What is Python buffer type for?

There is a `buffer` type in python, but I don't know how can I use it.

In the Python doc the description is:

> `buffer(object[, offset[, size]])`
>
> The object argument must be an object that supports the buffer call interface (such as strings, arrays, and buffers). A new buffer object will be created which references the object argument. The buffer object will be a slice from the beginning of object (or from the specified offset). The slice will extend to the end of object (or will have a length given by the size argument).

python python-2.7

edited Sep 1 '14 at 6:08
Arjun J Rao
567 4 22

asked Aug 6 '10 at 9:55
satoru
10.8k 12 47 96

2 Answers

An example usage:

```
>>> s = 'Hello world'
>>> t = buffer(s, 6, 5)
>>> t
<read-only buffer for 0x10064a4b0, size 5, offset 6 at 0x100634ab0>
>>> print t
world
```

The buffer in this case is a sub-string, starting at position 6 with length 5, and it doesn't take extra storage space - it references a slice of the string.

This isn't very useful for short strings like this, but it can be necessary when using large amounts of data. This example uses a mutable `bytearray`:

```
>>> s = bytearray(1000000)   # a million zeroed bytes
>>> t = buffer(s, 1)         # slice cuts off the first byte
>>> s[1] = 5                 # set the second element in s
>>> t[0]                     # which is now also the first element in t!
'\x05'
```

This can be very helpful if you want to have more than one view on the data and don't want to (or can't) hold multiple copies in memory.

Note that `buffer` has been replaced by the better named `memoryview` in Python 3, though you can use either in Python 2.7.

Note also that you can't implement a buffer interface for your own objects without delving into the C API, i.e. you can't do it in pure Python.

edited Aug 6 '16 at 15:17

answered Aug 6 '10 at 10:05
Scott Griffiths
13.6k 5 40 68

Thanks for your explanation. But I still don't quite understand what's the difference between buffering and simple slicing. Using `s[6:11]` doesn't take extra storage space either, am I wrong? – satoru Aug 6 '10 at 11:31

8 In general a slice will take extra storage, so yes `s[6:11]` will be a copy. If you set `t = s[6:11]` and then `del s`, it frees the memory that was taken by `s`, proving that `t` was copied. (To see this you need a bigger `s` and track Python's memory usage). It is however much more efficient just to make the copy if there isn't much data involved. – Scott Griffiths Aug 6 '10 at 12:11

Thank you very mush :) BTW, could you please tell me what tool can I use to track Python's memory usage? – satoru Aug 6 '10 at 12:48

3 @Saturo.Logic: I think you should accept is answer ;-) – Etienne Aug 6 '10 at 15:31

6 For Python noobs like me: buffer is memoryview in Python 3 – Dirk Bester Aug 6 '12 at 8:05

I think buffers are e.g. useful when interfacing python to native libraries. (Guido van Rossum explains `buffer` in this mailinglist post).

For example, numpy seems to use buffer for efficient data storage:

```
import numpy
a = numpy.ndarray(1000000)
```

the `a.data` is a:

```
<read-write buffer for 0x1d7b410, size 8000000, offset 0 at 0x1e353b0>
```

edited Apr 25 at 7:25
Chankey Pathak
13.4k 9 45 92

answered Aug 6 '10 at 19:59
Andre Holzner
11.6k 4 32 47

The mailing list is certainly worth a read. – Robino Nov 7 '16 at 20:09