
```

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Smart brackets
- Intellisense
- Toggle statements as comments
- Select elements and statements between parentheses
- Insert code snippets
  - Per extension
  - Per user
  - Can contain placeholders

```
ifp
ifp                    if progn express…
```

```
if progn expression (AutoCAD AutoLISP
Extension)

(if (testexpr)
    (progn
        (thenexpr)
    )
)
```

```
(if (testexpr)
  (progn
    (thenexpr)
  )
)
```

Edit LSP Files

You can use these editing features with a LSP file

- Syntax highlighting
- Smart brackets
- Intellisense
- Toggle statements as comments
- Select elements and statements between parentheses
- Insert code snippets

DEMO

Format LSP Statements

Selected statements

- Right-click and choose Format Selection

All statements

- Right-click and choose Format Document

```
(defun c:NINSERT ( / lastEnt)
(setq lastEnt (entlast))
(initdia)
(command "INSERT")
(while (not (zerop (getvar "CMDACTIVE")))
(command PAUSE)
)
(if (eq lastEnt (entlast))
(alert "No block was inserted.")
(alert "Block was inserted")
)
(princ)
)
```

```
(defun c:NINSERT (/ lastEnt)
  (setq lastEnt (entlast))
  (initdia)
  (command "INSERT")
  (while (not (zerop (getvar "CMDACTIVE")))
    (command PAUSE)
  )
  (if (eq lastEnt (entlast))
    (alert "No block was inserted.")
    (alert "Block was inserted")
  )
  (princ)
)
```

Configure Extension to Load and Debug Statements

Debug configurations must be setup to

- Load LSP files
- Execute AutoLISP statements
- Debug AutoLISP statements

Two debug configurations available

- AutoLISP Debug Attach (attachlisp)
- AutoLISP Debug Launch (launchlisp)

Load LSP Statements

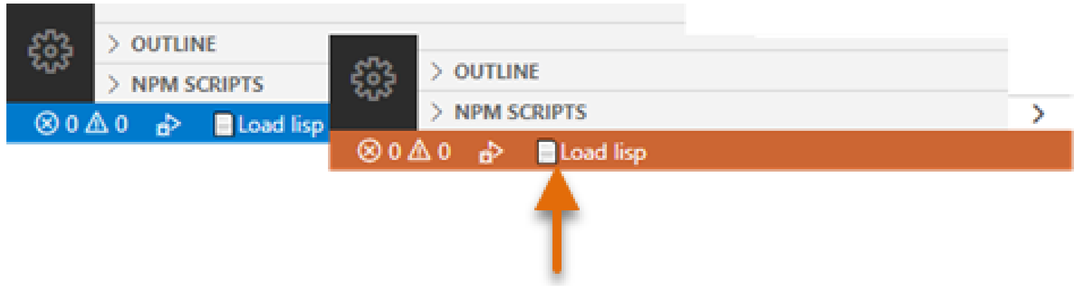
Debugging must first be started

Select statement

- Right-click and choose Evaluate in Debug Console

All statements

- Right-click and choose Load File in AutoCAD
- “Load Lisp” on the status bar



Load LSP Statements

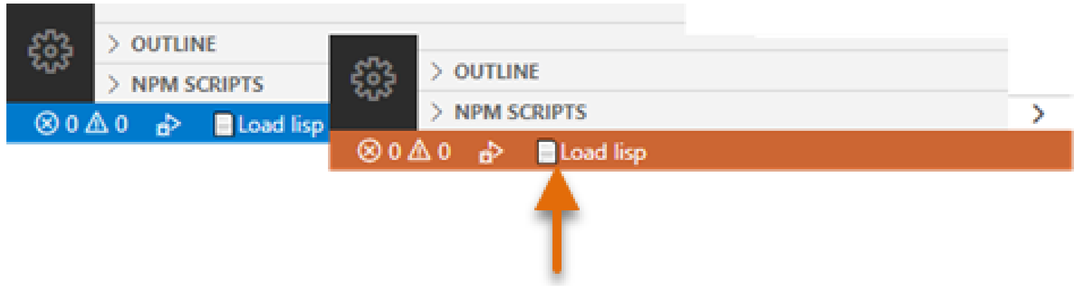
Debugging must first be started

Select statement

- Right-click and choose Evaluate in Debug Console

All statements

- Right-click and choose Load File in AutoCAD
- “Load Lisp” on the status bar



DEMO

Debug LSP Files

You can use these debugging features on a LSP file

Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints

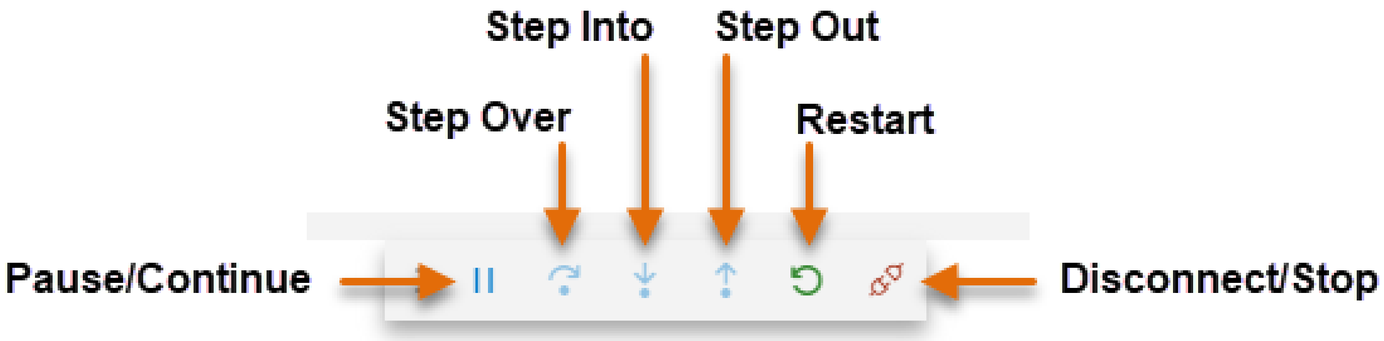
```
5
6   (foreach dir dirs
7     (if (and (/= dir ".") (/= dir ".."))
