Contact

Dr. James Shackleford
shack@drexel.edu
Bossone 211

Office Hours: 3 – 4 pm (Wednesday)
Course Website: http://learn.dcollege.net

Textbook (for this review)

Think Python
by Allen Downey
O'Reilly Press, 2015
ISBN-13: 978-1449330729
(Freely available in PDF format, check course website)

Grading

(subject to change)

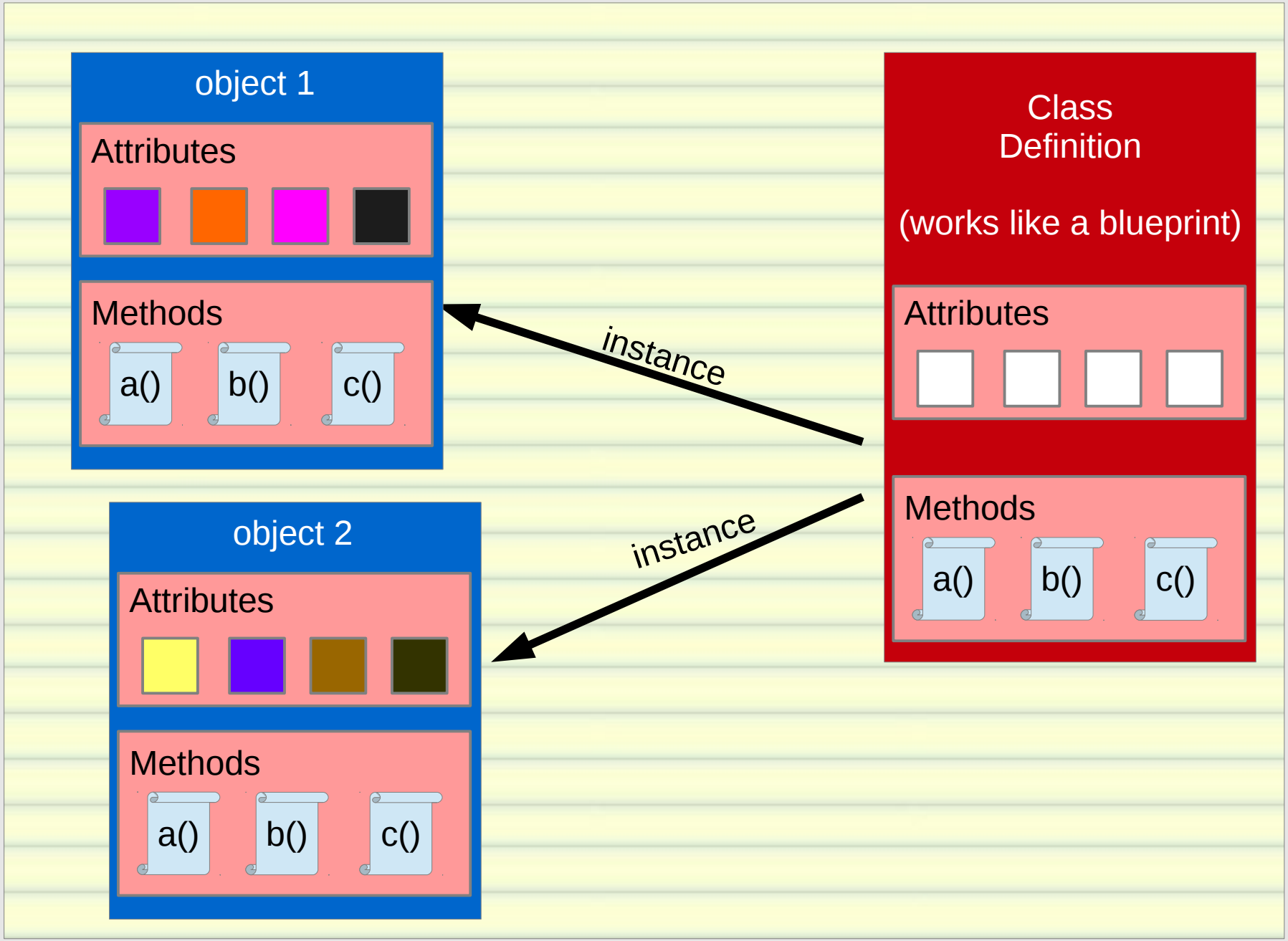
- 30% In-lab Programming Assignments
- 30% Take-Home Programming Assignments
- 40% Programming Projects

