array — Sequence of Fixed-type Data

> **Purpose:** Manage sequences of fixed-type numerical data efficiently.

The array module defines a sequence data structure that looks very much like a list, except that all of the members have to be of the same primitive type. The types supported are all numeric or other fixed-size primitive types such as bytes.

Refer to the table below for some of the supported types. The standard library documentation for array includes a complete list of type codes.

Type Codes for array Members

Code	Type	Minimum size (bytes)
b	int	1
B	int	1
h	signed short	2
H	unsigned short	2
i	signed int	2
I	unsigned int	2
l	signed long	4
L	unsigned long	4
q	signed long long	8
Q	unsigned long long	8
f	float	4
d	double float	8

Initialization

An array is instantiated with an argument describing the type of data to be allowed, and possibly an initial sequence of data to store in the array.

```
# array_string.py

import array
import binascii

s = b'This is the array.'
a = array.array('b', s)

print('As byte string:', s)
print('As array      :', a)
print('As hex        :', binascii.hexlify(a))
```

In this example, the array is configured to hold a sequence of bytes and is initialized with a simple byte string.

```
$ python3 array_string.py

As byte string: b'This is the array.'
As array      : array('b', [84, 104, 105, 115, 32, 105, 115, 32,
 116, 104, 101, 32, 97, 114, 114, 97, 121, 46])
As hex        : b'54686973206973207468652061727261792e'
```

Manipulating Arrays

An array can be extended and otherwise manipulated in the same ways as other Python sequences.

```
# array_sequence.py

import array
import pprint

a = array.array('i', range(3))
print('Initial :', a)

a.extend(range(3))
print('Extended:', a)

print('Slice   :', a[2:5])

print('Iterator:')
print(list(enumerate(a)))
```

The supported operations include slicing, iterating, and adding elements to the end.

```
$ python3 array_sequence.py

Initial : array('i', [0, 1, 2])
Extended: array('i', [0, 1, 2, 0, 1, 2])
Slice   : array('i', [2, 0, 1])
Iterator:
[(0, 0), (1, 1), (2, 2), (3, 0), (4, 1), (5, 2)]
```

Arrays and Files

The contents of an array can be written to and read from files using built-in methods coded efficiently for that purpose.

```
# array_file.py

import array
import binascii
import tempfile

a = array.array('i', range(5))
print('A1:', a)

# Write the array of numbers to a temporary file
output = tempfile.NamedTemporaryFile()
a.tofile(output.file)  # must pass an *actual* file
output.flush()

# Read the raw data
with open(output.name, 'rb') as input:
    raw_data = input.read()
    print('Raw Contents:', binascii.hexlify(raw_data))

    # Read the data into an array
    input.seek(0)
    a2 = array.array('i')
    a2.fromfile(input, len(a))
    print('A2:', a2)
```

This example illustrates reading the data “raw,” meaning directly from the binary file, versus reading it into a new array and converting the bytes to the appropriate types.

```
$ python3 array_file.py

A1: array('i', [0, 1, 2, 3, 4])
Raw Contents: b'0000000001000000020000000300000004000000'
A2: array('i', [0, 1, 2, 3, 4])
```

`tofile()` uses `tobytes()` to format the data, and `fromfile()` uses `frombytes()` to convert it back to an array instance.

```
# array_tobytes.py
```

```
import array
import binascii

a = array.array('i', range(5))
print('A1:', a)

as_bytes = a.tobytes()
print('Bytes:', binascii.hexlify(as_bytes))

a2 = array.array('i')
a2.frombytes(as_bytes)
print('A2:', a2)
```

Both tobytes() and frombytes() work on byte strings, not Unicode strings.

```
$ python3 array_tobytes.py

A1: array('i', [0, 1, 2, 3, 4])
Bytes: b'0000000001000000020000000300000004000000'
A2: array('i', [0, 1, 2, 3, 4])
```

Alternative Byte Ordering

If the data in the array is not in the native byte order, or if the data needs to be swapped before being sent to a system with a different byte order (or over the network), it is possible to convert the entire array without iterating over the elements from Python.

```
# array_byteswap.py

import array
import binascii


def to_hex(a):
    chars_per_item = a.itemsize * 2  # 2 hex digits
    hex_version = binascii.hexlify(a)
    num_chunks = len(hex_version) // chars_per_item
    for i in range(num_chunks):
        start = i * chars_per_item
        end = start + chars_per_item
        yield hex_version[start:end]


start = int('0x12345678', 16)
end = start + 5
a1 = array.array('i', range(start, end))
a2 = array.array('i', range(start, end))
a2.byteswap()

fmt = '{:>12} {:>12} {:>12} {:>12}'
print(fmt.format('A1 hex', 'A1', 'A2 hex', 'A2'))
print(fmt.format('-' * 12, '-' * 12, '-' * 12, '-' * 12))
fmt = '{!r:>12} {:12} {!r:>12} {:12}'
for values in zip(to_hex(a1), a1, to_hex(a2), a2):
    print(fmt.format(*values))
```

The byteswap() method switches the byte order of the items in the array from within C, so it is much more efficient than looping over the data in Python.

```
$ python3 array_byteswap.py

      A1 hex           A1       A2 hex           A2
------------ ------------ ------------ ------------
 b'78563412'    305419896  b'12345678'   2018915346
 b'79563412'    305419897  b'12345679'   2035692562
 b'7a563412'    305419898  b'1234567a'   2052469778
 b'7b563412'    305419899  b'1234567b'   2069246994
 b'7c563412'    305419900  b'1234567c'   2086024210
```

See also

- Standard library documentation for array
- `struct` – The `struct` module.
- Numerical Python – NumPy is a Python library for working with large data sets efficiently.
- Python 2 to 3 porting notes for array

collections.abc — Abstract Base Classes for Containers

heapq – Heap Sort Algorithm

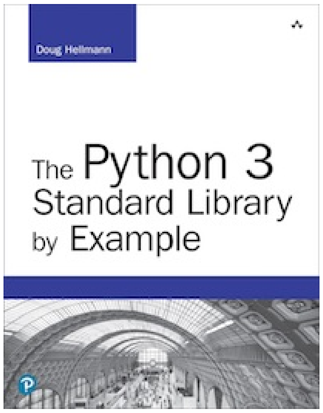
Quick Links

Initialization
Manipulating Arrays
Arrays and Files
Alternative Byte Ordering

This page was last updated 2017-01-28.

Navigation

collections.abc — Abstract Base Classes for Containers
heapq – Heap Sort Algorithm



The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

This Site

Module Index
Index

Other Writing

Blog
The Python Standard Library By Example