8       (progn
9         (setq indent "\n")
10
11        (repeat level
12          (setq indent (strcat indent "  "))
13        )
```

```
5
6   (foreach dir dirs
7     (if (and (/= dir ".") (/= dir ".."))
8       (progn
9         (setq indent "\n")
10
11        (repeat level
12          (setq indent (strcat indent "  "))
13        )
14
15        (prompt (strcat indent dir))
16        (recurse_folders (strcat path dir) (1+ level))
17      )
18    )
19  )
20  (princ)
21 )
```

Debug LSP Files

You can use these debugging features on a LSP file

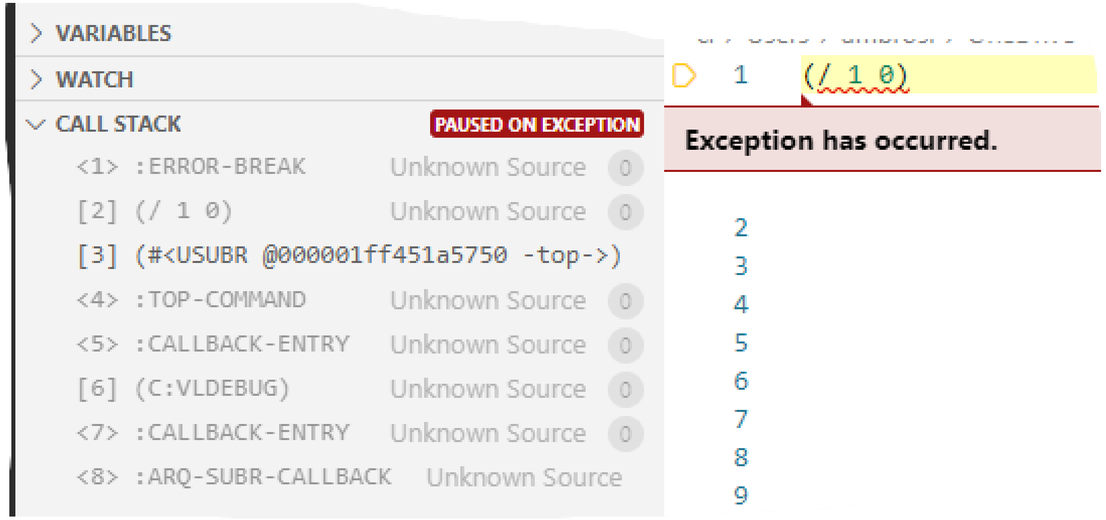
- **Breakpoints**
  - Debug toolbar



