6-面向对象基础

__init__()

作用：初始化对象

```
class Washer():
    def __init__(self):
        self.weight=800
        self.height=500

    def print_info(self):
        print(f'洗衣机宽度{self.weight},高度为{self.height}')

i = Washer()
i.print_info()
#result:洗衣机宽度800,高度为500
```

带参数的__init__()

```
class Washer():
    def __init__(self,weight,height):
        self.weight=weight
        self.height=height

    def print_info(self):
        print(f'洗衣机宽度{self.weight},高度为{self.height}')

i = Washer(10,20)
i.print_info()
#result:洗衣机宽度10,高度为20
```

__str__()

作用：使得print输出对象的内容

```
class Washer():
    def __init__(self,weight,height):
        self.weight=weight
        self.height=height
        self.title="haier"

    def print_info(self):
        print(f'洗衣机宽度{self.weight},高度为{self.height}')

     def __str__(self):
         return f"这是{self.title}品牌的洗衣机"

i = Washer(10,20)
print(i)
#result:这是haier品牌的洗衣机
```

6-1 继承

在Python类中，所有类默认继承object类，object是顶级类或基类；其他子类叫做派生类。

```
class Master():
    def __init__(self):
        self.skill = '[古法配方]'

```

```
    def make_cake(self):
        print(f'运用{self.skill}方法制作')


class School():
    def __init__(self):
        self.skill = '[经典配方]'

    def make_cake(self):
        print(f'运用{self.skill}方法制作')

class Prentice(Master,School):
    def __init__(self):
        self.skill = '[独创配方]'

    #如果先调用了父类的属性和方法，父类属性会覆盖子类属性，故调用前现调用自己子类初始化
    def make_cake(self):
        self.__init__()
        print(f'运用{self.skill}方法制作')

    #调用父类方法确保属性调用正确，现进行初始化
    def make_cake_master(self):
        Master.__init__(self)
        Master.make_cake(self)

    def make_cake_school(self):
        School.__init__(self)
        School.make_cake(self)
```

super()调用父类方法

```
class Master():
    def __init__(self):
        self.skill = '[古法配方]'

    def make_cake(self):
        print(f'运用{self.skill}方法制作')


class School(Master):
    def __init__(self):
        self.skill = '[经典配方]'

    def make_cake(self):
        print(f'运用{self.skill}方法制作')
        # super(School, self).__init__()
        # super(School, self).make_cake()
        super().__init__()
        super().make_cake()

class Prentice(School):
    def __init__(self):
        self.skill = '[独创配方]'

    #如果先调用了父类的属性和方法，父类属性会覆盖子类属性，故调用前现调用自己子类初始化
    def make_cake(self):
```

```
        self.__init__()
        print(f'运用{self.skill}方法制作')

    def make_old_cake(self):
        # 方法一：super(当前类名,self).函数()
        # super(Prentice, self).__init__()
        # super(Prentice, self).make_cake()

        # 方法二：无参数super
        super().__init__()
        super().make_cake()
```

6-2 私有权限

作用：属性、方法不可继承给子类。

设置方法：属性名和方法名前面加上两个下划线__。

获取和修改方法：使用get和set方法

```
class Prentice(Master,School):
    def __init__(self):
        self.skill = '[独创配方]'
        self.__money = 2000000

    def get_money(self):
        return self.__money

    def set_money(self,money):
        self.__money = money
```

6-3 多态

作用：传入不同的对象，产生不同的结果

定义：一种使用对象的方式，子类重写父类的方法

```
class Dog():
    def work(self):
        pass

class ArmyDog(Dog):
    def work(self):
        print('追击敌人...')

class DrugDog(Dog):
    def work(self):
        print('搜查毒品...')

class Person():
    def work_with_dog(self,dog):
        dog.work()
```

6-4 类方法

类方法：需要用装饰器@classmethod标识为类方法，第一个参数必须是类对象，一般以cls作为第一个参数

```
class Dog():
    __tooth = 10
    @classmethod
    def get_tooth(cls):
        return cls.__tooth
```

6-5 静态方法

特点：需要通过@staticmethod进行修饰，既不需要使用传递类对象也不需要传递实例对象

```
class Dog():
    @staticmethod
    def info_print():
        print('这是一个狗类')
```