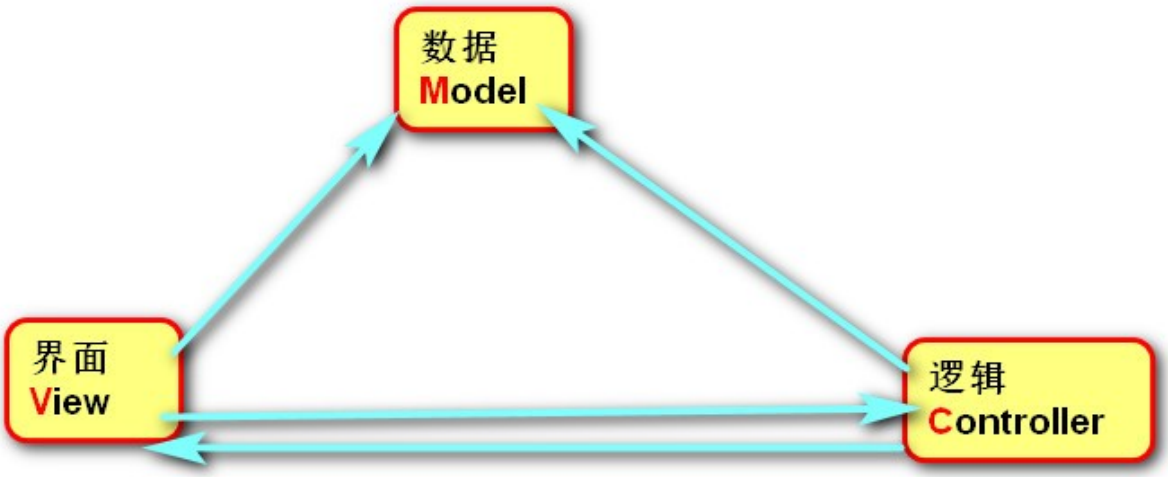
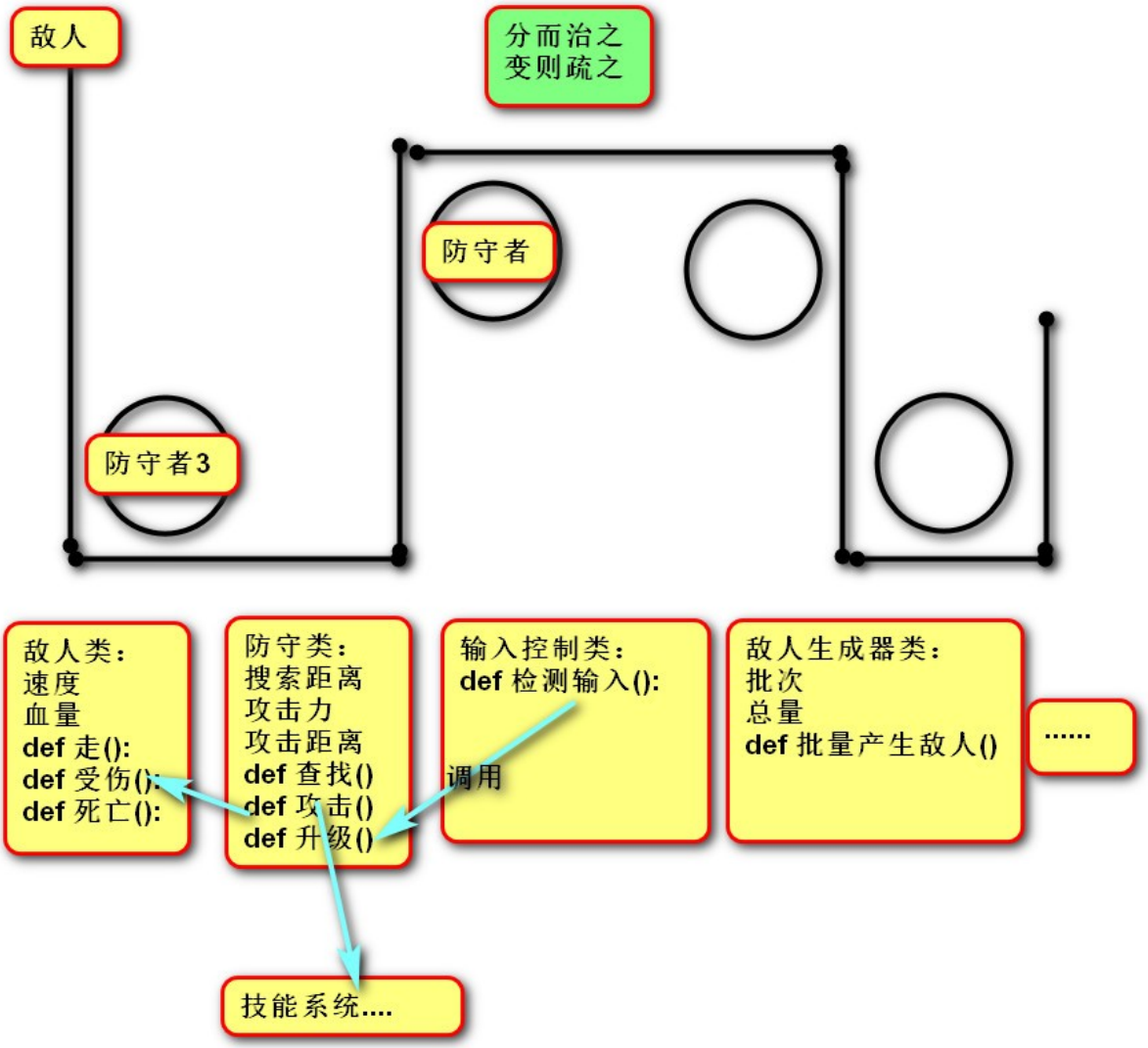
设计角度讲