Debug LSP Files

You can use these debugging features on a LSP file

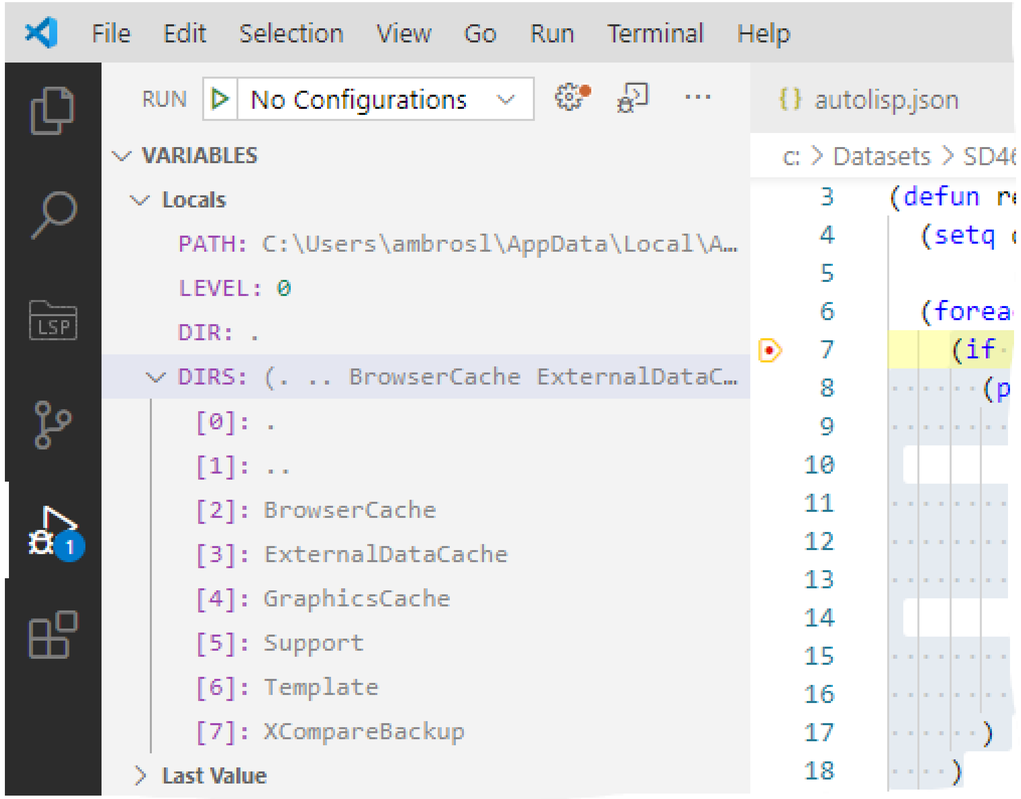
- Breakpoints
- Break on Error



Debug LSP Files

You can use these debugging features on a LSP file

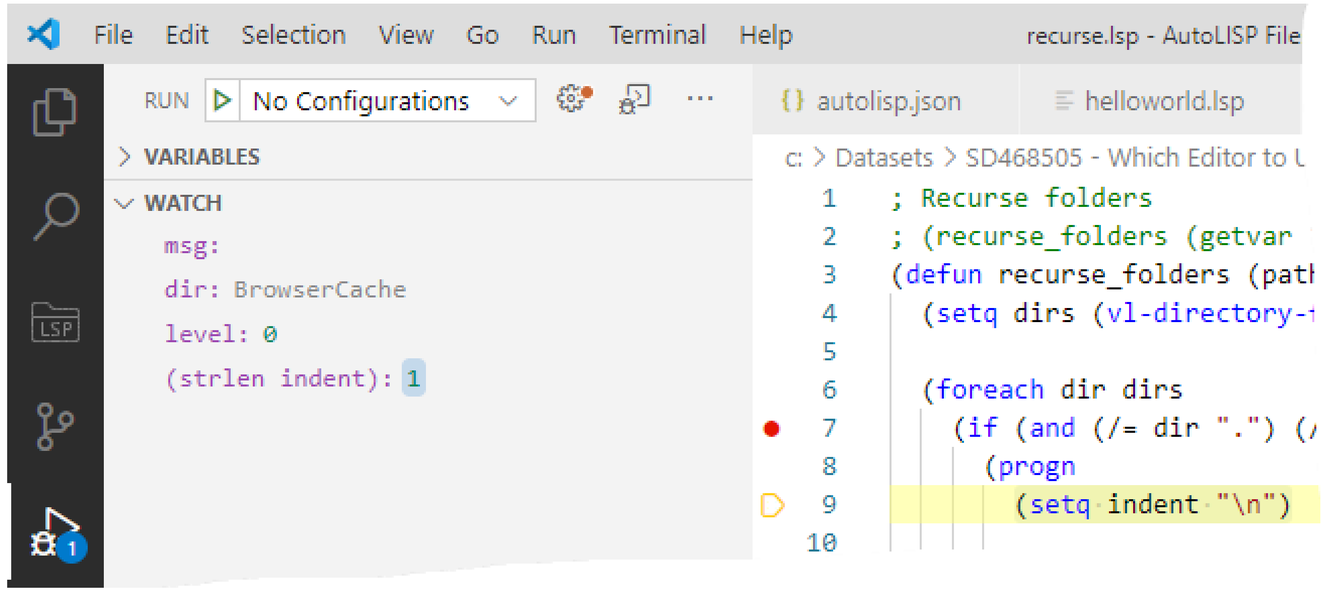
- Breakpoints
- Break on Error
- Local variables and last statement value



Debug LSP Files

You can use these debugging features on a LSP file

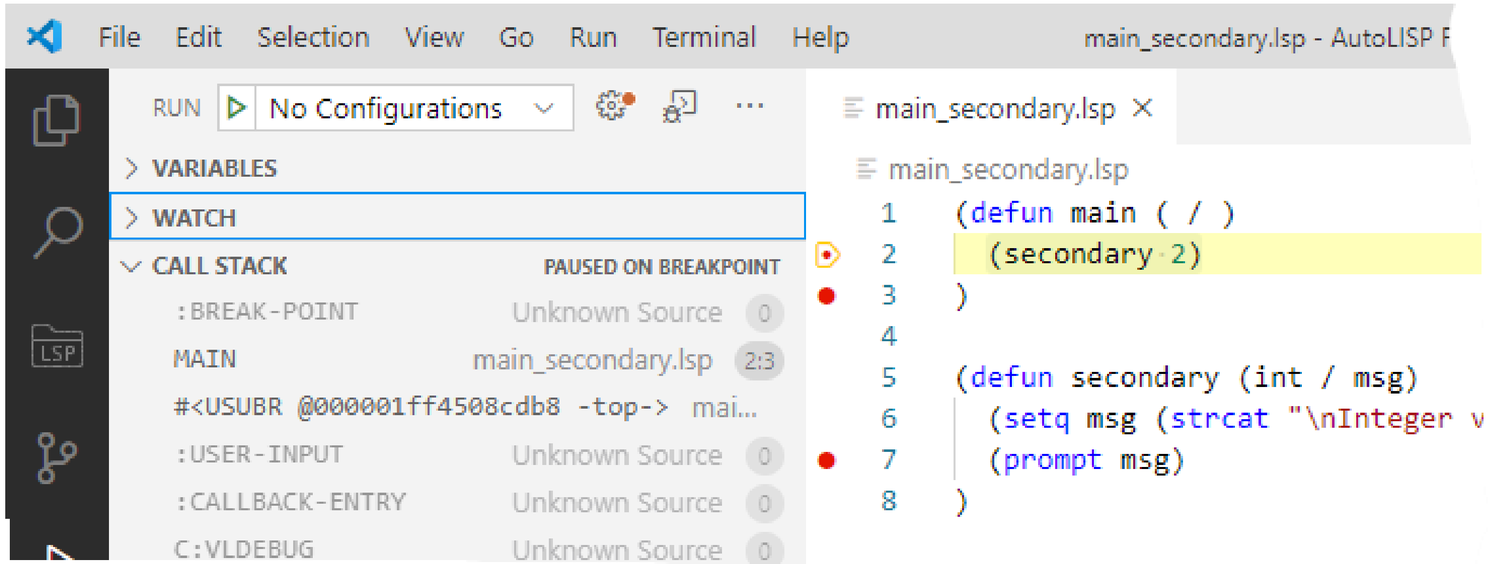
- Breakpoints
- Break on Error
- Local variables and last statement value
- Watches



Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error
- Local variables and last statement value
