School of Economics ang Management, Beihang University

现代程序设计技术

赵吉昌
jichang@buaa.edu.cn

密码

- 派生自 UNIX、有 40 多年历史的 BSD 称得上是古老的操作系统了，用今天的标准看它的安全性不会有多高。
- 2014 年，Leah Neukirchen 从 BSD3 源码树中发现了密码文件 /etc/passwd，包含了 UNIX/BSD 早期缔造者和开发者如 Dennis Ritchie、Ken Thompson、Brian W. Kernighan、Steve Bourne 和 Bill Joy 等人的密码。
- 这些密码使用了 Descrypt 哈希算法加密，在 40 年前这种 算法是当时最先进的哈希加密算法了，但今天早已过时，Neukirchen 也很轻松的把大部分密码都破解了出来：Dennis Ritchie 的密码是 dmac（他的中间名字是 MacAlistair），Eric Schmidt 的密码是 wendy!!（他妻子的名字），Stephen Bourne 的密码就是 bourne，Kernighan 的密码是 /.,/.,。但还有几个人的密码他没能破解，其中之一是 Ken Thompson。
- Neukirchen 在 The Unix Heritage Society 邮件列表上发帖请求其他人帮忙破解。10 月 9 日，Nigel Williams 破解了这个密码，他使用 AMD Radeon Vega64 显卡运行 hashcat 程序耗时四天多的时间将其破解。Ken Thompson 使用了一个不同寻常的密码——p/q2-q4!，这是他喜爱的国际象棋游戏一种常用开局的描述记数法。

关于作业的一些说明

- 作业跟课堂讲的内容的关系
  - 不能说没有，但特别微弱，不那么直接
- 这么设计的目的
  - 课程强调关键基础和设计模式等“务虚”内容
  - 作业主要是锻炼实际问题的解决能力
    - 问题理解+自主学习第三方库
  - 课程内容会帮助加速对第三库的理解
- 面向实际应用
  - 需要迅速地进行问题分析与技术选型
  - 需要通过例子和文档迅速地掌握新库或新方法

关于作业的一些说明

- 难度
  - 更加灵活：难度可以进一步提升
  - 更加自主：选择适当的程度完成
- 时间周期
  - 调整至三周左右的时间
  - 多花点时间琢磨
- 代码共享
  - 做的好作业会上传到课程中心
- 鼓励“回头看”
  - 根据当前掌握的技术重新审视之前的作业
  - 没有标准答案
  - 目的是能力培养不是填写答案

关于作业的一些说明

- 解决的一般过程
  - 问题理解
    - 背景，要素，意义
  - 探索视角
    - 维度？人或其他实体？行为？分类？综合？
  - 技术选型
    - 课堂上提到的内容，标准库或第三方库
  - 技术实现
    - 可能碰到技术难题，需要查阅reference
  - 结果分析
    - 注意可视化绝对不是唯一途径，紧扣视角及其综合
  - 结果总结
    - 什么现象？为什么？有什么管理启示？
    - 与问题理解照应

关于作业的一些说明

- 觉得简单怎么办
  - 迅速完成并找更复杂的问题自己钻研
  - 欢迎提供新思路成为下周作业
- 觉得困难怎么办
  - 初学阶段的困难是正常的
  - 仔细琢磨，不着急慌乱动手
  - 有问题多阅读reference
    - 这门课培养大家的核心能力之一
  - 有困惑多讨论
    - 我就在那里，要么沉默地刷着邮箱，要么呆坐在教室的前排

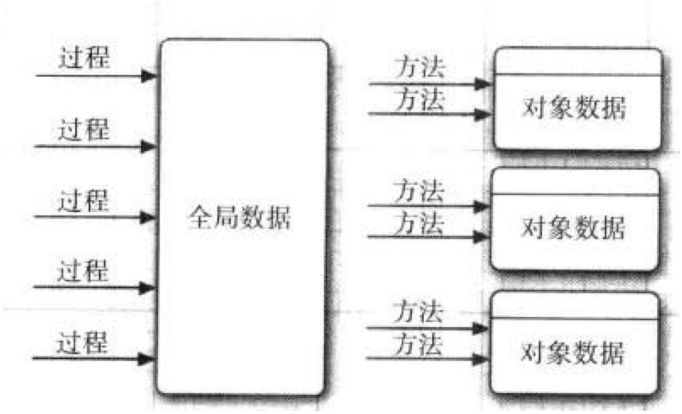
- 不要“怨天尤人”
  - 你们碰到的所谓“困难”还远没到这个地步
  - 少年人有什么可怕的

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
    - 类变量在整个实例化的对象中是公用的
    - 类变量定义在类中且在函数体之外
    - 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
  - 以n个参数的列表去调用一个方法就相当于将方法的对象插入到参数列表的最前面后，以这个列表去调用相应的函数
  - 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)
    - 并不是实例对象中所有的方法都是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

- 无。