Variables and Pointers

References to Objects

Author: Torey Hickman

Phase 1: Day 3

Why Use Variables?

- Store data for later use
- Improve readability

What is a Variable?

- A reference to an object

Variable Assignment

- What is the value of b?

```
a = “some string”
b = a

b
```

Variable Assignment

- What is the value of b?

```
a = “some string”
b = a

b
# => “some string”
```

Variable Assignment

- What is the value of b?

```
a = “some string”
b = a

a = “another string”

b
```

Variable Assignment

- What is the value of b?

```
a = “some string”
b = a

a = “another string”

b
# => “some string”
```

Variable Assignment

- What is the value of b?

```
a = “some string”
b = a

a << “!”
b
```

Variable Assignment

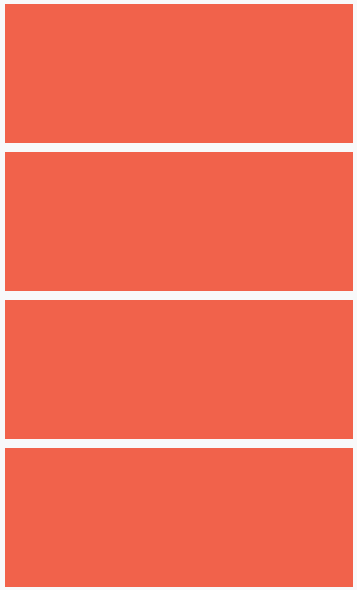
- What is the value of b?

```
a = “some string”
b = a

a << “!”

b
# => “some string!”
```

