What if you did very bad things with integers

or, Through CPython and back again

Andromeda Yelton https://andromedayelton.com @ThatAndromeda

Photo by Blake Cheek on Unsplash

```
>>> a = 1000

>>> b = 1000
```

```
>>> a == b
```

???

```
>>> a = 1000
>>> b = 1000
>>> a == b
```

```
True
```

```
>>> a = 10
```

```
>>> b = 10
```

```
>>> a == b
```

???

```
>>> a = 10
>>> b = 10
>>> a == b
True
```

```
>>> a = 1000
```

```
>>> b = 1000
```

```
>>> a is b
```

???

```
>>> a = 1000
>>> b = 1000
>>> a is b
```

```
False
```

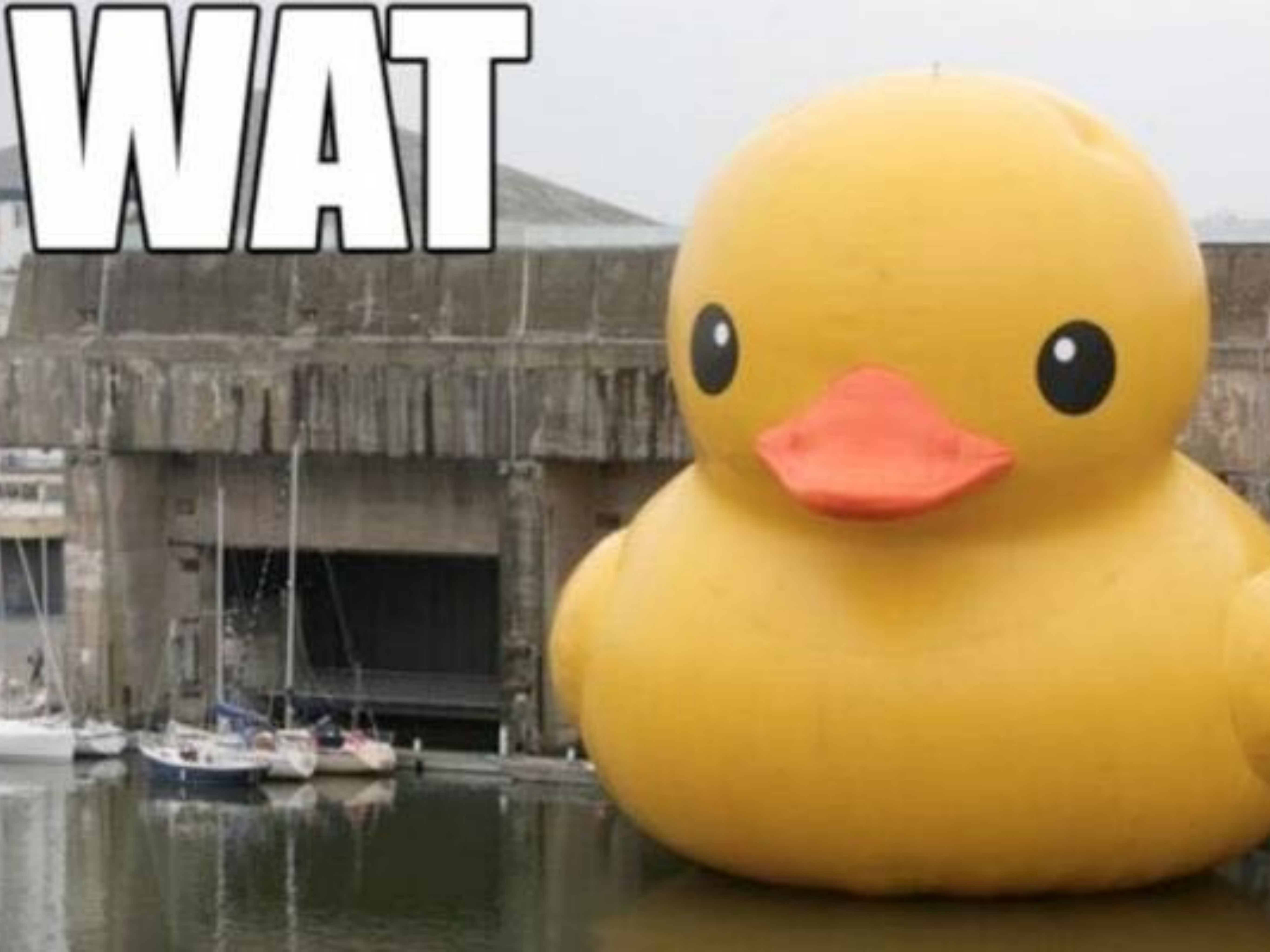
```
>>> a = 10
```

```
>>> b = 10
```

```
>>> a is b
```

