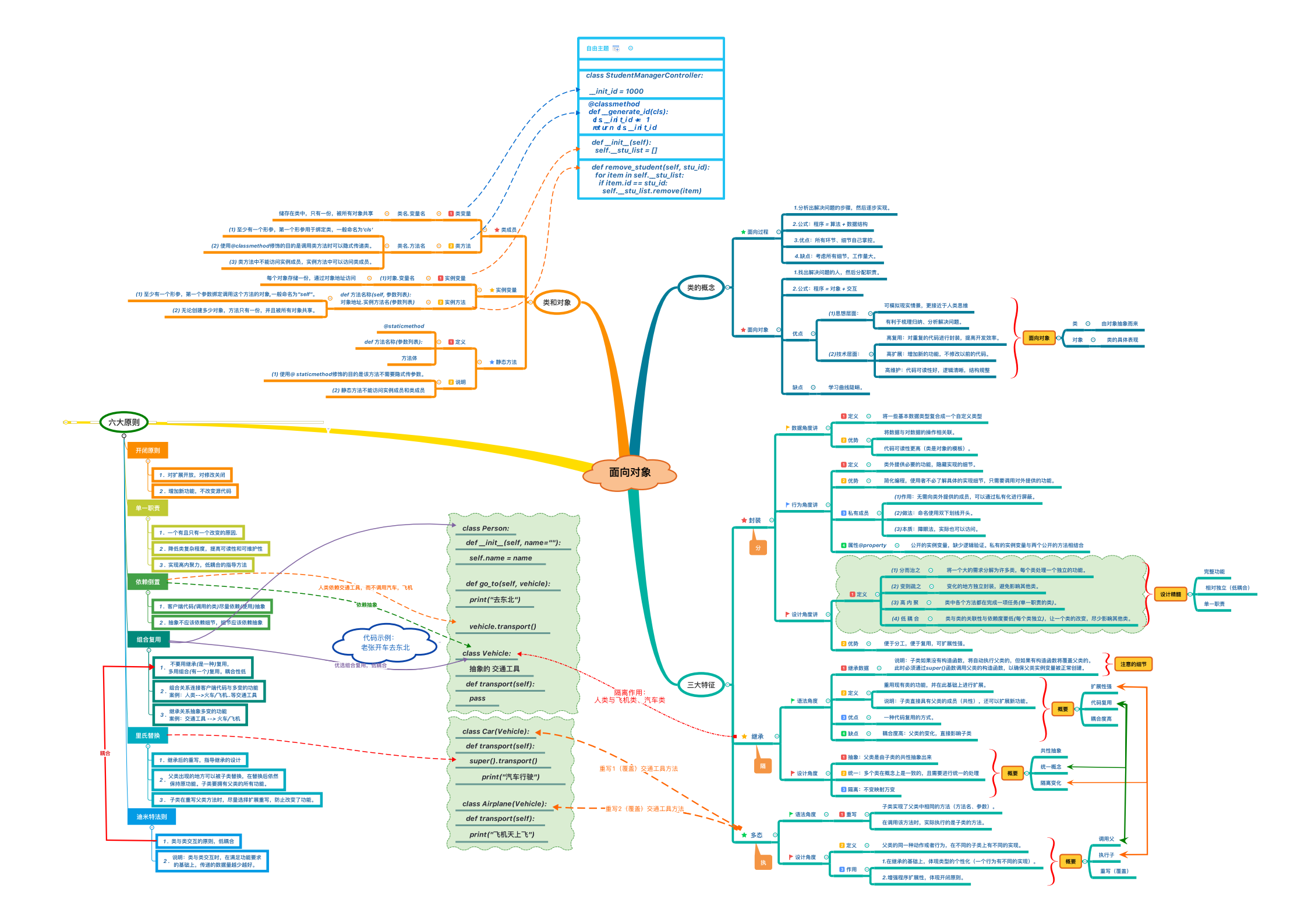
面向对象
类和对象
自由主题
class StudentManagerController:
__init_id = 1000
@classmethod
def __generate_id(cls):
cls.__init_id += 1
return cls.__init_id
def __init__(self):
self.__stu_list = []
def remove_student(self, stu_id):
for item in self.__stu_list:
if item.id == stu_id:
self.__stu_list.remove(item)
★类成员
1 类变量
储存在类中，只有一份，被所有对象共享
类名.变量名
2 类方法
(1) 至少有一个形参，第一个形参用于绑定类，一般命名为'cls'
(2) 使用@classmethod修饰的目的是调用类方法时可以隐式传递类。
(3) 类方法中不能访问实例成员，实例方法中可以访问类成员。
类名.方法名
★实例变量
1 实例变量
每个对象存储一份，通过对象地址访问
(1)对象.变量名
2 实例方法
(1) 至少有一个形参，第一个参数绑定调用这个方法的对象,一般命名为"self"。
(2) 无论创建多少对象，方法只有一份，并且被所有对象共享。
def 方法名称(self, 参数列表):
对象地址.实例方法名(参数列表)
★静态方法
@staticmethod
def 方法名称(参数列表):
方法体
1 定义
(1) 使用@ staticmethod修饰的目的是该方法不需要隐式传参数。
(2) 静态方法不能访问实例成员和类成员
2 说明
类的概念
★面向过程
1.分析出解决问题的步骤，然后逐步实现。
2.公式：程序 = 算法 + 数据结构
3.优点：所有环节、细节自己掌控。
4.缺点：考虑所有细节，工作量大。
★面向对象
1.找出解决问题的人，然后分配职责。
2.公式：程序 = 对象 + 交互
优点
(1)思想层面：
可模拟现实情景，更接近于人类思维
有利于梳理归纳、分析解决问题。
(2)技术层面：
高复用：对重复的代码进行封装，提高开发效率。
高扩展：增加新的功能，不修改以前的代码。
高维护：代码可读性好，逻辑清晰，结构规整
面向对象
类 由对象抽象而来
对象 类的具体表现
缺点 学习曲线陡峭。
六大原则
开闭原则
1．对扩展开放，对修改关闭
2．增加新功能，不改变源代码
单一职责
1．一个有且只有一个改变的原因.
