

# CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding  
Sep 30 2019, Lecture 15



# Announcement

---

- Quiz 1 grade posted
  - Average: 28.66/50
  - Highest : 50/50

# Agenda

---

- Agenda:
  - Quiz Solutions
  - Object Oriented Programming (OOP)
    - Inheritance, access control

# 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

```
1 class Car():
2     """A simple attempt to represent a car."""
3
4     def __init__(self, make, model, year):
5         """Initialize attributes to describe a car."""
6         self.make = make
7         self.model = model
8         self.year = year
9         self.odometer_reading = 0
10
11
12     def get_descriptive_name(self):
13         """Return a neatly formatted descriptive name."""
14         long_name = str(self.year) + " " + self.make + " " + self.model
15         return long_name.title()
16
17     def read_odometer(self):
18         """Print a statement showing the car's mileage"""
19         print("This car has " + str(self.odometer_reading) + " miles on it.")
20
21 my_new_car = Car("bmw", "m3", 2016)
22 print(my_new_car.get_descriptive_name())
23 my_new_car.read_odometer()
```

```
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())
```

# Inheritance

---

- Overriding methods from the parent class
  - when you think it does not fit for the child class

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

# Inheritance

- Instances as attributes

```
66 class Battery():
67     """A simple attempt to model a batter of a Electric Car"""
68
69     def __init__(self, battery_size = 70):
70         """Initialize the batter's attributes"""
71         self.battery_size = battery_size
72
73     def describe_battery(self):
74         """Print a statement describing the battery size"""
75         print("This car has a " + str(self.battery_size) + "-kwh battery")
76
77 class ElectricCar(Car):
78     """Represent aspects of a car, specific to electric car"""
79
80     def __init__(self, make, model, year):
81         """Initialize attributes of the parent class
82         Then initialize attributes specific to an electric car
83         """
84         super().__init__(make, model, year)
85         self.battery = Battery()
86
87 my_tesla = ElectricCar("tesla", "model s", "2019")
88
89 my_tesla.battery.describe_battery()
```

