

---

## IDACode

IDACode makes it easy to execute and debug Python scripts in your IDA environment without leaving Visual Studio Code. The VS Code extension can be found on the marketplace.

IDACode is still in a very early state and bugs are to be expected. Please open a new issue if you encounter any issues.

### Features

- **Speed:** Quickly create and execute scripts.
- **Debugging:** Attach a Python debugger at any time.
- **Compatibility:** IDACode does not require you to modify your scripts in a specific way. All scripts can be executed from within IDA without changes.
- **Modularity:** IDACode does not make extensive use of safe wrappers for thread synchronization, this allows you to import any module from any path at any given time. Instead IDACode synchronizes the script execution thread with IDA's main thread to avoid performance and unexpected issues.
- **Syncing:** As IDACode uses [debugpy](#) for communication, it syncs the output window naturally with VS Code's output panel.

IDACode supports both Python 2 and Python 3!

### Setup

To set up the dependencies for the IDA plugin run:

```
1 # make sure to use the correct Python version
2 # IDACode supports the latest debugpy as of version 3.0.0, make sure to
  upgrade!
3 python -m pip install --user debugpy tornado
```

Either clone this repository or download a release package from here. [ida.zip](#) reflects the contents of the [ida](#) folder in this repository. Copy all files into IDA's plugin directory.

The next step is to configure your settings to match your environment. Edit [idacode\\_utils/settings.py](#) accordingly:

- **HOST:** This is the host address. This is always `127.0.0.1` unless you want it to be accessible from a remote location. **Keep in mind that this plugin does not make use of authentication.**
- **PORT:** This is the port you want IDA to listen to. This is used for websocket communication between IDA and VS Code.

- 
- **DEBUG\_PORT**: This is the port you want to listen on for incoming debug sessions.
  - **PYTHON**: This is the absolute path to the Python distribution that your IDA setup uses.
  - **LOGGING**: Determines whether the debugger should log into files. This is especially useful when you are running into issues with IDACode. Please submit a new issue if you find anything. The files are always located in your temp directory (e.g. Windows: %TEMP%). The files are called `debugpy.*.log`.

You can now start the plugin by clicking on **IDACode** in the plugins menu.

The VS Code extension is available on the marketplace. To configure the extension please refer to the extension's README.

## Usage

### IDA

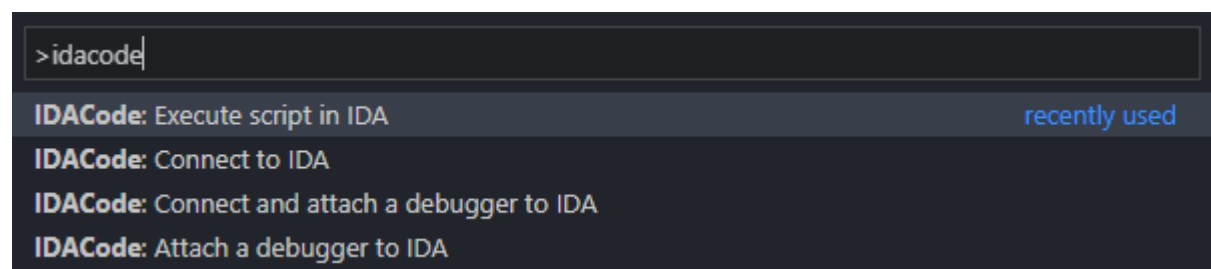
Hit **IDACode** in the plugin menu. You should be greeted with the following text:

```
1 IDACode listening on 127.0.0.1:7065
```

### VS Code

As of version 0.2.0 IDACode supports “Execute on save” which is enabled by default. VS Code will automatically execute your script in IDA as soon as you save the current document (for example with CTRL+S). This behavior can be disabled in the settings.

There are 4 commands at your disposal:



Once you have a folder open that you want to put your scripts in (you **must** specify the folder when VS Code asks you to!) you are ready to connect to IDA. You can do so by either executing **Connect to IDA** or **Connect and attach a debugger to IDA**. Please keep in mind that a debug session is permanent until you restart IDA. You can not change the workspace folder once the debugger has started.

---

Ensure that the workspace folder is the folder that your main scripts are located in. Once you are connected you are able to select [Execute script in IDA](#).

