Lecture

The is operator

Unexpected Results

Implementation Details

Memory Optimization

The is operator – unexpected results

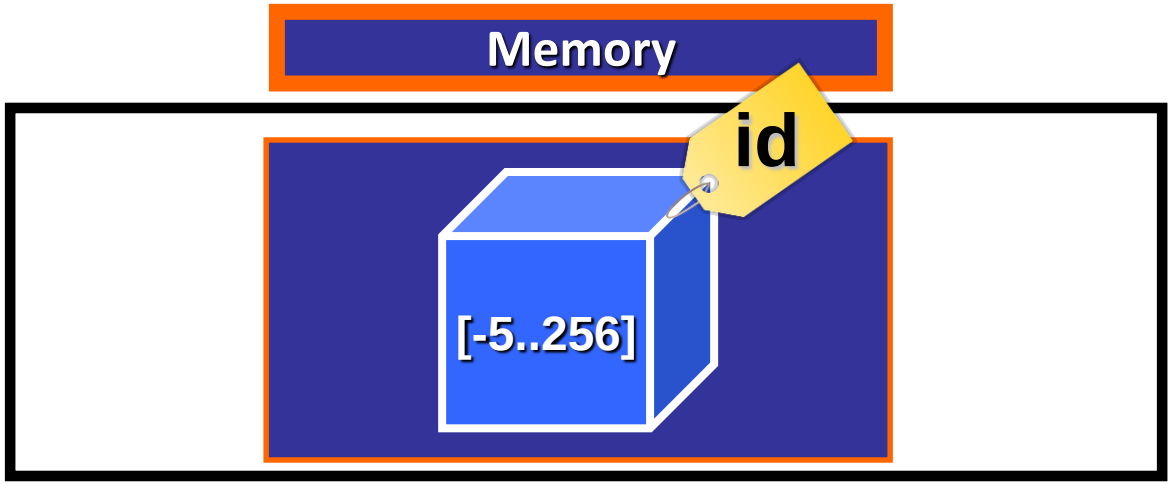
```
>>> a = 5
>>> b = 5
>>> a is b
True
```

Small Integers

[-5, 256]

The is operator – unexpected results

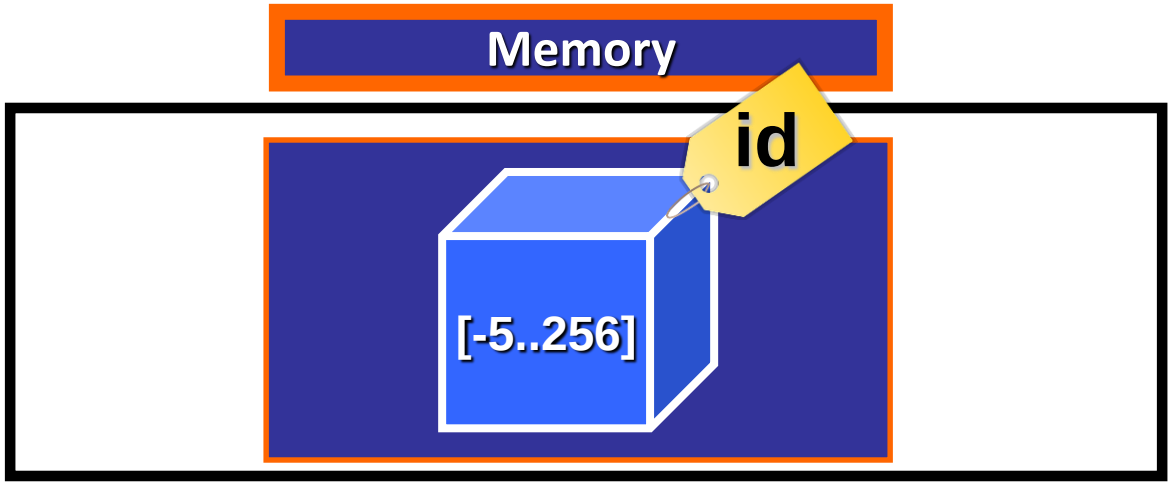
The current implementation keeps an array of integer objects for all integers between `-5` and `256`, when you create an int in that range you actually just get back a reference to the existing object. So it should be possible to change the value of `1`. I suspect the behaviour of Python in this case is undefined.