New Topic

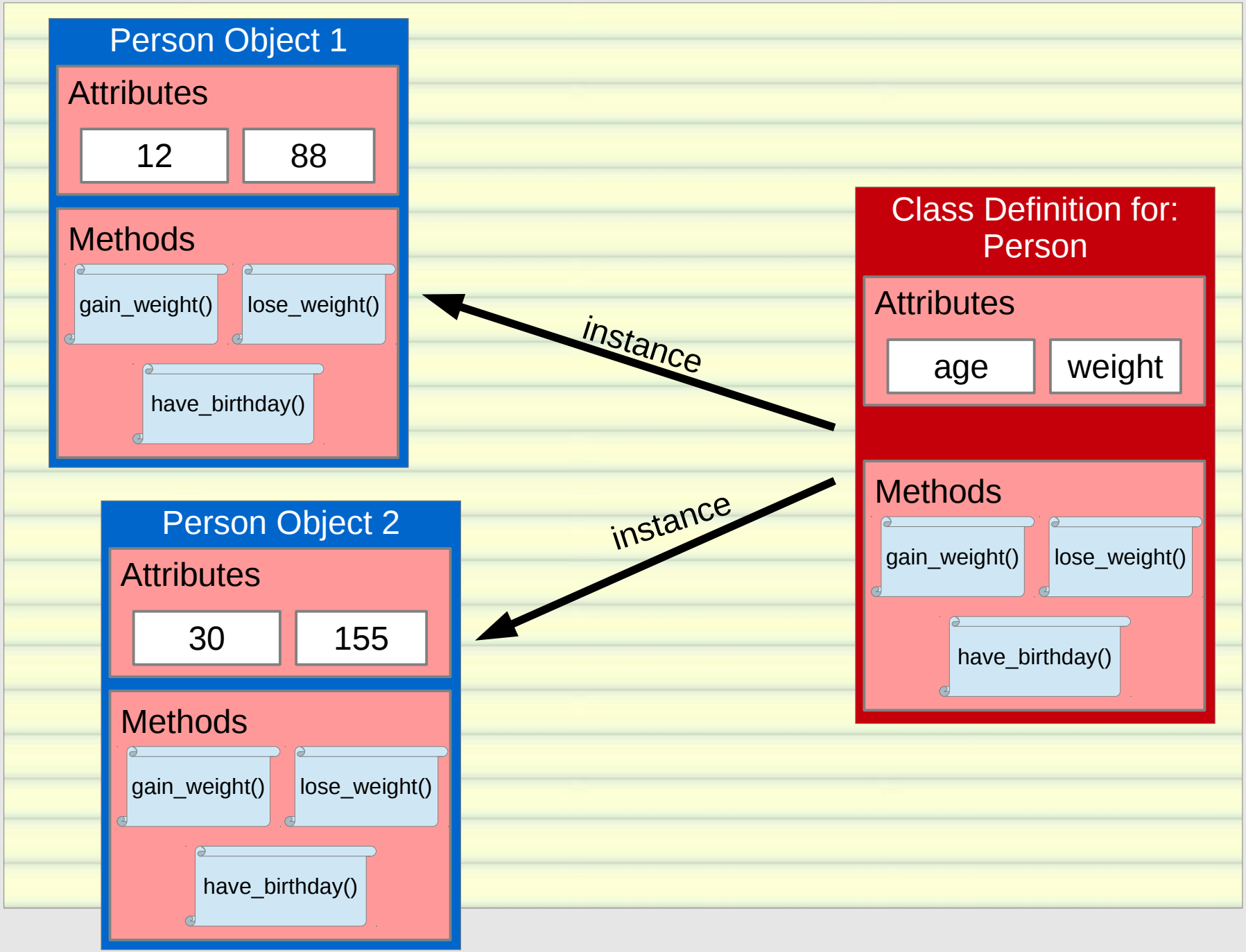
Review of basic
Object Oriented Programming

Classes, Objects, Instances,
Methods, and Attributes

The Big Idea


object 1
Attributes
Methods
a()
b()
c()
object 2
Attributes
Methods
a()
b()
c()
instance
instance
Class
Definition
(works like a blueprint)
Attributes
Methods
a()
b()
c()

The Big Idea


Person Object 1
Attributes
12
88
Methods
gain_weight()
lose_weight()
have_birthday()
instance
Class Definition for: Person
Attributes
age
weight
Methods
gain_weight()
lose_weight()
have_birthday()
instance
Person Object 2
Attributes
30
155
Methods
gain_weight()
lose_weight()
have_birthday()

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

class name

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

class docstring

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

methods

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

some instances

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

called when object is **created**

...or...

instantiated

Class Definition

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

Class Definition

???

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

Class Definition

this is the *instance*!

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

Instance/Object Attributes

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)
```

these are instance ***attributes***

Initializing Objects

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
```

we are
initializing them

Accessing Object Attributes

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
26
27 print joe.age
28 print bob.weight
```

accessing attributes is easy

Calling Methods

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
```

calling methods is also easy

Calling Methods

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

btw. this is the same (but uncommon)

Calling Methods

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

here we are explicitly passing the instance

Calling Methods

```python
class Person:
    """
    This is the docstring for the Person class!!

      The Person class (loosely) represent a person.
    """

    def __init__(self, age, weight):
        self.age = age
        self.weight = weight

    def gain_weight(self, amount):
        """Add 'amount' weight to a Person object"""
        self.weight += amount

    def lose_weight(self, amount):
        """Remove 'amount' weight from a Person object"""
        self.weight -= amount

    def have_birthday(self):
        """Increase the age of a Person object by 1"""
        self.age += 1

joe = Person(12, 88)
bob = Person(30, 155)

joe.have_birthday()
Person.have_birthday(joe)
```

here, it is passed implicitly

Calling Methods

```
1  class Person:
2      """
3      This is the docstring for the Person class!!
4
5        The Person class (loosely) represent a person.
6      """
7
8      def __init__(self, age, weight):
9          self.age = age
10         self.weight = weight
11
12     def gain_weight(self, amount):
13         """Add 'amount' weight to a Person object"""
14         self.weight += amount
15
16     def lose_weight(self, amount):
17         """Remove 'amount' weight from a Person object"""
18         self.weight -= amount
19
20     def have_birthday(self):
21         """Increase the age of a Person object by 1"""
22         self.age += 1
23
24 joe = Person(12, 88)
25 bob = Person(30, 155)
26
27 bob.lose_weight(5)
28 Person.lose_weight(bob, 5)
```

same stuff going on here, but with an argument

“Magic Methods”

