Speare Code Editor
The Small Python IDE for Python Development

Version 0.0.1
Update: 4 Mar 2020

Free, Lightweight, Open Source, Extendable Flexibility

Speare (http://www.sevenuc.com/en/speare.html) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++, Python and Django. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language in it.

For general information of debugging C and C++ extension for Python with Speare code editor, please refer this document:
http://www.sevenuc.com/download/Small_IDE_for_C_and_C++_development.pdf

Debug Python Projects

1. Show the debug toolbar

Click main menu "Navigate" → "Toggle Output".

2. Debug toolbar

From left to right: Start, Stop, Step Into, Step Out, Run To/Continue, Step Over/Step Next, Show Watches. The command "Run To" tell the debugger run to meet a breakpoint or an exception occurred or the program exited. On the rightmost there

are three other icons, they are, search items in the stackview, siding stackview, and clean debug output.

Search in the debug output

Click in the output area to let it got focus and use the shortcut key "Command + F" to do the searching.

3. Socket Port

You can set the socket communication port number both used by Debug Server and the editor. Open the Preferences of Speare and select the "Extra" panel and then input your number.

Note: Please remember to empty the port number when you switched to debugging with the default built-in programming languages with default port number.

4. Watches

Watch List: click debug toolbar "Show Watches" to manage watched variables and expressions, batches of watched items can be removed by multiple selection.

a. Whenever values of watched variables changed, debug session will pause at that point.
b. The values of evaluated expressions will be sent as a *"virtual stack node"* that paired the expression and its value together and representing as JSON key and value.

When watched variables and expressions display in stackview, their nodes normally has a different icon and text colour, and always placed on the top of stackview.

Add Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Watch Variable".

Remove Watches:

a. Click debug toolbar "Show Watches".
b. Right click stackview and select menu item "Remove Watch".

5. Run Extra Commands

a. Click main menu "Debug" → "Send Debug Command".

b. Right click stackview and select menu item "Send Command". When send an extra debug command, the input box will prompt in stackview, so please click stackview to let it got focus before select menu item.

6. Show Stackview Item Values

Right click stackview and select menu item "View Value AS", variables can be configured to display as "Hex", "Decimal", "Octal", "UTF-8" and "Unicode" sequences.

Caution:

Please ensure all source files have been dragged in the workspace before start a debug session, because macOS app can't be allowed to access files outside of its sandbox.

Startup Debugging Session

1. Launch a Terminal.app window or tab, and start Debug Server.
Note: *The content directly printed by the debugging program will be displayed in the terminal instead of in the debug "output view".*

2. Click start button on the "Debug toolbar" to start debugging session.

3. Continually click "Step Over" button several times on the "Debug toolbar" to ignore some initialising steps in debugging session until it reached the main entry point.

Python Debugger

The Python debugger of Speare code editor supports Python version 2.5, 2.6, 2.7 and 3.x, and MicroPython. You can enjoy debugging Python scripts as same as debugging web applications that based on web frameworks such as Flask and Django under the lightweight environment of Speare code editor.

Steps of Start Debugging Session:

1. Download Speare Debug Server:
http://sevenuc.com/download/python_debugger.tar.gz (30KB)

The source code of the Python debugger can be directly customised to meet special requirements of your projects.

2. Uncompress the tarball
Uncompress it to your local directory. e.g ~/Desktop and take a look at the readme.txt in it.

3. Start the debug server

```
$ cd ~/Desktop/debugger/2.x
$ python server.py
```

4. Debug session start
Click "Start" button on the debug toolbar of Speare code editor.
Add breakpoint, step in, step out, step next, watch stack trace ...

5. Add condition breakpoint
Right click on the breakpoint, on the prompt menu, → select "Condition" and then input expression or use empty string to remove the condition, left click outside of the input box to close it and execute the command. e.g. x>5 means: Pause on the breakpoint only if x>5 is true.

6. Run Extra Commands
Right click in the stackview (bottom left side) and then input extra command when the debugging session paused. Left click anywhere outside of the input box to close it and the command will be directly send to the debug server.

a. Remove all breakpoints
. clear: clear all breakpoints of current file.

b. Stack trace and frame operation
. where: Print stack trace, an arrow indicates the "current frame".
. up [count]: Move stack trace to older frame.
. down [count]: Move stack trace to newer frame.

c. Display argument list
. args: Print the argument list of the current function.

d. Display return value
. retval: Print the return value for the last return of a function.

e. Display value of expression
. p expression: Print the value of the expression.
. pp expression: Pretty-print the value of the expression.
. display expression: Display the value of the expression, Python 3.x only.
. undisplay: Clear all display expressions for the current frame, Python 3.x only.
. undisplay expression: Do not display the expression any more in the current frame, Python 3.x only.

f. Display argument type
. whatis argument: Print the type of the argument.

g. Display source code of object
. source expression: Python 3.x only.

h. Continue execution
. until: continue execution until the line number greater than the current is reached.
. until [lineno]: continue execution until line number greater or equal to the lineno is reached.

i. Add module search path
. basedir directory: insert a directory in sys.path.

Pretty Debug Output Printing

Speare allows keywords highlighting in console output, the colour definition file located in Speare.app/Contents/Resources/console.plist, you can directly add some keywords to support your special debugging output colour scheme.

Note: *valid colour value must a hex string that has seven characters, e.g , right click the editor → select "Colour Picker" then you can fetch such a HTML colour value conveniently.*

Switch Python Interpreter

You can directly switch CPython interpreter to MicroPython or your own self-compiled version of Python, or others such as PyPy, Jython and IronPython.

Extend Speare Code Editor for Python, Python Game and Python Web Application Development

Speare Code Editor can be easily extended to support any type of game engine or web framework that using Python as scripting language such as Flask and Django.

To add scripts to better support Python debugging in Speare code editor, please download the guide document and following the description in it.