CSc 110
Objects and References

Adriana Picoral (she/her/hers)

Announcements

- Individual exam on Wednesday (11%)
- Group exam on Friday (4%)
- Review session tomorrow (Tuesday, Nov 15) at 5pm
    - McClelland Park 105

Objects

- (Most) Of the values in python are **objects**!

  - Strings are objects `"This is an object"`
  - Integers are objects `134`
  - Lists are objects `[1, 4, 8, 2, 6]`
  - Dictionaries are objects `{ "and":10, "or":20 }`

- Entities that can be assigned to a variable or passed as an argument to a function are, typically, objects

Object Types

- Every **object** has a **type** (or **None**)

`"This is an object"`	`str`
`134`	`int`
`[1, 4, 8, 2, 6]`	`list`
`{ "and":10, "or":20 }`	`dict`

Object Types

- Some types of object are **mutable** and other are **immutable**
    - **Mutable object:** An object that can be changed once it is created
    - **Immutable object:** An object that cannot be changed once it is created

Object References

- When we are assigning a variable to an object, we are storing a **reference** to the object

- When we use the **variable name**, this “points us” to the object that is associated with the name

- A few examples . . .

Object References

What will this print?

```
title = "William"
name = title
print(name + " " + title)
title = "Josh"
print(name + " " + title)
name_b = name
name = "Stanley"
print(name + " " + name_b + " " + title)
```

Object References

Variable

Objects

What are the references?

```
title = "William"
name = title
print(name + " " + title)
title = "Josh"
print(name + " " + title)
name_b = name
name = "Stanley"
print(name + " " + name_b + " " + title)
```

Object References

What are the references?

```
numbers = [50, 30, 80]
more = numbers
more.append(70)
more = [80, 70, 60]
numbers.append(10)
numbers = "look, numbers!"
more = numbers
```

Object-Reference Diagram

Draw the object-reference diagram for this code.

```
a = [7, 4, 5]
b = {9, 8, 3}
c = a
a = b
a.add(10)
c.append('hi')
b = 'name'
```

Object References

- A variable does not actually hold the value of the object within it
- Instead, a ***reference*** to the object
  - The object is “sitting” somewhere in your computer’s memory (RAM)
- When you assign a value to a *new* variable, one of two things could happen
  - If you assign it to an existing object, the variable references that object
  - If you assign it to a new object, the object is created, placed in memory, and then the variable references it

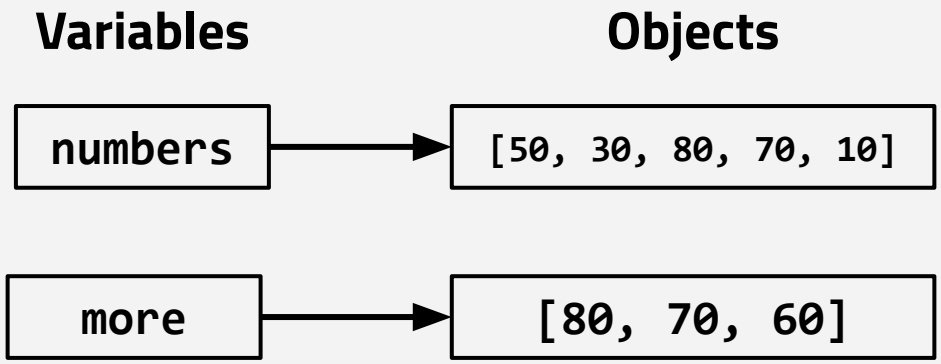
Garbage Collection

- What about those ***dangling*** objects? (see the last example)

