

课程：面向对象-继承

目标

- 继承的概念
- 单继承
- 多继承
- 子类重写父类的同名属性和方法
- 子类调用父类的同名属性和方法
- 多层继承
- super()
- 私有属性和私有方法

一. 继承的概念

生活中的继承，一般指的是子女继承父辈的财产。



- 拓展1：经典类或旧式类

不由任意内置类型派生出的类，称之为经典类。

```
1 class 类名:  
2     代码  
3     .....
```

- 拓展2：新式类

```
1 class 类名(object):  
2     代码
```

Python面向对象的继承指的是多个类之间的所属关系，即子类默认继承父类的所有属性和方法，具体如下：

```
1  # 父类A
2  class A(object):
3      def __init__(self):
4          self.num = 1
5
6      def info_print(self):
7          print(self.num)
8
9  # 子类B
10 class B(A):
11     pass
12
13
14 result = B()
15 result.info_print() # 1
```

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

二. 单继承

故事主线：一个煎饼果子老师傅，在煎饼果子界摸爬滚打多年，研发了一套精湛的摊煎饼果子的技术。师父要把这套技术传授给他的唯一的最得意的徒弟。

分析：徒弟是不是要继承师父的所有技术？

```
1  # 1. 师父类
2  class Master(object):
3      def __init__(self):
4          self.kongfu = '[古法煎饼果子配方]'
5
6      def make_cake(self):
7          print(f'运用{self.kongfu}制作煎饼果子')
8
9
10 # 2. 徒弟类
11 class Prentice(Master):
12     pass
13
14
15 # 3. 创建对象daqiu
16 daqiu = Prentice()
17 # 4. 对象访问实例属性
18 print(daqiu.kongfu)
```

```
19 # 5. 对象调用实例方法
20 daqiu.make_cake()
```

三. 多继承

故事推进：daqiu是个爱学习的好孩子，想学习更多的煎饼果子技术，于是，在百度搜索到黑马程序员，报班学习煎饼果子技术。

所谓多继承意思就是一个类同时继承了多个父类。

```
1 class Master(object):
2     def __init__(self):
3         self.kongfu = '[古法煎饼果子配方]'
4
5     def make_cake(self):
6         print(f'运用{self.kongfu}制作煎饼果子')
7
8
9 # 创建学校类
10 class School(object):
11     def __init__(self):
12         self.kongfu = '[黑马煎饼果子配方]'
13
14     def make_cake(self):
15         print(f'运用{self.kongfu}制作煎饼果子')
16
17
18 class Prentice(School, Master):
19     pass
20
21
22 daqiu = Prentice()
23 print(daqiu.kongfu)
24 daqiu.make_cake()
```

注意：当一个类有多个父类的时候，默认使用第一个父类的同名属性和方法。

四. 子类重写父类同名方法和属性

故事：daqiu掌握了师父和培训的技术后，自己潜心钻研出自己的独门配方的一套全新的煎饼果子技术。

```
1 class Master(object):
2     def __init__(self):
```

```

3         self.kongfu = '[古法煎饼果子配方]'
4
5     def make_cake(self):
6         print(f'运用{self.kongfu}制作煎饼果子')
7
8
9 class School(object):
10     def __init__(self):
11         self.kongfu = '[黑马煎饼果子配方]'
12
13     def make_cake(self):
14         print(f'运用{self.kongfu}制作煎饼果子')
15
16
17 # 独创配方
18 class Prentice(School, Master):
19     def __init__(self):
20         self.kongfu = '[独创煎饼果子配方]'
21
22     def make_cake(self):
23         print(f'运用{self.kongfu}制作煎饼果子')
24
25
26 daqiu = Prentice()
27 print(daqiu.kongfu)
28 daqiu.make_cake()
29
30 print(Prentice.__mro__)

```

子类 and 父类具有同名属性和方法，默认使用子类的同名属性和方法。

五. 子类调用父类的同名方法和属性

故事：很多顾客都希望也能吃到古法和黑马的技术的煎饼果子。

```

1 class Master(object):
2     def __init__(self):
3         self.kongfu = '[古法煎饼果子配方]'
4
5     def make_cake(self):
6         print(f'运用{self.kongfu}制作煎饼果子')
7
8
9 class School(object):
10     def __init__(self):
11         self.kongfu = '[黑马煎饼果子配方]'