Docs

The is operator – unexpected results

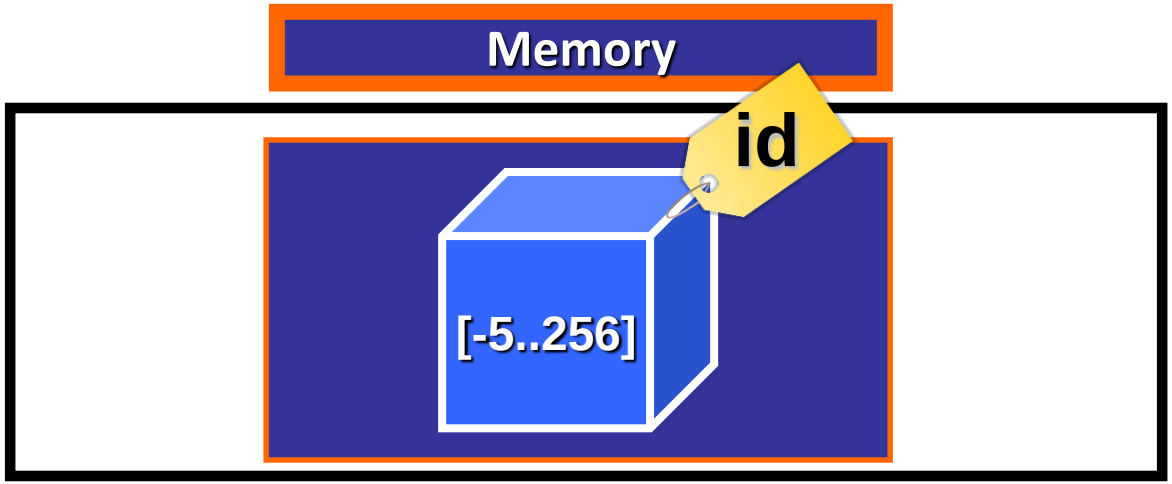
The current implementation keeps an array of integer objects for all integers between `-5` and `256`, when you create an int in that range you actually just get back a reference to the existing object. So it should be possible to change the value of `1`. I suspect the behaviour of Python in this case is undefined.



Docs

The is operator – unexpected results

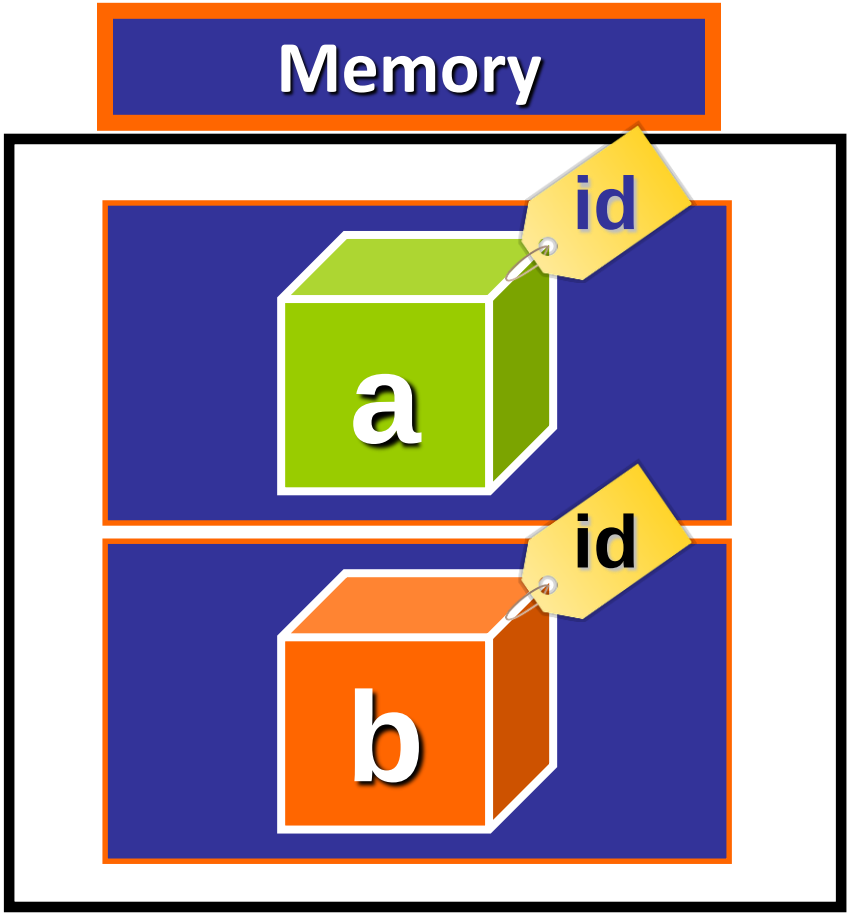
The current implementation keeps an array of integer objects for all integers between `-5` and `256`, when you create an int in that range you actually just get back a reference to the existing object. So it should be possible to change the value of `1`. I suspect the behaviour of Python in this case is undefined.