(other than `__init__`)

Common “Magic Methods”

Expression	“Magic” Method	Returns	Description
x + y	`__add__(self, y)`	object	Addition
x - y	__sub__(self, y)	object	Subtraction
x * y	__mul__(self, y)	object	Multiplication
x / y	__truediv__(self, y)	object	Division
x % y	__mod__(self, y)	object	Modulus
x ** y	__pow__(self, y)	object	Exponentiation
x == y	__eq__(self, y)	Boolean	Equal
x != y	__ne__(self, y)	Boolean	Not Equal
x < y	__lt__(self, y)	Boolean	Less Than
x <= y	__le__(self, y)	Boolean	Less Than OR Equal
x > y	__gt__(self, y)	Boolean	Greater Than
x >= y	__ge__(self, y)	Boolean	Greater Than OR Equal
-x	__neg__(self, y)	object	Unary minus
abs(x)	__abs__(self, y)	object	Absolute value
float(x)	__float__(self)	float	Convert to float
int(x)	__int__(self)	int	Convert to int
str(x)	__repr__(self)	string	Convert to string

Note: y is always to the “right hand side”

Common “Magic Methods”

Expression	“Magic” Method	Returns	Description
x + y	`__add__(self, y)`	object !!!	Addition
x - y	__sub__(self, y)	object	Subtraction
x * y	__mul__(self, y)	object	Multiplication
x / y	__truediv__(self, y)	object	Division
x % y	__mod__(self, y)	object	Modulus
x ** y	__pow__(self, y)	object	Exponentiation
x == y	__eq__(self, y)	Boolean	Equal
x != y	__ne__(self, y)	Boolean	Not Equal
x < y	__lt__(self, y)	Boolean	Less Than
x <= y	__le__(self, y)	Boolean	Less Than OR Equal
x > y	__gt__(self, y)	Boolean	Greater Than
x >= y	__ge__(self, y)	Boolean	Greater Than OR Equal
-x	__neg__(self, y)	object	Unary minus
abs(x)	__abs__(self, y)	object	Absolute value
float(x)	__float__(self)	float	Convert to float
int(x)	__int__(self)	int	Convert to int
str(x)	__repr__(self)	string !!!	Convert to string

Note: y is always to the “right hand side”

Common “Magic Methods”

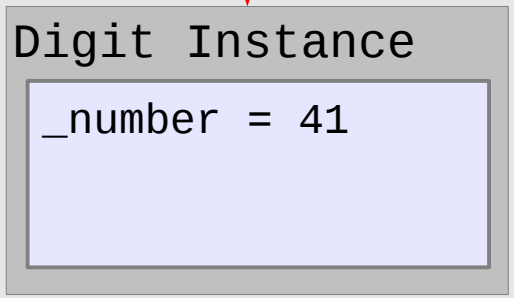
```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```

Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```

Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```



Common “Magic Methods”

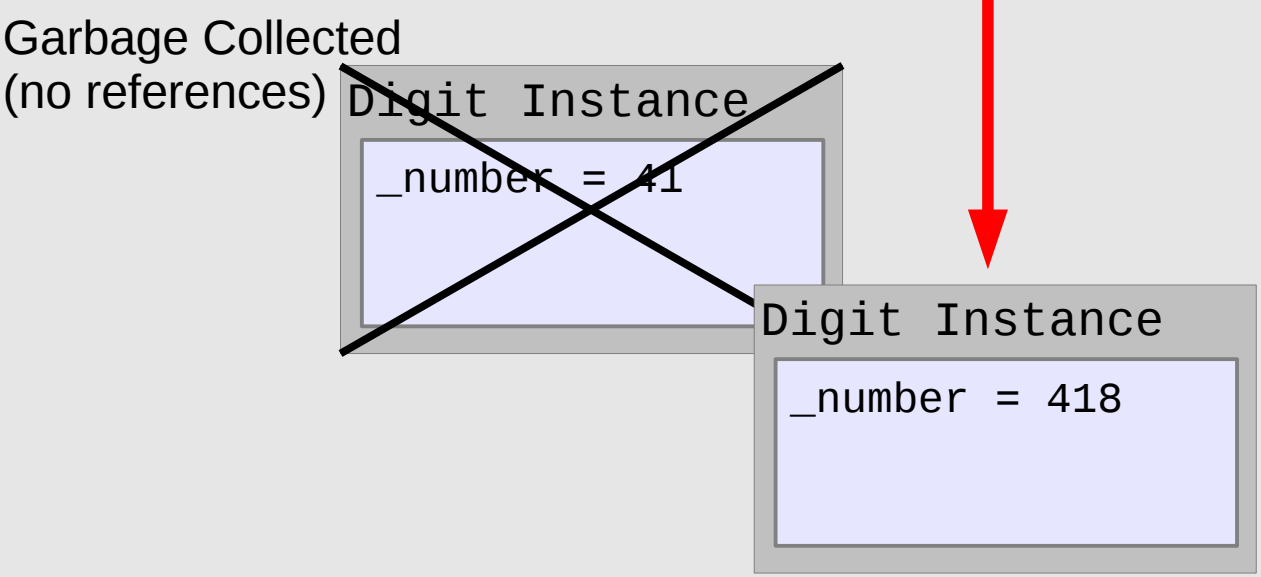
```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```