```

```

12
13     def make_cake(self):
14         print(f'运用{self.kongfu}制作煎饼果子')
15
16
17 class Prentice(School, Master):
18     def __init__(self):
19         self.kongfu = '[独创煎饼果子配方]'
20
21     def make_cake(self):
22         # 如果是先调用了父类的属性和方法，父类属性会覆盖子类属性，故在调用属性前，先调用
        自己子类的初始化
23         self.__init__()
24         print(f'运用{self.kongfu}制作煎饼果子')
25
26         # 调用父类方法，但是为保证调用到的也是父类的属性，必须在调用方法前调用父类的初始化
27     def make_master_cake(self):
28         Master.__init__(self)
29         Master.make_cake(self)
30
31     def make_school_cake(self):
32         School.__init__(self)
33         School.make_cake(self)
34
35
36 daqiu = Prentice()
37
38 daqiu.make_cake()
39
40 daqiu.make_master_cake()
41
42 daqiu.make_school_cake()
43
44 daqiu.make_cake()

```

六. 多层继承

故事：N年后，daqiu老了，想要把所有技术传承给自己的徒弟。

```

1 class Master(object):
2     def __init__(self):
3         self.kongfu = '[古法煎饼果子配方]'
4
5     def make_cake(self):
6         print(f'运用{self.kongfu}制作煎饼果子')
7

```

```

8
9 class School(object):
10     def __init__(self):
11         self.kongfu = '[黑马煎饼果子配方]'
12
13     def make_cake(self):
14         print(f'运用{self.kongfu}制作煎饼果子')
15
16
17 class Prentice(School, Master):
18     def __init__(self):
19         self.kongfu = '[独创煎饼果子配方]'
20
21     def make_cake(self):
22         self.__init__()
23         print(f'运用{self.kongfu}制作煎饼果子')
24
25     def make_master_cake(self):
26         Master.__init__(self)
27         Master.make_cake(self)
28
29     def make_school_cake(self):
30         School.__init__(self)
31         School.make_cake(self)
32
33
34 # 徒孙类
35 class Tusun(Prentice):
36     pass
37
38
39 xiaoqiu = Tusun()
40
41 xiaoqiu.make_cake()
42
43 xiaoqiu.make_school_cake()
44
45 xiaoqiu.make_master_cake()
46

```

七. super()调用父类方法

```

1 class Master(object):
2     def __init__(self):
3         self.kongfu = '[古法煎饼果子配方]'
4

```

```

5     def make_cake(self):
6         print(f'运用{self.kongfu}制作煎饼果子')
7
8
9     class School(Master):
10        def __init__(self):
11            self.kongfu = '[黑马煎饼果子配方]'
12
13        def make_cake(self):
14            print(f'运用{self.kongfu}制作煎饼果子')
15
16            # 方法2.1
17            # super(School, self).__init__()
18            # super(School, self).make_cake()
19
20            # 方法2.2
21            super().__init__()
22            super().make_cake()
23
24
25    class Prentice(School):
26        def __init__(self):
27            self.kongfu = '[独创煎饼果子技术]'
28
29        def make_cake(self):
30            self.__init__()
31            print(f'运用{self.kongfu}制作煎饼果子')
32
33        # 子类调用父类的同名方法和属性：把父类的同名属性和方法再次封装
34        def make_master_cake(self):
35            Master.__init__(self)
36            Master.make_cake(self)
37
38        def make_school_cake(self):
39            School.__init__(self)
40            School.make_cake(self)
41
42        # 一次性调用父类的同名属性和方法
43        def make_old_cake(self):
44            # 方法一：代码冗余；父类类名如果变化，这里代码需要频繁修改
45            # Master.__init__(self)
46            # Master.make_cake(self)
47            # School.__init__(self)
48            # School.make_cake(self)
49
50            # 方法二：super()
51            # 方法2.1 super(当前类名, self).函数()
52            # super(Prentice, self).__init__()
53            # super(Prentice, self).make_cake()

```

```

54
55     # 方法2.2 super().函数()
56     super().__init__()
57     super().make_cake()
58
59
60 daqiu = Prentice()
61
62 daqiu.make_old_cake()

