1. Programming paradigm：编程范式，是一种根据不同程序设计语言的特性对其进行分类的方法。常见的编程范式包括指令式和声明式，使用前者编程的程序员需要具体描述机器每一步骤的行为，使用后者则只需描述运算的逻辑，而不需要整个运算流程。指令编程式可进一步细分为过程式和面向对象，命令式则可分为函数式编程、逻辑编程等。有的编程语言只支持一种编程范式，还有的则可能支持好几种。
2. Structured Programming：结构化编程，一种采用结构化控制流（选择和循环）以提高程序质量、缩短开发时间、使程序更简洁易读的编程范式。结构化编程所使用的元素有控制结构（顺序执行、选择、迭代和递归），子过程和语句块。具有代表性结构化编程语言有ALGHOL，Pascal，Ada等，注意很多这类语言都不支持goto跳转语句。
3. Object-Oriented Programming：面向对象编程，一种基于“对象”概念的编程范式。数据以域（通常又叫属性、特性）的形式，代码以过程（通常又叫方法）的形式定义。面向对象编程的一个特点是对象可以通过自身的方法获取和修改自己的属性（通常以this或者self的形式）。在OOP程序里，对象之间进行交互。有很多不同的OOP，其中最常见的是基于类实现的，对象是类的实例，具有类定义赋予他们的特性。常用的面向对象编程语言包括Java，C++，Python，PHP等。OOP具有封装性（即类的外部可以调用类的方法，而不需要/不会直接看到内部的私有属性），继承（基类和派生类之间的关系），多态（由继承产生的相似但不同的方法），泛型（同一方法对不同数据类型的支持）。

为什么面向对象方法在模块化和复用性方面优于传统方法？

模块化：面向对象方法按照人们通常的思维方式建立问题领域的模型，相比传统方法提高了问题抽象的层次，也就更容易根据模型不同类之间的区别和联系进行模块的划分。传统方法只能在零散的变量和过程上进行划分，如划分成包，不同模块之间的界限比较模糊，而面向对象方法有效地实现了抽象、分类和封装，各类的私有属性和方法实现细节对外不可见，通过类之间的消息传递进行通讯，则模块化的程度大大提高了。比如一个贪吃蛇小游戏，可以很自然地定义蛇、食物、地图和游戏控制/玩家交互类，有助于厘清游戏的逻辑。

复用性：传统方法中零散的变量和过程复用起来工作比较琐碎，如修改变量、函数或添加新变量并为其提供功能相似的新函数。面向对象方法中类的继承、多态和泛型机制为代码复用提供了强力的支持。为类添加新的域、新的函数、进行操作符重载来使某一方法支持更多不同类型的参数，保留旧的基类并添加新的派生类，这些在面向对象方法中都很容易。