- Watches
- Call stack



Debug LSP Files

You can use these debugging features on a LSP file

- Breakpoints
- Break on Error
- Local variables and last statement value
- Watches
- Call stack

DEMO

Other Extensions

Extensions can be installed to improve editing functions

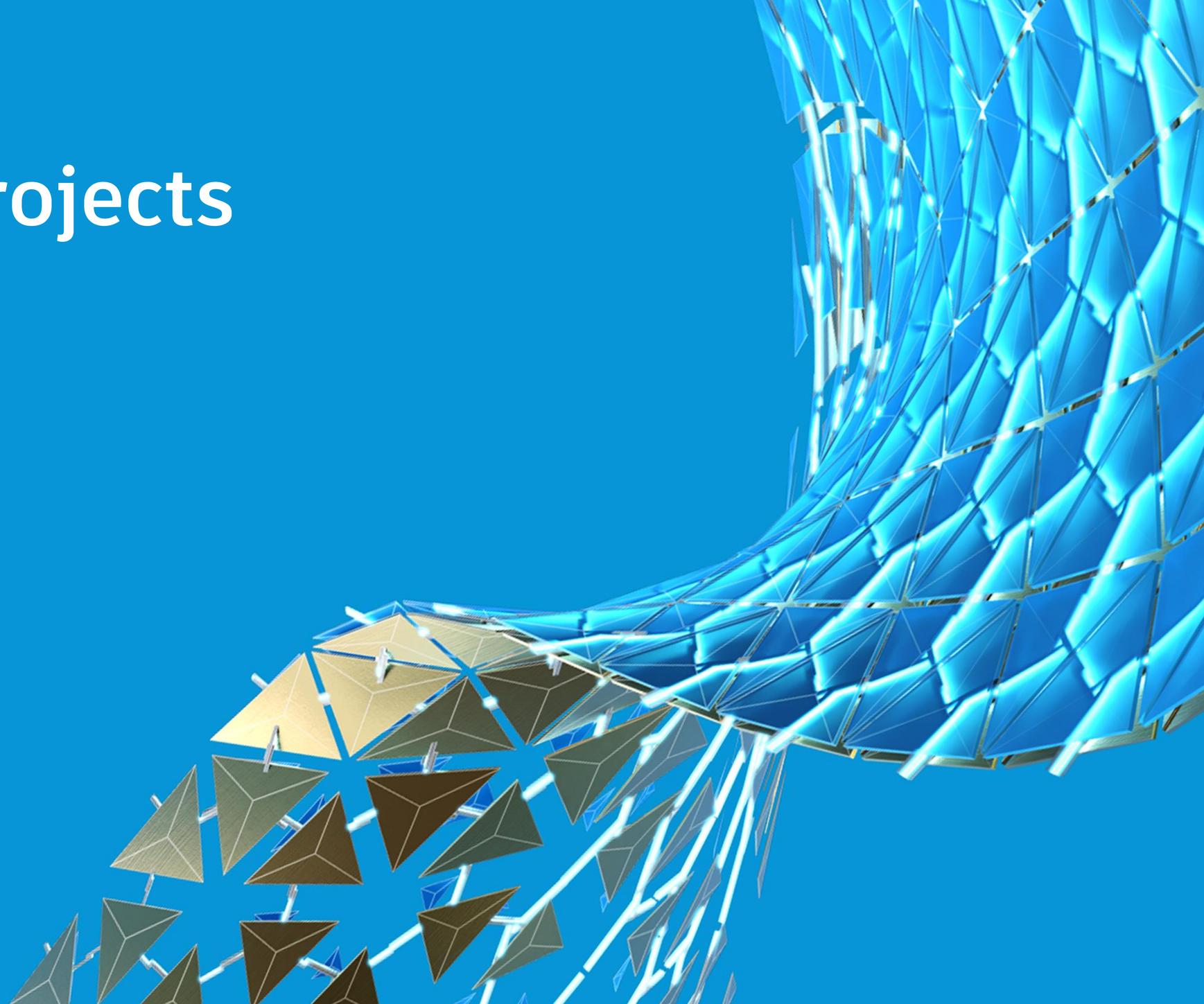
- Bracket Pair Colorizer 2 – Different parentheses coloring by nesting levels

```
; Recurse folders
; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
(defun recurse_folders (path level / indent dir dirs)
  (setq dirs (vl-directory-files path nil -1))

  (foreach dir dirs
    (if (and (/= dir ".") (/= dir ".."))