Variable Assignment

- Variables are not objects, just pointers to objects

The Stack and Heap

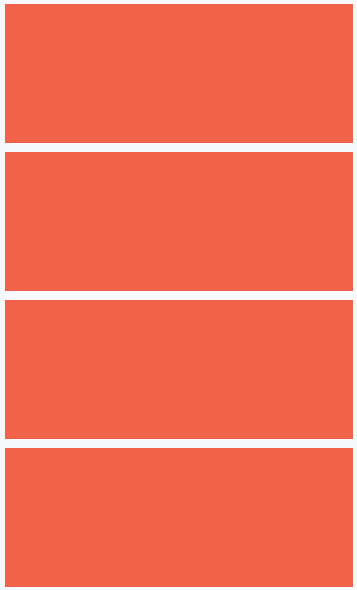


stack

system heap

The Stack and Heap

```
a = “some string”
```

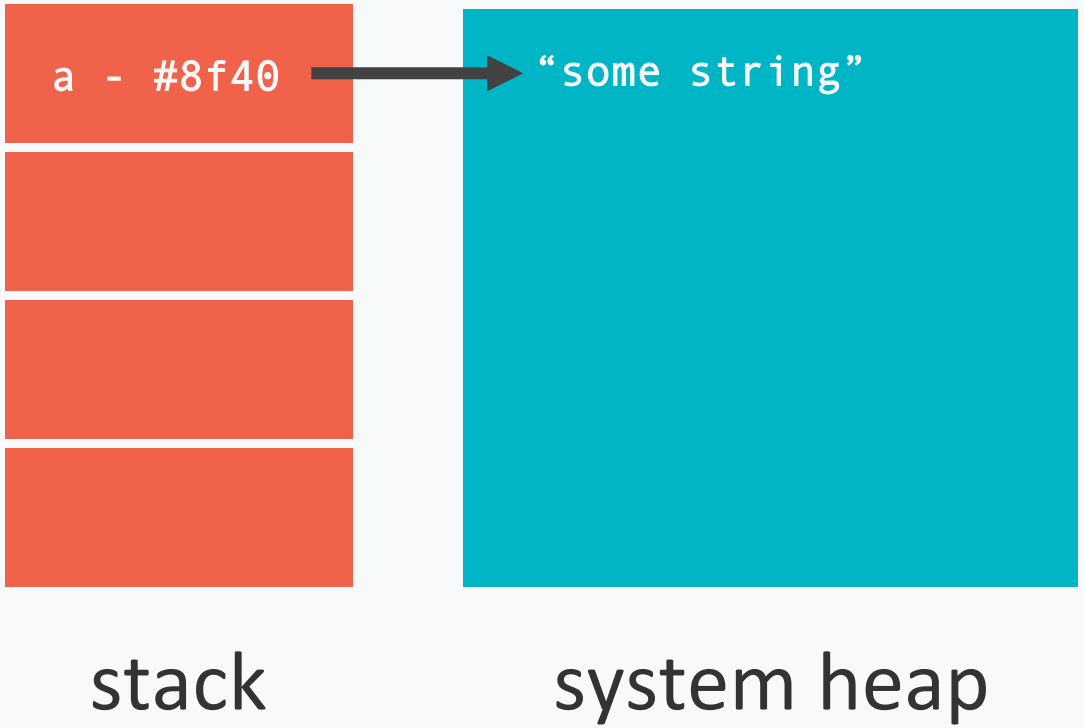


stack

system heap

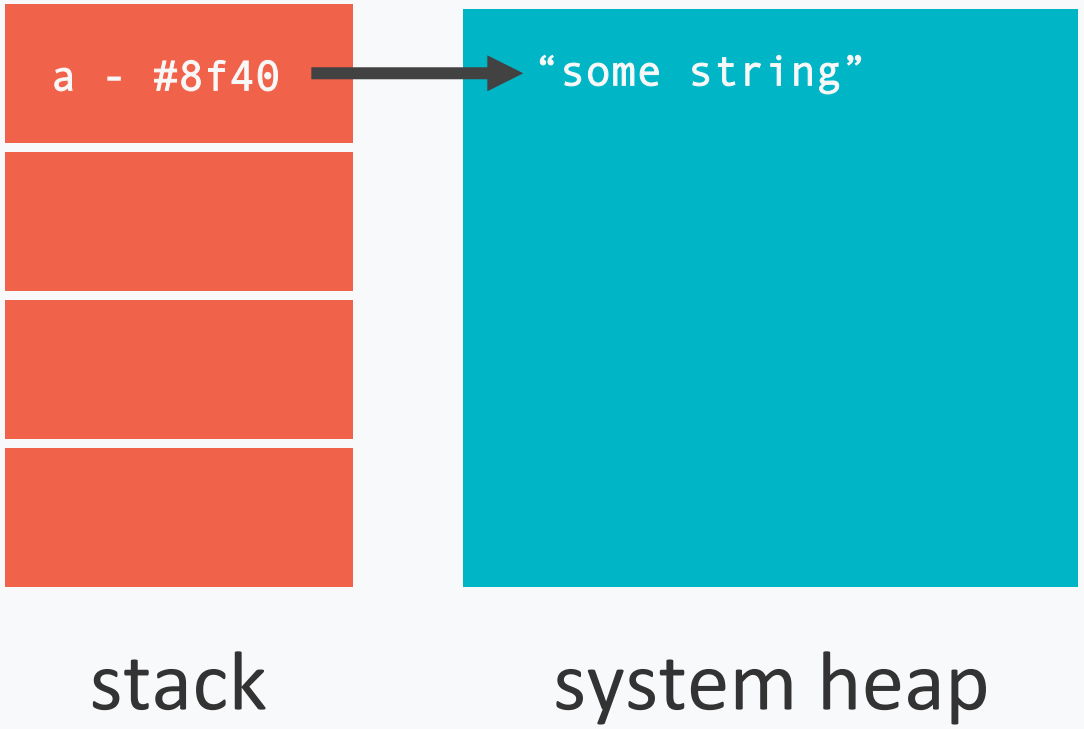
The Stack and Heap

```
a = “some string”
```



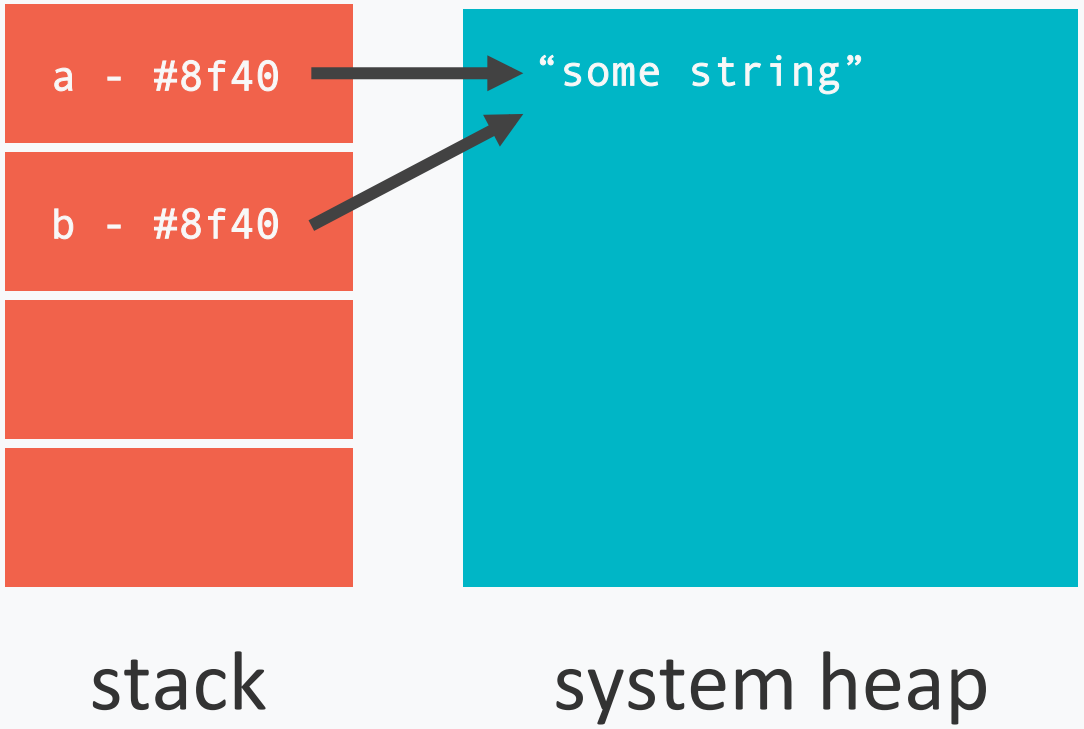
The Stack and Heap

```
a = “some string”
b = a
```



