Iterator Objects

Python provides two general-purpose iterator objects. The first, a sequence iterator, works with an arbitrary sequence supporting the `__getitem__()` method. The second works with a callable object and a sentinel value, calling the callable for each item in the sequence, and ending the iteration when the sentinel value is returned.

PyTypeObject **PySeqIter_Type**
: Type object for iterator objects returned by `PySeqIter_New()` and the one-argument form of the `iter()` built-in function for built-in sequence types.

int **PySeqIter_Check**(op)
: Return true if the type of *op* is `PySeqIter_Type`.

PyObject* **PySeqIter_New**(PyObject **seq*)
: *Return value: New reference.*
Return an iterator that works with a general sequence object, *seq*. The iteration ends when the sequence raises `IndexError` for the subscripting operation.

PyTypeObject **PyCallIter_Type**
: Type object for iterator objects returned by `PyCallIter_New()` and the two-argument form of the `iter()` built-in function.

int **PyCallIter_Check**(op)
: Return true if the type of *op* is `PyCallIter_Type`.

PyObject* **PyCallIter_New**(PyObject **callable*, PyObject **sentinel*)
: *Return value: New reference.*
Return a new iterator. The first parameter, *callable*, can be any Python callable object that can be called with no parameters; each call to it should return the next item in the iteration. When *callable* returns a value equal to *sentinel*, the iteration will be terminated.