Tuple Objects

`PyTupleObject`

This subtype of `PyObject` represents a Python tuple object.

PyTypeObject **`PyTuple_Type`**

This instance of `PyTypeObject` represents the Python tuple type; it is the same object as `tuple` in the Python layer.

int **`PyTuple_Check`**(PyObject **p*)

Return true if *p* is a tuple object or an instance of a subtype of the tuple type.

int **`PyTuple_CheckExact`**(PyObject **p*)

Return true if *p* is a tuple object, but not an instance of a subtype of the tuple type.

PyObject* **`PyTuple_New`**(Py_ssize_t *len*)

Return value: New reference.

Return a new tuple object of size *len*, or *NULL* on failure.

PyObject* **`PyTuple_Pack`**(Py_ssize_t *n*, ...)

Return value: New reference.

Return a new tuple object of size *n*, or *NULL* on failure. The tuple values are initialized to the subsequent *n* C arguments pointing to Python objects. `PyTuple_Pack(2, a, b)` is equivalent to `Py_BuildValue("(OO)", a, b)`.

Py_ssize_t **`PyTuple_Size`**(PyObject **p*)

Take a pointer to a tuple object, and return the size of that tuple.

Py_ssize_t **`PyTuple_GET_SIZE`**(PyObject **p*)

Return the size of the tuple *p*, which must be non-*NULL* and point to a tuple; no error checking is performed.

PyObject* **`PyTuple_GetItem`**(PyObject **p*, Py_ssize_t *pos*)

Return value: Borrowed reference.

Return the object at position *pos* in the tuple pointed to by *p*. If *pos* is out of bounds, return *NULL* and sets an `IndexError` exception.

PyObject* **`PyTuple_GET_ITEM`**(PyObject **p*, Py_ssize_t *pos*)

Return value: Borrowed reference.

Like `PyTuple_GetItem()`, but does no checking of its arguments.

PyObject* **`PyTuple_GetSlice`**(PyObject **p*, Py_ssize_t *low*, Py_ssize_t *high*)

Return value: New reference.
Take a slice of the tuple pointed to by *p* from *low* to *high* and return it as a new tuple.

int **`PyTuple_SetItem`**(PyObject **p*, Py_ssize_t *pos*, PyObject **o*)

Insert a reference to object *o* at position *pos* of the tuple pointed to by *p*. Return `0` on success.

Note: This function "steals" a reference to *o*.

void **`PyTuple_SET_ITEM`**(PyObject **p*, Py_ssize_t *pos*, PyObject **o*)

Like `PyTuple_SetItem()`, but does no error checking, and should *only* be used to fill in brand new tuples.

Note: This function "steals" a reference to *o*.

int **`_PyTuple_Resize`**(PyObject ***p*, Py_ssize_t *newsize*)

Can be used to resize a tuple. *newsize* will be the new length of the tuple. Because tuples are *supposed* to be immutable, this should only be used if there is only one reference to the object. Do *not* use this if the tuple may already be known to some other part of the code. The tuple will always grow or shrink at the end. Think of this as destroying the old tuple and creating a new one, only more efficiently. Returns `0` on success. Client code should never assume that the resulting value of `*p` will be the same as before calling this function. If the object referenced by `*p` is replaced, the original `*p` is destroyed. On failure, returns `-1` and sets `*p` to *NULL*, and raises `MemoryError` or `SystemError`.

int **`PyTuple_ClearFreeList`**()

Clear the free list. Return the total number of freed items.

Struct Sequence Objects

Struct sequence objects are the C equivalent of `namedtuple()` objects, i.e. a sequence whose items can also be accessed through attributes. To create a struct sequence, you first have to create a specific struct sequence type.

PyTypeObject* **`PyStructSequence_NewType`**
(PyStructSequence_Desc **desc*)

Create a new struct sequence type from the data in *desc*, described below. Instances of the resulting type can be created with `PyStructSequence_New()`.

void **`PyStructSequence_InitType`**(PyTypeObject **type*, PyStructSequence_Desc **desc*)

Initializes a struct sequence type *type* from *desc* in place.

int **`PyStructSequence_InitType2`**(PyTypeObject **type*, PyStructSequence_Desc **desc*)

The same as `PyStructSequence_InitType`, but returns `0` on success and `-1` on failure.

New in version 3.4.

`PyStructSequence_Desc`

Contains the meta information of a struct sequence type to create.

Field	C Type	Meaning
`name`	`char *`	name of the struct sequence type
`doc`	`char *`	pointer to docstring for the type or NULL to omit
`fields`	`PyStructSequence_Field *`	pointer to *NULL*-terminated array with field names of the new type
`n_in_sequence`	`int`	number of fields visible to the Python side (if used as tuple)

`PyStructSequence_Field`

Describes a field of a struct sequence. As a struct sequence is modeled as a tuple, all fields are typed as `PyObject*`. The index in the `fields` array of the `PyStructSequence_Desc` determines which field of the struct sequence is described.

Field	C Type	Meaning
`name`	`char *`	name for the field or *NULL* to end the list of named fields, set to PyStructSequence_UnnamedField to leave unnamed
`doc`	`char *`	field docstring or *NULL* to omit

char* **`PyStructSequence_UnnamedField`**

Special value for a field name to leave it unnamed.

PyObject* **`PyStructSequence_New`**(PyTypeObject **type*)

Creates an instance of *type*, which must have been created with `PyStructSequence_NewType()`.

PyObject* **`PyStructSequence_GetItem`**(PyObject **p*, Py_ssize_t *pos*)

Return the object at position *pos* in the struct sequence pointed to by *p*. No bounds checking is performed.

PyObject* **`PyStructSequence_GET_ITEM`**(PyObject **p*, Py_ssize_t *pos*)

Macro equivalent of `PyStructSequence_GetItem()`.

void **`PyStructSequence_SetItem`**(PyObject **p*, Py_ssize_t *pos*, PyObject **o*)

Sets the field at index *pos* of the struct sequence *p* to value *o*. Like `PyTuple_SET_ITEM()`, this should only be used to fill in brand new instances.

Note: This function "steals" a reference to *o*.

PyObject* **`PyStructSequence_SET_ITEM`**(PyObject **p*, Py_ssize_t **pos*, PyObject **o*)

Macro equivalent of `PyStructSequence_SetItem()`.

Note: This function "steals" a reference to *o*.