Digit Instance

_number = 41

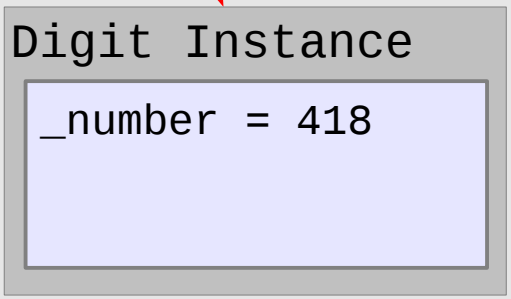
Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```



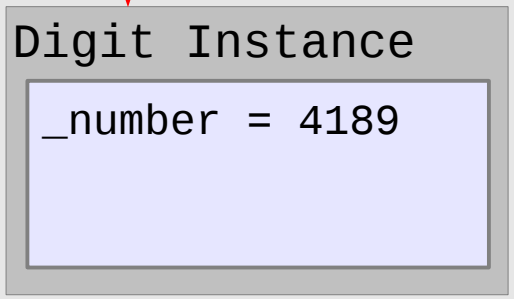
Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```



Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```



Done.

...almost. We still need to print the answer!

Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```

Digit Instance

_number = 4189

Common “Magic Methods”

```
1  class Digit(object):
2      def __init__(self, number):
3          self._number = number
4
5      def __repr__(self):
6          return str(self._number)
7
8      def __add__(self, right):
9          result = Digit(10*self._number + right._number)
10         return result
11
12 print Digit(4) + Digit(1) + Digit(8) + Digit(9)
13
14 # Output:
15 # 4189
```

print displays this return value (must be a string)

