

# IDA Python: User scripting for a complex application

Gergely Erdélyi

23 April 2008

**D-DOME.NET**

# Agenda

- Introduction
- IDA Pro
- Python
- Integrating Python into IDA Pro
- The IDAPython project
- Conclusion

# Introduction

- Software is getting more and more complex
- Yet, users want even more
- Application scripting can help

# IDA Pro

- Interactive Disassembler Pro
- Advanced disassembler application
- Extensive processor and file format support
- Programmable with built-in language IDC
- Extensible through plugins

Toolbar area containing icons for file operations (save, open, print), navigation (back, forward), and various tool-specific functions. A dropdown menu is currently set to 'Text'.

Navigation tabs for different views: IDA View-A, Hex View-A, Exports, Imports, Names, Functions, Strings, Structures, and Enums.

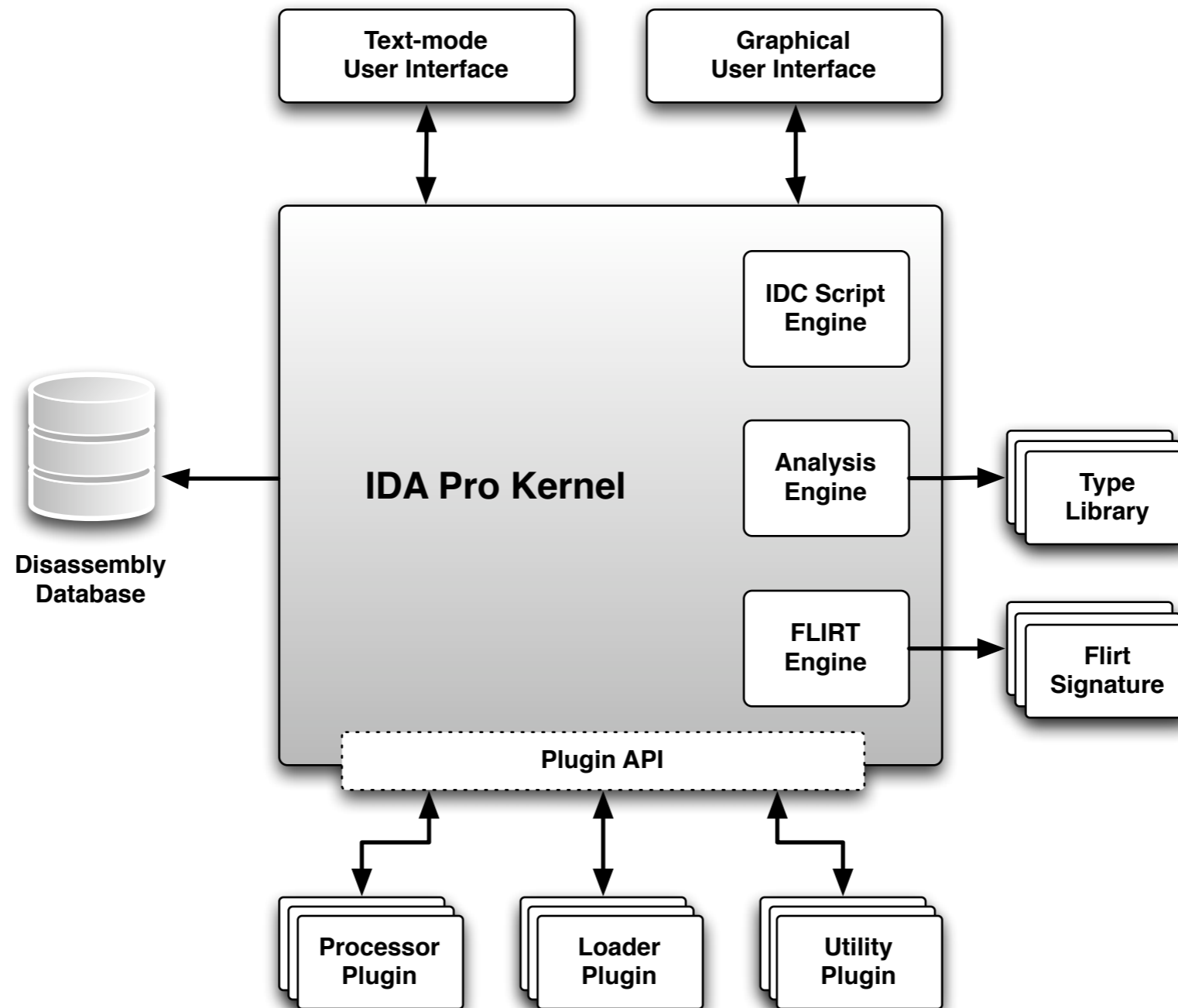
```

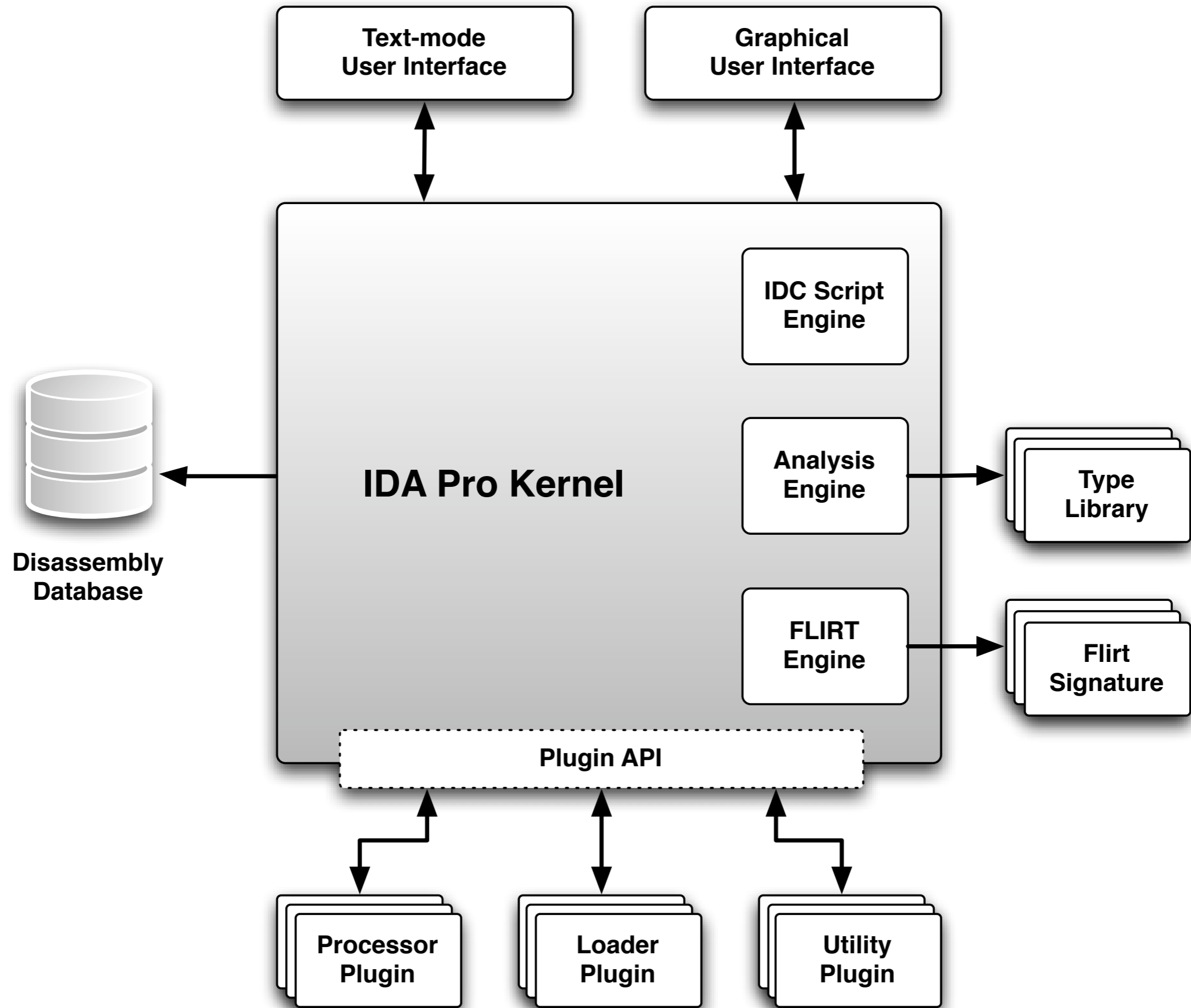
IDA View-A
.text:0040E6F0 var_5 = byte ptr -5
.text:0040E6F0 var_4 = dword ptr -4
.text:0040E6F0 arg_0 = dword ptr 4
.text:0040E6F0 arg_4 = dword ptr 8
.text:0040E6F0
* .text:0040E6F0 mov eax, [esp+arg_4]
* .text:0040E6F4 sub esp, 8
* .text:0040E6F7 push ebx
* .text:0040E6F8 push ebp ; int
* .text:0040E6F9 push esi ; int
* .text:0040E6FA push edi ; int
* .text:0040E6FB mov edi, [eax]
* .text:0040E6FD push 0
* .text:0040E6FF inc edi
* .text:0040E700 lea ecx, [edi+0Bh]
* .text:0040E703 push ecx
* .text:0040E704 push edi
* .text:0040E705 call sub_40E060
    
```

```

Executing function 'main'...
-----
IDAPython version 0.9.56 final (serial 0) initialized
Python interpreter version 2.5.0 final (serial 0)
-----
    
```

