# Module idautils

idautils.py - High level utility functions for IDA

| Classes | |
|---|---|
| | Strings<br>Allows iterating over the string list. |
| | peutils_t<br>PE utility class. |

| Functions | |
|---|---|
| | refs(ea, funcfirst, funcnext)<br>Generic reference collector - INTERNAL USE ONLY. |
| | CodeRefsTo(ea, flow)<br>Get a list of code references to 'ea' |
| | CodeRefsFrom(ea, flow)<br>Get a list of code references from 'ea' |
| | DataRefsTo(ea)<br>Get a list of data references to 'ea' |
| | DataRefsFrom(ea)<br>Get a list of data references from 'ea' |
| | XrefTypeName(typecode)<br>Convert cross-reference type codes to readable names |
| | XrefsFrom(ea, flags=0)<br>Return all references from address 'ea' |
| | XrefsTo(ea, flags=0)<br>Return all references to address 'ea' |
| | Threads()<br>Returns all thread IDs |
| | Heads(start=None, end=None)<br>Get a list of heads (instructions or data) |
| | Functions(start=None, end=None)<br>Get a list of functions |
| | Chunks(start)<br>Get a list of function chunks |
| | Modules()<br>Returns a list of module objects with name,size,base and the rebase_to attributes |
| | Names()<br>Returns a list of names |
| | Segments()<br>Get list of segments (sections) in the binary image |
| | Entries()<br>Returns a list of entry points |
| | FuncItems(start)<br>Get a list of function items |
| | Structs()<br>Get a list of structures |
| | StructMembers(sid)<br>Get a list of structure members information (or stack vars if given a frame). |
| | DecodePrecedingInstruction(ea)<br>Decode preceding instruction in the execution flow. |
| | DecodePreviousInstruction(ea)<br>Decodes the previous instruction and returns an insn_t like class |
| | DecodeInstruction(ea)<br>Decodes an instruction and returns an insn_t like class |
| | GetDataList(ea, count, itemsize=1)<br>Get data list - INTERNAL USE ONLY |
| | PutDataList(ea, datalist, itemsize=1)<br>Put data list - INTERNAL USE ONLY |
| | MapDataList(ea, length, func, wordsize=1)<br>Map through a list of data words in the database |
| | GetInputFileMD5()<br>Return the MD5 hash of the input binary file |
| | GetIdbDir()<br>Get IDB directory |
| | GetRegisterList()<br>Returns the register list |
| | GetInstructionList()<br>Returns the instruction list of the current processor module |
| | Assemble(ea, line)<br>Assembles one or more lines (does not display an message dialogs) If line is a list then this function will attempt to assemble all the lines This |

| | |
|---|---|
| | function will turn on batch mode temporarily so that no messages are displayed on the screen |
| | ProcessUiActions(actions, flags=0)<br>Returns: Boolean. |

| Variables | |
|---|---|
| | cpu = <idautils._cpu object><br>This is a special class instance used to access the registers as if they were attributes of this object. |
| | procregs = <idautils._procregs object><br>This object is used to access the processor registers. |
| | __package__ = None |

Function Details

### CodeRefsTo(ea, flow)

Get a list of code references to 'ea'

Parameters:

- **ea** - Target address
- **flow** (Boolean (0/1, False/True)) - Follow normal code flow or not

Returns:
list of references (may be empty list)

Example:

```
for ref in CodeRefsTo(get_screen_ea(), 1):
    print ref
```

### CodeRefsFrom(ea, flow)

Get a list of code references from 'ea'

Parameters:

- **ea** - Target address
- **flow** (Boolean (0/1, False/True)) - Follow normal code flow or not

Returns:
list of references (may be empty list)

Example:

```
for ref in CodeRefsFrom(get_screen_ea(), 1):
    print ref
```

### DataRefsTo(ea)

Get a list of data references to 'ea'

Parameters:

- **ea** - Target address

Returns:
list of references (may be empty list)

Example:

```
for ref in DataRefsTo(get_screen_ea()):
    print ref
```

### DataRefsFrom(ea)

Get a list of data references from 'ea'

Parameters:

- **ea** - Target address

Returns:
list of references (may be empty list)

Example:

```
for ref in DataRefsFrom(get_screen_ea()):
    print ref
```

### XrefTypeName(typecode)

Convert cross-reference type codes to readable names

Parameters:

- **typecode** - cross-reference type code

### XrefsFrom(ea, flags=0)

Return all references from address 'ea'

Parameters:

