

Super Classes

Textbook

Super Classes



We can even create a class to put other classes into! This is called a super class. In this case, the children class (the classes inside the big class) inherit the attributes or behaviors from the parent class.

Let's explore an example of some simple shape equations to help us understand superclasses.

```
1 class Rectangle:
2     def __init__(self, width, height):
3         self.width = width
4         self.height = height
5
6     def area(self):
7         return self.width * self.height
8
9     def perimeter(self):
10        return 2 * self.width + 2 * self.height
11
12 class Square:
13     def __init__(self, width, height):
14         self.width = width
```

```

15     self.height = height
16
17     def area(self):
18         return self.width * self.height
19
20     def perimeter(self):
21         return 2 * self.width + 2 * self.height
22
23 myRectangle = Rectangle(3, 10)
24 myRectangle.area()
25
26 mySquare = Square(5, 5)
27 mySquare.area()
28

```

We have created two classes for finding the area and perimeter of a rectangle and a square. You'll notice that there is a lot of repeated code. This is because finding the area and perimeter of a square is a lot like finding the area a perimeter of a rectangle.

Wouldn't it be a good idea to just use the same few lines of code for both places?

This is a good situation to use inheritance. [Inheritance](#) allows a class to use properties of a different class.

Super Class Inheritance

Now let's use `super()` to condense how much code we need to write.

Notice that the class named `Square` has parentheses with the class name `Rectangle` inside. This allows this class to access the `Rectangle` class.

```

1 class Rectangle:
2     def __init__(self, width, height):
3         self.width = width
4         self.height = height
5
6     def area(self):
7         return self.width * self.height
8
9     def perimeter(self):
10        return 2 * self.width + 2 * self.height
11
12 class Square(Rectangle):
13     def __init__(self, width, height):
14         super().__init__(width, height)
15
16
17 mySquare = Square(5, 5)
18 mySquare.area()

```

Try it!

This will print out `25`.

Now, inside the class named `Square`, we still have the `__init__()` function which is always needed. But below that, we used `super().__init__(width, height)` after it. This line of code gives the class named `Square` access to the functions inside the class named `Rectangle`.

So now, at the bottom, we could create an object named `mySquare` by using the `Square` class. And we can also use the `area()` function from the class named `Rectangle`.

- The class named `Rectangle` is the superclass.
- The class named `Square` is the subclass.

This is an example of inheritance. The `Square` class inherited the `area()` function from the `Rectangle` class.

Multiple Inheritance

Classes can actually inherit content from multiple other classes! Let's say we wanted to find the area of a house shape. This house shape is a square and a triangle put together. The triangle is the same height as the square.



Let's say we have already created the code to find the area of the square and the area of the triangle. Now you need to find the area of this shape. It would be efficient to simply use the code you already made to find the area of the two separate shapes and combine them! This can be done with [multiple inheritance](#).

```
1 class Square:
2     def __init__(self, width, height):
3         self.width = width
4         self.height = height
5
6     def areaSquare(self):
7         return self.width * self.height
8
9     def perimeter(self):
10        return 2 * self.width + 2 * self.height
11
12 class Triangle():
```

```

13     def __init__(self, width, height):
14         self.width = width
15         self.height = height
16
17     def areaTriangle(self):
18         return self.width * self.height / 2
19
20
21 class House(Square, Triangle):
22     def __init__(self, width, height):
23         super().__init__(width, height)
24
25     def areaHouse(self):
26         areaBottom = super().areaSquare()
27         areaTop = super().areaTriangle()
28         return areaBottom + areaTop
29
30 myHouse = House(5, 5)
31 myHouse.areaHouse()
32

```

Try it!

So let's unpack this code a bit.