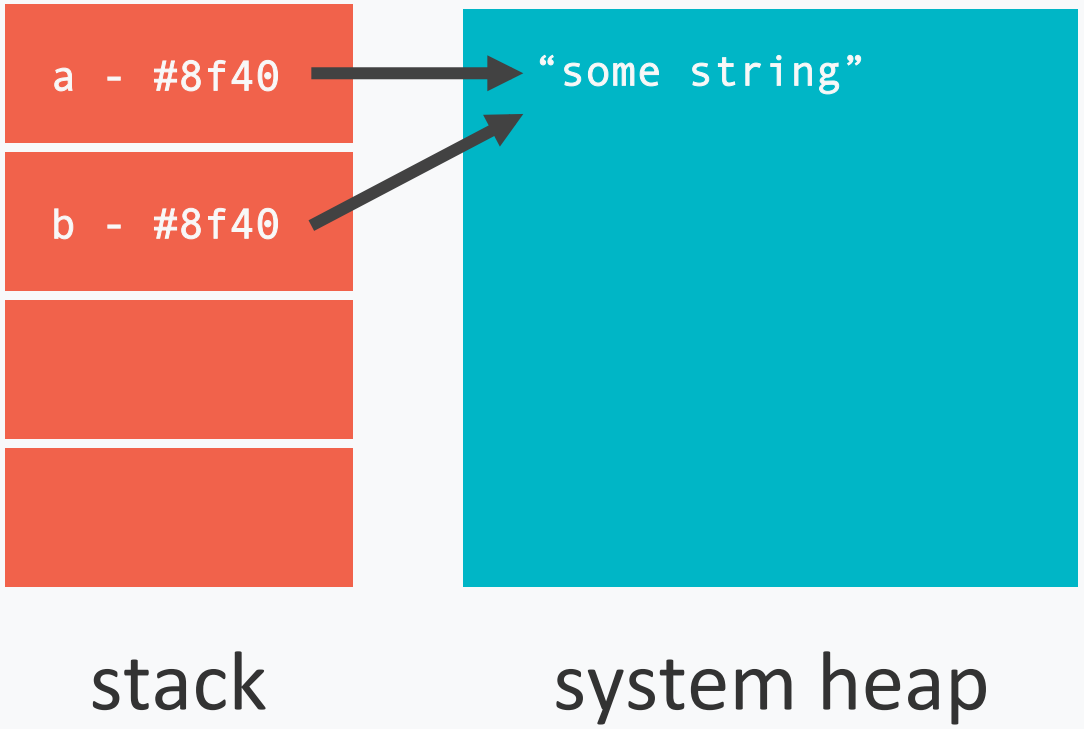
The Stack and Heap

```
a = “some string”
b = a
```



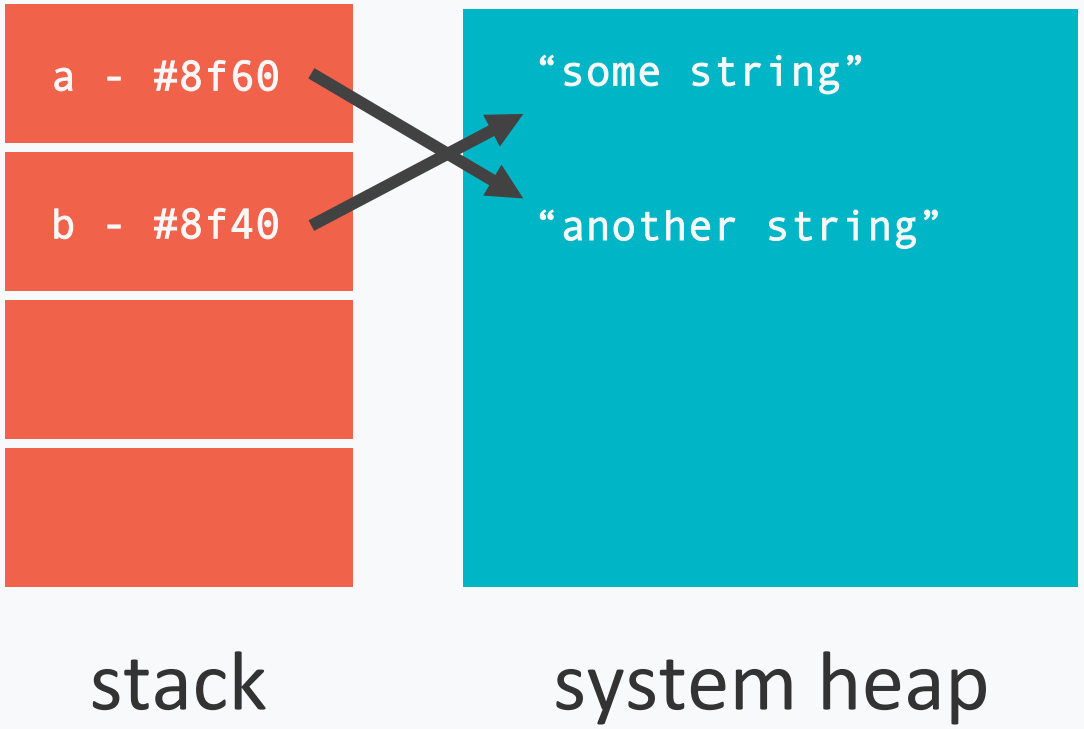
The Stack and Heap

```
a = “some string”
b = a

a = “another string”
```