Digit Instance

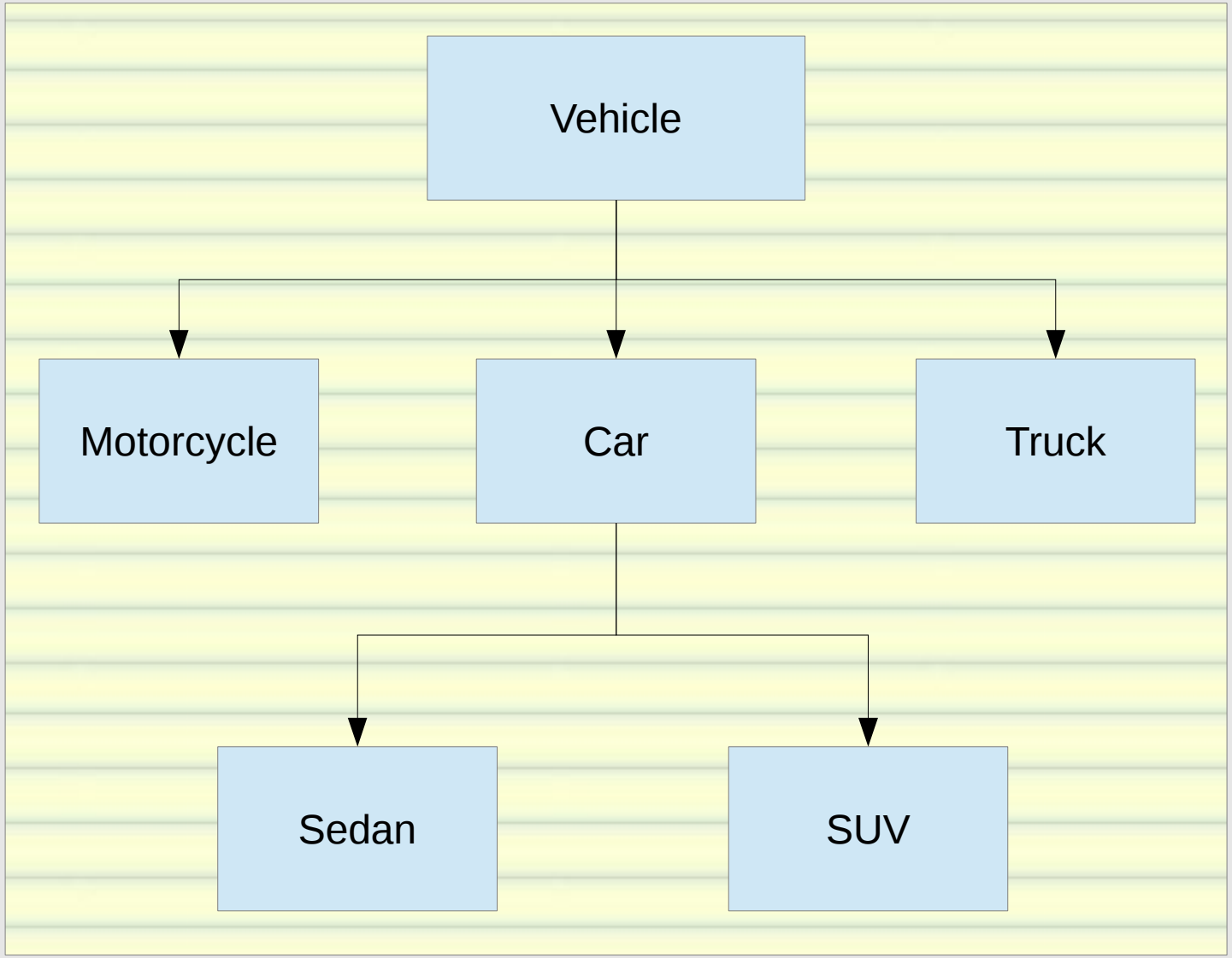
_number = 4189

Inheritance

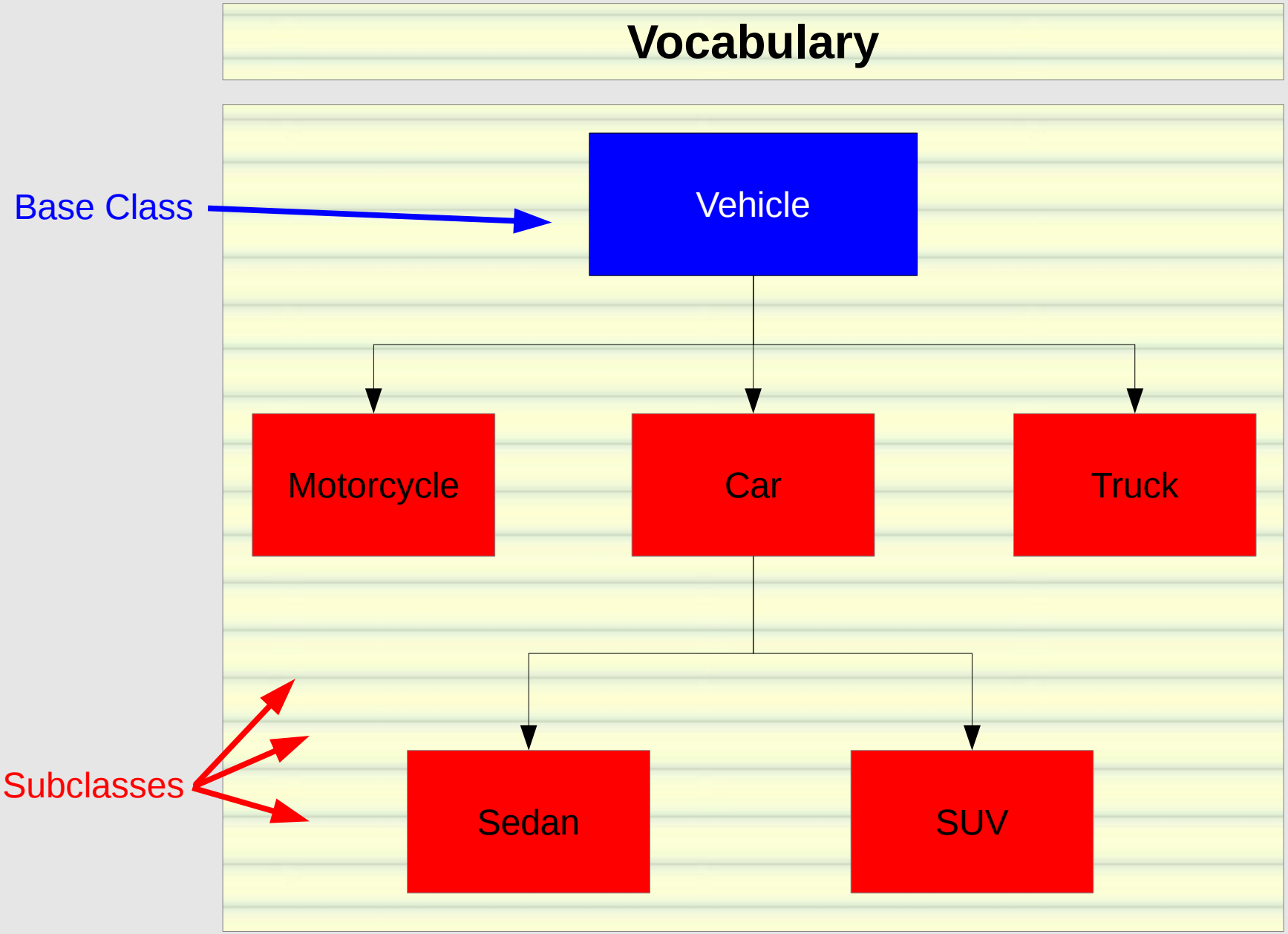
“Basing new classes off of other classes”

Inheritance – “Basing new classes off of other classes”

