School of Economics ang Management, Beihang University

现代程序设计技术

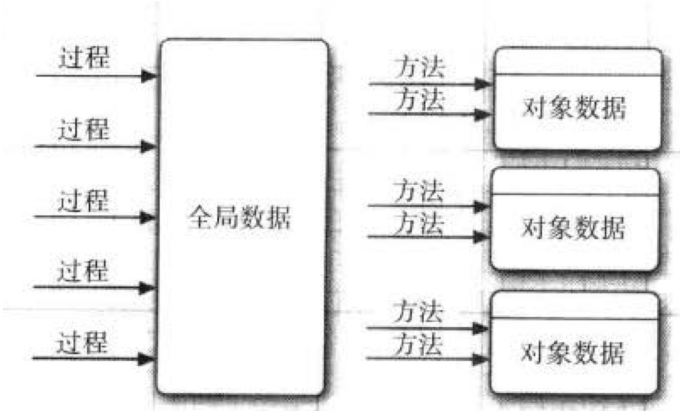
赵吉昌
jichang@buaa.edu.cn

本周内容

- Python基础
  - 面向对象程序设计
  - 自定义类

面向对象程序设计

- OO
  - 一种程序设计范式
  - 程序由对象组成，每个对象包含对用户公开的特定功能和隐藏的实现部分
  - 对象是数据与相关行为的集合
  - 不必关心对象的具体实现，只要能满足用户的需求即可
- 与结构化程序设计的差异
  - 将数据搁在第一位
  - 更加适用于规模大的问题



面向对象程序设计

- 类
  - 对象的类型，用来描述对象
  - 构造对象的模板
  - 定义了该集合中每个对象所共有的属性和方法
  - 由类构造对象的过程称之为实例化，或创建类的实例
- 一些重要概念
  - 多态
    - 可以对不同类型的对象执行相同的操作
  - 封装
    - 将数据和行为组合，并对外隐藏数据的实现方式
    - 对象的状态：当下实例域的值的特定组合
  - 继承
    - 通过扩展一个类来建立另外一个类

面向对象程序设计

- 对象的特性
  - 行为
    - 通过可调用的方法定义
  - 状态
    - 保存描述当前特征的信息
    - 对象状态的改变必须通过调用方法实现
  - 标识
    - 每个对象实例的标识应该是不同的

面向对象程序设计

表 4-1 表达类关系的 UML 符号

关　系	UML 连接符
继承	
接口实现	
依赖	
聚合	
关联	
直接关联	

- 类与类之间的关系
  - 依赖
    - 一个类的方法操纵另一个类的实例
    - 耦合度及其最小化
  - 聚合
    - 类A包含类B的实例对象
  - 继承
    - 类A由类B扩展而来
    - 如果类A扩展类B，类A不但包含从类B继承的方法，还会拥有一些额外的功能
  - UML

面向对象程序设计

- 使用预定义类
  - 标准库
  - 第三方库
  - 文档
- 使用自定义类
  - 识别类
  - 名词对应类
  - 动词对应方法
  - 类间的关系
  - 个人的经验

自定义类

- 创建类
  - 使用 class 语句来创建一个新类，class 之后为类的名称并以冒号结尾：
  - class ClassName:
  - """类的信息"""
  - <statement-1>
  - .
  - .
  - .
  - <statement-N>

自定义类

- 创建类
  - 类的文档信息可以通过`ClassName.__doc__`查看
  - 类的定义要先执行才能生效
    - 可以在函数中或`if`等中进行类定义
  - 类会创建一个新的命名空间

自定义类

- 辨析
  - 类对象
    - 类对象支持属性引用和实例化
    - 属性引用可以是类变量，也可以是类方法
  - 实例对象
    - 实例对象仅能进行属性引用（数据或方法）

自定义类

- 辨析
  - 类变量
    - 类变量在所有实例化的对象中是公用的
    - 类变量定义在类中且在类方法之外
    - 类变量通常不作为实例变量使用
    - 类比于Java中的静态属性
  - 实例变量
    - 每个实例独有
    - 第一次使用时自动生成
      - 与类变量重名时仍作为实例变量

自定义类

- 类的数据属性
  - 只要能避免冲突，可以向一个实例对象添加自定义的数据属性，而不会影响方法的正确性
    - 内置类的实例对象并不支持
    - Python类不能用来实现纯净的数据类型
  - 可以通过del删除
    - `del c.name`
  - 应该谨慎地使用数据属性
    - 与方法重名或与类变量重名？
  - 从方法内部引用数据属性（或其他方法）并没有快捷方式
    - 往往需要通过实例对象的引用来间接访问

