Discussion 05:
Object Oriented Programming

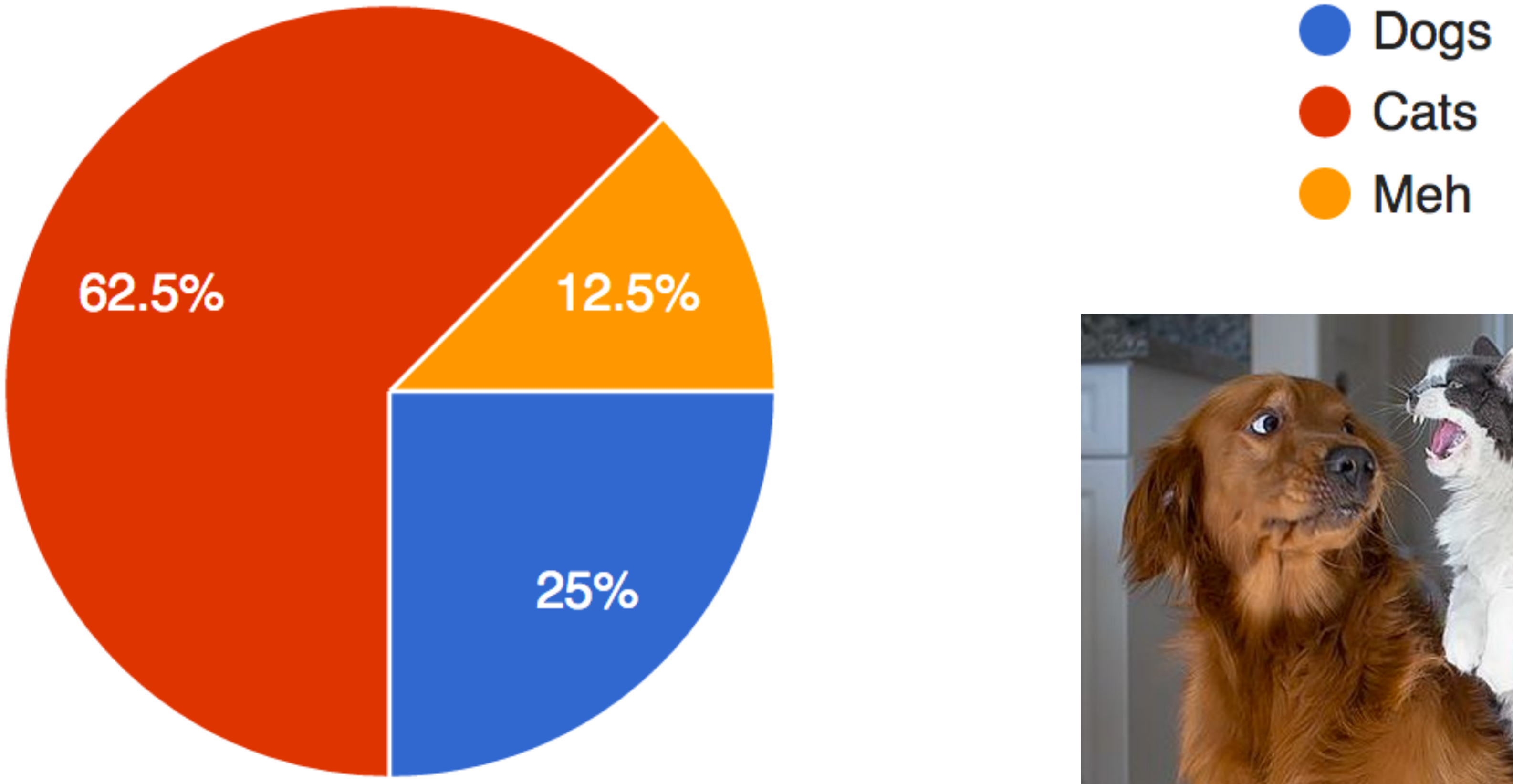
TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Agenda

1. Attendance
2. Announcements
3. Check Your Understanding
4. OOP

Feedback

I need more responses!



Attendance

Sign in at bit.do/jerrydisc

OR

Come to me for check-in

Announcements

Ants due next Friday (bonus point for 1 day early)