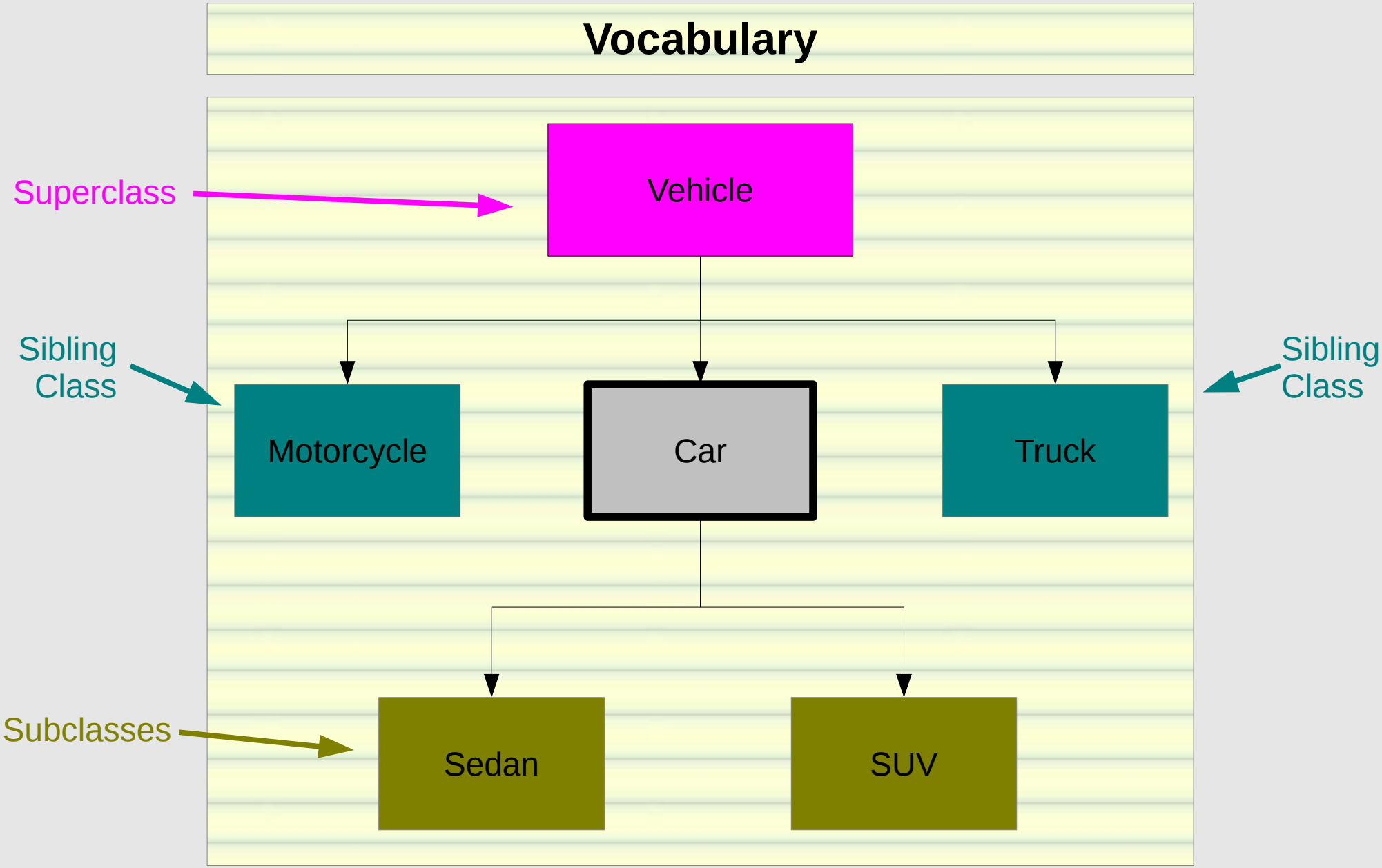
Allows for the creation of “Class Hierarchies”



Inheritance – “Basing new classes off of other classes”