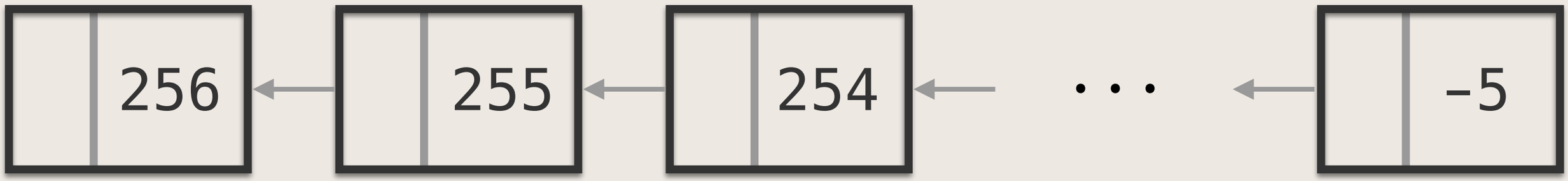
???

```
>>> a = 10
>>> b = 10
>>> a is b
True
```

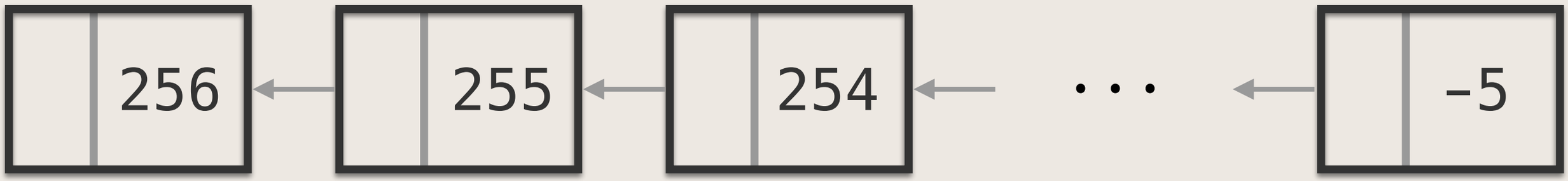

WAT

```c
#if NSMALLNEGINTS + NSMALLPOSINTS > 0
/* Small integers are preallocated in this array so that they
   can be shared.
   The integers that are preallocated are those in the range
   -NSMALLNEGINTS (inclusive) to NSMALLPOSINTS (not inclusive).
*/
static PyLongObject small_ints[NSMALLNEGINTS + NSMALLPOSINTS];
#ifdef COUNT_ALLOCS
Py_ssize_t quick_int_allocs, quick_neg_int_allocs;
#endif

static PyObject *
get_small_int(sdigit ival)
{
    PyObject *v;
    assert(-NSMALLNEGINTS <= ival && ival < NSMALLPOSINTS);
    v = (PyObject *)&small_ints[ival + NSMALLNEGINTS];
    Py_INCREF(v);
#ifdef COUNT_ALLOCS
    if (ival >= 0)
        quick_int_allocs++;
    else
        quick_neg_int_allocs++;
#endif
    return v;
}
```

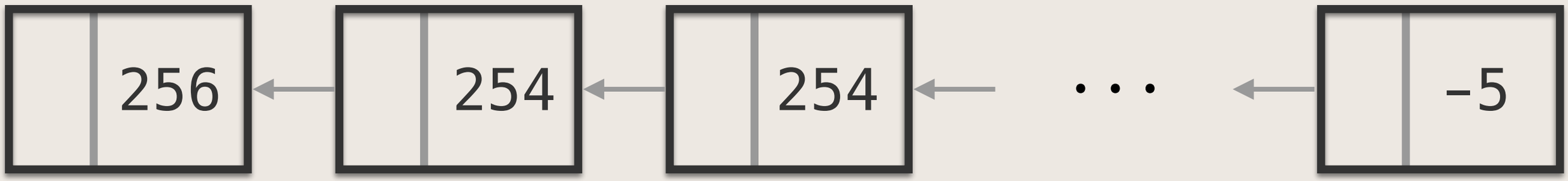
Woo yeah small integers



What if you were evil…?



