lesson_10

September 27, 2022

1 Lesson 10 ~ Inheritance

[8]:
```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self, users_name):
        return f"Hello {users_name}, how are you?"

    def speak(self):
        return "Woof, woof!"
```

[9]:
```
dog_1 = Dog("Bob", 12)
```

[10]:
```
dog_1.greet("Henry")
```

[10]:
```
'Hello Henry, how are you?'
```

[11]:
```
class Cat:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self, users_name):
        return f"Hello {users_name}, how are you?"

    def speak(self):
        return "Meow, meow!"
```

[12]:
```
cat_1 = Cat("Luna", 5)
```

[13]:
```
cat_1.greet("Henry")
```

[13]:
```
'Hello Henry, how are you?'
```

[49]:
```python
class Animal:
    def __init__(self, name, age):
        self.name = name
        if age <= 0:
            raise ValueError("age can't be less than 0")
        self.age = age

    def greet(self, user_name):
        return f"Hello {user_name}, how are you?"
```

[50]:
```python
class Dog(Animal):
    def speak(self):
        return "Woof, woof!"
```

[51]:
```python
class Cat(Animal):
    def speak(self):
        return "Meow, meow!"
```

[52]:
```python
dog_2 = Dog("Jack", 24)
cat_2 = Cat("Tom", 12)
```

[53]:
```python
dog_2.greet("Henry")
```

[53]: 'Hello Henry, how are you?'

[54]:
```python
dog_2.speak()
```

[54]: 'Woof, woof!'

[55]:
```python
cat_2.speak()
```

[55]: 'Meow, meow!'

[56]:
```python
class BullDog(Dog):
    pass
```

[57]:
```python
dog_3 = BullDog("Adam", 42)
```

[58]:
```python
dog_3.speak()
```

[58]: 'Woof, woof!'

[59]:
```python
dog_3.greet("Henry")
```

[59]: 'Hello Henry, how are you?'

[88]:
```
class BullDog(Dog):
    def __init__(self, name, age, color):
        super().__init__(name, age)
        self.color = color

    def speak(self):  # override
        return "Haff, haff!"

    def greet(self, user_name):
        print("Starting")
        return super().greet(f"Mr. {user_name}")
```

[89]:
```
dog_4 = BullDog("Cooper", 12, "White")
```

[90]:
```
dog_4.name
```

[90]: 'Cooper'

[91]:
```
dog_4.color
```

[91]: 'White'

[92]:
```
dog_4.speak()
```

[92]: 'Haff, haff!'

[93]:
```
dog_4.greet("Henry")
```

```
Starting
```

[93]: 'Hello Mr. Henry, how are you?'

[94]:
```
a = 42
```

[97]:
```
type(a) is int
```

[97]: True

[99]:
```
isinstance(a, int)
```

[99]: True

[100]:
```
dog_1
```

[100]: <__main__.Dog at 0x105475af0>

[101]:
```
dog_4
```

[101]:
```
<__main__.BullDog at 0x1053e5f10>
```

[102]:
```
isinstance(dog_4, BullDog)
```

[102]:
```
True
```

[103]:
```
isinstance(dog_4, Dog)
```

[103]:
```
True
```

[104]:
```
isinstance(dog_4, Animal)
```

[104]:
```
True
```

[105]:
```
isinstance(dog_4, Cat)
```

[105]:
```
False
```

[106]:
```
isinstance(dog_4, object)
```

[106]:
```
True
```

[110]:
```
type(dog_4)
```

[110]:
```
__main__.BullDog
```

[111]:
```
isinstance(True, int)
```

[111]:
```
True
```

[112]:
```
type(True)
```

[112]:
```
bool
```

[114]:
```
from functools import total_ordering

@total_ordering
class Rectangle:
    def __init__(self, width, length):
        if width < 0 or length < 0:
            raise Exception("width and length should be positive numbers")
        self.width = width
        self.length = length

    def area(self):
        return self.calculate_area(self.width, self.length)
```

```
    @staticmethod
    def calculate_area(width, length):
        return width * length

    def __repr__(self):
        return f"Rectangle({self.width}, {self.length})"

    def __lt__(self, other):
        return self.area() < other.area()

    def __eq__(self, other):
        return self.area() == other.area()

    def __bool__(self):
        return self.area() > 0

    def __add__(self, other):
        if isinstance(other, (int, float)):
            return self.area() + other
        elif isinstance(other, self.__class__):
            return self.area() + other.area()
        else:
            raise TypeError(f"Can't add Rectangle with {other.__class__}")

    def __radd__(self, other):
        return self.__add__(other)

    def __iadd__(self, other):
        raise TypeError("+= not supported for this object")

    def __int__(self):
        return int(self.area())
```

[115]:
```
class Square(Rectangle):
    def __init__(self, width):
        super().__init__(width, width)
```

[116]:
```
sq = Square(10)
```

[117]:
```
sq
```

[117]:
```
Rectangle(10, 10)
```

[118]:
```
sq.area()
```

[118]:
```
100
```

[119]:
```
sq + sq
```

[119]:
```
200
```

[197]:
```
@total_ordering
class Shape:
    def area(self):
        raise NotImplementedError()

    @staticmethod
    def calculate_area(*args):
        raise NotImplementedError()

    def __lt__(self, other):
        return self.area() < other.area()

    def __eq__(self, other):
        return self.area() == other.area()

    def __bool__(self):
        return self.area() > 0

    def __add__(self, other):
        if isinstance(other, (int, float)):
            return self.area() + other
        elif isinstance(other, self.__class__):
            return self.area() + other.area()
        else:
            raise TypeError(f"Can't add Rectangle with {other.__class__}")

    def __radd__(self, other):
        return self.__add__(other)

    def __iadd__(self, other):
        raise TypeError("+= not supported for this object")

    def __int__(self):
        return int(self.area())

    def __repr__(self):
        value_mapping = [f'{attr_name}={attr_value}' for attr_name, attr_value␣
 ↪in vars(self).items()]
        return f"{self.__class__.__name__}({', '.join(value_mapping)})"
```

[198]:
```
class Rectangle(Shape):
    def __init__(self, width, length):
        if width < 0 or length < 0:
```

```
            raise Exception("width and length should be positive numbers")
        self.width = width
        self.length = length

    def area(self):
        return self.calculate_area(self.width, self.length)

    @staticmethod
    def calculate_area(width, length):
        return width * length
```

[199]:
```
rectangle_1 = Rectangle(12, 24)
```

[196]:
```
rectangle_1
```

[196]:
```
Rectangle(width=12, length=24)
```

[185]:
```
rectangle_1 + rectangle_1
```

[185]:
```
576
```

[186]:
```
import math


class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return self.calculate_area(self.radius)

    @staticmethod
    def calculate_area(radius):
        return math.pi * radius **2
```

[187]:
```
circle_1 = Circle(1)
```

[188]:
```
circle_1 + circle_1
```

[188]:
```
6.283185307179586
```

[189]:
```
circle_1
```

[189]:
```
Circle(radius=1)
```

[190]:
```
class Square(Rectangle):
    def __init__(self, width):
```

```
        super().__init__(width, width)
```

[191]:
```
sq = Square(10)
```

[192]:
```
sq
```

[192]:
```
Square(width=10, length=10)
```

[ ]:
```

```