CS 142
Inheritance & Polymorphism

Rhodes College

Announcements

- Reminder
  - Program 2 has been assigned, due 2/10

Overriding Methods

```
petsList = [Dog("Spot"), Cat("Kitty"), Cat("Abbey"), Dog("Khan"), Pet("Tweety")]
for pet in petsList:
    print(pet.getName(), "says", end=' ')
    pet.makeSound()
```

Output:

```
Spot says Woof, woof!
Kitty says Meow
Abbey says Meow
Khan says Woof, woof!
Tweety says Grr...
```

Polymorphism

- Definition: An object's ability to take many forms.
- Polymorphic behavior:
  - Ability to define a method in a superclass and override it in a subclass
    - Subclass defines method with the same name
  - Ability to call the correct version of overridden method depending on the type of object that called for it

Polymorphism

```
def show_pet_info(aPet):
      print(aPet.getName(), "the", aPet.getSpecies(), "says", end=' ')
      aPet.makeSound()

def main():
      petList = [Dog("Spot", True), Cat("Kitty", False), Cat("Abbey", True),
                 Dog("Khan", False), Bird("Tweety", True)]
      petList.append(Pet("Harry", "Hamster"))

      for pet in petList:
           show_pet_info(pet)

main()
```

What happens if someone adds a line of code that says `petList.append("I am a String!")`

Output:

```
Spot the Dog says Woof, woof
Kitty the Cat says Meow
Abbey the Cat says Meow
Khan the Dog says Woof, woof
Tweety the Bird says chirp, chirp
Harry the Hamster says Grr....
```

Polymorphism

```
def show_pet_info(aPet):
      if isinstance(aPet, Pet):
            print(aPet.getName(), "the", aPet.getSpecies(), "says", end=' ')
            aPet.makeSound()
      else:
            print("That is not a Pet.")

def main():
      petList = [Dog("Spot", True), Cat("Kitty", False), Cat("Abbey", True),
                 Dog("Khan", False), Bird("Tweety", True)]
      petList.append(Pet("Harry", "Hamster"))

      petList.append("I am a String!")

      for pet in petList:
           show_pet_info(pet)


main()
```

isinstance function: determines whether object is an instance of a class
- –Format: `isinstance(`*`object`*`, `*`class`*`)`

Output:

```
Spot the Dog says Woof, woof
Kitty the Cat says Meow
Abbey the Cat says Meow
Khan the Dog says Woof, woof
Tweety the Bird says chirp, chirp
Harry the Hamster says Grr....
That is not a Pet.
```

Overriding Methods

The method can call the superclass equivalent and add to it, or do something completely different

```
class Pet(object):
    def __init__(self, name='', species=''):
        self.__name = name
        self.__species = species

    def makeSound(self):
        print("Grr...")

class Cat(Pet):
    def __init__(self, name, hasClaws):
        Pet.__init__(self, name, "Cat")
        self.__hasClaws = hasClaws

    def makeSound(self):
        print("Meow")

class Dog(Pet):
    def __init__(self, name, hatesCats):
        Pet.__init__(self, name, "Dog")
        self.__hatesCats = hatesCats

    def makeSound(self):
        print("Woof, woof")
```

```
class Automobile(object):
    def __init__(self, make, model, mileage, price):
        self.__make = make
        self.__model = model
        self.__mileage = mileage
        self.__price = price

    def __str__(self):
        return 'The following car is in inventory:\n' + \
               'Make: ' + self.__make + '\n' + \
               'Model: ' + str(self.__model) + '\n' + \
               'Mileage: ' + str(self.__mileage) + '\n' + \
               'Price: ' + str(self.__price) + '\n'

class Car(Automobile):
    def __init__(self, make, model, mileage, price, doors):
        Automobile.__init__(self, make, model, mileage, price)
        self.__doors = doors

    def __str__(self):
        return super().__str__() + "Number of doors: " + str(self.__doors) + "\n"
```

In-Class Activity

- – Work with a partner
- – Hand in 1 copy for each group
- – Counts as a quiz
  - Hand in at beginning of next class if you don't finish it today.