      (progn
        (setq indent "\n")

        (repeat level
          (setq indent (strcat indent "  "))
        )
```

```
; Recurse folders
; (recurse_folders (getvar "LOCALROOTPREFIX") 0)
(defun recurse_folders (path level / indent dir dirs)
  (setq dirs (vl-directory-files path nil -1))

  (foreach dir dirs
    (if (and (/= dir ".") (/= dir ".."))
      (progn
        (setq indent "\n")

        (repeat level
          (setq indent (strcat indent "  "))
        )
```

- Quick and Simple Text Selection – Improved selection of elements between parentheses
- vscode-position – Shows character position of the cursor in the status bar

AutoLISP Projects

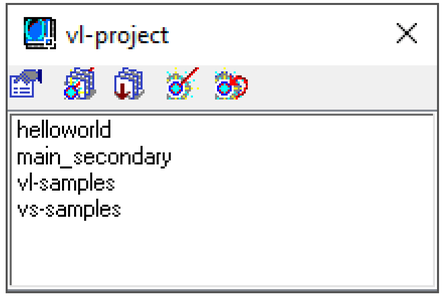


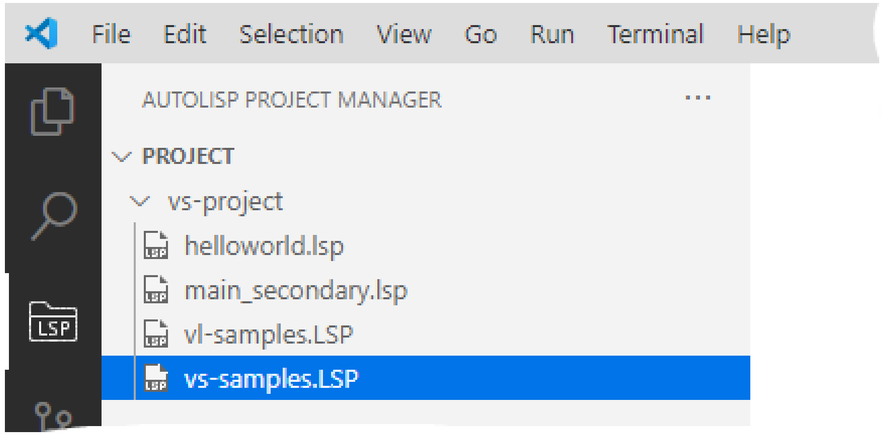
AutoLISP Projects

Allow you to

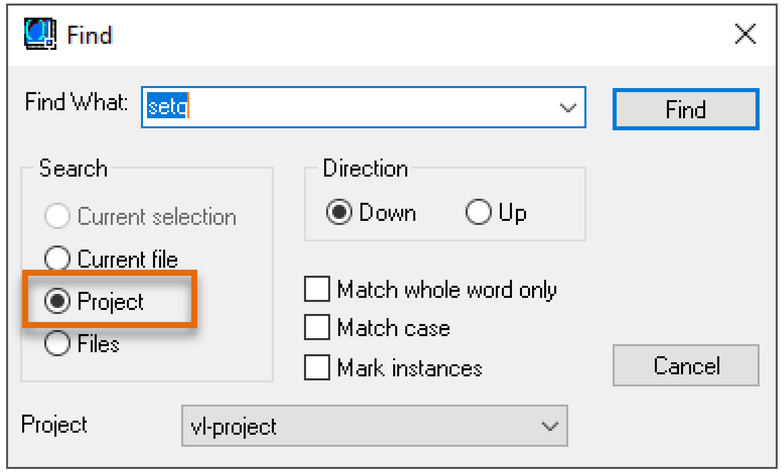
- Manage LSP files
- Open a LSP file
- Find/replace text
- Compile and protect LSP files

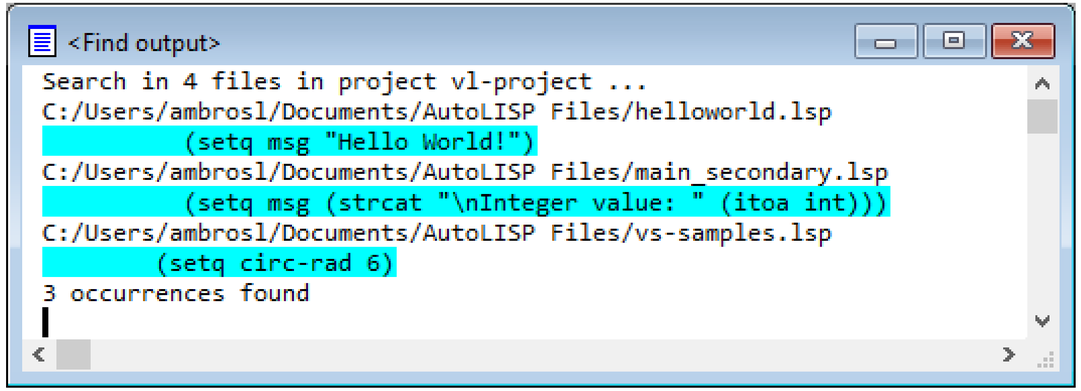
Manage and Open LSP Files

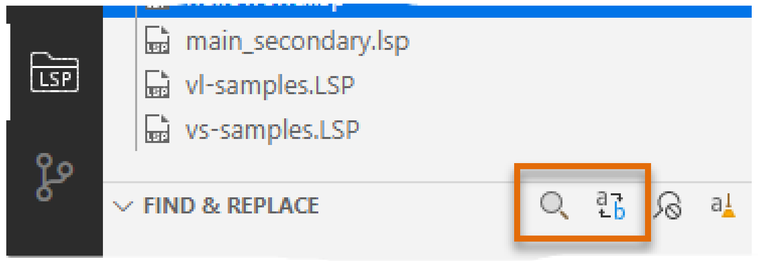


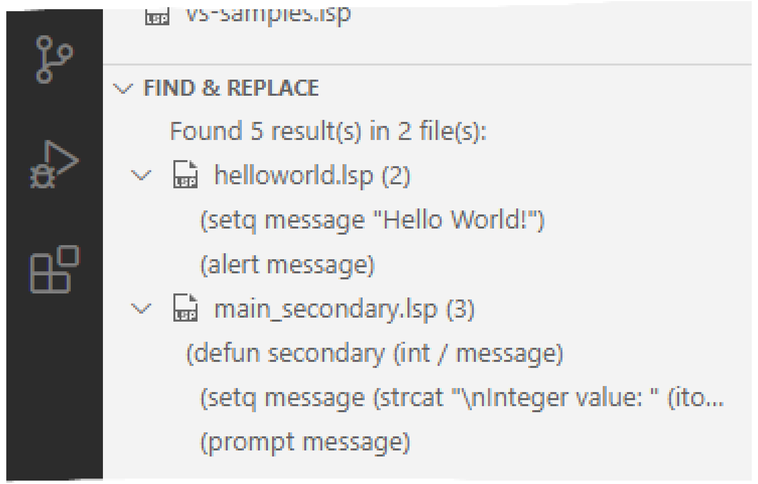


Find and Replace Text in LSP Files









Compile and Protect LSP Files