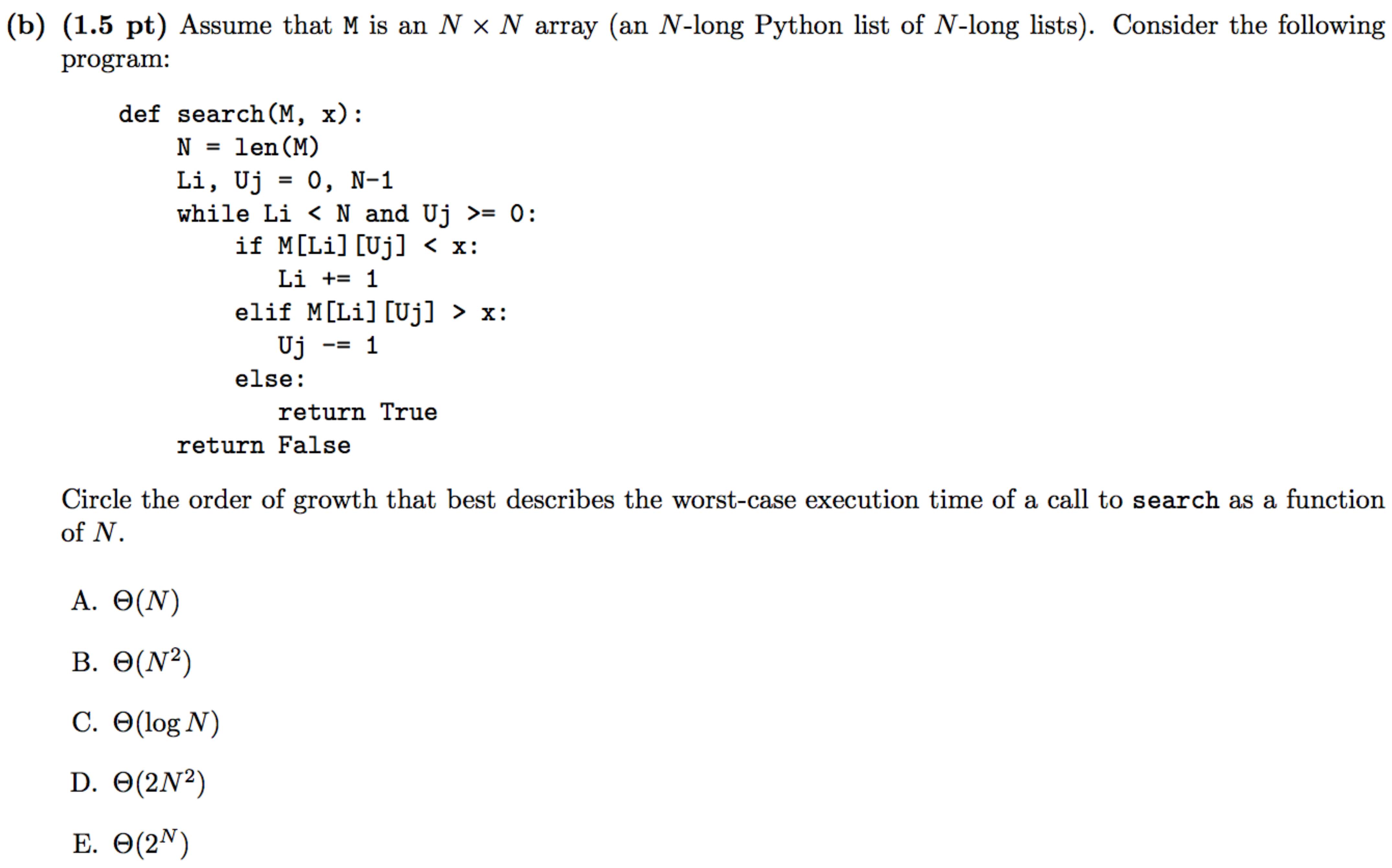
Hw 6 due Today

Hw 7 due next Tuesday

Lab feedback: bit.do/jerrylabfb

Discussion feedback: bit.do/jerrydiscfb

Check Your Understanding



(b) (1.5 pt) Assume that `M` is an $N \times N$ array (an N-long Python list of N-long lists). Consider the following program:

```
def search(M, x):
    N = len(M)
    Li, Uj = 0, N-1
    while Li < N and Uj >= 0:
        if M[Li][Uj] < x:
           Li += 1
        elif M[Li][Uj] > x:
           Uj -= 1
        else:
           return True
    return False
```

Circle the order of growth that best describes the worst-case execution time of a call to **search** as a function of N.