# IDA Pro Architecture



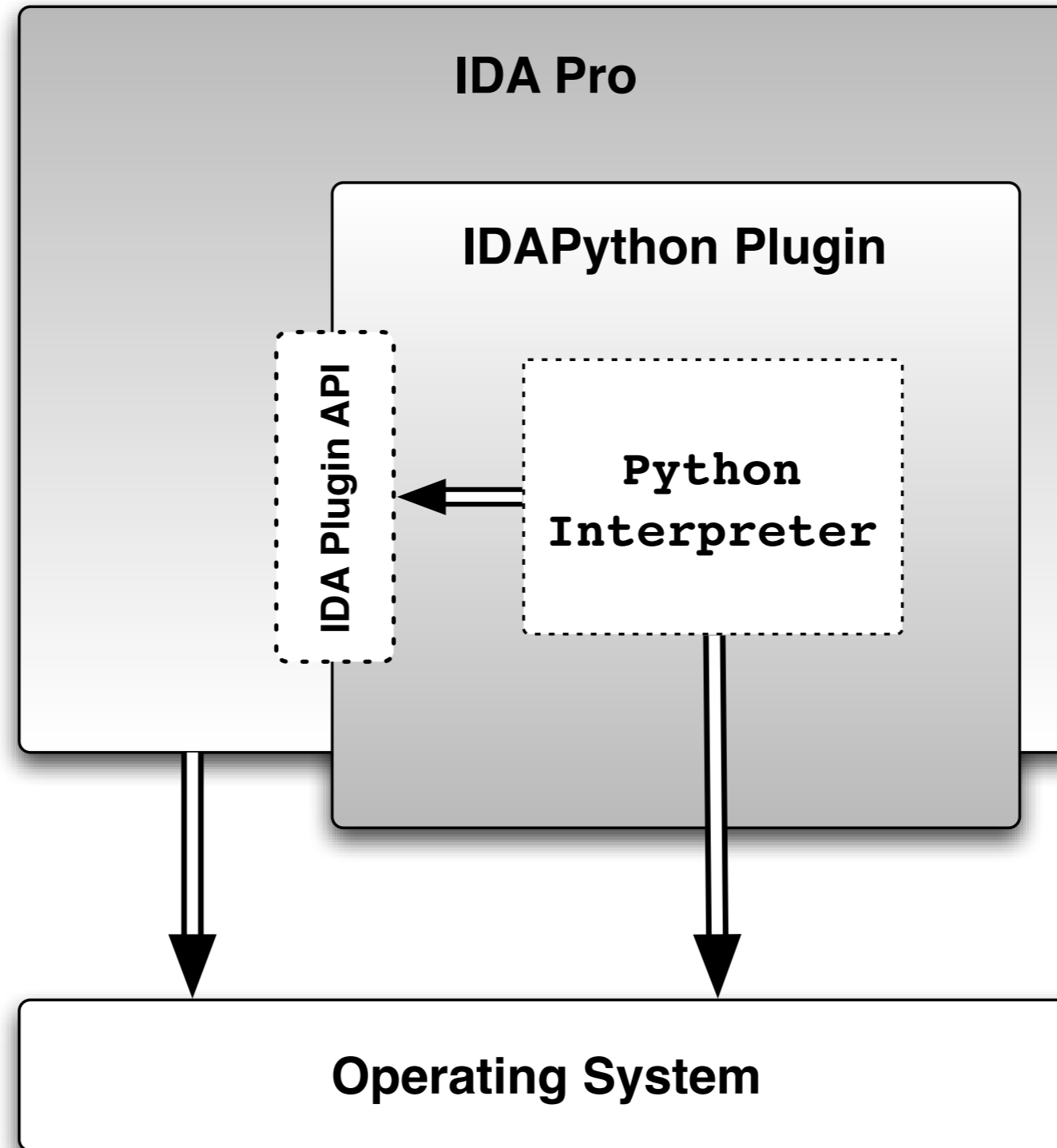


# The Python Language

- High-level scripting language
- Dynamically typed, garbage collected
- Imperative, functional and object-oriented
- Runs on all major operating systems
- Batteries included
- Freely available



# Embedding Python into IDA Pro



# Basic Function Wrapping

## **C function:**

```
inline ea_t idaapi get_item_head(ea_t ea);
```

## **Python version:**

```
def get_item_head(ea_t ea):  
    ...
```

```
a = get_item_head(0x1234)
