

Object-Oriented Programming (OOP)

Making Game with Python (1)

Zhihong (John) Zeng & Andrew Zeng



Agenda

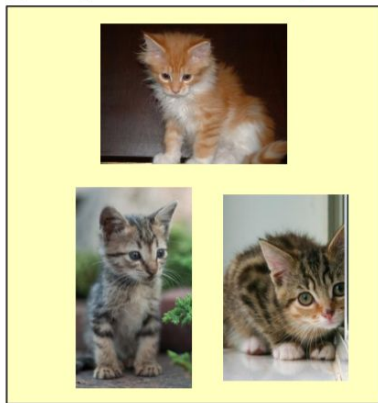
- What is OOP
- Class and Instance
- Example 1:
 - Animal Object
 - Cat Object
 - Rabbit Object
- Example 2:
 - Num Object

What is OOP?

- Mimic real life (object is used to bundle data into a package with attributes and functions)
- Group different objects in a hierarchical way

Animal (attribute: age, etc.; behavior/functions: speak, etc.)

Cat



Rabbit



Image Credits, clockwise from top: Image Courtesy [Harald Wehner](#), in the public Domain. Image Courtesy [MTSOfan](#), CC-BY-NC-SA. Image Courtesy [Carlos Solana](#), license CC-BY-NC-SA. Image Courtesy [Rosemarie Banghart-Kovic](#), license CC-BY-NC-SA. Image Courtesy [Paul Reynolds](#), license CC-BY. Image Courtesy [Kenny Louie](#), license CC-BY

Class and Instance

- Class:
 - Define class name
 - Define attributes
 - Define behavior/functions
- Instance:
 - Create a instance of a class
 - do operations on the instance

Animal Object

```
class Animal(object):  
    def __init__(self, age):  
        self.age = age  
  
    def speak(self):  
        print('not defined')  
  
    def __str__(self):  
        return f'Animal: age ({self.age})'  
  
obj_animal = Animal(1)  
print(obj_animal)  
obj_animal.speak()
```

Cat Object

```
class Cat(Animal):
    def __init__(self, age, color='BW'):
        super().__init__(age)
        self.color = color

    def speak(self):
        print('Meow')

    def __str__(self):
        return f'Cat: age ({self.age}), color ({self.color})'

obj_cat = Cat(2, 'Brown')
print(obj_cat)
obj_cat.speak()
```

Rabbit Object

```
class Rabbit(Animal):
    def __init__(self, age, weight):
        super().__init__(age)
        self.weight = weight

    def speak(self):
        print('Squeak')

    def __str__(self):
        return f'Rabbit: age ({self.age}), weight ({self.weight})'

obj_rabbit = Rabbit(1, '2LB')
print(obj_rabbit)
obj_rabbit.speak()
```

Special operators of python base object

- Operators: +, -, ==, <, >, len(), print
- Overwrite the following functions to change their behaviors
 - `__add__(self, other)`
 - `__sub__(self, other)`
 - `__eq__(self, other)`
 - `__lt__(self, other)`
 - `__len__(self)`
 - `__str__(self)`

Num Object

```
class Num(object):  
    def __init__(self, data):  
        self.data = data  
  
    def __str__(self):  
        return f'Num: {self.data}'  
  
    def __add__(self, other):  
        sum = self.data + other.data  
        return Num(sum)  
  
x = Num(3)  
print(x)  
y = Num(5)  
print(y)  
sum = x + y  
print(sum)
```