第二周借助矩形菱形的打印介绍了计算机计算过程的流程控制以及面向过程与面向对象两种代码组织方式。

其中面向过程的编程使用整体命名法，段头使用“def 名字：”，段体使用相同的缩行做区分，缩行相似于大括号的作用，需要非常细心区分，此外可以在段头的名字后加（）用于数据输入，可以在段体结尾加入return用于数据输出，调用时只需“名字（）”即可，在括号内输入条件值时可以待参数名也可以不带，但一定保持统一，当存在多个参数时，可使用keyword参数使参数的顺序可以随意；但不使用keyword参数，则必须按照def的顺序设置。

面向对象的编程则将数据与过程分隔，”def \_init\_(self,参数):”作为构造器采用“self.数据=值”定义数据，之后是处理过程。调用时需要先定义对象初始化数据，之后再调用对象的相应过程处理数据。

之后老师就作业提交方式介绍了git工具的三种应用场景以及项目工程代码管理方法。第一个应用场景是第一次创建仓库初始化，提交master公共文件。第一次创建工作目录并在命令行中使用cd切换到工作路径后再使用git init初始化，第一次使用时需设置个人名字与邮件，因为没有做好初始化，我在后续提交作业时遇到了“please tell me who are you”的错误提示。第二个应用场景是建立分支并提交代码到分支，建立与删除服务器代理，拉取仓库中的代码，查看、创建、切换分支时需非常细心，最好每次操作都通过git branch查看当前分支。提交时使用“git add \*”“ git commit -m 更新了GUI -a”“git push --set-upstream origin 分支名”三个连续步骤即可。第三个场景是后期分支代码合并，克隆服务器仓库并拉去分支代码，之后在主分支中“git master 分支名”实现合并到master，最后用同方法提交到仓库。

结束python语言初步学习后，进入Qt编程学习。学习中在很多地方直接使用他人已经定义好的类，因此安装了很多第三方模块，在安装与实践中不断遇到问题，不断尝试解决进而更深入地理解路径、系统。使用类的过程中，老师提出不必过度纠结一些代码原理，善于使用帮助手册辅助学习的观点。首先创建QApplication，再使用from、import引入类，其中QtWidgets，QtCore，QtGui，QThread是其中重点类。在这一过程中老师穿插了面向对象的代码复用方式——继承。

Qt的高级编程涉及人机交互，信号和槽的理解有一定难度，但在一步步地模仿，尤其是在后续大嘴鱼的使用中建立了一定的模糊概念。Qt应用模版更是简单易理解，QDailog构造窗体模块、QApplication应用模块、main.py主模块、用户调用脚本既实现数据封装代码高级，又方便用户使用。为了更好的人机交互，使用Qt提供的键盘事件，Qpen绘制图像。但多线程的理解与使用还不太熟练，有待在后续实践中提高。