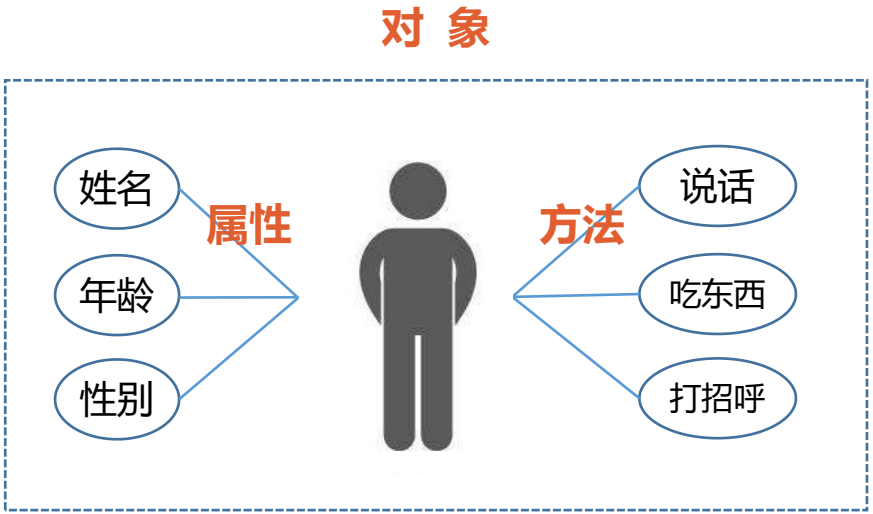
神经网络&深度学习
TENSORFLOW
Google

4.3 Python面向对象的编程

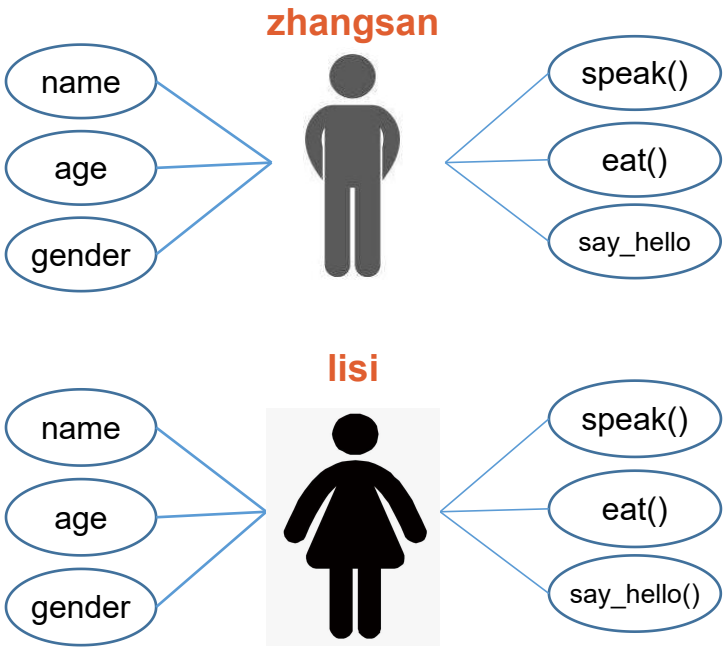
中国大学MOOC

- 面向对象的程序设计（Object-oriented programming，OOP）
- 面向过程的程序设计（Process-oriented programming, POP）



对象（object）：将属性和方法封装在一起。

对象的属性和方法



对象名 属性

zhangsan . name
zhangsan . age
zhangsan . gender

成员变量

zhangsan . speak()
zhangsan . say_hello()
zhangsan . call()