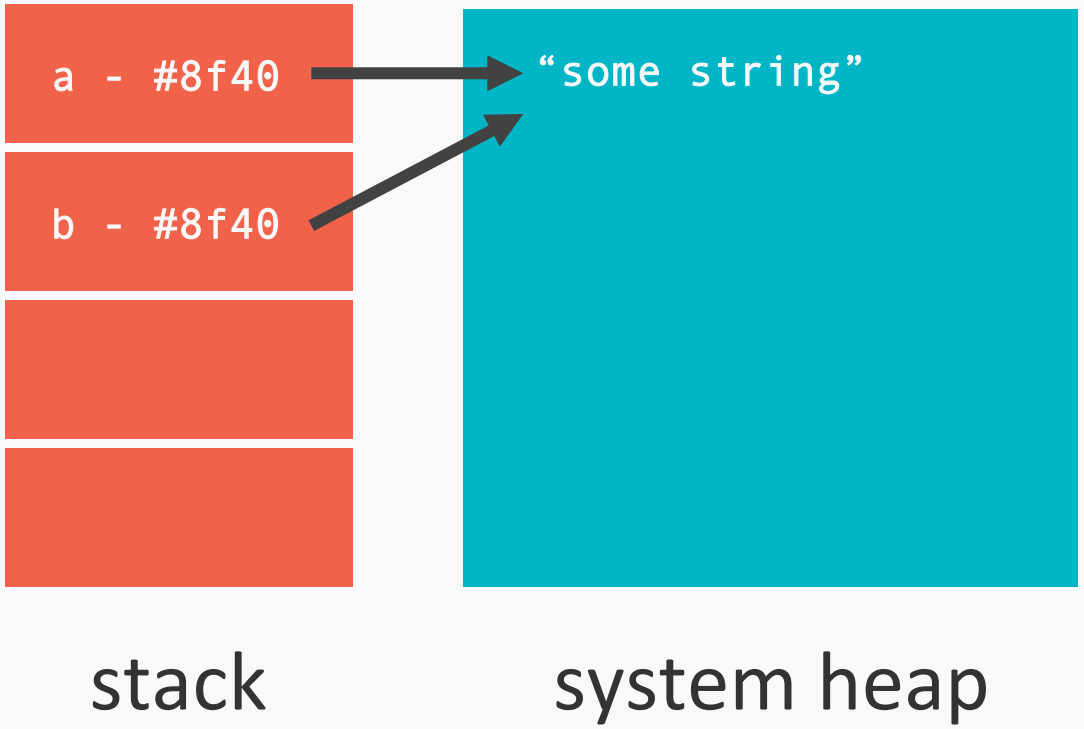
The Stack and Heap

```
a = “some string”
b = a

a = “another string”
```



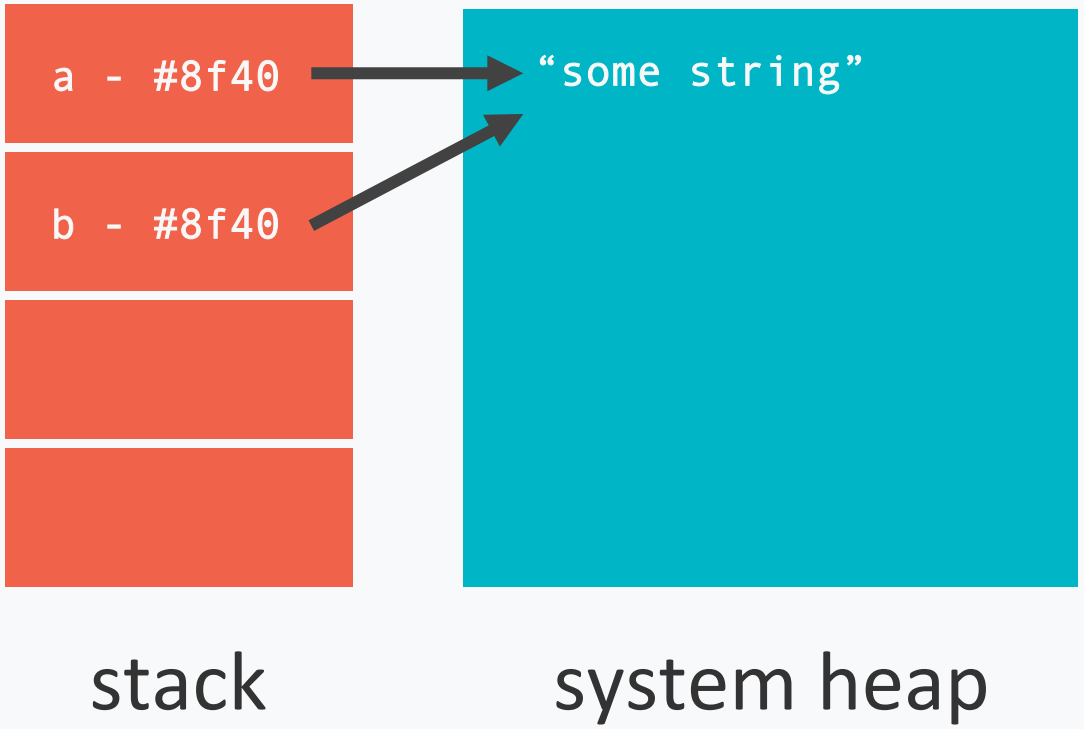
The Stack and Heap

```
a = “some string”
b = a
```



