Python面向对象编程

类与继承

目录

- 类定义与实例
  - 示例1
  - 示例2

- 类继承
  - 基类与派生类
  - super方法
  - call方法

Python类

示例-1

Python类定义

```
class MyClass:
"""一个简单的类实例"""
    i = 12345
    def f(self):
        return 'hello world'

# 实例化类
x = MyClass()
# 访问类的属性和方法
print("MyClass 类的属性 i 为：", x.i)
print("MyClass 类的方法 f 输出为：", x.f())
```

类

属性

方法

self -> this指针

实例化为对象

x.f() -> MyClass.f(x)

Python类__init__函数()

Python 类的构造函数

```
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart

x = Complex(3.0, -4.5)
print(x.r, x.i) # 输出结果：3.0 -4.5
```

类

构造函数

调用构造函数

Python类

示例-2

类定义示例-Vehicle

类初始化（Initialize）：使用具体的参数初始化类（class）

获得一个实例（instance）

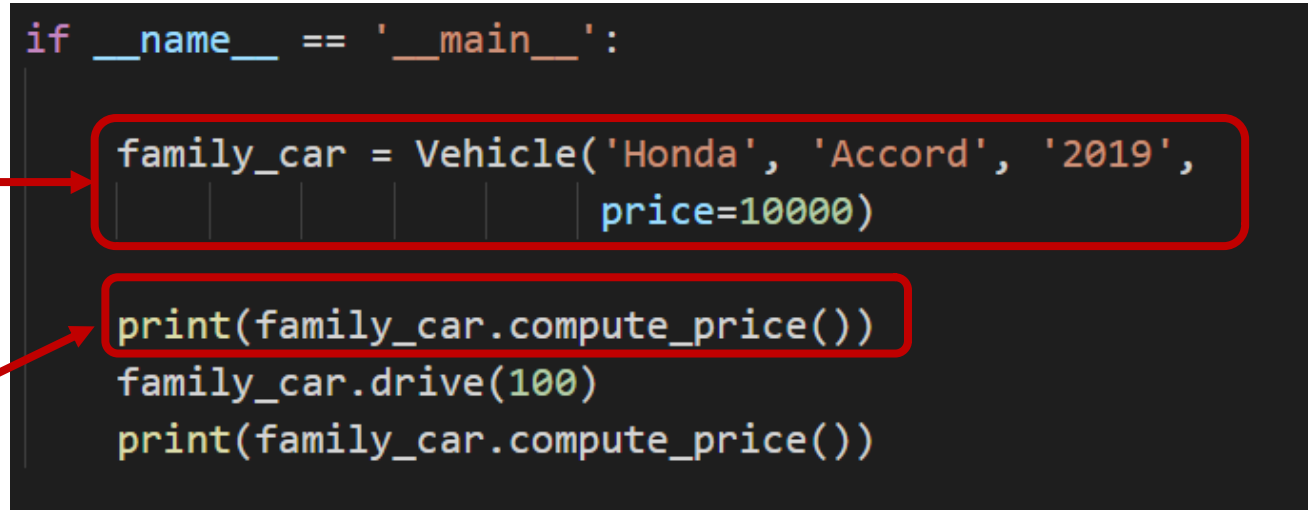
- 实例的变量（Instance variable）
- 实例的动作（Does something with the instance）

```python
class Vehicle:

    def __init__(self, make, name, year,
                 is_electric=False, price=100):
        self.name = name
        self.make = make
        self.year = year
        self.is_electric = is_electric
        self.price = price

        self.odometer = 0

    def drive(self, distance):
        self.odometer += distance

    def compute_price(self):
        if self.is_electric:
            price = self.price / (self.odometer * 0.8)
        else:
            price = self.price / self.odometer
        return price
```

类Class的使用

实例化（Instantiate）一个类，获得一个具体的实例（instance）

调用实例的方法（instance method）



```
if __name__ == '__main__':

    family_car = Vehicle('Honda', 'Accord', '2019',
                         price=10000)

    print(family_car.compute_price())
    family_car.drive(100)
    print(family_car.compute_price())
```

类定义示例-Dog

类初始化（Initialize）：使用具体的参数初始化类（class）

获得一个实例（instance）

实例的动作（Does something with the instance）

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

    def sit_down(self):
        print(self.name.title()+' is now sitting down!')
    def roll_over(self):
        print(self.name.title()+' rolled over!')
```

```
cmq_dog=Dog('tuan tuan',0.2)
print(cmq_dog.name)
cmq_dog.sit_down()
cmq_dog.roll_over()
```

想一想，练一练

- 有一只猫，有颜色color、有动作jump，如何用类来描述它呢？
- 实例化一只黑猫，跳了2步。
- 用ipynb文件记录一下你的python代码。
- 提示：1.必须有init函数

Python类的继承

示例-3

Python类继承-基类与派生类

基类/父类

```
class people:
    #定义构造方法
    def __init__(self,n,a):
        self.name = n
        self.age = a

    def speak(self):
        print("%s 说：我 %d 岁。" %(self.name,self.age))
```

派生类/子类

```
class student(people):
    def __init__(self,n,a,w,g):
        #调用父类的构函
        people.__init__(self,n,a)
        self.grade = g

    #覆写父类的方法
    def speak(self):
        print("%s 说：我 %d 岁了，我在读 %d 年级"%(self.name,self.age,self.grade))
```

覆写(overridden)父类的同名方法

用法：

```
s = student('ken',10,60,3)
s.speak()
```

Python类super() 方法

子类对象调用父类已被覆盖的方法

```
class Parent: # 定义父类
    def myMethod(self):
        print ('调用父类方法')

class Child(Parent): # 定义子类
    def myMethod(self):
        print ('调用子类方法')

c = Child() # 子类实例
c.myMethod() # 子类调用重写方法
super(Child,c).myMethod() #用子类对象调用父类已被覆盖的方法
```

方法重载

super() 超类，父类

想一想，练一练

- 实现基类：猫，实现基类的方法：speak
- 实现派生类：黑猫，重写基类的speak方法
- 实例化一个黑猫对象，调用派生类和基类的speak方法
- 请将代码和运行结果截图，投稿

Python 类 call方法

- 在创建python类的时，同时写了__call__（）方法

- 类实例化出实例后，
- 实例名()将调用call（）方法。

Python 类 call方法-示例1

- 在创建python类的时，同时写了__call__()方法

- 类实例化出实例后，
- 实例名()将调用call()方法。

```
class people:
    #定义构造方法
    def __init__(self,n,a):
        self.name = n
        self.age = a

    def __call__(self):
        print('hello '+self.name)

class student(people):
    def __init__(self,n,a,w,g):
        #调用父类的构函
        people.__init__(self,n,a)
        self.grade = g
```

```
p = people('alice',10)
p()

s = student('ken',10,60,3)
s()
```

谢谢指正！