LSP files can be compiled into

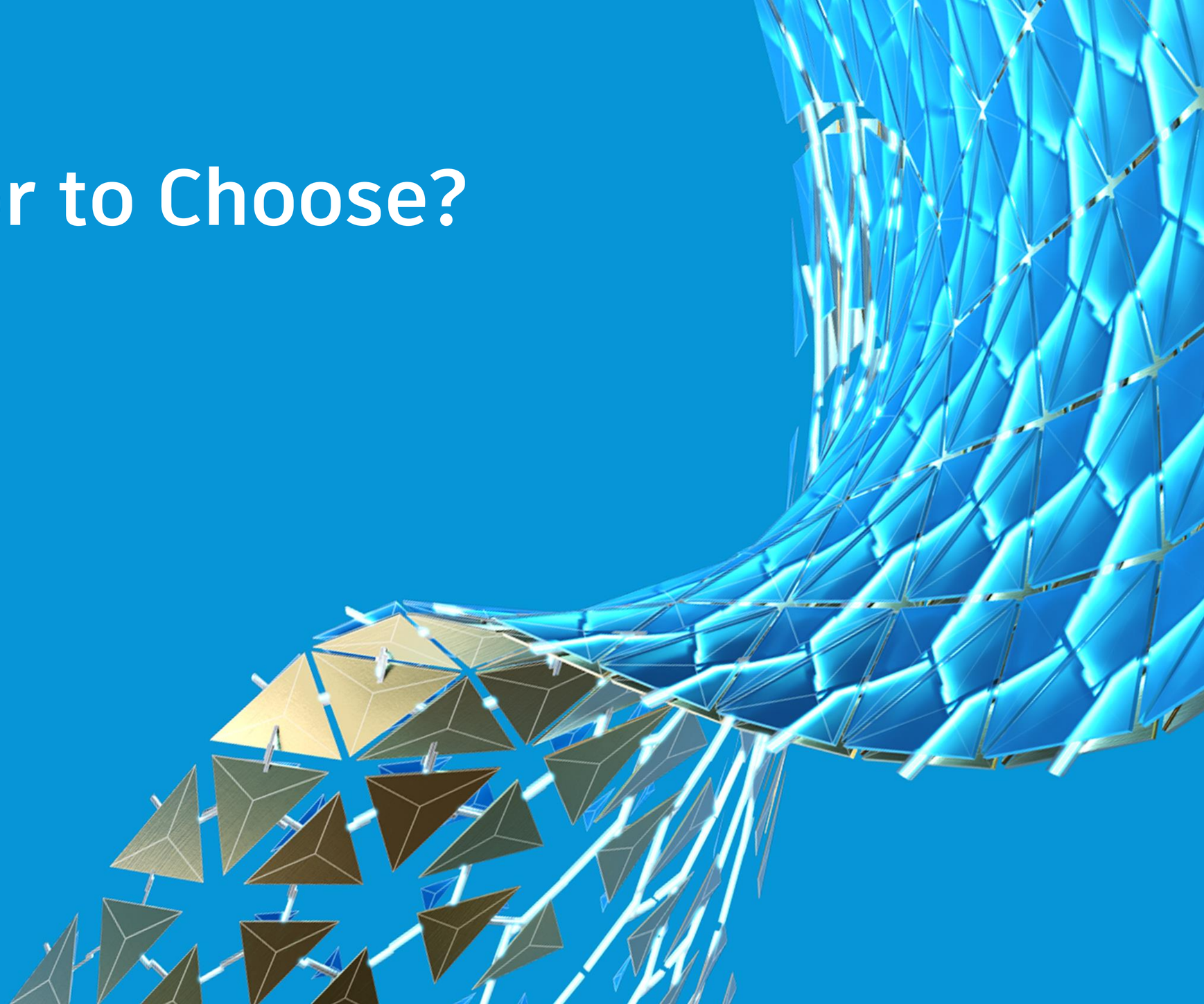
- FAS – FASt load AutoLISP files (Windows and Mac OS)
- VLX – Visual LISP Executable file (Windows only)

AutoLISP projects can make it easier to compile multiple LSP files

These approaches can be used to compile LSP files

- In Visual LISP, File menu > Make Application > New Application Wizard
- MAKELISPAPP command when LISPSYS is set to 1 or 2

Which Editor to Choose?



Which Editor to Choose?

If you are using

- Visual LISP – AutoCAD 2020 or earlier (Windows only)
- VS Code – AutoCAD 2021 (Windows or Mac OS)

VS Code can be used with AutoCAD 2020 or earlier

- Debugging isn’t supported

Which Editor to Choose?

See the handout for a list of editor feature differences

Feature	Visual LISP	AutoLISP Extension for VS Code
Unicode support		X
Autocomplete function/variable name	X	X
Code snippets	/, Save Block As and Insert File	X, snippets file
Auto-indentation while typing	X	X
Smart brackets		X
Find and replace in current editor window	X	X
Find and replace in current AutoLISP Project (PRJ) file	X	X
Toggle comment	X	X
