1. **编码规范**
2. 变量

局部变量必须以小写开头。

|  |
| --- |
| pkt = {};  pkt[*"msg"*] = *"createPlayer"*;  pkt[*"player"*] = net.\_player;  *self*.sendEvent(pkt); |

成员变量必须以\_+小写开头。

|  |
| --- |
| *self*.\_db = None; # 数据库操作类  *self*.\_logic = player\_logic.PlayerLogic.\_instance; # 玩家逻辑 |

1. 常量/枚举

常量/枚举必须全部使用大写，并且用\_分割单词。

|  |
| --- |
| ## 月份最大值  MONTH\_MAXCOUNT = 12;  ## 能力最大值  ABI\_MAXCOUNT = 10;  ## 等级最大值  LEVEL\_MAXCOUNT = 100;  ## 男性  SEX\_MALE = 1;  ## 女性  SEX\_FEMALE = 2; |

枚举用类型开头。

|  |
| --- |
| ## 男性  SEX\_MALE = 1;  ## 女性  SEX\_FEMALE = 2; |

所有数字和字符串都应该定义成一个常量。

|  |
| --- |
| ## 月份最大值  MONTH\_MAXCOUNT = 12; |

1. 函数

函数名用小写开头。

|  |
| --- |
| ## 创建初始数据  # @param net 网络连接  # @param name 名字  # @param model 模型  # @param month 出生月份  # @return 成功返回None，否则返回唯一字符错误码  def **createData**(*self*, net, name, model, month): |

1. 类

类名必须以大写开头。

|  |
| --- |
| ## 玩家逻辑  class **PlayerLogic**: |

1. 缩写词

缩写词最多只使用首字母大写。

|  |
| --- |
| ## 创佳你html文档  def **createHtml**(*self*): |

1. 测试值

检测时尽量不要省略右值。应为python中0和None是不同的数据

|  |
| --- |
| if playerId == None: # id有可能为0  return *"PLAYERLOGIC\_REMOVE\_NTLOGIN"*; |

1. 留空

操作符，关键字，冒号分号之类的前后注意使用空格留空。

|  |
| --- |
| # 检查是否可以叠加  if t.\_stackable == False:  return *"ITEMLOGIC\_SPLIT\_UNSTACKABLE"*; |

逻辑及不同组的变量之间用换行分隔开。

|  |
| --- |
| # 检查是否可以叠加  if t.\_stackable == False:  return *"ITEMLOGIC\_SPLIT\_UNSTACKABLE"*;  # 检查物品数  if srcCount <= 0 or dstCount <= 0:  return *"ITEMLOGIC\_SPLIT\_COUNTINVALID"*;  if srcCount + dstCount != srcItem[item\_define.ITEMDATA\_COUNT]:  return *"ITEMLOGIC\_SPLIT\_COUNTINVALID"*; |

1. **注释风格**
2. 文件头

|  |
| --- |
| # -\*- coding:GBK -\*-  ## 游戏服务器  # @author Dashi Bai  # @date 2011-06-21  # @note 额外说明  # @bug 要修改的BUG  # @todo 还需要做的事情 |

note，bug，todo标记是非必须的

1. 类

|  |
| --- |
| ## 游戏服务器  # @note 额外说明  # @par \_players 玩家数据  # @bug 要修改的BUG  # @todo 还需要做的事情  class **GameServer**(game\_module.GameModule): |

par为public的成员变量,格式为# @par 变量名 说明

note，bug，todo标记是非必须的

1. 枚举/常量

|  |
| --- |
| ## 月份最大值  MONTH\_MAXCOUNT = 12;  ## 能力最大值  ABI\_MAXCOUNT = 10;  ## 等级最大值  LEVEL\_MAXCOUNT = 100;  ## 男性  SEX\_MALE = 1;  ## 女性  SEX\_FEMALE = 2; |

1. 变量

|  |
| --- |
| ## 实例  \_instance = None;  ## 构造函数  # @param slotId 打卡数据操作下标  # @param effect 奖励  def **\_\_init\_\_**(*self*, slotId, effect):  *self*.\_slotId = slotId; # 打卡数据操作下标  *self*.\_effect = effect; # 奖励 |

由于python成员变量是不需要在类中声明的，所以直接写在构造函数中就行了。即\_\_init\_\_函数内的self.\_slotId = slotId;这块。注意由于成员变量不对外暴露，所以只用一个#号

而类静态变量需要在类中声明下，如果他暴露在外面则在前面用两个##,如果不希望他暴露则在类后面用一个#号表示，如成员变量所示

1. 函数

|  |
| --- |
| ## 创建玩家数据  # @param player 玩家信息  # @param sex 性别  # @param month 出生月份  # @param model 模型  # @return 返回新创建的玩家  # @note 额外说明  # @bug 要修改的BUG  # @todo 还需要做的事情  def **createPlayerData**(player, sex, month, model): |

param标记表示变量参数

return 标记表示返回的数据

note，bug，todo标记是非必须的

1. 代码片段

|  |
| --- |
| ## 烹饪  # @param player 要操作的玩家  # @param id 触发器ID  # @return 成功返回None，否则返回唯一字符错误码  def **cook**(*self*, player, id):  # 取得烹饪  dat = player[*"dat"*];  profession = dat[*"pro"*];  c = None;  for p in profession:  if p[profession\_define.PROFESSIONDATA\_TYPE] == profession\_define.PROFESSIONTYPE\_COOK:  c = p;  break;  if c == None:  return *"COOKLOGIC\_COOK\_COOKNONE"*;    # 取得触发器  t = *self*.getTrigger(id);    # 检查条件  err = t.testCook(player);  if err != None:  return err;    # 烹饪  t.cook(player);    # 增加熟练度  c[profession\_define.PROFESSIONDATA\_EXP] = c[profession\_define.PROFESSIONDATA\_EXP]+1;    # 设置修改值  modify = player[*"modify"*];  modify[*"pro"*] = profession; |

如果不是原子操作，则必须对代码进行详细的注释。务必写清楚各块的功能。注意，由于不需要暴露，所以用一个#号