Docs

The is operator – unexpected results

```
>>> a = 257
>>> b = 257
>>> a is b
False
```



The is operator – unexpected results

```
>>> a = 3
>>> b = 3
>>> a is b
True
```

Memory

Strings

String Interning

The is operator – unexpected results

```
>>> a = "H"
>>> b = "H"
>>> a is b
True
```

The is operator – unexpected results


```
>>> a = "Hello, World"
>>> a[5] = "a"
Traceback (most recent call last):
  File "<pyshell#124>", line 1, in <module>
    a[5] = "a"
TypeError: 'str' object does not support item assignment
```

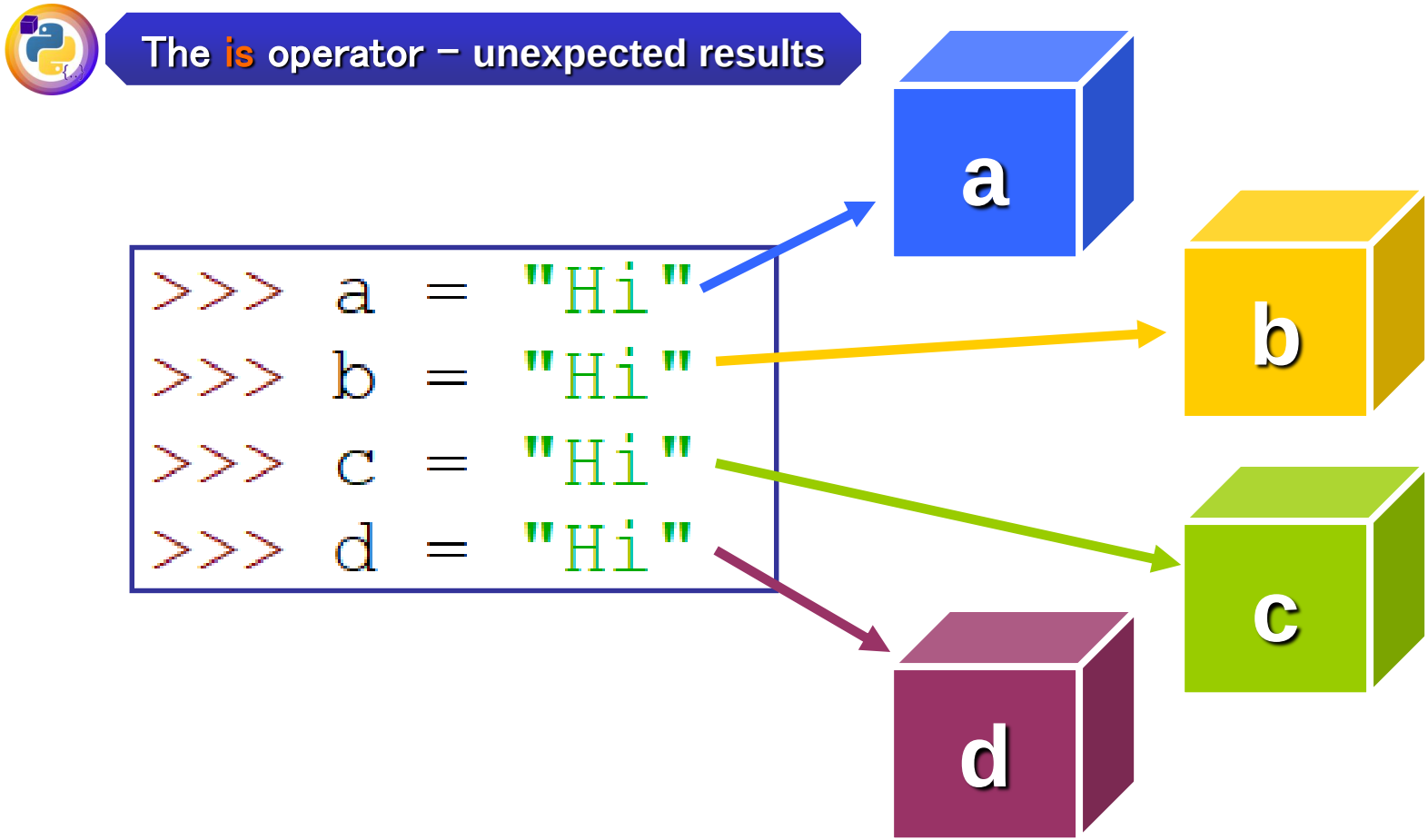
The is operator – unexpected results


```
>>> a = "Hello, World"
>>> a[5] = "a"
Traceback (most recent call last):
  File "<pyshell#124>", line 1, in <module>
    a[5] = "a"
TypeError: 'str' object does not support item assignment
```