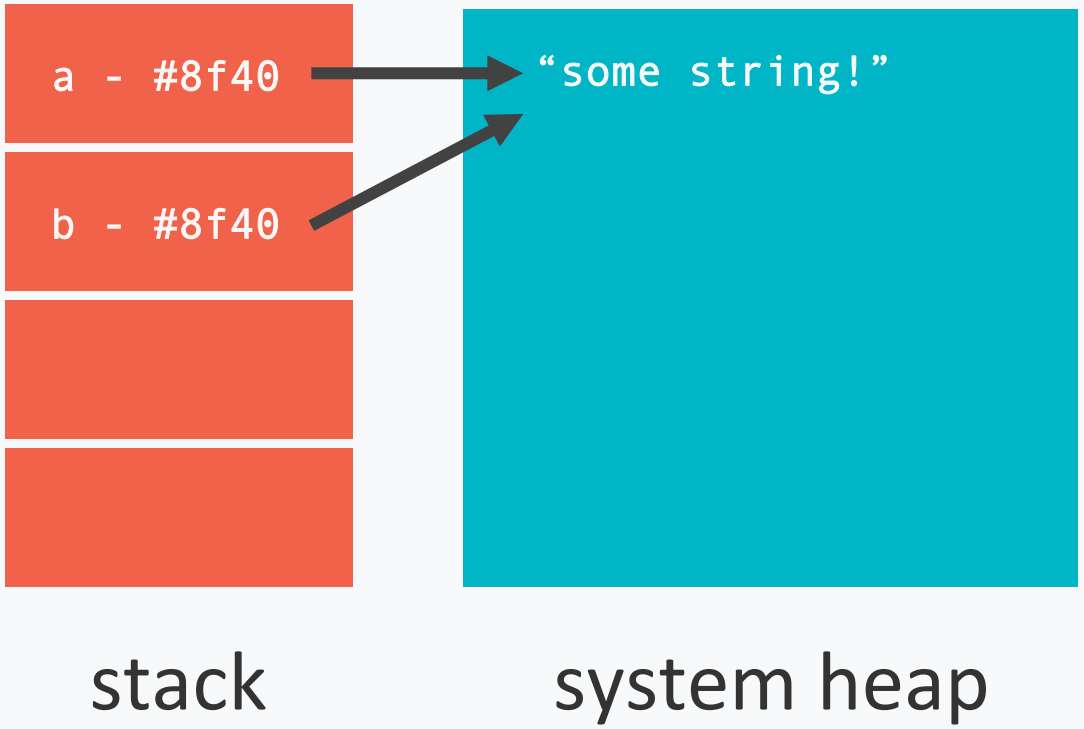
The Stack and Heap

```
a = “some string”
b = a

a << “!”
```



The Stack and Heap

```
a = “some string”
b = a

a << “!”
```

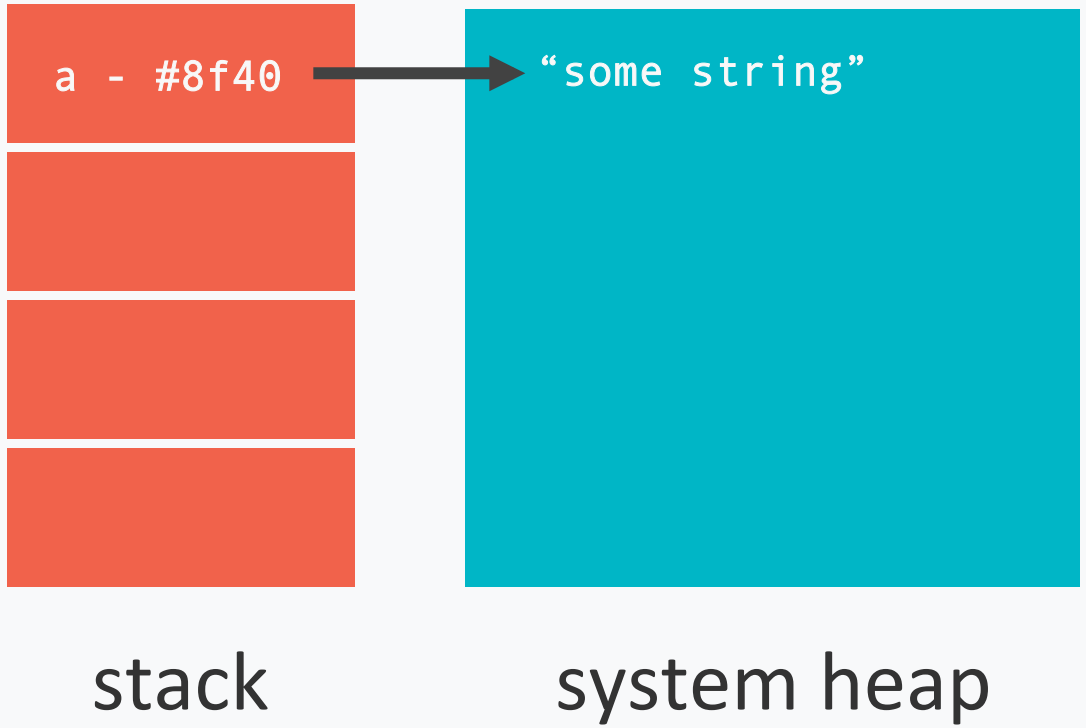


#dup and #clone

- The `#dup` and `#clone` methods will make new objects similar to the old objects

