

PEP 688: Typing for the buffer protocol


Jelle Zijlstra

Quora


The buffer protocol

```
import numpy as np
```

```
def need_buffer(buffer):  
    mv = memoryview(buffer)  
    ...
```

```
need_buffer(b"abc") # 
```

```
need_buffer(np.array(...)) # 
```

```
need_buffer("abc") # 
```


The buffer protocol: Types?

```
import numpy as np
```


```
def need_buffer(buffer: ???) -> None:
```

```
    mv = memoryview(buffer)
```

```
    ...
```

```
need_buffer(b"abc") # 
```

```
need_buffer(np.array(...)) # 
```

```
need_buffer("abc") # 
```

PEP 688 v1: types.Buffer

```
from types import Buffer
```

```
issubclass(bytes, Buffer) # ✓
```

```
issubclass(np.ndarray, Buffer) # ✓
```

```
issubclass(str, Buffer) # ✗
```

```
def need_buffer(buffer: Buffer) -> None:
```

```
    mv = memoryview(buffer)
```

```
    ...
```

PEP 688 v1: Problems!

```
from types import Buffer

# We can't have a Buffer that's also a Protocol
def need_buffer(buffer: Buffer-but-also-Sized?) -> None:
    print(len(buffer))
    mv = memoryview(buffer)
    ...
```

Idea 1: Define buffers in Python!

```
b""".__buffer__(flags) # memoryview
class MyBuffer:
    def __buffer__(self, flags: int) -> memoryview:
        return memoryview(...)
```

Pros:

- PyPy does this
- Useful outside typing

Cons:

- Not what I signed up for
- What about the `bf_releasebuffer` slot?

Idea 2: Just set a flag

```
bytes.__buffer__ # True  
str.__buffer__ # AttributeError
```

Pros:

- Simpler

Cons:

- No other dunder works this way
- Easy to lie

Feedback?

- I'd love to hear it!
- Come find me in person
- PEP 688 thread on discuss.python.org
- jelle@quora.com