```

注意：使用super() 可以自动查找父类。调用顺序遵循 `__mro__` 类属性的顺序。比较适合单继承使用。

八. 私有权限

8.1 定义私有属性和方法

在Python中，可以为实例属性和方法设置私有权限，即设置某个实例属性或实例方法不继承给子类。

故事：daqiu把技术传承给徒弟的同时，不想把自己的钱(2000000个亿)继承给徒弟，这个时候就要为 `钱` 这个实例属性设置私有权限。

设置私有权限的方法：在属性名和方法名 前面 加上两个下划线 `__`。

```

1  class Master(object):
2      def __init__(self):
3          self.kongfu = '[古法煎饼果子配方]'
4
5      def make_cake(self):
6          print(f'运用{self.kongfu}制作煎饼果子')
7
8
9  class School(object):
10     def __init__(self):
11         self.kongfu = '[黑马煎饼果子配方]'
12
13     def make_cake(self):
14         print(f'运用{self.kongfu}制作煎饼果子')
15
16
17 class Prentice(School, Master):
18     def __init__(self):
19         self.kongfu = '[独创煎饼果子配方]'
20         # 定义私有属性
21         self.__money = 2000000
22

```



```

23     # 定义私有方法
24     def __info_print(self):
25         print(self.kongfu)
26         print(self.__money)
27
28     def make_cake(self):
29         self.__init__()
30         print(f'运用{self.kongfu}制作煎饼果子')
31
32     def make_master_cake(self):
33         Master.__init__(self)
34         Master.make_cake(self)
35
36     def make_school_cake(self):
37         School.__init__(self)
38         School.make_cake(self)
39
40
41     # 徒孙类
42     class Tusun(Prentice):
43         pass
44
45
46     daqiu = Prentice()
47     # 对象不能访问私有属性和私有方法
48     # print(daqiu.__money)
49     # daqiu.__info_print()
50
51     xiaoqiu = Tusun()
52     # 子类无法继承父类的私有属性和私有方法
53     # print(xiaoqiu.__money) # 无法访问实例属性__money
54     # xiaoqiu.__info_print()

```

注意：私有属性和私有方法只能在类里面访问和修改。

8.2 获取和修改私有属性值

在Python中，一般定义函数名 `get_xx` 用来获取私有属性，定义 `set_xx` 用来修改私有属性值。

```

1     class Master(object):
2         def __init__(self):
3             self.kongfu = '[古法煎饼果子配方]'
4
5         def make_cake(self):
6             print(f'运用{self.kongfu}制作煎饼果子')
7
8
9     class School(object):

```

```
10     def __init__(self):
11         self.kongfu = '[黑马煎饼果子配方]'
12
13     def make_cake(self):
14         print(f'运用{self.kongfu}制作煎饼果子')
15
16
17 class Prentice(School, Master):
18     def __init__(self):
19         self.kongfu = '[独创煎饼果子配方]'
20         self.__money = 2000000
21
22     # 获取私有属性
23     def get_money(self):
24         return self.__money
25
26     # 修改私有属性
27     def set_money(self):
28         self.__money = 500
29
30     def __info_print(self):
31         print(self.kongfu)
32         print(self.__money)
33
34     def make_cake(self):
35         self.__init__()
36         print(f'运用{self.kongfu}制作煎饼果子')
37
38     def make_master_cake(self):
39         Master.__init__(self)
40         Master.make_cake(self)
41
42     def make_school_cake(self):
43         School.__init__(self)
44         School.make_cake(self)
45
46
47 # 徒孙类
48 class Tusun(Prentice):
49     pass
50
51
52 daqiu = Prentice()
53
54 xiaoqiu = Tusun()
55 # 调用get_money函数获取私有属性money的值
56 print(xiaoqiu.get_money())
57 # 调用set_money函数修改私有属性money的值
58 xiaoqiu.set_money()
```

九. 总结

- 继承的特点
 - 子类默认拥有父类的所有属性和方法
 - 子类重写父类同名方法和属性
 - 子类调用父类同名方法和属性
- `super()`方法快速调用父类方法
- 私有限制
 - 不能继承给子类的属性和方法需要添加私有限制
 - 语法

```
1 class 类名():
2     # 私有属性
3     __属性名 = 值
4
5     # 私有方法
6     def __函数名(self):
7         代码
```