1. The class named `House` pulls information from the class named `Square` and the class named `Triangle`, as can be found in the parentheses after `House`.
2. Inside the class named house, we have the normal `__init__()` function with the parameters that match the parameters in the previous classes.
3. We also have the code to access other classes.
4. `super().__init__(width, height)`
5. The class named `House` also has another method named `areaHouse`. This method has some variables named `areaBottom` and `areaTop`. These variables are set equal to function calls in the other classes. These other functions are named `areaSquare()` and `areaTriangle()`. These values are then added together and returned.
6. Towards the bottom we create a new object called `myHouse` and set it equal to the class named `House` with specific parameters.
7. The last line of code then asks for the specific function named `areaHouse()` inside the class `myHouse` to run. which returns the area of the house shape.

Checkpoint

Super Classes

Practice creating a super class! First we will make 2 classes, and one will use a method from the other. These classes will help calculate free time once homework time has been taken away.

1. Create a class named `Time`.

2. Inside the class named `Time` , create the `__init__()` function with three parameters: `self` , `free` , and `homework` . Make sure to assign these values to self. These values will represent free time and time spent on homework.
3. Inside the class named `Time` , create a method named `evening()` . Inside the method named `evening()` , create a return statement that returns `self.free - self.homework` . Remember to include the `self` parameter in the method.
4. Create another class named `Student` . Make sure to include parentheses and include the class named `Time` inside. This makes the class named Time a super class. Now we can use methods from the class named `Time` .
5. Inside the class named `Student` , create the `__init__()` function with three parameters: `self` , `free` , and `homework` . Now we don't need to assign these to self because they are assigned to a different class.
6. Inside the `__init__()` function, include `super().__init__(free, homework)` . This means that we want to use methods from the designated super class.
7. After the two classes have been made, create an object named `wednesday` by using the class named `Student` and the parameters `(4, 3)` .
8. Finally call the method named `evening` on the object named `wednesday` .

Requirements:

- Create a class named Time.
- Inside the class named Time, create the `__init__()` function with three parameters: self, free, and homework. Make sure to assign these values to self. These values will represent free time and time spent on homework.
- Inside the class named Time, create a method named `evening()`. Inside the method named `evening()`, create a return statement that returns `self.free - self.homework`. Remember to include the self parameter in the method.
- Create another class named Student. Make sure to include parentheses and include the class named Time inside. This makes the class named Time a super class. Now we can use methods from the class named Time.
- Inside the class named Student, create the `__init__()` function with three parameters: self, free, and homework. Now we don't need to assign these to self because they are assigned to a different class.
- Inside the `__init__()` function, include `super().__init__(free, homework)`. This means that we want to use methods from the designated super class.
- After the two classes have been made, create an object named wednesday by using the class named Student and the parameters (4, 3).
- Finally call the method named evening on the object named wednesday.

Questions (3)

1. Which is the superclass in the following code?

MULTIPLE CHOICE

```
class Rectangle: def __init__(self, width, height): self.width = width self.height = height def area(self): return self.width * self.height def perimeter(self): return 2 * self.width + 2 * self.height class Square(Rectangle): def __init__(self, width, height): super().__init__(width, height) mySquare = Square(5, 5) mySquare.area()
```

Choose the correct answer:

- A. Square
- B. Rectangle
- C. area()
- D. perimeter()
- E. mySquare

2. What is inheritance?

MULTIPLE CHOICE

Choose the correct answer:

- A. Inheritance allows a class to use properties of a different class.
- B. Inheritance is when you don't need to make an object
- C. Inheritance allows a method to use an attribute from the object
- D. Inheritance allows one class to make multiple objects

3. True or False: Classes can inherit from multiple other classes.

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

Challenges (1)

1. Cube Volume

Use a superclass to find the volume of a cube. One class will be for a Square and the other class will be for a Cube.