A. $\Theta(N)$

B. $\Theta(N^2)$

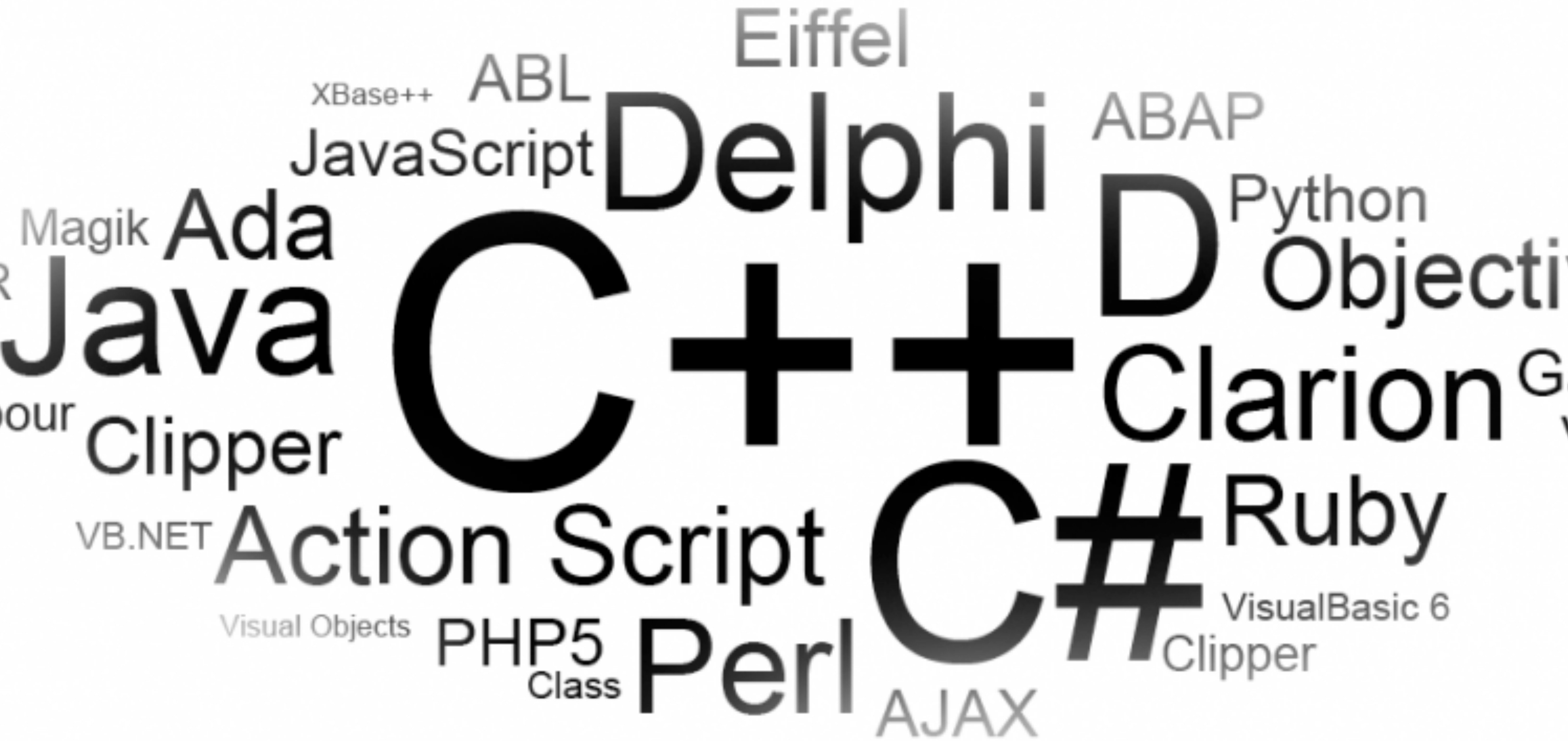
C. $\Theta(\log N)$

D. $\Theta(2N^2)$

E. $\Theta(2^N)$

http://inst.eecs.berkeley.edu/~cs61a/sp16/assets/pdfs/61a-sp16-mt2.pdf

Object Oriented Programming



Objects/Classes

Objects

- A (hopefully) more intuitive way of **representing data**
- A commonly used method of **organizing a program**
- Formally split "global state" and "local state"

Objects/Classes

Classes

- A **“blueprint”**
- Objects are an **instance** of a class

Objects

Attributes - **data!**

- **Class attributes** is shared by the class
- **Instance attributes** belong to an instance

Methods - **behavior!**

- Callable by instances

Attributes

```
class Car:
    headlights = 2 # Class attributes
    wheels = 0

    def __init__(self, make):
        self.make = make # Instance attribute
        self.wheels = 4  # Override class here!
```

Class vs Instance

Differences between **class** and **instance:**

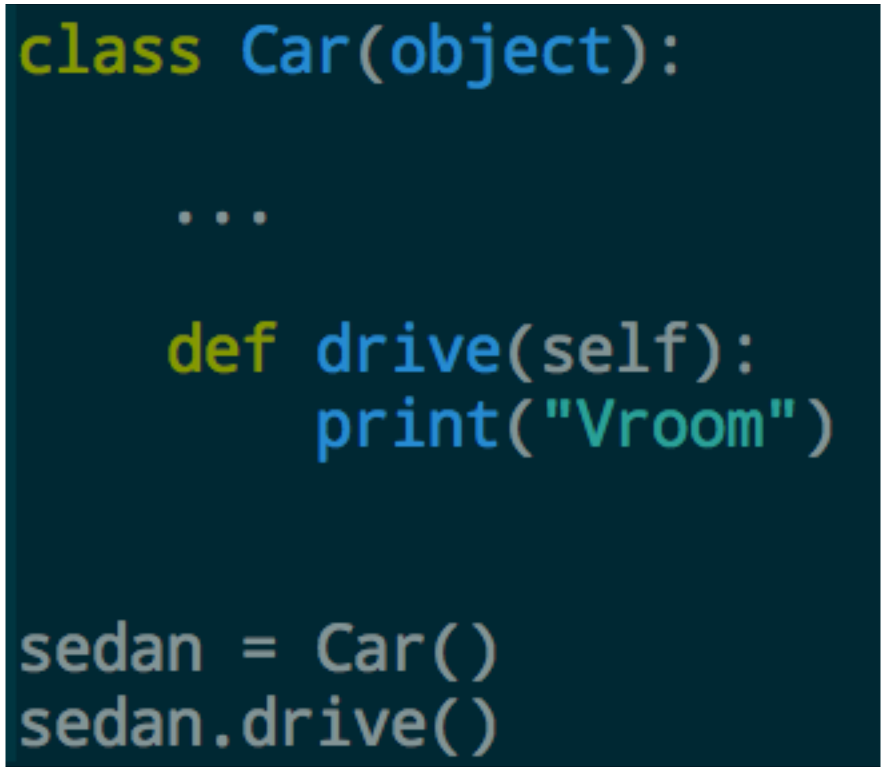
- **Instance attributes take precedence** over class attributes