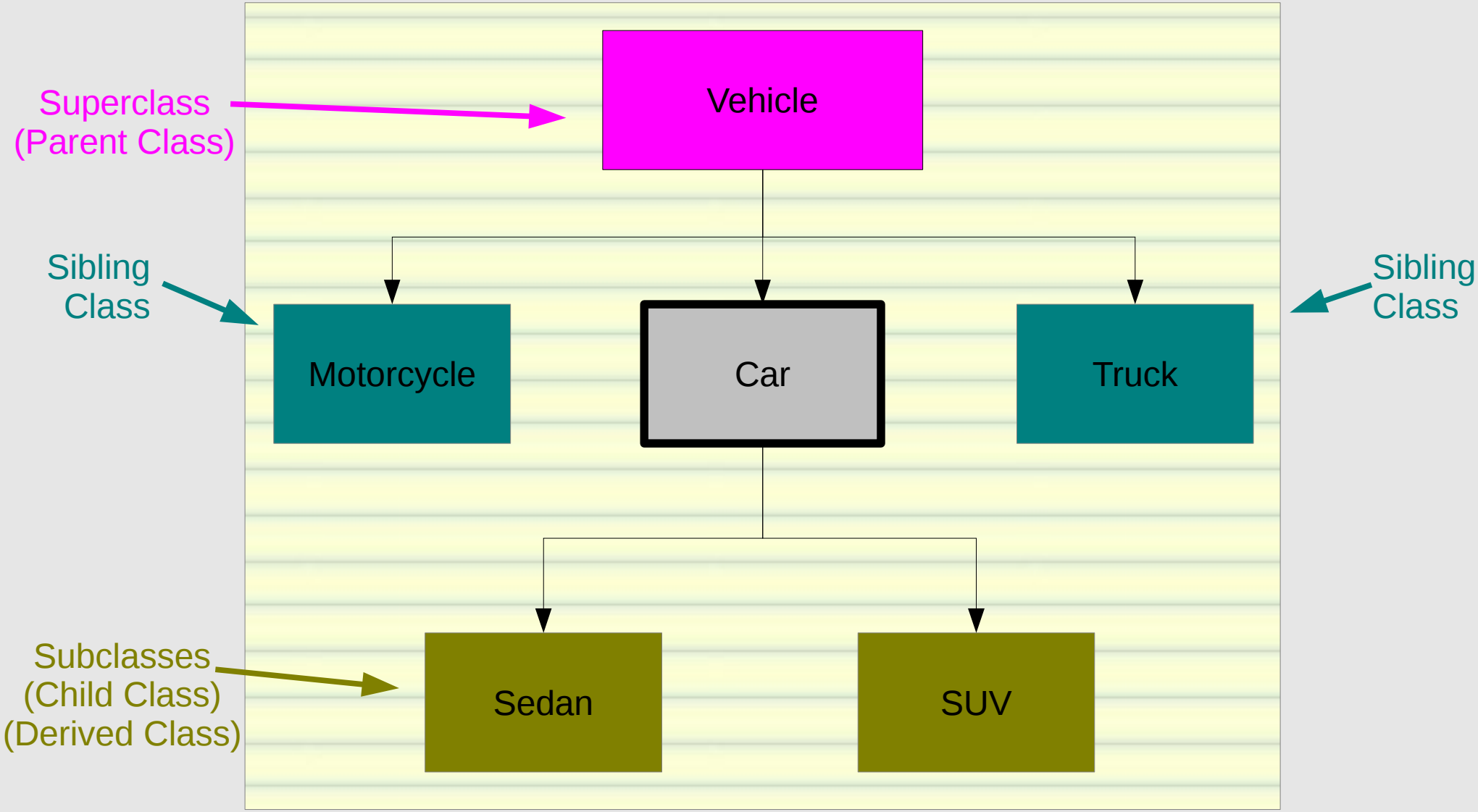


Inheritance – “Basing new classes off of other classes”



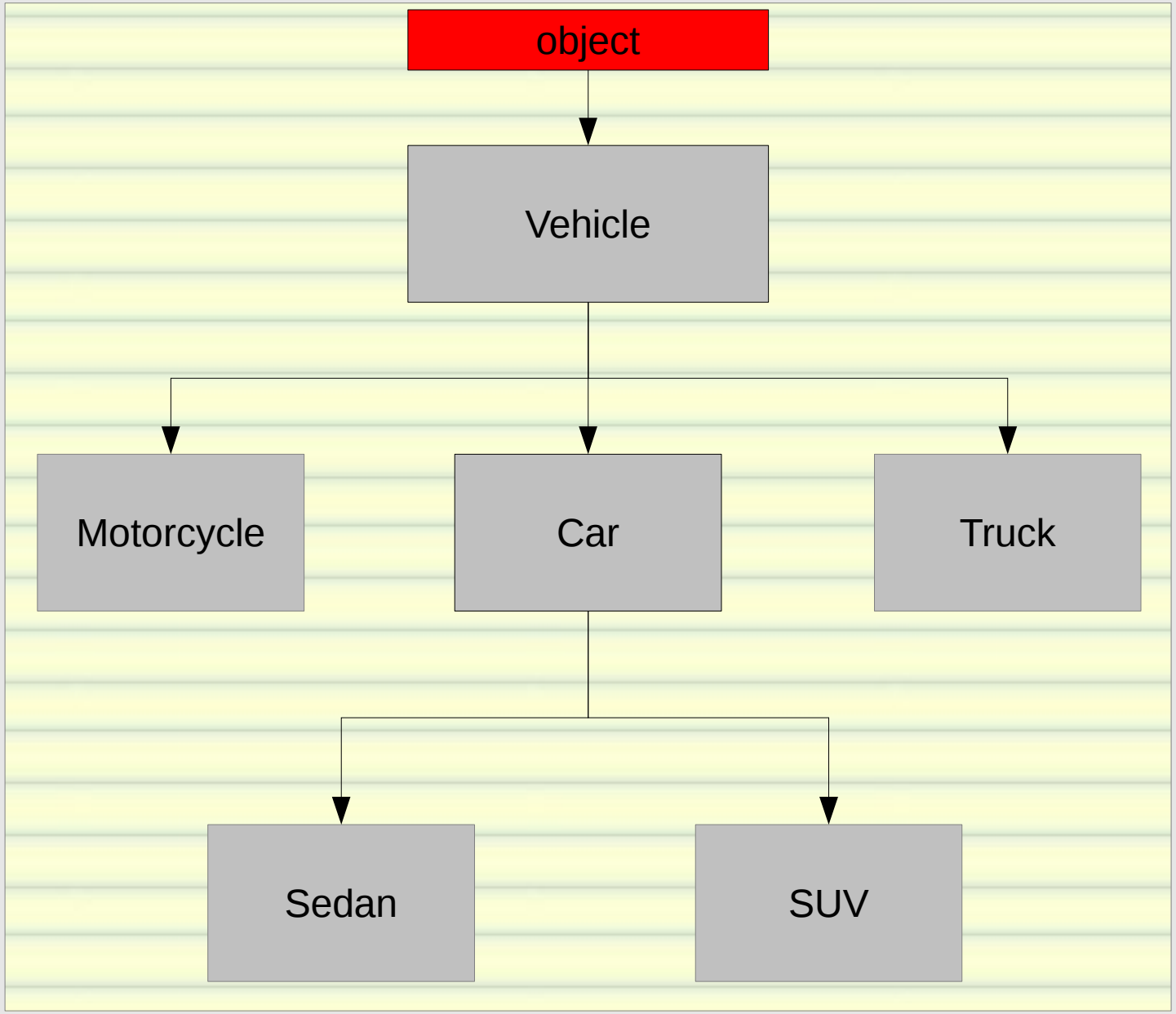
Inheritance – “Basing new classes off of other classes”

Also Seen Vocabulary



Inheritance – “Basing new classes off of other classes”