The is operator – unexpected results

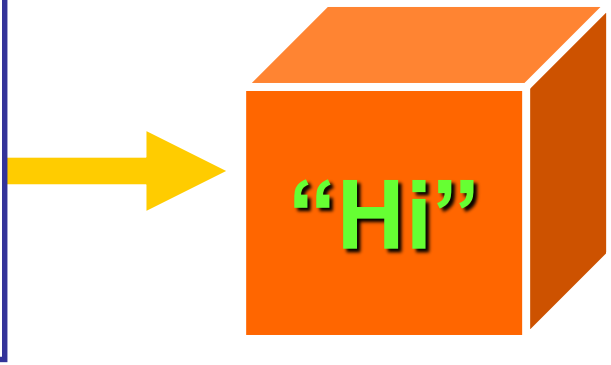
Immutable

```
>>> a = "Hello, World"
>>> a[5] = "a"
Traceback (most recent call last):
  File "<pyshell#124>", line 1, in <module>
    a[5] = "a"
TypeError: 'str' object does not support item assignment
```


The is operator – unexpected results
>>> a = "Hi"
>>> b = "Hi"
>>> c = "Hi"
>>> d = "Hi"
a
b
c
d

The is operator – unexpected results

```
>>> a = "Hi"
>>> b = "Hi"
>>> c = "Hi"
>>> d = "Hi"
```



The is operator – unexpected results

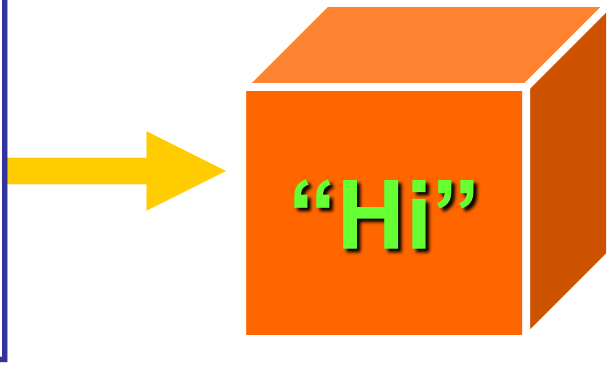
```
>>> a = "Hi"
>>> id(a)
48310656
>>> b = "Hi"
>>> id(b)
48310656
>>> c = "Hi"
>>> id(c)
48310656
>>> d = "Hi"
>>> id(d)
48310656
```

The is operator – unexpected results

```
>>> a = "Hi"
>>> id(a)
48310656
>>> b = "Hi"
>>> id(b)
48310656
>>> c = "Hi"
>>> id(c)
48310656
>>> d = "Hi"
>>> id(d)
48310656
```

The is operator – unexpected results

```
>>> a = "Hi"
>>> b = "Hi"
>>> c = "Hi"
>>> d = "Hi"
```



Rules of String Interning

- All length 0 and length 1 strings are interned.
- Strings that are not composed exclusively of ASCII letters (a-z, A-Z), digits (0-9), or underscores are not interned.

Rules of String Interning

- All length 0 and length 1 strings are interned.
- Strings that are not composed exclusively of ASCII letters (a-z, A-Z), digits (0-9), or underscores are not interned.

Rules of String Interning

- All length 0 and length 1 strings are interned.
- Strings that are not composed exclusively of ASCII letters (a-z, A-Z), digits (0-9), or underscores are not interned.

Rules of String Interning

- All length 0 and length 1 strings are interned.
- Strings that are not composed exclusively of ASCII letters (a-z, A-Z), digits (0-9), or underscores are not interned.

The is operator – unexpected results

```
>>> a = "Z"
>>> b = "Z"
>>> a is b
True
```

The is operator – unexpected results

```
>>> a = ""
>>> b = ""
>>> a is b
True
```

The is operator – unexpected results

```
>>> a = "@Hello"
>>> b = "@Hello"
>>> a is b
False
```

The is operator – unexpected results

```
>>> a = "Hello"
>>> b = "Hello"
>>> a is b
True
```

The is operator – unexpected results

```
>>> a = "32424"
>>> b = "32424"
>>> a is b
True
```

The is operator – unexpected results

```
>>> a = "ab" + "ab"
>>> b = "ab" + "ab"
>>> a is b
True
```

The is operator – unexpected results

```
>>> a = "abcde"
>>> b = "".join(["a", "b", "c", "d", "e"])
>>> a is b
False
```

Working with Objects