Cell Objects

"Cell" objects are used to implement variables referenced by multiple scopes. For each such variable, a cell object is created to store the value; the local variables of each stack frame that references the value contains a reference to the cells from outer scopes which also use that variable. When the value is accessed, the value contained in the cell is used instead of the cell object itself. This de-referencing of the cell object requires support from the generated byte-code; these are not automatically de-referenced when accessed. Cell objects are not likely to be useful elsewhere.

`PyCellObject`

The C structure used for cell objects.

PyTypeObject **`PyCell_Type`**

The type object corresponding to cell objects.

int **`PyCell_Check`**(ob)

Return true if *ob* is a cell object; *ob* must not be *NULL*.

PyObject* **`PyCell_New`**(PyObject **ob*)

Return value: New reference.

Create and return a new cell object containing the value *ob*. The parameter may be *NULL*.

PyObject* **`PyCell_Get`**(PyObject **cell*)

Return value: New reference.

Return the contents of the cell *cell*.

PyObject* **`PyCell_GET`**(PyObject **cell*)

Return value: Borrowed reference.

Return the contents of the cell *cell*, but without checking that *cell* is non-*NULL* and a cell object.

int **`PyCell_Set`**(PyObject **cell*, PyObject **value*)

Set the contents of the cell object *cell* to *value*. This releases the reference to any current content of the cell. *value* may be *NULL*. *cell* must be non-*NULL*; if it is not a cell object, `-1` will be returned. On success, `0` will be returned.

void **`PyCell_SET`**(PyObject **cell*, PyObject **value*)

Sets the value of the cell object *cell* to *value*. No reference counts are adjusted, and no checks are made for safety; *cell* must be non-*NULL* and must be a cell object.