



22.4. Invoking the Parent Class's Method

Sometimes the parent class has a useful method, but you just need to execute a little extra code when running the subclass's method. You can override the parent class's method in the subclass's method with the same name, but also invoke the parent class's method. Here's how.

Say you wanted the `Dog` subclass of `Pet` to say "Arf! Thanks!" when the `feed` method is called, as well as executing the code in the original method.

Here's the original `Pet` class again.

```
1 from random import randrange
2
3 # Here's the original Pet class
4 class Pet():
5     boredom_decrement = 4
6     hunger_decrement = 6
7     boredom_threshold = 5
8     hunger_threshold = 10
9     sounds = ['Mrrip']
10    def __init__(self, name = "Kitty"):
11        self.name = name
12        self.hunger = randrange(self.hunger_threshold)
13        self.boredom = randrange(self.boredom_threshold)
14        self.sounds = self.sounds[:] # copy the class attribute, so that when we m
15
```

Activity: 1 -- ActiveCode (inheritance_pet_class_copy)

And here's a subclass that overrides `feed()` by invoking the the parent class's `feed()` method; it then also executes an extra line of code. Note the somewhat inelegant way of invoking the parent class' method. We explicitly refer to `Pet.feed` to get the method/function object. We invoke it with parentheses. However, since we are not invoking the method the normal way, with `<obj>.methodname`, we have to explicitly pass an instance as the first parameter. In this case, the variable `self` in `Dog.feed()` will be bound to an instance of `Dog`, and so we can just pass `self`: `Pet.feed(self)`.

```
1 from random import randrange
2
3 class Dog(Pet):
4     sounds = ['Woof', 'Ruff']
5
6     def feed(self):
7         Pet.feed(self)
8         print("Arf! Thanks!")
9
10 d1 = Dog("Astro")
11
12 d1.feed()
13
```

Arf! Thanks!

Activity: 2 -- ActiveCode (feed_me_example)

Note

There's a better way to invoke a superclass's method. Unfortunately, the implementation of python in our ActiveCode windows doesn't support it, so we aren't using it here. In that alternative method, we would call `super().feed()`. This is nice because it's easier to read, and also because it puts the specification of the class that `Dog` inherits from in just one place, `class Dog(Pet)`. Elsewhere, you just refer to `super()` and python takes care of looking up that the parent (super) class of `Dog` is `Pet`.

This technique is very often used with the `__init__` method for a subclass. Suppose that some extra instance variables are defined for the subclass. When you invoke the constructor, you pass all the regular parameters for the parent class, plus the extra ones for the subclass. The subclass' `__init__` method then stores the extra parameters in instance variables and calls the parent class' `__init__` method to store the common parameters in instance variables and do any other initialization that it normally does.

Let's say we want to create a subclass of `Pet`, called `Bird`, and we want it to take an extra parameter, `chirp_number`, with a default value of 2, and have an extra instance variable, `self.chirp_number`. Then, we'll use this in the `hi` method to make more than one sound.

```

1 class Bird(Pet):
2     sounds = ["chirp"]
3     def __init__(self, name="Kitty", chirp_number=2):
4         Pet.__init__(self, name) # call the parent class's constructor
5         # basically, call the SUPER -- the parent version -- of the constructor, w
6         self.chirp_number = chirp_number # now, also assign the new instance varia
7
8     def hi(self):
9         for i in range(self.chirp_number):
10             print(self.sounds[randrange(len(self.sounds))])
11             self.reduce_boredom()
12
13 b1 = Bird('tweety', 5)
14 b1.teach("Polly wanna cracker")
15 b1.hi()

```

```

Polly wanna cracker
chirp
Polly wanna cracker
Polly wanna cracker
Polly wanna cracker

```

Activity: 3 -- ActiveCode (super_methods_1)

Check your understanding

Intro-9-1: What will print when `print(b1.sounds)` is run?

- ☐ A. 5
- ☐ B. ["Mrp"]
- ☒ C. ["chirp"]
- ☐ D. Error

Check me

Compare me

✓ The interpreter finds the value in the class variable for the class Bird.

Activity: 4 -- Multiple Choice (question_inheritance_4)

Intro-9-2: For the Dog class defined in the earlier activecode window, what would happen when `d1.feed()` is run if the `Pet.feed(self)` line was deleted?

- ☐ A. Error when invoked
- ☐ B. The string would not print out but d1 would have its hunger reduced.
- ☒ C. The string would print but d1 would not have its hunger reduced.
- ☐ D. Nothing would be different. It is the same as the current code.

Check me

Compare me

✓ Since we are no longer calling the parent Pet class's method in the Dog subclass's method definition, the class definition will override the parent method.

Activity: 5 -- Multiple Choice (question_inheritance_5)

You have attempted 6 of 5 activities on this page

22.3. Overriding Methods">

22.5. Tamagotchi Revisited">

Overriding Methods">

✓ Completed. Well Done!

22.5. Tamagotchi Revisited">Next Section - 22.5. Tamagotchi Revisited