## Debugging

IDACode uses VS Code's remote debugger to connect to IDA. All VS Code features are supported. However, you have to specify the scripts entrypoint by using Python builtin functionality: [breakpoint](#). This instruction tells the debugger to pause execution, if there's no debugger present it will just ignore the function. IDACode imports a helper package called [dbg](#) which implements an overload of [breakpoint](#) called [bp](#). This function supports logging and conditionals:

```
1 name = idc.get_segm_name(segment)
2 dbg.bp(name==".text", f"found {name} at {segment}")
```

Please also note that a [breakpoint\(\)](#) call should never occur at the end of a file, it must always be before any other line of code as it breaks on the *next* instruction in your code. Also note that if you decide to use the [dbg](#) package you must either remove all references or use the variable `__idacode__` as conditional before executing it as a normal IDA script.

It is also important that attaching a debugger will create a new debugger instance. In most cases this is not what you want. If you disconnect from the debugger use VS Code's remote debugger to connect back.

## Known Issues

- Imported module doesn't reload after changes, refer to this for a work around.

The image shows a Windows desktop with two applications open. On the left is a Notepad++ editor window titled 'sub8\_v8\_decrypt.py'. It contains a Python script for decrypting a file. The script defines functions for decrypting a single byte, a string, and the entire file. It uses a key 'I0L0v3Y0u0V1rUs' and a constant 255 for XOR encryption. The file 'off\_405000fo' is decrypted and saved as 'off\_405000fo\_decrypt.py'. On the right is the Immunity Debugger window, showing a memory dump at address 00405000. The dump displays hexadecimal values in the left column and their ASCII representations in the right column. The ASCII column shows the decrypted string 'I.am.now.far.far.away.com' and a file path 'data:lpLibFileName'. The debugger interface includes a menu bar, a toolbar, and a status bar at the bottom indicating the current address and disassembly.

```
1 from ida_bytes import *
2 from helpers import *
3
4 def decrypt(key, data):
5     out = ''
6     for d in data:
7         for k in key:
8             d = d ^ ord(k)
9             out += chr((d) & 255)
10    return out
11
12 def decrypt_string(ea):
13     key = 'I0L0v3Y0u0V1rUs'
14     data = b''
15     for _ in range(500):
16         b = get_bytes(ea, 1)
17         if ord(b) == 0:
18             break
19         data += b
20     ea += 1
21     out = decrypt(key, data)
22     return out
23
24 def decrypt_all(ea, ea_end):
25     while ea <= ea_end:
26         b = get_bytes(ea, 1)
27         if ord(b) == 0:
28             ea += 1
29             continue
30         new_str = decrypt_string(ea)
31         if is_ascii(new_str):
32             print(new_str)
33             replace_string(ea, new_str)
34             ea = len(new_str)
35
36 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

File Edit View Search Window Debugger Lambda Options Windows Help

Stack View x Peasbook x New View-1 x Stackview x External console x Lambda Window x