```

# Calling a Wrapped Function

1. Function is called from Python
2. Arguments are converted from Python to C
3. C function is called with converted arguments
4. Return value is converted from C to Python
5. Value returned to Python

# Function Returning a String

## C declaration:

```
char *get_manual_insn(ea_t ea,  
                      char *buf, size_t bufsize);
```

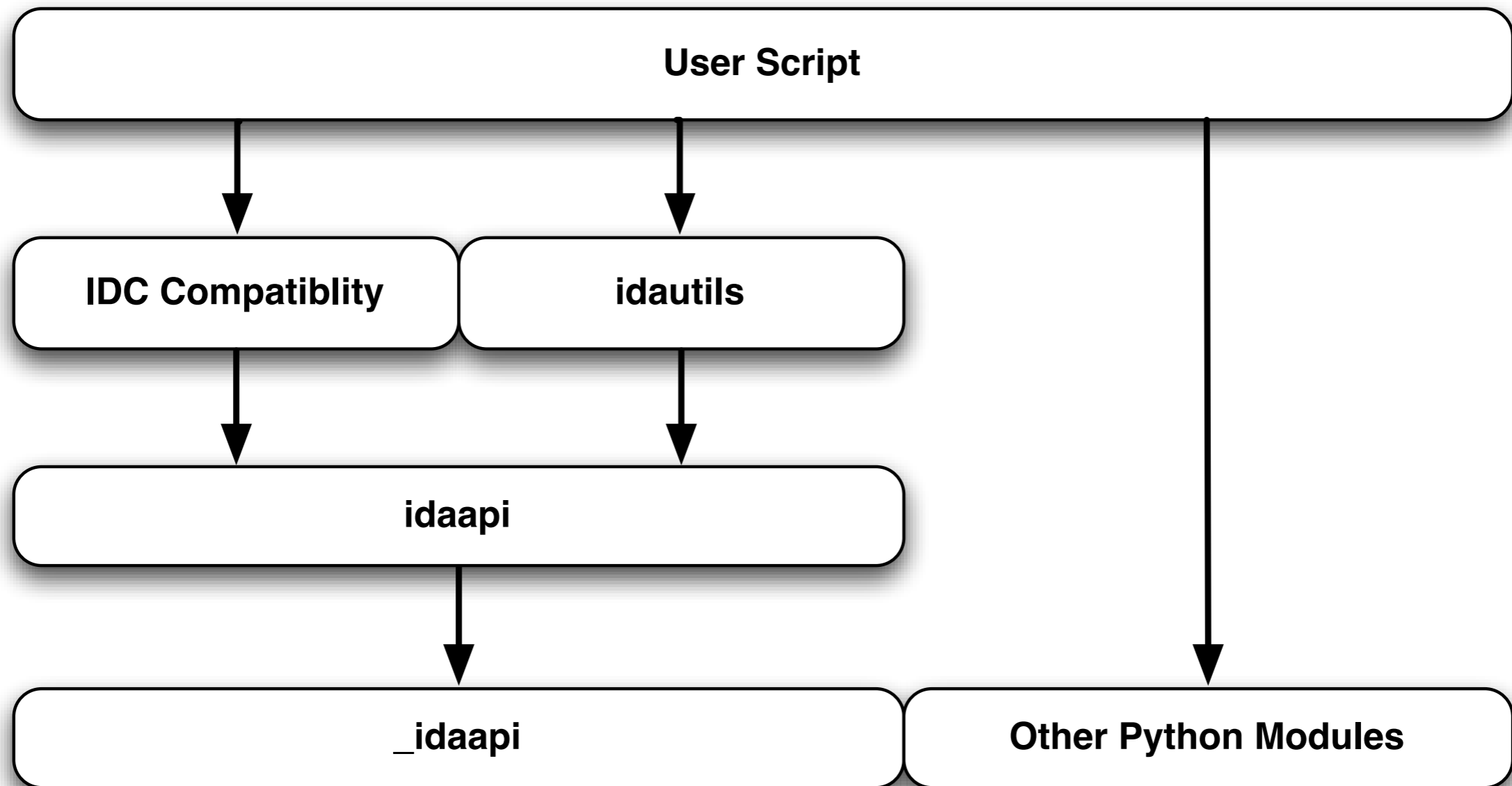
## Python version:

```
def get_manual_insn(ea):  
    ...  
  
ins = get_manual_insn(0x1234)
```