What if you were evil…?



```
>>> import ctypes
>>> def mutate_int(an_int, new_value):
...     ctypes.memmove(id(an_int) + 24, \
          id(new_value) + 24, 8)
>>> mutate_int(7, 10)
```

thanks, kate.io

```
>>> for i in range(0, 10):
...     print(i)
0
1
2
3
4
5
6
10
8
9
```

Math “works” too

```
>>> 7 + 1

11
```

what is truth…?

```
>>> isinstance(True, int)
True
```

if you redefine 1…

can you redefine `True`?

```
>>> mutate_int(1, 5)
>>> 1
```

Photo by Rye Jessen on Unsplash

your repl
segfaults

Photo by Rye Jessen on Unsplash

```
import ctypes

def mutate_int(an_int, new_value):
    ctypes.memmove(id(an_int) + 24, id(new_value) + 24, 8)

mutate_int(1, 0)

print(False == 1)
print(True == 1)
print(bool(1))
```

```
False
```

```
import ctypes

def mutate_int(an_int, new_value):
    ctypes.memmove(id(an_int) + 24, id(new_value) + 24, 8)

mutate_int(1, 0)

print(False == 1)
print(True == 1)
print(bool(1))
```

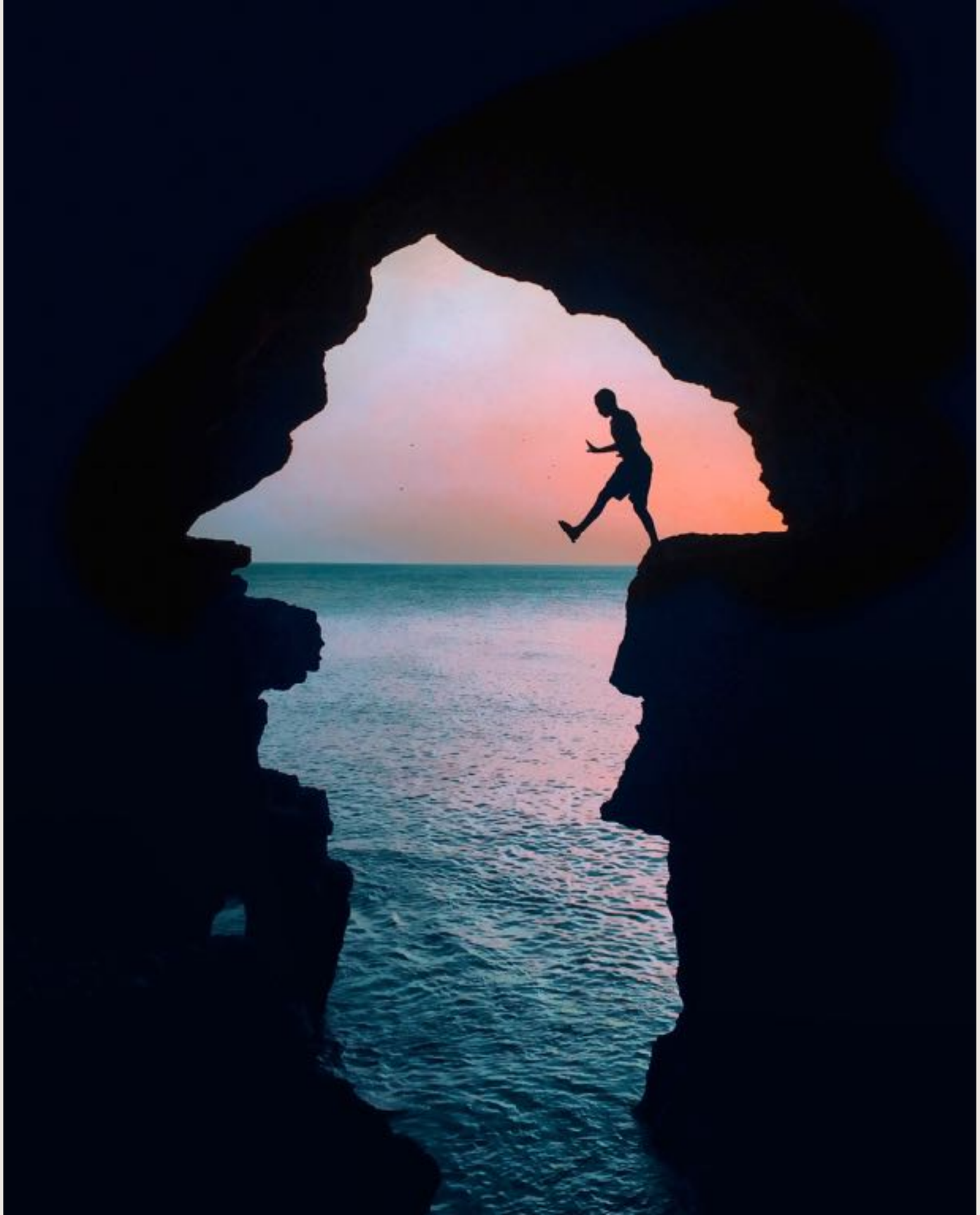
```
False
```

```
False
```

```python
import ctypes

def mutate_int(an_int, new_value):
    ctypes.memmove(id(an_int) + 24, id(new_value) + 24, 8)

mutate_int(1, 0)

print(False == 1)
print(True == 1)
print(bool(1))
```

```
False

False

True
```

```
>>> True == 1
True
>>> True is 1
False
```