```
00405000 db 0 ; DATA XREF: .data:off_405000fo
00405001 db 0
00405002 db 0
00405003 db 0
00405004 db 0
00405005 db 0
00405006 db 0
00405007 db 0
00405008 db 0
00405009 db 0
0040500A db 0
0040500B db 0
0040500C db 0
0040500D db 0
0040500E db 0
0040500F db 0
00405010 db 0
00405011 db 0
00405012 db 0
00405013 db 0
00405014 db 0
00405015 db 0
00405016 db 0
00405017 db 0
00405018 db 0
00405019 db 0
0040501A db 0
0040501B db 0
0040501C db 0
0040501D db 0
0040501E db 0
0040501F db 0
00405020 db 0
00405021 db 0
00405022 db 0
00405023 db 0
00405024 db 0
00405025 db 0
00405026 db 0
00405027 db 0
00405028 db 0
00405029 db 0
0040502A db 0
0040502B db 0
0040502C db 0
0040502D db 0
0040502E db 0
0040502F db 0
00405030 db 0
00405031 db 0
00405032 db 0
00405033 db 0
00405034 db 0
00405035 db 0
00405036 db 0
00405037 db 0
00405038 db 0
00405039 db 0
0040503A db 0
0040503B db 0
0040503C db 0
0040503D db 0
0040503E db 0
0040503F db 0
00405040 db 0
00405041 db 0
00405042 db 0
00405043 db 0
00405044 db 0
00405045 db 0
00405046 db 0
00405047 db 0
00405048 db 0
00405049 db 0
0040504A db 0
0040504B db 0
0040504C db 0
0040504D db 0
0040504E db 0
0040504F db 0
00405050 db 0
00405051 db 0
00405052 db 0
00405053 db 0
00405054 db 0
00405055 db 0
00405056 db 0
00405057 db 0
00405058 db 0
00405059 db 0
0040505A db 0
0040505B db 0
0040505C db 0
0040505D db 0
0040505E db 0
0040505F db 0
00405060 db 0
00405061 db 0
00405062 db 0
00405063 db 0
00405064 db 0
00405065 db 0
00405066 db 0
00405067 db 0
00405068 db 0
00405069 db 0
0040506A db 0
0040506B db 0
0040506C db 0
0040506D db 0
0040506E db 0
0040506F db 0
00405070 db 0
00405071 db 0
00405072 db 0
00405073 db 0
00405074 db 0
00405075 db 0
00405076 db 0
00405077 db 0
00405078 db 0
00405079 db 0
0040507A db 0
0040507B db 0
0040507C db 0
0040507D db 0
0040507E db 0
0040507F db 0
00405080 db 0
00405081 db 0
00405082 db 0
00405083 db 0
00405084 db 0
00405085 db 0
00405086 db 0
00405087 db 0
00405088 db 0
00405089 db 0
0040508A db 0
0040508B db 0
0040508C db 0
0040508D db 0
0040508E db 0
0040508F db 0
00405090 db 0
00405091 db 0
00405092 db 0
00405093 db 0
00405094 db 0
00405095 db 0
00405096 db 0
00405097 db 0
00405098 db 0
00405099 db 0
0040509A db 0
0040509B db 0
0040509C db 0
0040509D db 0
0040509E db 0
0040509F db 0
004050A0 db 0
004050A1 db 0
004050A2 db 0
004050A3 db 0
004050A4 db 0
004050A5 db 0
004050A6 db 0
004050A7 db 0
004050A8 db 0
004050A9 db 0
004050AA db 0
004050AB db 0
004050AC db 0
004050AD db 0
004050AE db 0
004050AF db 0
004050B0 db 0
004050B1 db 0
004050B2 db 0
004050B3 db 0
004050B4 db 0
004050B5 db 0
004050B6 db 0
004050B7 db 0
004050B8 db 0
004050B9 db 0
004050BA db 0
004050BB db 0
004050BC db 0
004050BD db 0
004050BE db 0
004050BF db 0
004050C0 db 0
004050C1 db 0
004050C2 db 0
004050C3 db 0
004050C4 db 0
004050C5 db 0
004050C6 db 0
004050C7 db 0
004050C8 db 0
004050C9 db 0
004050CA db 0
004050CB db 0
004050CC db 0
004050CD db 0
004050CE db 0
004050CF db 0
004050D0 db 0
004050D1 db 0
004050D2 db 0
004050D3 db 0
004050D4 db 0
004050D5 db 0
004050D6 db 0
004050D7 db 0
004050D8 db 0
004050D9 db 0
004050DA db 0
004050DB db 0
004050DC db 0
004050DD db 0
004050DE db 0
004050DF db 0
004050E0 db 0
004050E1 db 0
004050E2 db 0
004050E3 db 0
004050E4 db 0
004050E5 db 0
004050E6 db 0
004050E7 db 0
004050E8 db 0
004050E9 db 0
004050EA db 0
004050EB db 0
004050EC db 0
004050ED db 0
004050EE db 0
004050EF db 0
004050F0 db 0
004050F1 db 0
004050F2 db 0
004050F3 db 0
004050F4 db 0
004050F5 db 0
004050F6 db 0
004050F7 db 0
004050F8 db 0
004050F9 db 0
004050FA db 0
004050FB db 0
004050FC db 0
004050FD db 0
004050FE db 0
004050FF db 0
00405100 db 0
00405101 db 0
00405102 db 0
00405103 db 0
00405104 db 0
00405105 db 0
00405106 db 0
00405107 db 0
00405108 db 0
00405109 db 0
0040510A db 0
0040510B db 0
0040510C db 0
0040510D db 0
0040510E db 0
0040510F db 0
00405110 db 0
00405111 db 0
00405112 db 0
00405113 db 0
00405114 db 0
00405115 db 0
00405116 db 0
00405117 db 0
00405118 db 0
00405119 db 0
0040511A db 0
0040511B db 0
0040511C db 0
0040511D db 0
0040511E db 0
0040511F db 0
004051
```

- mrexodia
- MeitarR
- Plutoberth
- OevreFlataeker
- RolfRolles