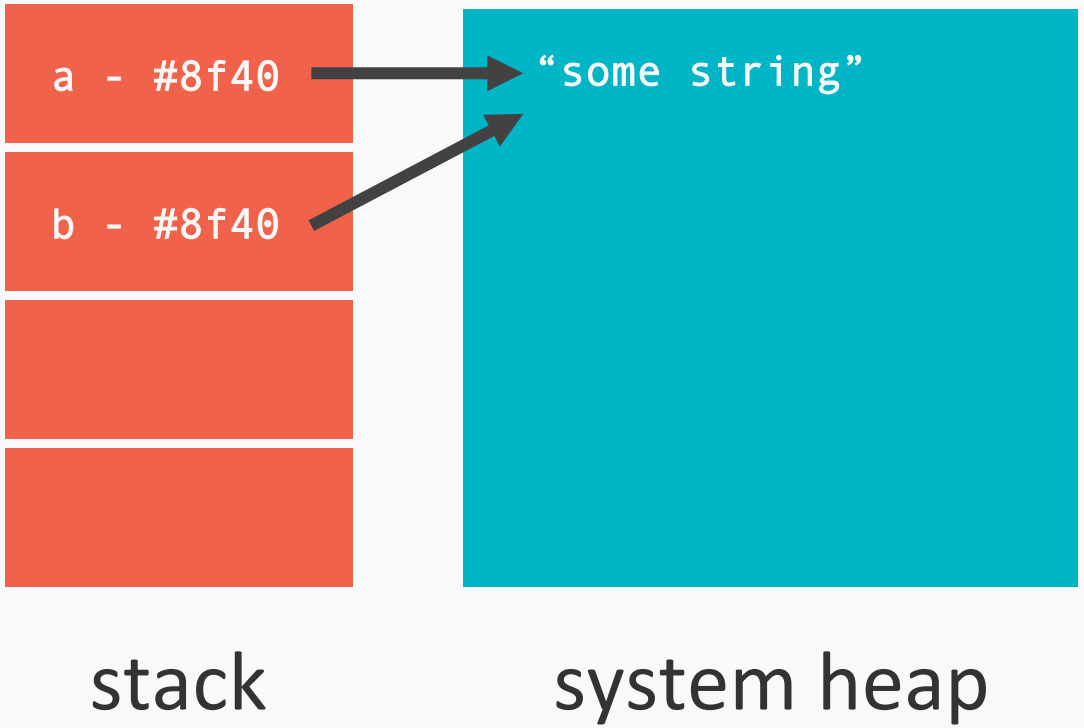
#dup and #clone

```
a = “some string”
```



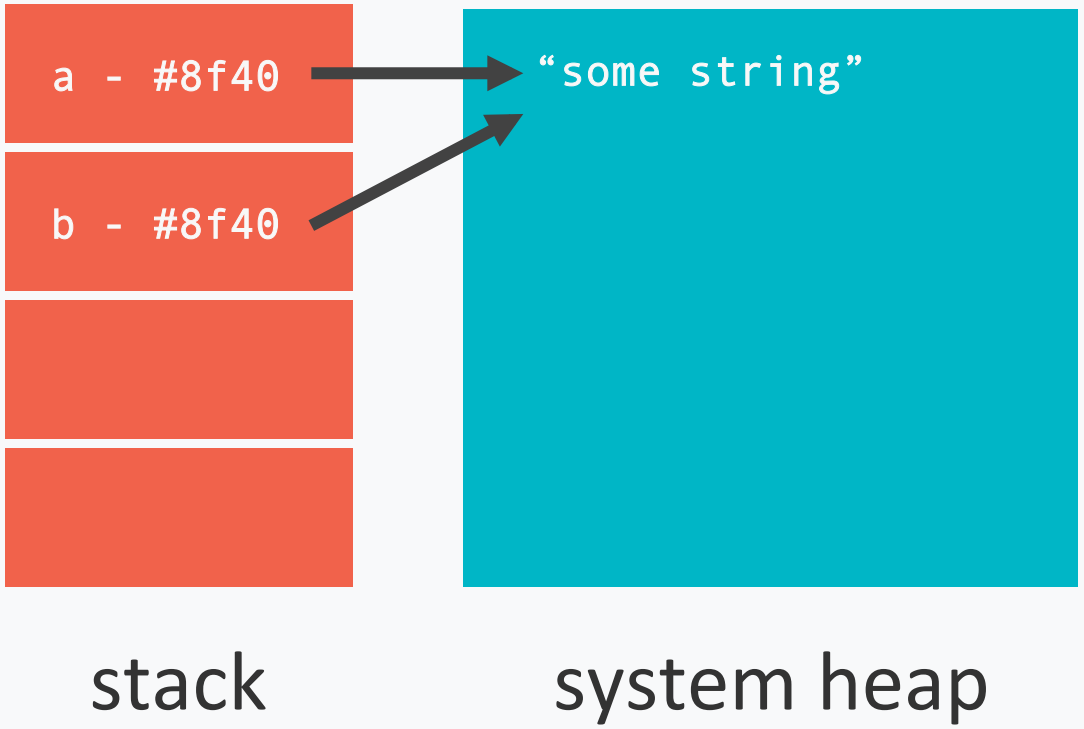
#dup and #clone

```
a = “some string”
b = a
```