成员函数

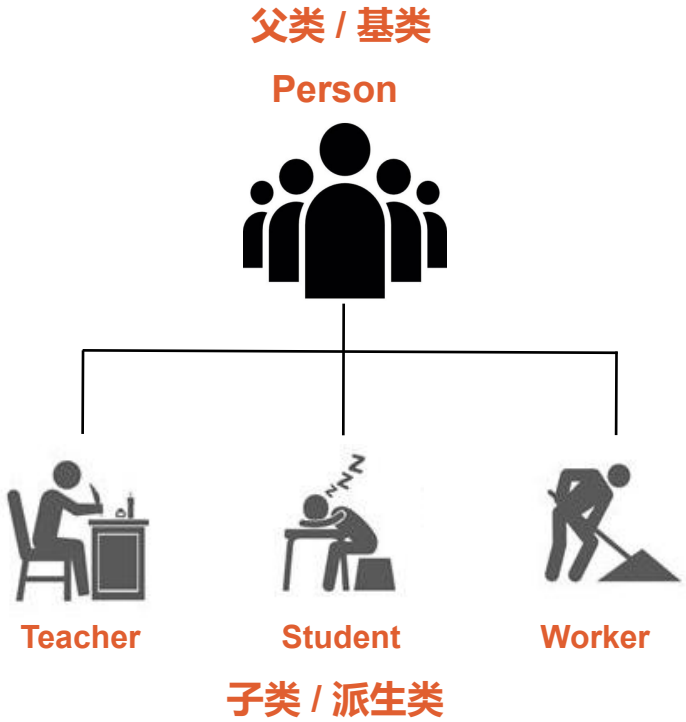
对象名 方法

- **类**（class）：具有相同的属性和方法的对象集合。
  - **对象**是类的**实例**
  - 子类**继承**了父类的**全部属性和方法**，

并且也有自己**特有的属性和方法**。

  - **继承**描述了类之间的**层次关系**

类和对象并非仅限于具体的事物，它也可以是一种**抽象的概念或者规则**。



声明类

```
class 类名:
    类属性=初值
    方法(参数列表)
```

访问类属性

```
类名.类属性
对象名.类属性
```

- 类属性是类中**所有对象共同拥有**的属性
- 它在内存中只存在**一个**副本
- 可以通过**类名**访问，也可以被类的所有**对象**访问
- 在类定义之后，可以通过类名**添加类属性**，新增的类属性也被**类和所有对象共有**。

类的特殊属性

类属性	含 义
__name__	类的名字（字符串）
__doc__	类的文档字符串
__bases__	类的所有父类组成的元组
__dict__	类的属性组成的字典
__module__	类所属的模块
__class__	类对象的类型

声明类

```
class 类名:
    类属性=初值
    方法(参数列表)
```

self， 参数2， 参数3， ...

- 方法中必须有一个参数self，而且它是参数列表中的第一个参数
- 由同一个类可以生成很多个对象,每一个对象，都有一个专属的self，代表这个对象自身

创建对象

```
对象名 = 类名()
```

```
1  class Person():
2      money=10000
3      def say_hello(self):
4          print("Hello!")
5
6  zhangsan = Person()
7
8  print(zhangsan.money)
9  zhangsan.say_hello()
```

对象拥有Person类中的所有属性和方法

运行结果：

```
10000
Hello!
```

- 创建对象之后，动态添加对象属性

```
1  class Person():
2      money=10000
3      def say_hello(self):
4          print("Hello!")
5
6  zhangsan = Person()
7
8  zhangsan.major="computer"
9  print("zhangsan's major is",zhangsan.major)
```

这是一个动态添加的实例属性，它只属于zhangsan自己，如果重新创建其他的Person()对象，是没有这个属性的。

运行结果：

```
zhangsan's major is computer
```

```
1   class Person():
2       money=10000
3       def say_hello(self):
4           print("Hello!")
5
6   zhangsan = Person()
7   zhangsan.major="computer"
8   print("zhangsan's major is",zhangsan.major)
9
10  lisi = Person()
11  print("lisi's major is",lisi.major)
```

运行结果：

```
zhangsan's major is computer
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-5-5cf93aa47a05> in <module>()
      9
     10 lisi = Person()
---> 11 print("lisi's major is",lisi.major)

AttributeError: 'Person' object has no attribute 'major'
```

错误信息

```
1  class Person():
2      money=10000
3      def say_hello(self):
4          print("Hello!")
5
6  zhangsan = Person()
7  lisi=Person()
8
9  print(Person.money)
10 print(zhangsan.money)
11 print(lisi.money)
12
13 Person.money=9000
14
15 print(Person.money)
16 print(zhangsan.money)
17 print(lisi.money)
18
19 zhangsan.money=5000
20
21 print(Person.money)
22 print(zhangsan.money)
23 print(lisi.money)
```

类属性，所有的实例共同享有。

修改类属性，所有实例的值都会发生变化。

zhangsan自己又创建了一个**实例属性**money，它已经不是原来的类属性money了。

运行结果：

```
10000
10000
10000
9000
9000
9000
9000
5000
9000
```

删除对象

```
del 对象名
```

```
del zhangsan
```

在执行完这条语句后，zhangsan对象就不存在了，如果再访问zhangsan对象，就会出现错误提示：

```
>>>zhangsan.major
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-2c73fb3b1bdb> in <module>()
----> 1 zhangsan.major

NameError: name 'zhangsan' is not defined
```