- However, **new instance defaults to the class attributes** unless they are changed in the constructor or somehow modified elsewhere.

Methods

A **bound method** combines a function and an instance

Dot expressions used to pass in an instance into “self”



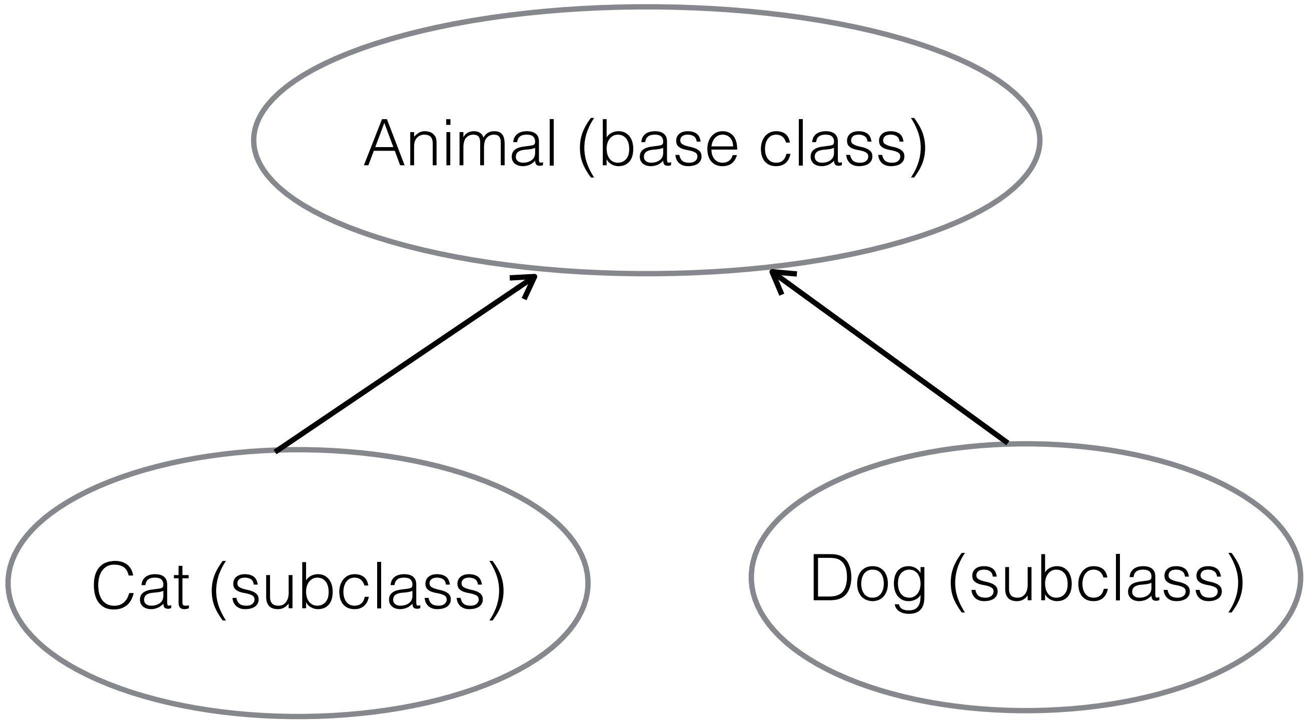
```
class Car(object):

    ...

    def drive(self):
        print("Vroom")


sedan = Car()
sedan.drive()
```

