@ http://www.voidspace.org.uk/python/weblog/arch_d7_2006_11_18.shtml

| | | |
|---|---|---|
| frame | **f_back** | next outer frame object (this frame's caller) |
| | **f_builtins** | builtins namespace seen by this frame |
| | **f_code** | code object being executed in this frame |
| | **f_exc_traceback** | traceback if raised in this frame, or **None** |
| | **f_exc_type** | exception type if raised in this frame, or **None** |
| | **f_exc_value** | exception value if raised in this frame, or **None** |
| | **f_globals** | global namespace seen by this frame |
| | **f_lasti** | index of last attempted instruction in bytecode |
| | **f_lineno** | current line number in Python source code |
| | **f_locals** | local namespace seen by this frame |
| | **f_restricted** | 0 or 1 if frame is in restricted execution mode |
| | **f_trace** | tracing function for this frame, or **None** |
| | | |
| code | **co_argcount** | number of arguments (not including * or ** args) |
| | **co_cellvars** | tuple containing the names of local variables that are referenced by nested functions |
| | **co_code** | string of raw compiled sequence of bytecode instructions |
| | **co_consts** | tuple of literal constants used in the bytecode |
| | **co_filename** | name of file in which this code object was created |
| | **co_firstlineno** | number of first line in Python source code |
| | **co_flags** | bitmap: 1=optimized \| 2=newlocals \|4=*arg \| 8=**arg |
| | **co_freevars** | tuple containing the names of free variables |
| | **co_lnotab** | encoded mapping of line numbers to bytecode indices |
| | **co_name** | name with which this code object was defined |
| | **co_names** | tuple of names of local variables used by bytecode |
| | **co_nlocals** | number of local variables used by the function (including arguments) |
| | **co_stacksize** | virtual machine stack space required |
| | **co_varnames** | tuple containing the names of the local variables (starting with the argument names) |

```
v = 0
def function1(a, b, c=1):
    w = 2
    x = 3
```

```
    w += 1
    print w
    print x
    print a
    print b
    print c
    print v
    def function2(d, e, f=4):
        ...
    return function2
```

```
Looking at code1.
co_argcount     3

co_cellvars     ('a', 'b', 'c', 'x', 'w')
      * all the names created in the scope of function1, including the arguments
but not the nested function definition

co_code         <binary data>

co_consts       (None, 2, 3, 1, 4, <code object function2>)
      * values used in the function, including the function object function2.
The leading None means the function has no docstring

co_filename     bytcodetest.py
co_firstlineno  2
co_flags        3
co_freevars     ()      => empty
co_lnotab       <binary data>
co_name         function1

co_names        ('w', 'x', 'a', 'b', 'c', 'v', 'function2')
      * all the names used in the scope of function1

co_nlocals      6
co_stacksize    7
co_varnames     ('a', 'b', 'c', 'function2', 'w', 'x')
      * all the local variables. It includes the nested function (which is
local), but not 'v' which is global

---
In Python, and in a given scope, a free variable is one that is defined in
some outer scope and a cell variable is one that is referenced in some inner
scope (i.e. is a free var for some inner scope).



In a more formal setting, globals would be free variables too.
```