- **构造函数**：在创建对象时，用来完成初始化操作。

```
__init__(self， 参数2， 参数3， ...)
```

  - 当创建对象时，系统会自动调用构造函数。
  - 可以把对成员变量赋初值的语句写在构造函数中，从而保证每个变量都有合适的初始值。

- **析构函数**：在清除对象时，回收和释放对象所占用的资源。

```
__del__( )
```

在Python中，构造函数和析构函数也**可以省略**。

```python
class Person:
    def __init__(self,name,age,gender="男"):
        self.name=name
        self.age=age
        self.gender=gender
    def __del__(self):
        print("Bye bye—from",self.name)
    def printInfo(self):
        print("姓名:",self.name,"年龄:",self.age, "性别: ",self.gender)

zhangsan=Person("张三",18)
lisi=Person("李四",19,"女")

zhangsan.printInfo()
lisi.printInfo()

del zhangsan
del lisi
```

运行结果：

```
姓名: 张三 年龄: 18 性别:  男
姓名: 李四 年龄: 19 性别:  女
Bye bye—from 张三
Bye bye—from 李四
```

静态方法和类方法

类方法

```
class 类名:
    @classmethod
    def 类方法名(cls,...):
        方法体
```

- 可以通过类名或对象名调用。
- 不能访问实例属性，但可以访问类属性。

静态方法

```
class 类名:
    @staticmethod
    def 类方法名( ):
        方法体
```

- 可以通过类名或对象名调用。
- 不能访问实例属性，也不能直接访问类属性。但是可以通过类名引用类属性。

公有变量和私有变量

- 公有变量：可以在类的**外部**访问。
- 保护变量：只能允许其**本身**和**子类**进行访问。
- 私有变量：不允许在类的外部访问。

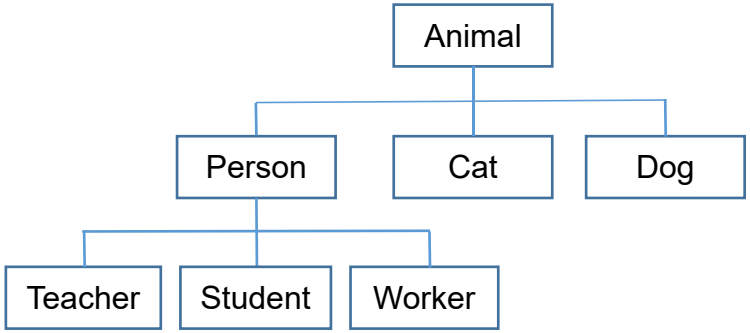
__xxx　　：私有变量

_xxx　　：保护变量

__xxx__　：专有变量，方法

- **继承**(inheritance)：子类能够继承父类中所有**非私有的**成员变量和成员函数。

```
class 子类名(父类名)
    类属性=初值
    方法(参数列表)
```




```python
class Person():
    money=10000
    def say_hello(self):
        print("Hello!")

class Teacher(Person):
    pass

amy = Teacher()

print(amy.money)
amy.say_hello()
```

运行结果：

```
10000
Hello!
```

- **类**(class)：对具有**相同属性和方法**的一组对象的描述，它定义了所有对象所共有的属性和方法。
- **对象**(object)：是类中的一个具体的**实例**(instance)。
- **属性**(attribute)：是类和对象中的**变量**。
  - **类属性**：定义在类的内部，方法的外部。是类中所有对象**共同拥有**的属性。
  - **实例属性**：也叫做**成员变量**，在每个对象中都有自己的副本。

- **方法**(method)：是在类中所定义的**函数**，它描述了对象能执行的**操作。**
  - **实例方法/成员函数**：只能通过**对象名**调用，第一个参数必须是self。
    构造函数和**析构函数**
  - **类方法**：可以通过**类名**或**对象名**调用，第一个参数必须是“cls”。
  - **静态方法**：通过类名或对象名调用，没有“self”或“cls”参数。
- **成员变量**：公有变量、保护变量、和私有变量。
- **继承**：是是**子类**自动的**共享父类中的属性和方法**的机制。