2．降低类复杂程度，提高可读性和可维护性
3．实现高内聚力，低耦合的指导方法
依赖倒置
1．客户端代码(调用的类)尽量依赖(使用)抽象
2．抽象不应该依赖细节，细节应该依赖抽象
组合复用
1．不要用继承(是一种)复用，多用组合(有一个)复用。耦合性低
2．组合关系连接客户端代码与多变的功能
案例：人类-->火车/飞机..等交通工具
3．继承关系抽象多变的功能
案例：交通工具 --> 火车/飞机
里氏替换
1．继承后的重写，指导继承的设计
2．父类出现的地方可以被子类替换，在替换后依然保持原功能。子类要拥有父类的所有功能。
3．子类在重写父类方法时，尽量选择扩展重写，防止改变了功能。
迪米特法则
1．类与类交互的原则，低耦合
2．说明：类与类交互时，在满足功能要求的基础上，传递的数据量越少越好。
耦合
人类依赖交通工具，而不调用汽车，飞机
依赖抽象
代码示例：
老张开车去东北
优选组合复用，低耦合
class Person:
def __init__(self, name=""):
self.name = name
def go_to(self, vehicle):
print("去东北")
vehicle.transport()
class Vehicle:
抽象的 交通工具
def transport(self):
pass
class Car(Vehicle):
def transport(self):
super().transport()
print("汽车行驶")
class Airplane(Vehicle):
def transport(self):
print("飞机天上飞")
隔离作用：
人类与飞机类、汽车类
重写1（覆盖）交通工具方法
重写2（覆盖）交通工具方法
三大特征
★封装
分
数据角度讲
1 定义 将一些基本数据类型复合成一个自定义类型
2 优势
将数据与对数据的操作相关联。
代码可读性更高（类是对象的模板）。
行为角度讲
1 定义 类外提供必要的功能，隐藏实现的细节。
2 优势 简化编程，使用者不必了解具体的实现细节，只需要调用对外提供的功能。
3 私有成员
(1)作用：无需向类外提供的成员，可以通过私有化进行屏蔽。
(2)做法：命名使用双下划线开头。
(3)本质：障眼法，实际也可以访问。
4 属性@property 公开的实例变量，缺少逻辑验证。私有的实例变量与两个公开的方法相结合
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。
设计精髓
完整功能
相对独立（低耦合）
单一职责
2 优势 便于分工，便于复用，可扩展性强。
★继承
隔
语法角度
1 继承数据 说明：子类如果没有构造函数，将自动执行父类的。但如果有构造函数将覆盖父类的。此时必须通过super()函数调用父类的构造函数，以确保父类实例变量被正常创建。
注意的细节
2 定义
重用现有类的功能，并在此基础上进行扩展。
说明：子类直接具有父类的成员（共性），还可以扩展新功能。
3 优点 一种代码复用的方式。
4 缺点 耦合度高：父类的变化，直接影响子类
概要
扩展性强
代码复用
耦合度高
设计角度
1 抽象：父类是由子类的共性抽象出来
2 统一：多个类在概念上是一致的，且需要进行统一的处理
3 隔离：不变映射万变
概要
共性抽象
统一概念
隔离变化
★多态
执
语法角度
1 重写 子类实现了父类中相同的方法（方法名、参数）。
在调用该方法时，实际执行的是子类的方法。
设计角度
2 定义 父类的同一种动作或者行为，在不同的子类上有不同的实现。
3 作用
1.在继承的基础上，体现类型的个性化（一个行为有不同的实现）。
2.增强程序扩展性，体现开闭原则。
概要
调用父
执行子
重写（覆盖）