we can still fail

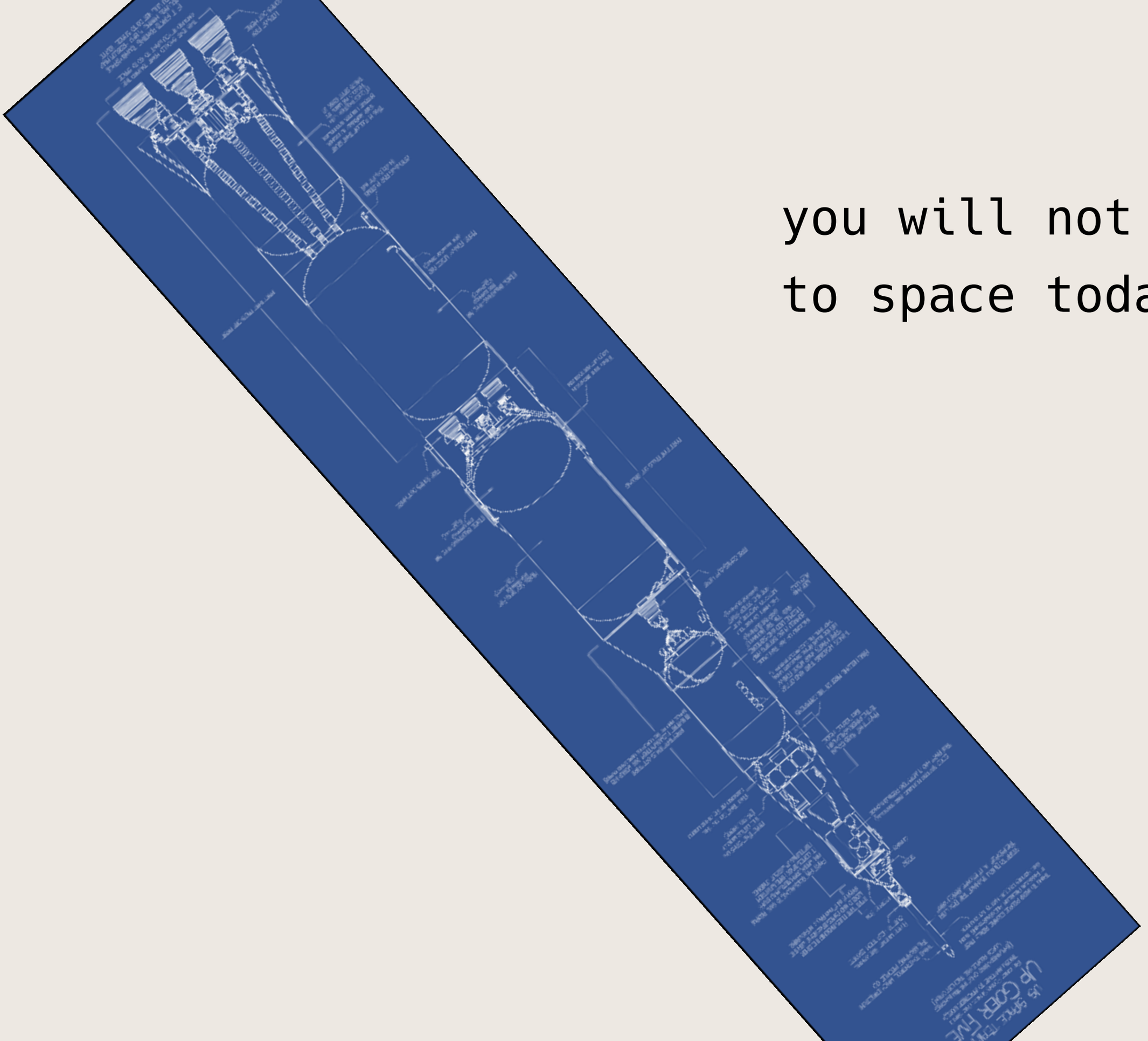
Photo by Meor Mohamad on Unsplash

Failing with Python internals

```
>>> mutate_int(1, 2):
>>> for i in range(0, 5):
...     print(i)
0
2
4
```