- **ea** - Reference address
- **flags** - any of ida_xref.XREF_* flags

Example:

```
for xref in XrefsFrom(here(), 0):
    print xref.type, XrefTypeName(xref.type),                        'from', hex(xref.frm), 'to', hex(xref.to)
```

### XrefsTo(ea, flags=0)

Return all references to address 'ea'

Parameters:

- **ea** - Reference address
- **flags** - any of ida_xref.XREF_* flags

Example:

```
for xref in XrefsTo(here(), 0):
    print xref.type, XrefTypeName(xref.type),                        'from', hex(xref.frm), 'to', hex(xref.to)
```

### Heads(start=None, end=None)

Get a list of heads (instructions or data)

Parameters:

- **start** - start address (default: inf.min_ea)
- **end** - end address (default: inf.max_ea)

Returns:
list of heads between start and end

### Functions(start=None, end=None)

Get a list of functions

Parameters:

- **start** - start address (default: inf.min_ea)
- **end** - end address (default: inf.max_ea)

Returns:
list of heads between start and end

**Note:** The last function that starts before 'end' is included even if it extends beyond 'end'. Any function that has its chunks scattered in multiple segments will be reported multiple times, once in each segment as they are listed.

### Chunks(start)

Get a list of function chunks

Parameters:

- **start** - address of the function

Returns:
list of funcion chunks (tuples of the form (start_ea, end_ea)) belonging to the function

### Names()

Returns a list of names

Returns:
List of tuples (ea, name)

### Segments()

Get list of segments (sections) in the binary image

Returns:
List of segment start addresses.

### Entries()

Returns a list of entry points

Returns:
List of tuples (index, ordinal, ea, name)

### FuncItems(start)

Get a list of function items

Parameters:

- **start** - address of the function

Returns:
ea of each item in the function

### Structs()

Get a list of structures

Returns:
List of tuples (idx, sid, name)

### StructMembers(sid)

Get a list of structure members information (or stack vars if given a frame).

Parameters:

- **sid** - ID of the structure.

Returns:
List of tuples (offset, name, size)

**Notes:**

- If 'sid' does not refer to a valid structure, an exception will be raised.
- This will not return 'holes' in structures/stack frames; it only returns defined structure members.

### DecodePrecedingInstruction(ea)

Decode preceding instruction in the execution flow.

Parameters:

- **ea** - address to decode

Returns:
(None or the decode instruction, farref) farref will contain 'true' if followed an xref, false otherwise

### DecodePreviousInstruction(ea)

Decodes the previous instruction and returns an insn_t like class

Parameters:

- **ea** - address to decode

Returns:
None or a new insn_t instance

### DecodeInstruction(ea)

Decodes an instruction and returns an insn_t like class

Parameters:

- **ea** - address to decode

Returns:
None or a new insn_t instance

### MapDataList(ea, length, func, wordsize=1)

Map through a list of data words in the database

Parameters:

- **ea** - start address
- **length** - number of words to map
- **func** - mapping function
- **wordsize** - size of words to map [default: 1 byte]

Returns:
None

### GetInputFileMD5()

Return the MD5 hash of the input binary file

Returns:
MD5 string or None on error

### GetIdbDir()

Get IDB directory

This function returns directory path of the current IDB database

### Assemble(ea, line)

Assembles one or more lines (does not display an message dialogs) If line is a list then this function will attempt to assemble all the lines This function will turn on batch mode temporarily so that no messages are displayed on the screen

Parameters:

- **ea** - start address

Returns:
(False, "Error message") or (True, asm_buf) or (True, [asm_buf1, asm_buf2, asm_buf3])

### ProcessUiActions(actions, flags=0)

Parameters:

- **actions** - A string containing a list of actions separated by semicolon, a list or a tuple
- **flags** - flags to be passed to process_ui_action()

Returns:
Boolean. Returns False if the action list was empty or execute_ui_requests() failed.

## Variables Details

### cpu

```
This is a special class instance used to access the registers as if they were attributes of this object.
For example to access the EAX register:
    print "%x" % cpu.Eax
```

Value:

```
<idautils._cpu object>
```

**procregs**

```
This object is used to access the processor registers. It is useful when decoding instructions and you want to see which instruction is which.
For example:
    x = idautils.DecodeInstruction(here())
    if x[0] == procregs.Esp:
        print "This operand is the register ESP
```

Value:

```
<idautils._procregs object>
```



| Home | Products | Support | Forum | Blog | News | About us | Contact | Site Map |
Copyright (c) 2017 Hex-Rays SA. Terms of use and privacy policy; updated at Thursday, 18-Oct-2018 05:36:38 EDT