lesson_8

September 4, 2022

1 Lesson 8 ~ Methods

[1]:
```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def speak(self):
        return "woof"
```

[2]:
```
dog_1 = Dog("Bob", 12)
```

[3]:
```
dog_1.speak()
```

[3]:
```
'woof'
```

[4]:
```
class Parrot:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def speak(self, persons_name):
        return f"Hello {persons_name}"
```

[5]:
```
parrot_1 = Parrot("Jack", 12)
```

[7]:
```
parrot_1.speak("Henry")
```

[7]:
```
'Hello Henry'
```

[12]:
```
class Parrot:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def speak(self, persons_name):
```

```
        return f"Hello {persons_name}, I'm {self.name} and I am {self.age}␣
↪years old"
```

[13]:
```
parrot_2 = Parrot("Kesha", 12)
```

[14]:
```
parrot_2.speak("Henry")
```

[14]:
```
"Hello Henry, I'm Kesha and I am 12 years old"
```

[15]:
```
parrot_2
```

[15]:
```
<__main__.Parrot at 0x121d6bc10>
```

[23]:
```
class Employee:
    company_name = "ACA"
    company_domain = "aca.am"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    @classmethod
    def get_company_info(cls):
        return f"This employee works at {cls.company_name} (website: {cls.
↪company_domain})"
```

[24]:
```
employee_1 = Employee("Adam Smith", 42)
```

[25]:
```
employee_1.get_company_info()
```

[25]:
```
'This employee works at ACA (website: aca.am)'
```

[26]:
```
Employee.get_company_info()
```

[26]:
```
'This employee works at ACA (website: aca.am)'
```

[27]:
```
Dog.speak()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [27], in <cell line: 1>()
----> 1 Dog.speak()

TypeError: speak() missing 1 required positional argument: 'self'
```

[28]:
```
employee_1.company_name = "Google"
```

[29]:
```
employee_1.get_company_info()
```

[29]:
```
'This employee works at ACA (website: aca.am)'
```

[45]:
```
class Pizza:
    def __init__(self, name, ingredients):
        self.name = name
        self.ingredients = ingredients

    @classmethod
    def make_margarita(cls):
        return cls(
            name="Margarita",
            ingredients=["mozarella", "tomato"],
        )

    @classmethod
    def make_pepperoni(cls):
        return cls(
            name="Pepperoni",
            ingredients=["pepperoni", "mozarella", "parmezan"],
        )
```

[46]:
```
custom_pizza = Pizza("ACA", ["pepperoni", "parmezan", "lori"])
```

[47]:
```
custom_pizza.ingredients
```

[47]:
```
['pepperoni', 'parmezan', 'lori']
```

[48]:
```
margarita = Pizza.make_margarita()
```

[49]:
```
margarita.ingredients
```

[49]:
```
['mozarella', 'tomato']
```

[50]:
```
class Rectangle:
    def __init__(self, width, length):
        self.width = width
        self.length = length

    def area(self):
        return self.width * self.length
```

[51]:
```
rectangle_1 = Rectangle(10, 12)
```

[52]:
```
rectangle_1.area()
```

[52]:
```
120
```

[61]:
```
class Rectangle:
    def __init__(self, width, length):
        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
```

[62]:
```
Rectangle.calculate_area(20, 50)
```

[62]:
```
1000
```

[63]:
```
import math
```

[64]:
```
class Cycle:
    def __init__(self, radius):
        self.radius = radius

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

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

[65]:
```
rectangle_2 = Rectangle(20, 50)
```

[66]:
```
rectangle_2.area()
```

[66]:
```
1000
```

1.1 Magic Methods

[67]:
```
class Employee:
    company_name = "ACA"
    company_domain = "aca.am"

    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
    @classmethod
    def get_company_info(cls):
        return f"This employee works at {cls.company_name} (website: {cls.
↪company_domain})"
```

[68]:
```
employee_1 = Employee("Adam Smith", 24)
```

[70]:
```
print(employee_1)
```

```
<__main__.Employee object at 0x12435d850>
```

[83]:
```
class Employee:
    company_name = "ACA"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self):
        return f"Employee: {self.name} (from {self.company_name})"
```

[84]:
```
employee = Employee("Adam Smith", 24)
```

[85]:
```
employee
```

[85]:
```
Employee: Adam Smith (from ACA)
```

[86]:
```
employee_1
```

[86]:
```
<__main__.Employee at 0x12435d850>
```

[87]:
```
employees = [employee, employee_1]
```

[88]:
```
employees
```

[88]:
```
[Employee: Adam Smith (from ACA), <__main__.Employee at 0x12435d850>]
```

[94]:
```
class Rectangle:
    def __init__(self, width, length):
        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):
        # area = self.area()
        # return f"Rectangle {area} m^2"
        return f"Rectangle({self.width}, {self.length})"
```

[95]:
```
rect = Rectangle(20, 50)
```

[96]:
```
rect
```

[96]:
```
Rectangle(20, 50)
```

[97]:
```
print(f"this is -> {rect}")
```

```
this is -> Rectangle(20, 50)
```

[98]:
```
str(rect)
```

[98]:
```
'Rectangle(20, 50)'
```

[141]:
```
from functools import total_ordering
```

[183]:
```
@total_ordering
class Rectangle:
    def __init__(self, width, length):
        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
```

[184]:
```
rectangle_1 = Rectangle(7, 23)
rectangle_2 = Rectangle(8, 21)
rectangle_3 = Rectangle(7, 24)
rectangle_4 = Rectangle(6, 21)
```

[185]:
```
rectangle_1.area() < rectangle_2.area()
```

[185]: True

[186]:
```
rectangle_1 < rectangle_2
```

[186]: True

[187]:
```
rectangle_1 == rectangle_2
```

[187]: False

[188]:
```
rectangle_1 > rectangle_2
```

[188]: False

[189]:
```
rectangle_2 == rectangle_3
```

[189]: True

[190]:
```
rectangle_1 > rectangle_4
```

[190]: True

[191]:
```
rectangle_1 <= rectangle_2
```

[191]: True

[192]:
```
rectangle_2 <= rectangle_3
```

[192]: True

[193]:
```
rectangle_1 >= rectangle_4
```

[193]: True

[194]:
```
bool(rectangle_3)
```

[194]: True

[195]:
```
rectangle_0 = Rectangle(0, 0)
```

[196]:
```
bool(rectangle_0)
```

[196]: False

[197]:
```
if rectangle_1:
    print("yes")
```

```
yes
```

[198]:
```
if rectangle_0:
    print("yes")
```

[ ]:
```

```