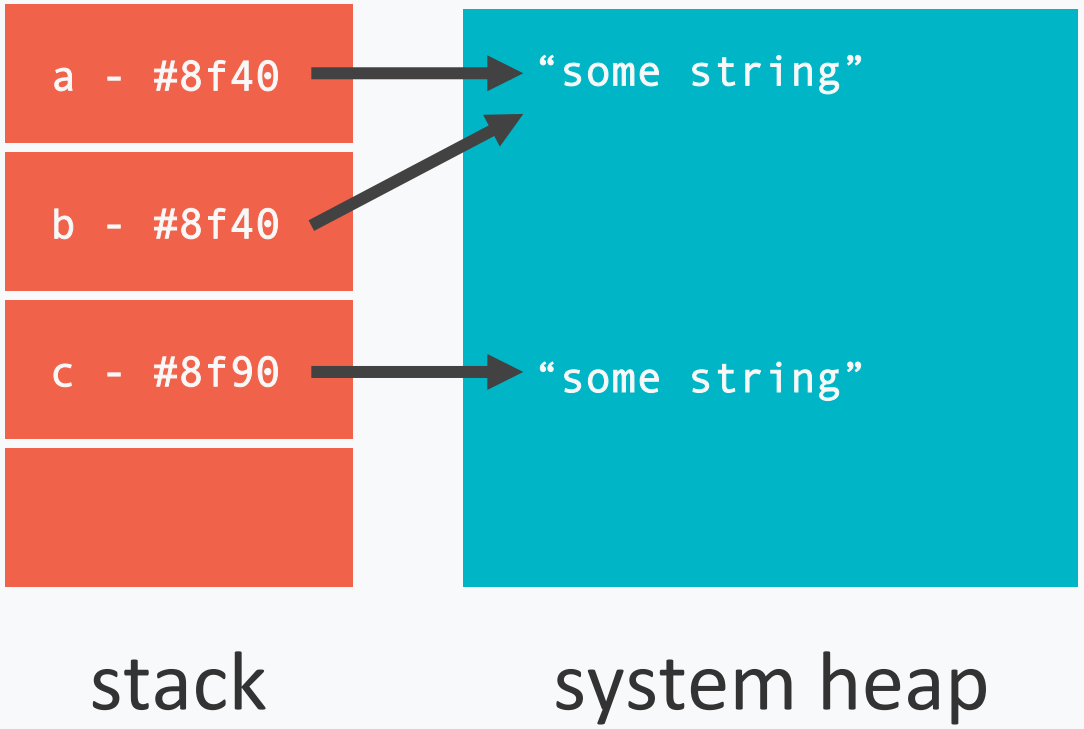
#dup and #clone

```
a = “some string”
b = a
c = a.dup
```



#dup and #clone

```
a = “some string”
b = a
c = a.dup
```



Arrays and Pointers

- Arrays have pointers, too

Arrays and Pointers

- An array keeps track of pointers to objects, not objects themselves.

Arrays and Pointers

- What happens if I alter an object that is already in an array?

```
a = “xyz”
b = a
array = [a, b]
```

Arrays and Pointers

- What happens if I alter an object that is already in an array?

```
a = “xyz”
b = a
array = [a, b]

a << “!”

array
```

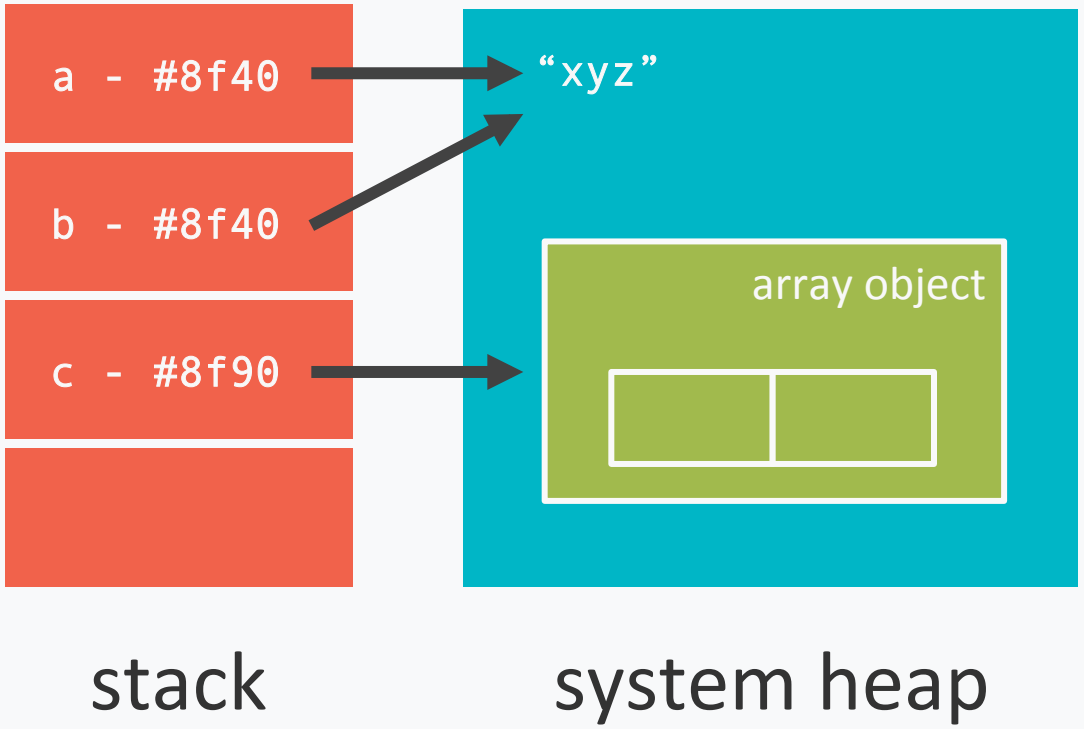
Arrays and Pointers

- What happens if I alter an object that is already in an array?

```
a = “xyz”
b = a
array = [a, b]

a << “!”

array
# => [“xyz!”, “xyz!”]
```