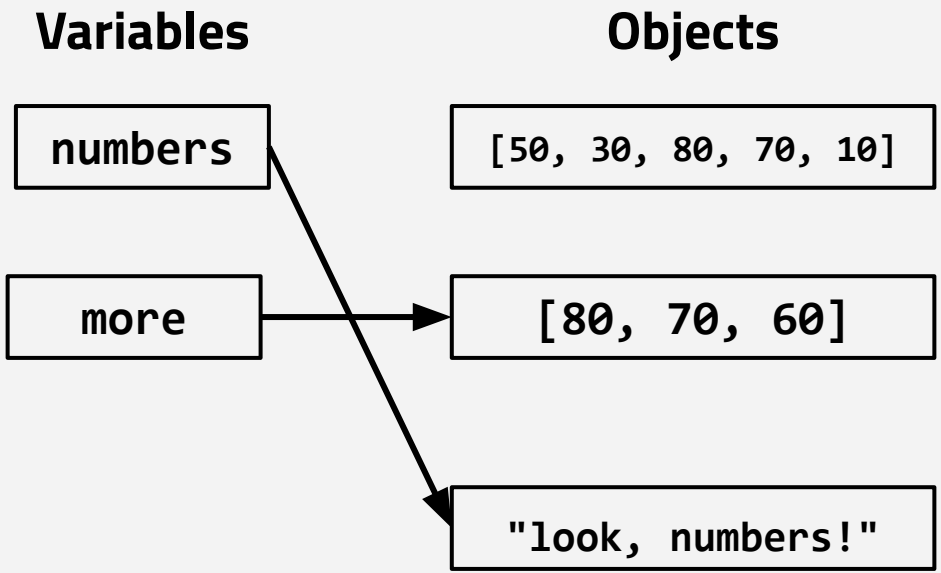
Garbage Collection

- What about those ***dangling*** objects? (see the last example)
- Taken care of by the **Garbage Collector (GC)**
- The **GC** is a part of python that automatically cleans up these stray objects as the program executes
- As the programmer, you don’t need to worry about them
- In some languages (like C), there is no built-in automatic GC
    - The programmer is responsible for managing memory!