Failing with Python internals

```
>>> mutate_int(1, 0):
>>> for i in range(0, 5):
...     print(i)
```



you will not go
to space today :(

Failing by confusing ourselves

```
>>> mutate_int(1, 0):
>>> count = 10
>>> while count < 15:
...     print('infinite loop')
...     count += 1
```

Photo by Madhu Shesharam on Unsplash

Bibliography

https://kate.io/blog/2017/08/22/weird-python-integers/

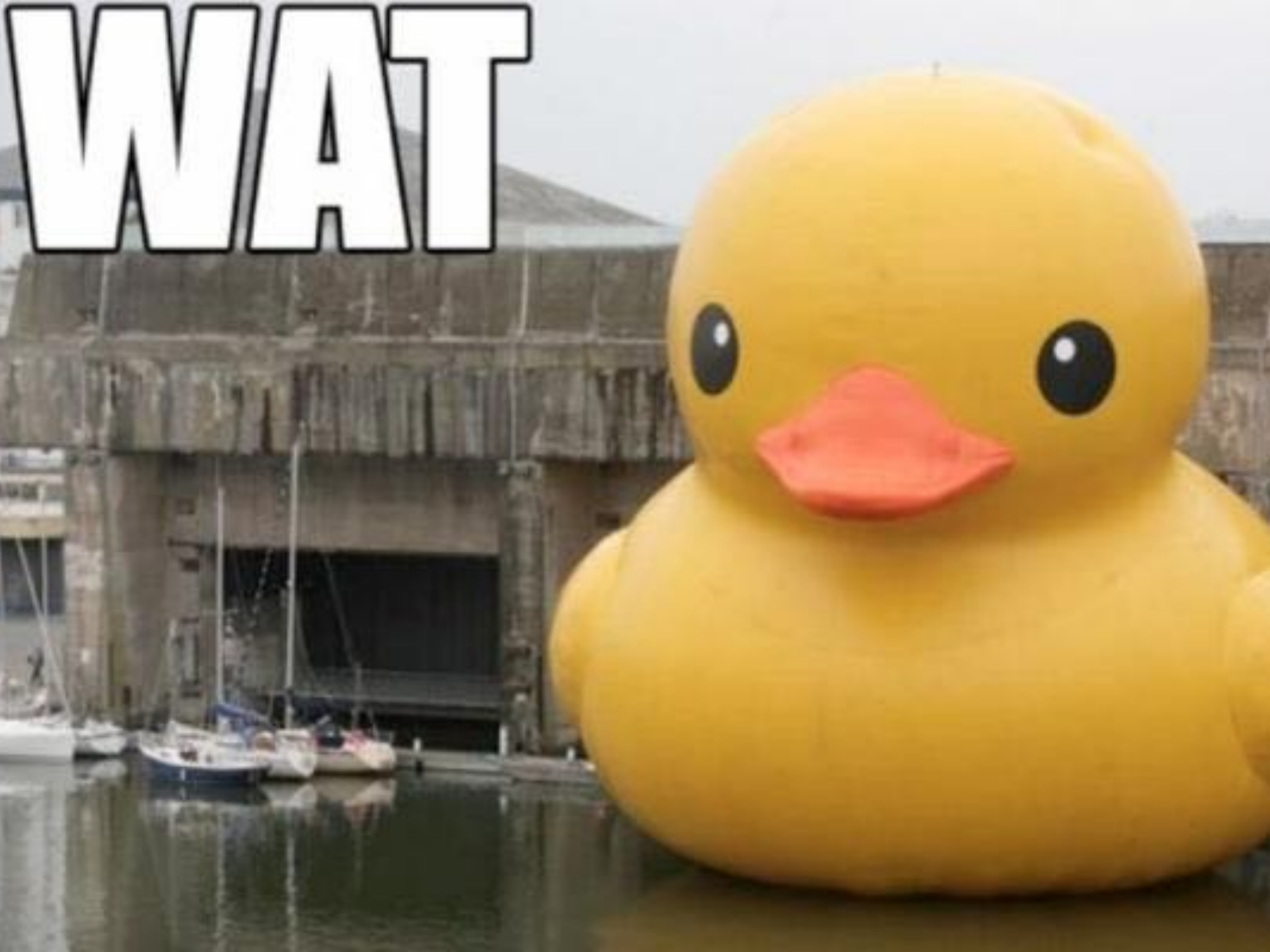
https://kate.io/blog/2017/08/24/python-constants-in-bytecode/

http://www.laurentluce.com/posts/python-integer-objects-implementation/

https://github.com/python/cpython/

https://docs.python.org/3/c-api/intro.html

Rob Conery — *The Impostor's Handbook*


WAT