CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding
Feb 25 2020, Lecture 11

Agenda

- Agenda:
  - Object Oriented Programming

Inheritance

- Inheritance allows us to define a class that inherits all the method and properties from another class

- **Parent class** is the class being inherited from, also called the base class

- **Child class** is the class that inherits from another class, also called derived class

Inheritance

- An example

```python
class Car():
    """A simple attempt to represent a car."""

    def __init__(self, make, model, year):
        """Initialize attributes to describe a car."""
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0


    def get_descriptive_name(self):
        """Return a neatly formatted descriptive name."""
        long_name = str(self.year) + " " + self.make + " " + self.model
        return long_name.title()

    def read_odometer(self):
        """Print a statement showing the car's mileage"""
        print("This car has " + str(self.odometer_reading) + " miles on it.")

my_new_car = Car("bmw", "m3", 2016)
print(my_new_car.get_descriptive_name())
my_new_car.read_odometer()
```

```python
class ElectricCar(Car):
    """Represent aspects of a car, specific to electric vehicles."""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class."""
        super().__init__(make, model, year)


my_tesla = ElectricCar("tesla", "model s", 2019)
print(my_tesla.get_descriptive_name())
```

Inheritance

- The __init__() method for a Child Class

  - when you add the __init__() function, the child class will no longer inherit the parent's __init__() function

  - To initialize the inherited attributes, use super().__init__()

```
41    class ElectricCar(Car):
42        """Represent aspects of a car, specific to electric vehicles."""
43
44        def __init__(self, make, model, year):
45            """Initialize attributes of the parent class."""
46            super().__init__(make, model, year)
47
48
49    my_tesla = ElectricCar("tesla", "model s", 2019)
50    print(my_tesla.get_descriptive_name())
```

Note: The fact is the child's __init__() function override inheritance from the parent's __init__() function

Inheritance

- But, since the child class inherits all the attributes from the parent, what if we do not use super().__init__(), but do the following instead?

```python
41    class ElectricCar(Car):
42        """Represent aspects of a car, specific to electric vehicles."""
43
44        def __init__(self, make, model, year):
45            """Initialize attributes of the parent class."""
46            self.make = make
47            self.model = model
48            self.year = year
49
50
51    my_tesla = ElectricCar("tesla", "model s", 2019)
52    print(my_tesla.get_descriptive_name())
```

Inheritance

- Add new attributes and methods for the child class to differentiate the child class from the parent class

```python
40
41  class ElectricCar(Car):
42      """Represent aspects of a car, specific to electric vehicles."""
43
44      def __init__(self, make, model, year):
45          """Initialize attributes of the parent class."""
46          super().__init__(make, model, year)
47          self.battery_size = 70
48
49      def describe_battery(self):
50          """Print a statement describing the battery size."""
51          print("This car has  a " + str(self.battery_size) + "-kWh battery.")
52
53
54  my_tesla = ElectricCar("tesla", "model s", 2019)
55  print(my_tesla.get_descriptive_name())
56  my_tesla.describe_battery()
57
```

```
classes ×
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week5/classes.py
2019 Tesla Model S
This car has  a 70-kWh battery.

Process finished with exit code 0
```

Inheritance

- Overriding methods from the parent class

  - when you think it does not fit for the child class

```python
class ElectricCar(Car):
    """Represent aspects of a car, specific to electric vehicles."""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class."""
        super().__init__(make, model, year)
        self.battery_size = 70

    def describe_battery(self):
        """Print a statement describing the battery size."""
        print("This car has  a " + str(self.battery_size) + "-kWh battery.")

    def fill_gas_tank(self):
        """Electric cars don't have gas tanks"""
        print("This car doesn't need a gas tank")
```

Inheritance

- Instances as attributes

```python
class Battery():
    """A simple attempt to model a batter of a Electric Car"""

    def __init__(self, battery_size = 70):
        """Initialize the batter's attributes"""
        self.battery_size = battery_size

    def describe_battery(self):
        """Print a statement describing the battery size"""
        print("This car has a " + str(self.battery_size) + "-kwh battery")

class ElectricCar(Car):
    """Represent aspects of a car, specific to electric car"""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class
        Then initialize attributes specific to an electric car
        """
        super().__init__(make, model, year)
        self.battery = Battery()

my_tesla = ElectricCar("tesla", "model s", "2019")

my_tesla.battery.describe_battery()
```