sedan is implicitly “self” →

Inheritance

Write once, reuse forever

Reuse code by **applying “is-a” relationships**



Cat **is an** Animal and Dog **is an** Animal but Cat is not a Dog

Inheritance

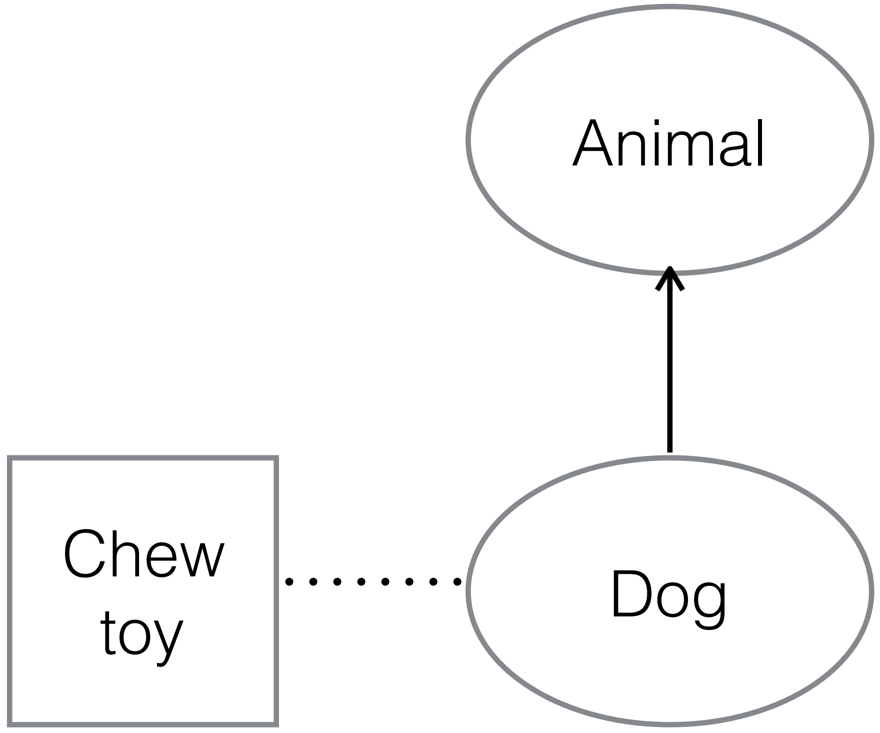
Can access/use **attributes** and **methods** from your parent class

- Don’t have to use them, can choose to **override**
- However, **parent’s behavior is present by default**

Inheritance

Beware: not everything should be inherited ("is-a")!

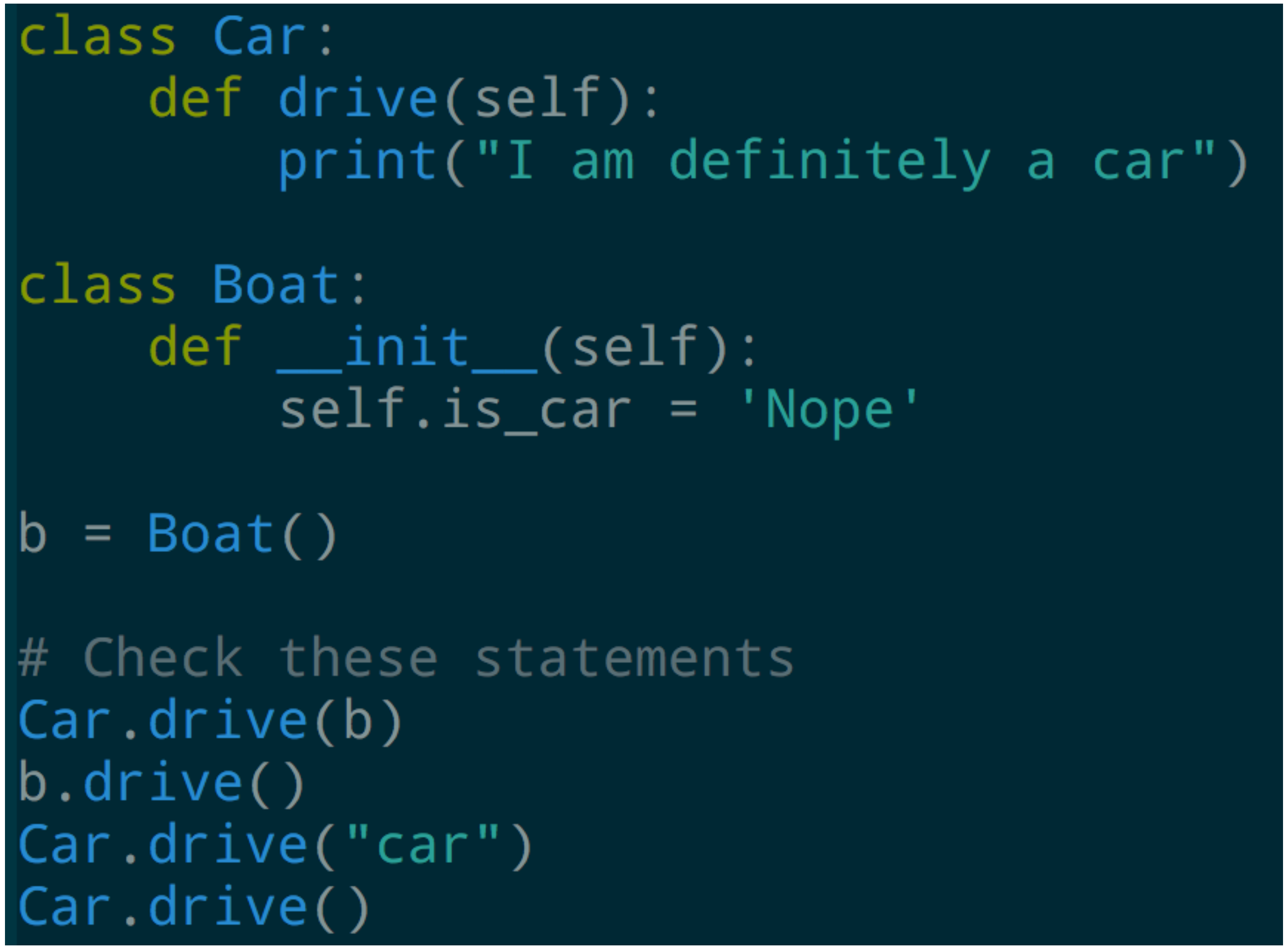
Sometimes, composition or **"has-a"** relationships are better.



