#5 Object Oriented Programming

TA: Jerry Chen (jerry.c@berkeley.edu)

Did you hear about the new object oriented get-rich-quick scheme? It's called "inheritance."
Who would win in a fight? Hog or Maps?
28 responses

- Hog: 64.3%
- Maps: 25%
- Ants: 5.3%
- Tom: 4.3%
- Neither :( 1.4%
I. Most Popular Technologies

<table>
<thead>
<tr>
<th>Language</th>
<th>2016</th>
<th>2015</th>
<th>2014</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>55.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL (or SQL Server)</td>
<td>49.1%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>36.3%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#</td>
<td>30.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>25.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>24.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>19.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>15.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node.js</td>
<td>17.2%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AngularJS</td>
<td>17.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ruby</td>
<td>8.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective-C</td>
<td>6.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

49,397 responses
OOP in Python

Some vocabulary

- Class - a "template" for an object
- Object - an instance of a class
- Attribute - data
  - **Instance attributes** are specific to an object
  - **Class attributes** are shared with objects of the class
- Method - a function that has been bound to an object
class Instructor:

degree = "PhD (Magic)"

def __init__(self, name):
    self.name = name

def lecture(self, topic):
    print("Today we're learning about " + topic)
class Instructor:

degree = "PhD (Magic)"

def __init__(self, name):
    self.name = name

def lecture(self, topic):
    print("Today we're learning about " + topic)


dumbledore = Instructor("Dumbledore")
class Instructor:
    degree = "PhD (Magic)"
    def __init__(self, name):
        self.name = name
    def lecture(self, topic):
        print("Today we're learning about " + topic)

dumbledore = Instructor("Dumbledore")
class Instructor:

degree = "PhD (Magic)"

def __init__(self, name):
    self.name = name

def lecture(self, topic):
    print("Today we're learning about " + topic)


dumbledore = Instructor("Dumbledore")
class Instructor:

degree = "PhD (Magic)"

def __init__(self, name):
    self.name = name

def lecture(self, topic):
    print("Today we're learning about " + topic)

dumbledore = Instructor("Dumbledore")

dumbledore.lecture("Magic")
class Instructor:
    degree = "PhD (Magic)"
    def __init__(self, name):
        self.name = name
    def lecture(self, topic):
        print("Today we're learning about " + topic)

dumbledore = Instructor("Dumbledore")
dumbledore.lecture("Magic")
class Car:
    def drive(self):
        print("I am definitely a car")

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

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

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

b = Boat()

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

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

b = Boat()

Car.drive(b)  # ✔️
b.drive()  # ✗
Car.drive("car")
Car.drive()
```python
class Car:
    def drive(self):
        print("I am definitely a car")

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

b = Boat()

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

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

b = Boat()

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("I am definitely... a boatcar")

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar")

b = BoatCar()
b.drive()
BoatCar.drive()  # deskewed
```
class Car:
    def __init__(not_self):
        not_self.tires = 10

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar")

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar")

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

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar"

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar"

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar")

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

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

class BoatCar(Boat):
    def drive():
        print("I am definitely... a boatcar")

b = BoatCar()
b.drive()  # ✔️
BoatCar.drive()  # ❌
```
class Car:
    def __init__(self, not_self):
        not_self.tires = 10

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

class BoatCar(Boat):
    def drive(self):
        print("I am definitely... a boatcar")

b = BoatCar()
b.drive()