Slice Objects

PyTypeObject **PySlice_Type**

The type object for slice objects. This is the same as `slice` in the Python layer.

int **PySlice_Check**(PyObject *ob)

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

PyObject* **PySlice_New**(PyObject *start, PyObject *stop, PyObject *step)

Return value: New reference.

Return a new slice object with the given values. The *start*, *stop*, and *step* parameters are used as the values of the slice object attributes of the same names. Any of the values may be *NULL*, in which case the `None` will be used for the corresponding attribute. Return *NULL* if the new object could not be allocated.

int **PySlice_GetIndices**(PyObject *slice, Py_ssize_t *length*, Py_ssize_t *start, Py_ssize_t *stop, Py_ssize_t *step*)

Retrieve the start, stop and step indices from the slice object *slice*, assuming a sequence of length *length*. Treats indices greater than *length* as errors.

Returns `0` on success and `-1` on error with no exception set (unless one of the indices was not `None` and failed to be converted to an integer, in which case `-1` is returned with an exception set).

You probably do not want to use this function.

Changed in version 3.2: The parameter type for the *slice* parameter was `PySliceObject*` before.

int **PySlice_GetIndicesEx**(PyObject *slice, Py_ssize_t *length*, Py_ssize_t *start, Py_ssize_t *stop, Py_ssize_t *step, Py_ssize_t *slicelength*)

Usable replacement for `PySlice_GetIndices()`. Retrieve the start, stop, and step indices from the slice object *slice* assuming a sequence of length *length*, and store the length of the slice in *slicelength*. Out of bounds indices are clipped in a manner consistent with the handling of normal slices.

Returns `0` on success and `-1` on error with exception set.

Changed in version 3.2: The parameter type for the *slice* parameter was `PySliceObject*` before.

Ellipsis Object

PyObject *__Py_Ellipsis__

The Python `Ellipsis` object. This object has no methods. It needs to be treated just like any other object with respect to reference counts. Like `Py_None` it is a singleton object.