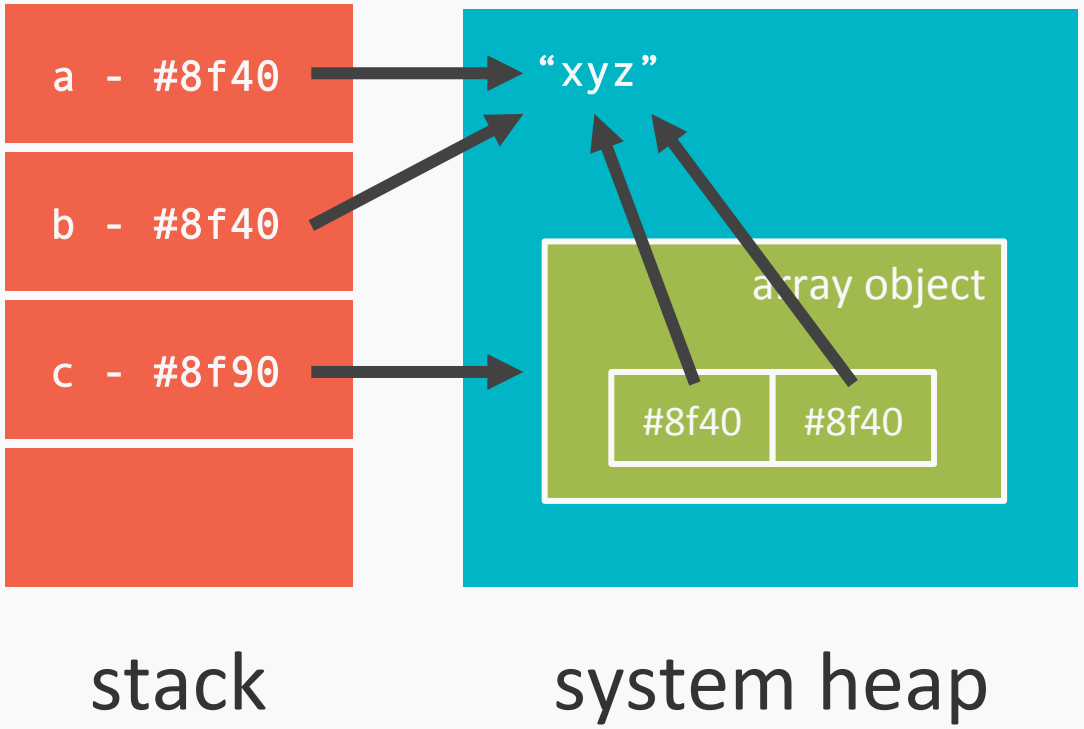
Arrays and Pointers

```
a = “xyz”
b = a
array = [a, b]
```



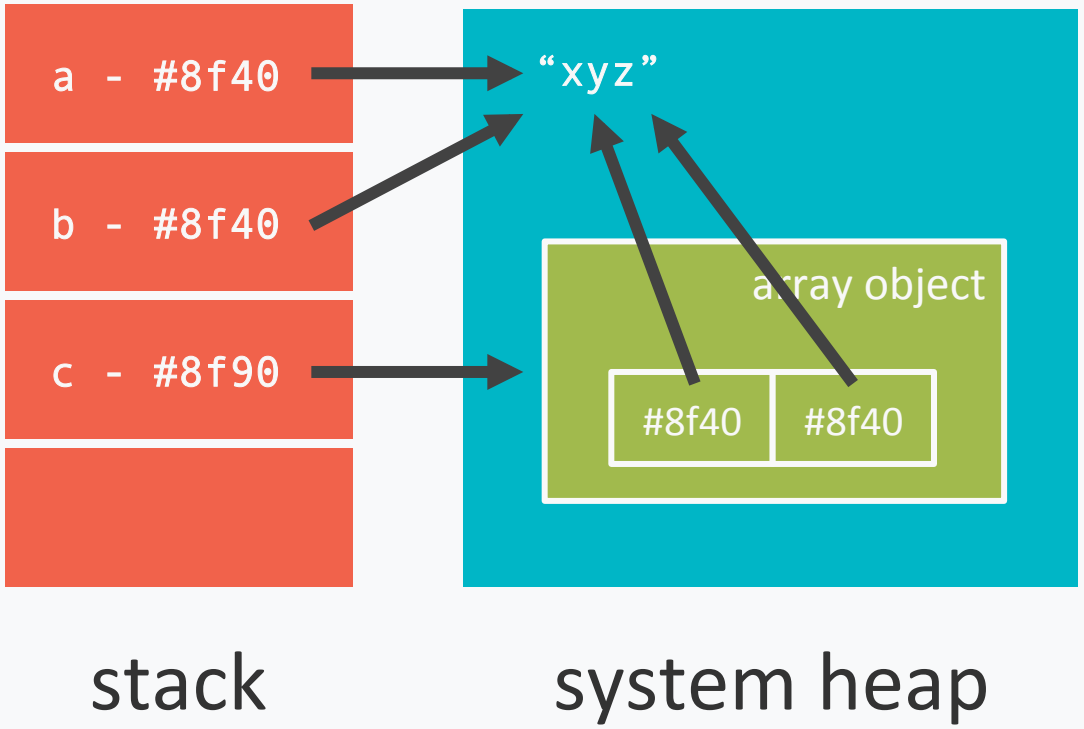
Arrays and Pointers

```
a = “xyz”
b = a
array = [a, b]
```



Arrays and Pointers

```
a = “xyz”
b = a
array = [a, b]

a << “!”
```



Arrays and Pointers

```
a = “xyz”
b = a
array = [a, b]

a << “!”

array
# => [“xyz!”, “xyz!”]
```