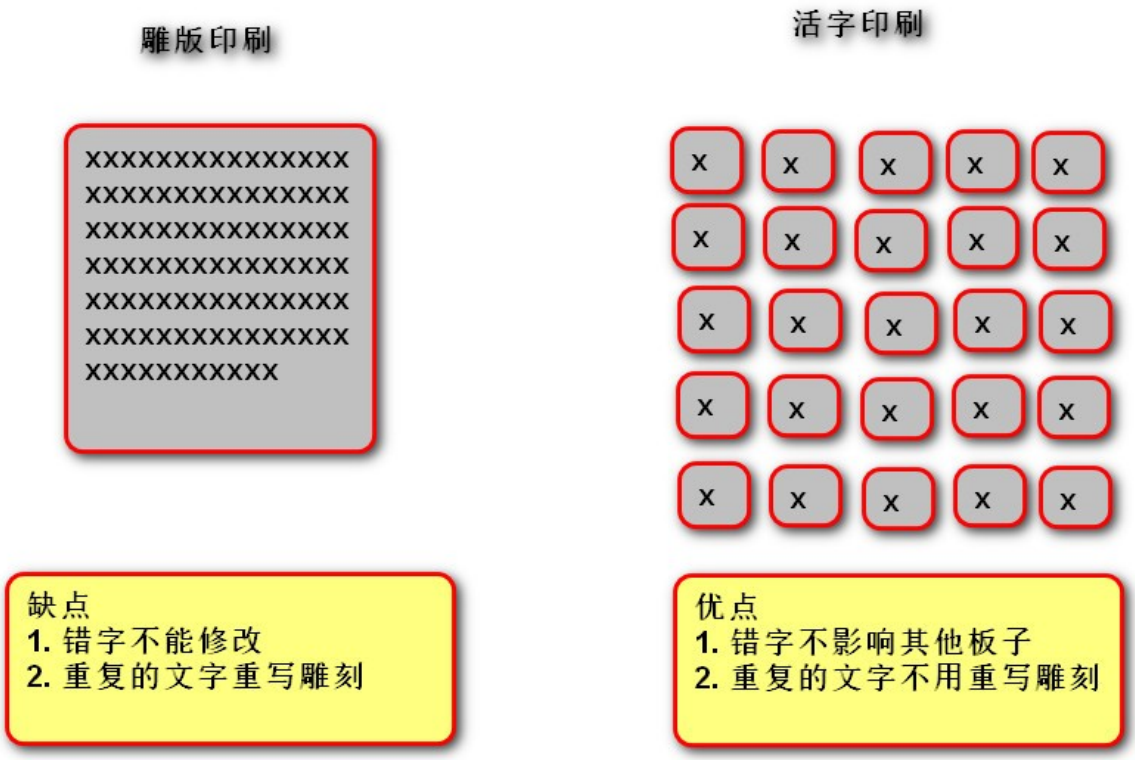
1. 定义：
   (1) 分而治之
   将一个大的需求分解为许多类，每个类处理一个独立的功能。
   (2) 变则疏之
   变化的地方独立封装，避免影响其他类。
   (3) 高 内 聚
   类中各个方法都在完成一项任务(单一职责的类)。
   (4) 低 耦 合
   类与类的关联性与依赖度要低(每个类独立)，让一个类的改变，尽少影响其他类。






```
"""
    封装
        设计思想
"""
# 以面向对象的思想,描述下列场景:
# 老张开车去东北.
class Person:
    def __init__(self, name=""):
        self.name = name
    # 需求:人的go_to方法,调用车的run方法
    # 技术:实例方法 -- 通过对象调用
    def go_to(self,str_position,type):
        print("去:",str_position)
        type.run()
```

```python
class Car:
    def run(self):
        print("走你....")
c01 = Car()
lz = Person("老张")
lz.go_to("东北",c01)
```



2. 优势：

便于分工，便于复用，可扩展性强。实现对学生信息的增加、删除、修改和查询。

分析

界面可能使用控制台，也可能使用 Web 等等。

1. 识别对象：界面视图类　逻辑控制类　数据模型类

2. 分配职责：

界面视图类：负责处理界面逻辑，比如显示菜单，获取输入，显示结果等。
逻辑控制类：负责存储学生信息，处理业务逻辑。比如添加、删除等
数据模型类：定义需要处理的数据类型。比如学生信息。

3. 建立交互：

界面视图对象 <----> 数据模型对象 <----> 逻辑控制对象

设计

数据模型类：StudentModel

数据：编号 id,姓名 name,年龄 age,成绩 score

逻辑控制类：StudentManagerController

数据：学生列表 __stu_list

行为：获取列表 stu_list,添加学生 add_student，删除学生 remove_student，修改学生 update_student，根据成绩排序 order_by_score。

界面视图类：StudentManagerView

数据：逻辑控制对象__manager

行为：显示菜单__display_menu，选择菜单项__select_menu_item，入口逻辑 main，输入学生__input_students，输出学生__output_students，删除学生__delete_student，修改学生信息__modify_student