# Introducing SWIG

- Simplified Interface Wrapper Generator
- Automates the process of wrapping
- Interfaces C/C++ code to scripting languages
- Supports over 15 scripting languages
- Has good support for C/C++ features
- Can use C/C++ header directly

# IDA Python Modules



# Simple IDC Script

```
#include <idc.idc>

static main()
{
    auto ea, func, ref;

    // Get current ea
    ea = ScreenEA();
    // Loop from start to end in the current segment
    for (func=SegStart(ea);
        func != BADADDR && func < SegEnd(ea);
        func=NextFunction(func))
    {
        // If the current address is function process it
        if (GetFunctionFlags(func) != -1)
        {
            Message("Function %s at 0x%x\n", GetFunctionName(func), func);
            // Find all code references to func
            for (ref=RfirstB(func); ref != BADADDR; ref=RnextB(func, ref))
            {
                Message("    called from %s(0x%x)\n", GetFunctionName(ref), ref);
            }
        }
    }
}
```

# IDA Python Version

```
from idutils import *

# Get current ea
ea = ScreenEA()

# Loop from start to end in the current segment
for funcea in Functions(SegStart(ea), SegEnd(ea)):
    print "Function %s at 0x%x" \
          (GetFunctionName(funcea), funcea)
# Find all code references to funcea
for ref in CodeRefsTo(funcea, 1):
    print "    called from %s(0x%x)" % \
          (GetFunctionName(ref), ref)
```

# The IDAPython Project

- Project was started in 2004
- Five stable releases so far
- Runs on Windows, Linux and Mac OS X
- Freely available with source under BSD license
- Latest version has seen over 3500 downloads
- Has a small but dedicated user base

# IDAPython in Comparison

	IDC	Binary Plugins	IDAPython
Development model	Interactive and write-debug	Write-compile-link-debug	Interactive and write-debug
User error tolerance	good: only error message	limited: likely crash on user error	good: only error message
Low-level functions	-	~2000	~1050
High-level functions	466	-	475
Operating system access	limited	extensive	extensive
Available libraries	few	many	many

# Availability and Resources

Main distribution site

<http://www.d-dome.net/idapython/>

Development project site

<http://code.google.com/p/idapython/>

Discussion Group

<http://groups.google.com/group/idapython>

# Conclusions

- Scripting makes an application more flexible
- There are many small details to consider
- Implementing application scripting is feasible
- One developer can manage a complex integration
- Open source development is a lot of work
- ...but fun!

# Resources

Main distribution site

<http://www.d-dome.net/idapython/>

Development project site

<http://code.google.com/p/idapython/>

Discussion Group

<http://groups.google.com/group/idapython>