1. Create a class named `Square`.
2. Inside the class named `Square`, create the `__init__()` function with two parameters: `self`, `length`. Make sure to assign the value of length to self. This will represent the length of a side of your square.
3. Inside the class named `Square`, create a method named `side()`. Inside the method named `side()`, create a return statement that simply returns `self.length`. Remember to include the `self` parameter in the method.
4. Inside the class named `Square`, create another method named `area()`. Inside the method named `area()`, create a return statement that returns the calculation for the area. Remember to include the `self` parameter in the method.
5. Inside the class named `Square`, create another method named `perimeter()`. Inside the method named `perimeter()`, create a return statement that returns the calculation for the perimeter (`self.length * 4`). Remember to include the `self` parameter in the method.
6. Create another class named `Cube`. Make sure to include parentheses and include the class named `Square` inside. This makes the class named `Square` a super class. Now we can use methods from the class named `Square`.
7. Inside the class named `Cube`, create the `__init__()` function with two parameters: `self`, and `length`. Now we don't need to assign these to self because they are assigned to a different class.
8. Inside the `__init__()` function, include `super().__init__(length)`. This means that we want to use methods from the designated super class.
9. After the two classes have been made, create an object named `myCube` by using the class named `Cube` and the parameter `(10)`.
10. Create a variable named `volume` and set it to the function call to get the area multiplied by the function call to get the side.
11. Print the variable named `volume`.

Requirements:

- Create a class named Square.
- Inside the class named Square, create the `__init__()` function with two parameters: `self`, `length`. Make sure to assign the value of length to self. This will represent the length of a side of your square.
- Inside the class named Square, create a method named `side()`. Inside the method named `side()`, create a return statement that simply returns `self.length`. Remember to include the `self` parameter in the method.
- Inside the class named Square, create another method named `area()`. Inside the method named `area()`, create a return statement that returns the calculation for the area. Remember to include the `self` parameter in the method.
- Inside the class named Square, create another method named `perimeter()`. Inside the method named `perimeter()`, create a return statement that returns the calculation for the perimeter (`self.length * 4`). Remember to include the `self` parameter in the method.
- Create another class named Cube. Make sure to include parentheses and include the class named Square inside. This makes the class named Square a super class. Now we can use methods from the class named Square.
- Inside the class named Cube, create the `__init__()` function with two parameters: `self`, and `length`. Now we don't need to assign these to self because they are assigned to a different class.
- Inside the `__init__()` function, include `super().__init__(length)`. This means that we want to use methods from the designated super class.
- After the two classes have been made, create an object named `myCube` by using the class named Cube and the parameter `(10)`.
- Create a variable named `volume` and set it to the function call to get the area multiplied by the function call to get the side.

- Print the variable named volume.

Answer Keys & Solutions

Checkpoint Solutions

Super Classes

```
1 class Time:
2     def __init__(self, free, homework):
3         self.free = free
4         self.homework = homework
5
6     def evening(self):
7         return self.free - self.homework
8
9 class Student(Time):
10     def __init__(self, free, homework):
11         super().__init__(free, homework)
12
13
14
15 wednesday = Student(4, 3)
16 wednesday.evening()
```

Questions

1. Which is the superclass in the following code?

MULTIPLE CHOICE

Correct Answer:

- A. Square ✗ Incorrect
- B. Rectangle ✓ Correct
- C. area() ✗ Incorrect
- D. perimeter() ✗ Incorrect
- E. mySquare ✗ Incorrect

Explanation:

The superclass is the class that other classes pull methods from.

2. What is inheritance?

MULTIPLE CHOICE

Correct Answer:

- A. Inheritance allows a class to use properties of a different class. ✓ Correct
- B. Inheritance is when you don't need to make an object ✗ Incorrect
- C. Inheritance allows a method to use an attribute from the object ✗ Incorrect
- D. Inheritance allows one class to make multiple objects ✗ Incorrect

Explanation:

Inheritance allows classes to use properties from other classes.

3. True or False: Classes can inherit from multiple other classes.

MULTIPLE CHOICE

Correct Answer:

- A. True ✓ Correct
- B. False ✗ Incorrect

Explanation:

This is called multiple inheritance

Challenges

1. Cube Volume

Solution:

```
1 class Square:
2     def __init__(self, length):
3         self.length = length
4
5     def side(self):
6         return self.length
7
8     def area(self):
9         return self.length * self.length
10
11    def perimeter(self):
12        return self.length * 4
13
14 class Cube(Square):
15     def __init__(self, length):
16         super().__init__(length)
17
18
19 myCube = Cube(10)
20 volume = myCube.area() * myCube.side()
```

21

22 `print(volume)`