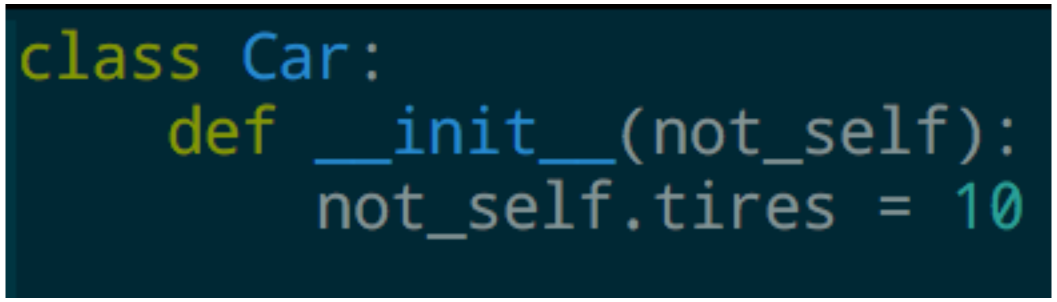
Dog **is an** Animal and **has a** chew toy.

Odds & Ends

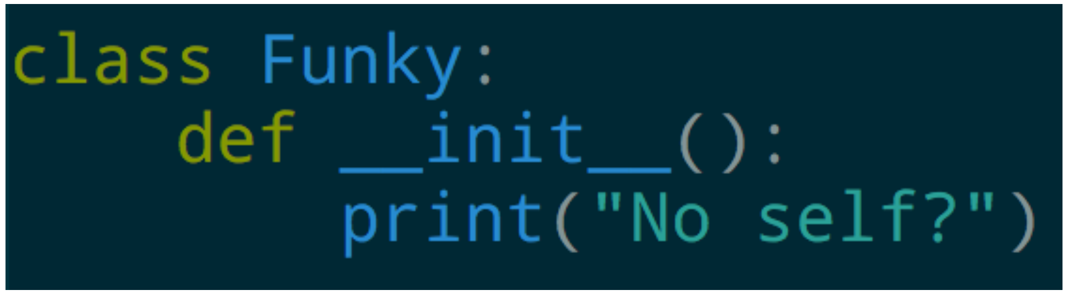
Which of the following are ok?



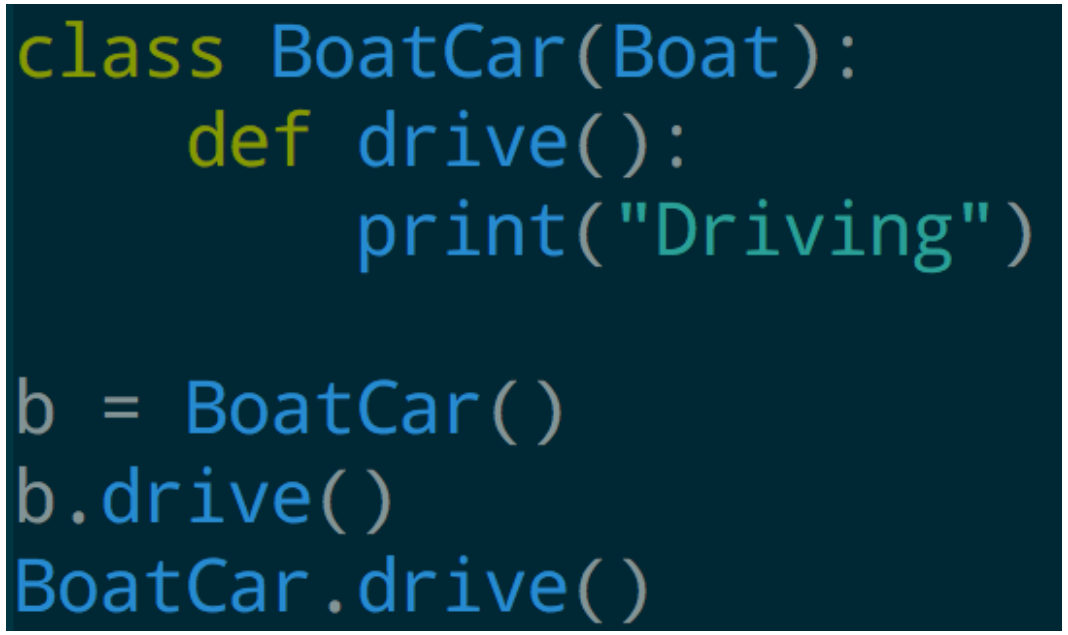
```
class Car:
    def drive(self):
        print("I am definitely a car")

class Boat:
    def __init__(self):
        self.is_car = 'Nope'

b = Boat()

# Check these statements
Car.drive(b)
b.drive()
Car.drive("car")
Car.drive()
```



```
class Car:
    def __init__(not_self):
        not_self.tires = 10
```