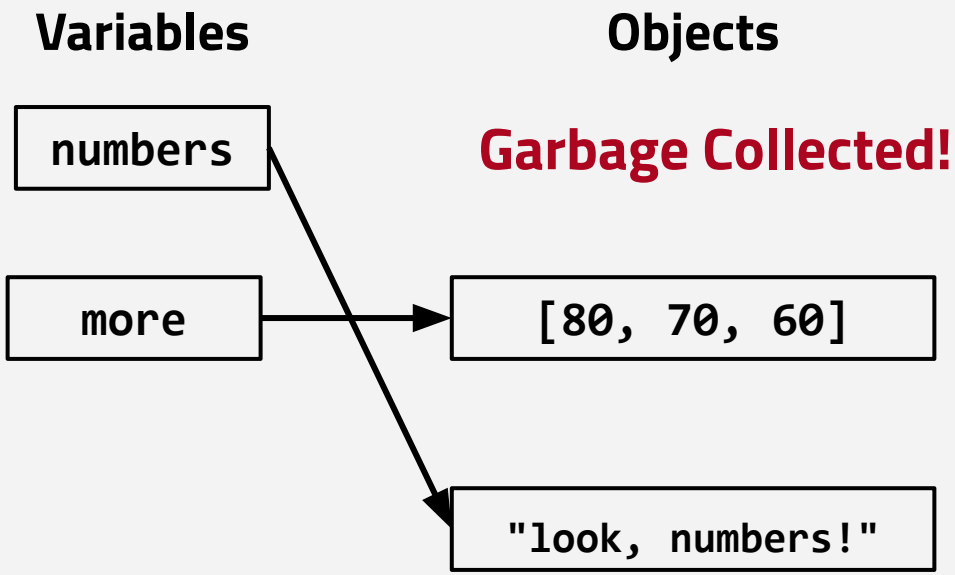
Garbage Collection Example



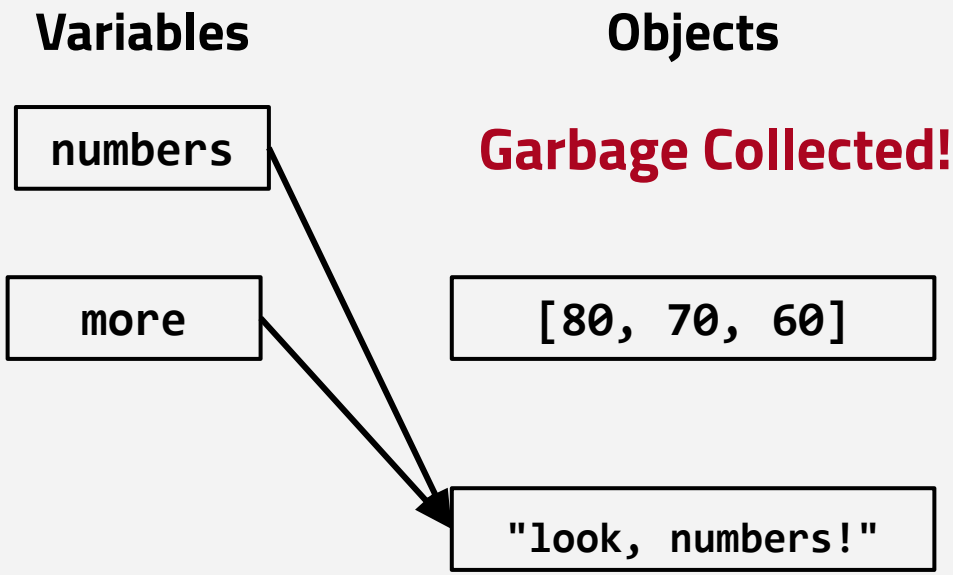
Garbage Collection Example



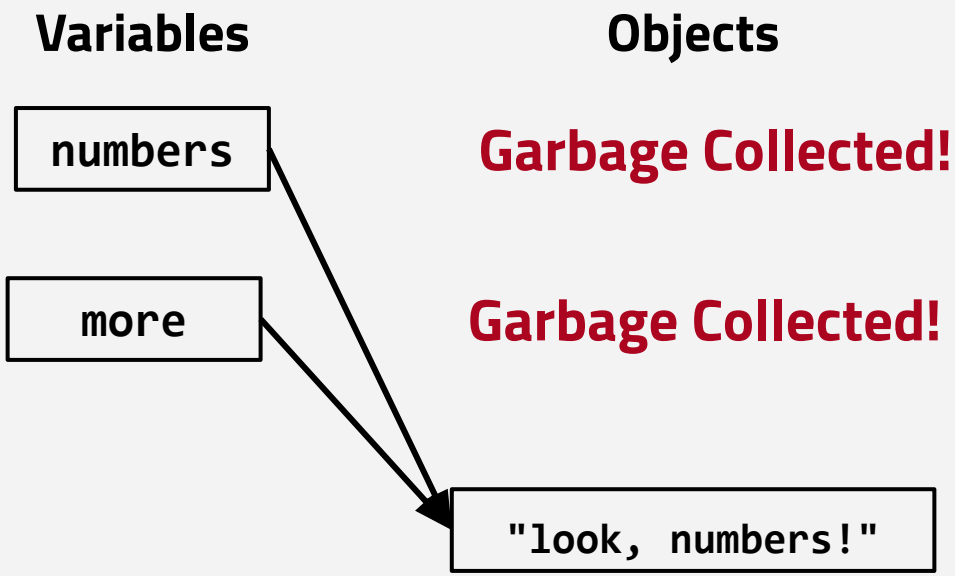
Garbage Collection Example



Garbage Collection Example



Garbage Collection Example



Mutable and Immutable

- This matters when it comes to variable references
  - Especially when it comes to passing a variable into a function via a function parameter

- When you pass a variable as an argument to a function, the parameter variable is a reference to the same object that was at the call-site
  - If the object type is mutable, the function can mutate it
  - If immutable, the function cannot mutate it

Passing Immutable Object by Reference

What will this program print out when executed?

```
def append_stuff(param):
    param = param + " stuff"
    print(param)
    param = "NEW!"
    print(param)

name = "Earl Button"
append_stuff(name)
print(name)
```

Passing Immutable Object by Reference

What are the references? How many elements GCed?

```
def append_stuff(param):
    param = param + " stuff"
    print(param)
    param = "NEW!"
    print(param)

name = "Earl Button"
append_stuff(name)
print(name)
```

Passing Mutable Object by Reference

What will this program print out when executed?

```
def append_stuff(param):
    param.append("Max")
    print(param)
    param = "STRING!"
    print(param)

items = ["Ben", "Sam", "Kim"]
append_stuff(items)
print(items)
```

Passing Mutable Object by Reference

What are the references?

```
def append_stuff(param):
    param.append("Max")
    print(param)
    param = "STRING!"
    print(param)

items = ["Ben", "Sam", "Kim"]
append_stuff(items)
print(items)
```

Draw the reference diagram

```
position = 'Dir of Videography'
sp = position.split(' ')
p2 = position.strip('ypgrha')
```

Draw the reference diagram

```
position = 'Dir of Videography'
sp = position.split(' ')
p2 = position.strip('ypgrha')
p3 = p2
p2 = position
position = p3
p2 = p2.strip('oe')
```

Draw the reference diagram

```
def update_list(e):
    e2 = e
    e = []
    for i in range(0, 3):
        e.append(i)
        e2.append(i+1)

numbers = [5, 15, 10]
update_list(numbers)
print(numbers)
```

Draw the reference diagram

```
def update(elements, label):
    elements.sort()
    label_2 = label
    label = label.strip('aeiou')
    return label

numbers = [5, 15, 10]
word = 'aerospace'
word = update(numbers, word)
print(numbers, word)
```