Color-code syntax	X	X

Which Editor to Choose?

Visual LISP IDE to Visual Studio Code Feature Access Comparison in the online help for a transitioning resource

Edit Tools

Feature	Visual LISP IDE	Visual Studio Code
Parentheses Matching	Edit > Parentheses Matching > Match Forward Edit > Parentheses Matching > Match Backward	Go > Go to Bracket Click next to an open or closing parenthesis
Select Matching Parentheses	Edit > Parentheses Matching > Select Forward Edit > Parentheses Matching > Select Backward	Click to the right of an open parenthesis or left of a closing parenthesis, and press Shift+Alt+RightArrow 2 (or 3) times
Comment Block	Edit > Extra Commands > Comment Block	Edit > Toggle Line Comment Edit > Toggle Block Comment
Uncomment Block	Edit > Extra Commands > Uncomment Block	Edit > Toggle Line Comment Edit > Toggle Block Comment
Format Code in Selection	Tools > Format Code in Selection	Right-click > Format Selection
Format Code in Editor	Tools > Format Code in Editor	Right-click > Format Document
Format Settings	Tools > Environment Options > Visual LISP Format Options	File > Preferences > Settings > Extensions > AutoCAD AutoLISP Configuration

Deploy AutoLISP Files

Deploy AutoLISP Files

Independent of the editor you choose, except file format

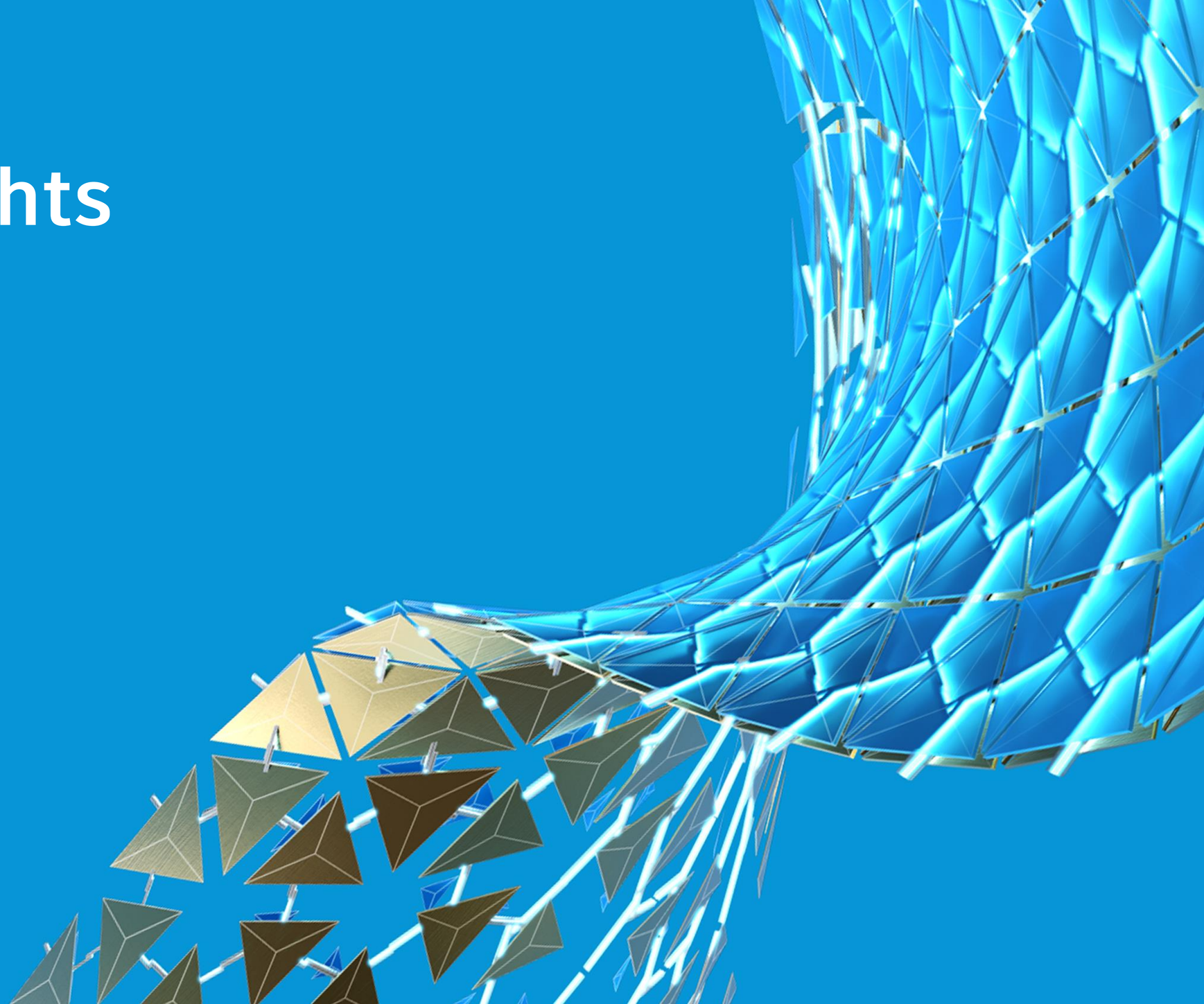
AutoLISP (LSP) files saved in

- Unicode file format can only be loaded in AutoCAD 2021
- ASCII file format can be loaded in AutoCAD 2021 and earlier

FAS or VLX files compiled with AutoCAD 2021

- LISPSYS = 0 or 2 can be loaded in AutoCAD 2021 and earlier
- LISPSYS = 1 can only be loaded in AutoCAD 2021

Final Thoughts



Final Thoughts

Customization and programming can:

- Enhance productivity
- Improve or introduce new workflows

Customizing has many similarities to Wonderland in *Lewis Caroll's Alice's Adventures*.

Both

- Are virtually endless
- Hold many mysteries just waiting to be discovered

Final Thoughts

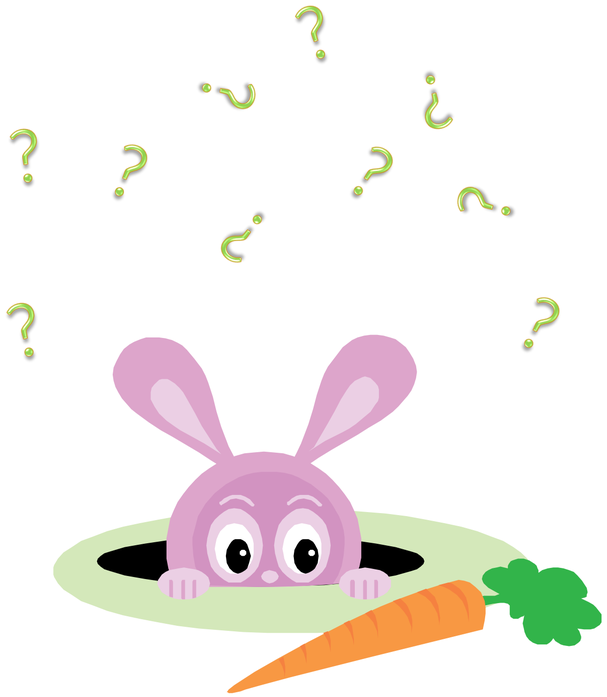
Questions? Questions? Questions?

If you have any further questions, feel free to contact me via

email: lee.ambrosius@autodesk.com

twitter: @leeambrosius

Thanks for watching this session!



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2020 Autodesk. All rights reserved.