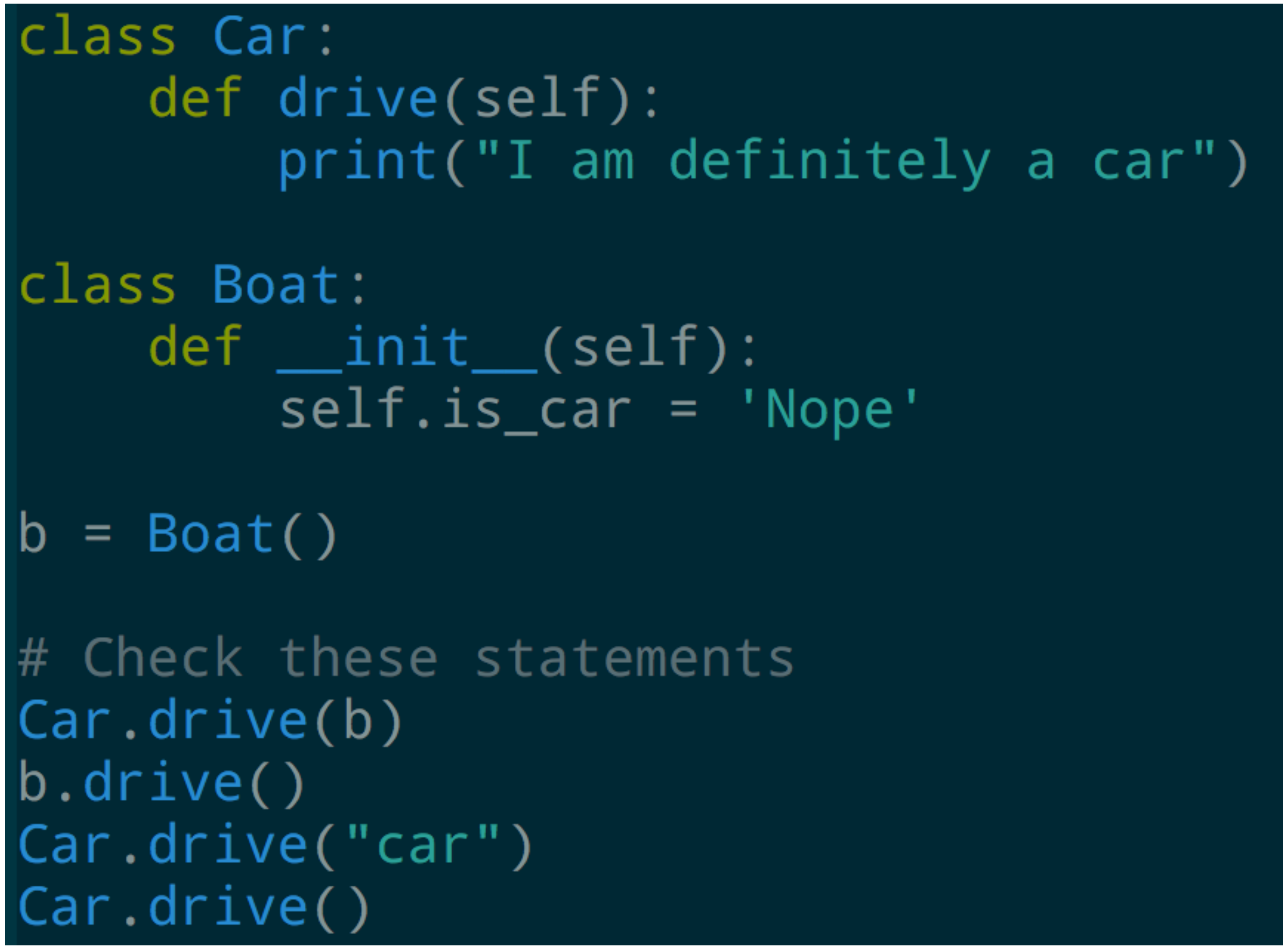
```
class Funky:
    def __init__():
        print("No self?")
```



```
class BoatCar(Boat):
    def drive():
        print("Driving")

b = BoatCar()
b.drive()
BoatCar.drive()
```