自定义类

- 补充：类的数据属性
  - 默认情况下通过字典__dict__维护数据属性
  - 能否像内置类一样不允许增加属性？
  - 定义__slots__，元组，类数据属性的描述
  - 类实例只能拥有slots中定义的数据属性，不能再增加新的数据属性
  - __dict__不能再使用

自定义类

- 类的方法
  - 在类的内部，使用 `def` 关键字来定义一个方法
  - 与一般函数定义不同，类方法必须包含参数 `self`，且为第一个参数
    - `self`并非`python`关键字，可写其他名称
    - 但应该按惯例写作`self`
    - `self`代表类的实例
    - 在调用时不必传入相应的参数
  - 同名的数据属性会覆盖方法

自定义类

- 类的方法
  - 通过实例调用一个方法就相当于将该实例插入到参数列表的最前面，并通过类对象来调用相应的函数
  - class D:
    - def f(self,a,b):
      - self.a=a
      - self.b=b
      - pass
  - d=D()
  - d.f(a,b) 等价于 D.f(d,a,b)吗？

自定义类

- 类的方法
  - 对于类对象，为函数(function)
  - 对于实例对象，为方法(method)
  - ```
    class T:
        def f(self):
            pass
    ```
  - ```
    print(type(T.f)) #function
    ```
  - ```
    t=T()
    ```
  - ```
    print(type(t.f)) #method
    ```

自定义类

- 类的方法
  - 方法定义不一定只在类中，也可以将一个函数对象赋值给类中的一个局部变量

```
def f1(self, x, y):
    return min(x, y)
class C:
    f = f1
c=C()
print(c.f(2,3))
print(type(c.f))#是method还是function?
```

自定义类

- 类的方法
  - 构造方法
    - `__new__()`为构造方法，参数为将要构造的类，无`self`参数，返回新创建的实例对象
    - 极少使用
  - 初始化方法
    - `__init__()`为初始化方法，在类实例化时会自动调用
    - 有`self`参数
    - 也可以有除`self`外的其他参数
    - 只能返回`None`
    - 较为常用

自定义类

- 类的私有性
  - 严格来讲，python类的属性和方法都是对外公开的
    - 通过类对象或实例对象可以访问
  - 通过一些“约定”来约束外部的访问
    - 在属性或方法前加_
    - 在属性或方法前加__
    - 命名改装（name mangling），外部访问时需要加上_<类名>的前缀
    - 但仅为约定
      - __dict__可以查看

本周作业

- 实现文本处理的Tokenizer类
  - 深度学习处理自然语言时，会常常用到Tokenizer（https://huggingface.co/transformers/tokenizer_summary.html）。简单来说，就是按照预先定义好的词典，把文本编码成整数序列的过程。深度学习模型进行文本挖掘任务时会经常需要处理这种编码过的序列。在构建过程中，有时候我们希望句子的长度能够整齐，所以会规定一个特殊的号码[PAD]=0，代表填充位。具体地，例如词典为 {'[PAD]': 0, '我' :1, '是': 2, '北京': 3, '大学生': 4, '的': 5 }，那么"我是北京的大学生" 将被编码为 [1, 2, 3, 5, 4] ；如果需要句子长度为8，那么我们在后面填充[PAD]，得到 [1,2,3,5,4,0,0,0]。现在请编程实现一个基础的中文Tokenizer类，分别实现按字（深度学习中也常用该方式）或按词（通过jieba分词）进行编码，并在前次作业提供的评论或微博文本数据集上进行测试。
  - 1. __init__(self, chars,coding='c',PAD=0) 输入将要需要操作的文本（一个字符串的列表），这里需要完成词典的构建（即汉字到正整数的唯一映射的确定）。注意构建词典一是要根据coding来选择按词构建（coding='w')，还是按字构建，默认按字构建；PAD默认为0。
  - 2. tokenize(self, sentence) 输入一句话，返回分词(字）后的字符列表(list_of_chars)。
  - 3. encode(self, list_of_chars) 输入字符(字或者词）的字符列表，返回转换后的数字列表(tokens)。
  - 4. trim(self, tokens, seq_len) 输入数字列表tokens，整理数字列表的长度。不足seq_len的部分用PAD补足，超过的部分截断。
  - 5. decode(self, tokens) 将模型输出的数字列表翻译回句子。如果有PAD，输出'[PAD]'。
  - 6. encode(self，seq_len) 返回所有文本（chars)的长度为seq_len的tokens。
  - 7. 附加：seq_len是句子的长度，实际任务中一般怎么来确定一个合适的长度，请以前次作业中的评论文本或微博文本为例，通过文本的长度分布来进行观察和讨论。
  - 8. 附加：了解一下基于Tokenizer编码后的序列在序列模型（如RNN, LSTM, GRU, Transformer)中一般是如何使用的，可用来完成哪些任务。