Classes and methods

Defining classes and methods

```
class ClassName:
    def method_name(self, other_parameters):
        body_of_method
```

Special methods

- Special methods start and end with `__`.
- Special methods have specific names, like `__init__` for the constructor or `__str__` for the conversion to string.
- The methods `__str__` and `__repr__` allow you to define human-readable and unambiguous string representations of your objects, respectively.
- By defining methods like `__eq__`, `__ne__`, `__lt__`, `__gt__`, `__le__`, and `__ge__`, you can control how *objects* of your *class* are compared.

With the `__init__` method:

???????????????? self.XXX

```
class Apple:
    def __init__(self, color, flavor):
        self.color = color
        self.flavor = flavor

honeycrisp = Apple("red", "sweet")
fuji = Apple("red", "tart")
print(honeycrisp.flavor)
print(fuji.flavor)
```

With the `__str__` method:

When you `print()` something, Python calls the object's `__str__()` method and outputs whatever that method returns

```
class Apple:
    def __init__(self, color, flavor):
        self.color = color
        self.flavor = flavor

    def __str__(self):
        return "an apple which is {} and {}".format(self.color, self.flavor)

honeycrisp = Apple("red", "sweet")
print(honeycrisp)

# prints "an apple which is red and sweet"
```

With the custom method

```
class Triangle:
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def area(self):
        return 0.5 * self.base * self.height
    def __add__(self, other):
        return self.area() + other.area()

triangle1 = Triangle(10, 5)
triangle2 = Triangle(6, 8)
print("The area of triangle 1 is", triangle1.area())
print("The area of triangle 2 is", triangle2.area())
print("The area of both triangles is", triangle1 + triangle2)
```

Revision #7
Created 2024-11-18 15:56:41 CST by A-Lang (Admin)
Updated 2024-11-18 16:28:58 CST by A-Lang (Admin)