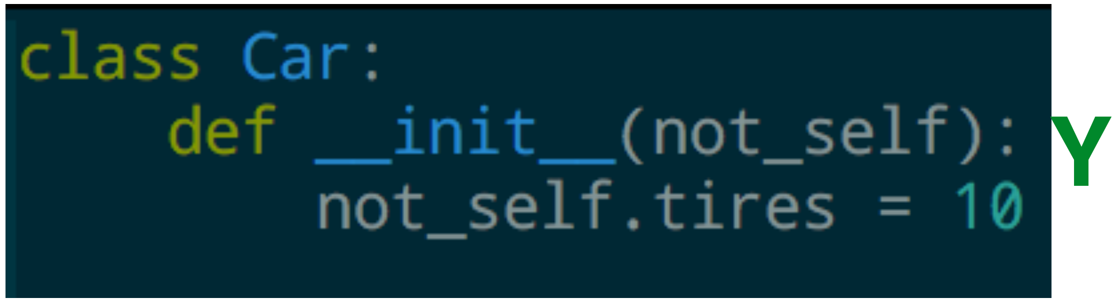
Odds & Ends

Which of the following are ok?



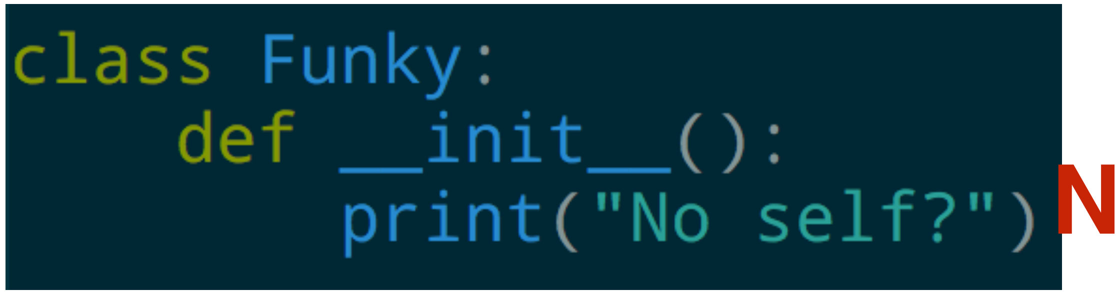
```python
class Car:
    def drive(self):
        print("I am definitely a car")

class Boat:
    def __init__(self):
        self.is_car = 'Nope'

b = Boat()

# Check these statements
Car.drive(b)
b.drive()
Car.drive("car")
Car.drive()
```

Y
N
Y
N



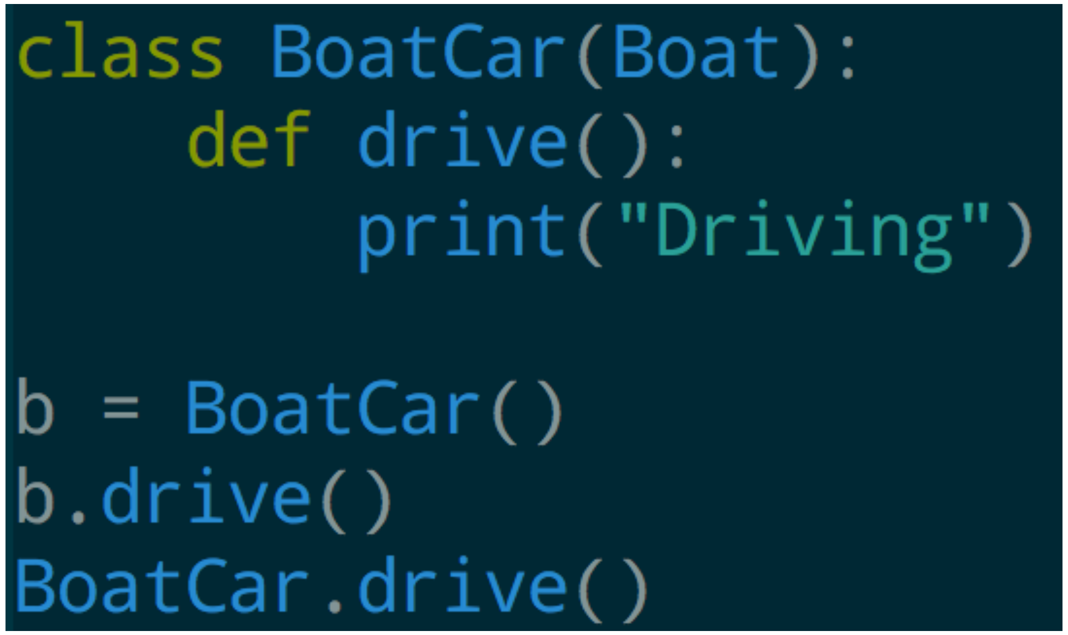
```python
class Car:
    def __init__(not_self):
        not_self.tires = 10
```

Y



```python
class Funky:
    def __init__():
        print("No self?")
```

N



```python
class BoatCar(Boat):
    def drive():
        print("Driving")

b = BoatCar()
b.drive()
BoatCar.drive()
```

N
Y