Everything is derived from *object*



Designing the Base Class

Vehicle inherits from object

Constructor

Methods inherited by subclasses

Methods overridden by subclasses

```
1  class Vehicle(object):
2      def __init__(self, numberOfTires):
3          self._numberOfTires = numberOfTires
4
5      def getNumberOfTires(self):
6          return self._numberOfTires
7
8      def setNumberOfTires(self, numberOfTires):
9          self._numberOfTires = numberOfTires
10
11     def getDescription(self):
12         return "A vehicle with %i tires" % self._numberOfTires
```

Implementing a Subclass

```
1  class Vehicle(object):
2      def __init__(self, numberOfTires):
3          self._numberOfTires = numberOfTires
4
5      def getNumberOfTires(self):
6          return self._numberOfTires
7
8      def setNumberOfTires(self, numberOfTires):
9          self._numberOfTires = numberOfTires
10
11     def getDescription(self):
12         return "A vehicle with %i tires" % self._numberOfTires
13
14 class Car(Vehicle):
15     def __init__(self):
16         super(Car, self).__init__(4)
17         self._plateNumber = None
18
19     def setLicensePlate(self, plateNumber):
20         self._plateNumber = plateNumber
21
22     def getDescription(self):
23         return "A CAR with %i tires" % self._numberOfTires
```

Car inherits from Vehicle

Constructor

Additional method

Method overridden by subclasses

What methods can a Car instance use?

```
1  class Vehicle(object):
2      def __init__(self, numberOfTires):
3          self._numberOfTires = numberOfTires
4
5      def getNumberOfTires(self):
6          return self._numberOfTires
7
8      def setNumberOfTires(self, numberOfTires):
9          self._numberOfTires = numberOfTires
10
11     def getDescription(self):
12         return "A vehicle with %i tires" % self._numberOfTires
13
14 class Car(Vehicle):
15     def __init__(self):
16         super(Car, self).__init__(4)
17         self._plateNumber = None
18
19     def setLicensePlate(self, plateNumber):
20         self._plateNumber = plateNumber
21
22     def getDescription(self):
23         return "A CAR with %i tires" % self._numberOfTires
```

Inheritance – “Basing new classes off of other classes”

Parent constructors are not called automatically!

```
1  class Vehicle(object):
2      def __init__(self, numberOfTires):
3          self._numberOfTires = numberOfTires
4
5      def getNumberOfTires(self):
6          return self._numberOfTires
7
8      def setNumberOfTires(self, numberOfTires):
9          self._numberOfTires = numberOfTires
10
11     def getDescription(self):
12         return "A vehicle with %i tires" % self._numberOfTires
13
14 class Car(Vehicle):
15     def __init__(self):
16         super(Car, self).__init__(4)
17         self._plateNumber = None
18
19     def setLicensePlate(self, plateNumber):
20         self._plateNumber = plateNumber
21
22     def getDescription(self):
23         return "A CAR with %i tires" % self._numberOfTires
```

Manual call to Superclass constructor(s)

Example: Assuming our classes are in vehicle.py

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from vehicle import Vehicle
>>> from vehicle import Car
>>> myVehicle = Vehicle(2)
>>> myVehicle.getDescription()
'A vehicle with 2 tires'
>>> myVehicle.setNumberOfTires(100)
>>> myVehicle.getNumberOfTires()
100
>>> myVehicle.getDescription()
'A vehicle with 100 tires'
>>> myCar = Car()
>>> myCar.getDescription()
'A CAR with 4 tires'
>>> myCar.setNumberOfTires(6)
>>> myCar.getNumberOfTires()
6
>>> myCar.getDescription()
'A CAR with 6 tires'
```

import our classes

Example: Assuming our classes are in vehicle.py

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from vehicle import Vehicle
>>> from vehicle import Car
>>> myVehicle = Vehicle(2)
>>> myVehicle.getDescription()
'A vehicle with 2 tires'
>>> myVehicle.setNumberOfTires(100)
>>> myVehicle.getNumberOfTires()
100
>>> myVehicle.getDescription()
'A vehicle with 100 tires'
>>> myCar = Car()
>>> myCar.getDescription()
'A CAR with 4 tires'
>>> myCar.setNumberOfTires(6)
>>> myCar.getNumberOfTires()
6
>>> myCar.getDescription()
'A CAR with 6 tires'
```

- ★ instantiate a Vehicle with 2 tires
- ★ get its description
- ★ set its # of tires to 100
- ★ read back the # of tires
- ★ get its new description

Example: Assuming our classes are in vehicle.py

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from vehicle import Vehicle
>>> from vehicle import Car
>>> myVehicle = Vehicle(2)
>>> myVehicle.getDescription()
'A vehicle with 2 tires'
>>> myVehicle.setNumberOfTires(100)
>>> myVehicle.getNumberOfTires()
100
>>> myVehicle.getDescription()
'A vehicle with 100 tires'
>>> myCar = Car()
>>> myCar.getDescription()
'A CAR with 4 tires'
>>> myCar.setNumberOfTires(6)
>>> myCar.getNumberOfTires()
6
>>> myCar.getDescription()
'A CAR with 6 tires'
```

- ★ instantiate a Car with 2 tires
- ★ get its description (overridden method)
- ★ set its # of tires to 100 (inherited method)
- ★ read back the # of tires (inherited method)
- ★ get its new description (overridden method)