**广东海洋大学学生实验报告书（学生用表）**

**GDOU-B-11-112**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | Python面向对象程序设计 | | | | 课程名称 | | 高级程序设计语言 | | | | 课程号 | |  |
| 学院(系) | 数学与计算机学院 | | 专业 | 物联网工程 | | | | | 班级 | | 物联1181 | | |
| 学生姓名 | 陈用林 | 学号 | 201811672102 | | | 实验地点 | | 二教A501 | | 实验日期 | | 2020.11.2 | |

1. **实验内容：**

**1、自定义一个三维向量类，并定义相应的特殊方法实现两个该类对象之间的加、减运算（要求支持运算符+、- ），实现该类对象与标量的乘、除运算（要求支持运算符（\*、/ ），以及向量长度的计算（要求使用属性实现）。**

**（1）程序代码：**



class Vector:

def \_\_init\_\_(self,x,y,z):

self.x = x

self.y = y

self.z = z

def add(self,another):

x = self.x + another.x

y = self.y + another.y

z = self.z + another.z

print('加法运算的结果为：({},{},{})'.format(x, y, z))

def sub(self,another):

x = self.x - another.x

y = self.y - another.y

z = self.z - another.z

print('减法运算的结果为：({},{},{})'.format(x, y, z))

def mul(self,n):

x = self.x \* n

y = self.y \* n

z = self.z \* n

print('乘法运算的结果为：({},{},{})'.format(x, y, z))

def tru(self,n):

x = self.x / n

y = self.y / n

z = self.z / n

print('除法运算的结果为：({},{},{})'.format(x,y,z))

def length(self):

a = pow(pow(self.x, 2) + pow(self.y, 2) + pow(self.z, 2), 0.5)

print('长度为：{}'.format(round(a, 3)))

v1 = Vector(1,2,3)

v2 = Vector(2,3,4)

v1.add(v2)

v1.sub(v2)

v1.mul(2)

v1.tru(2)

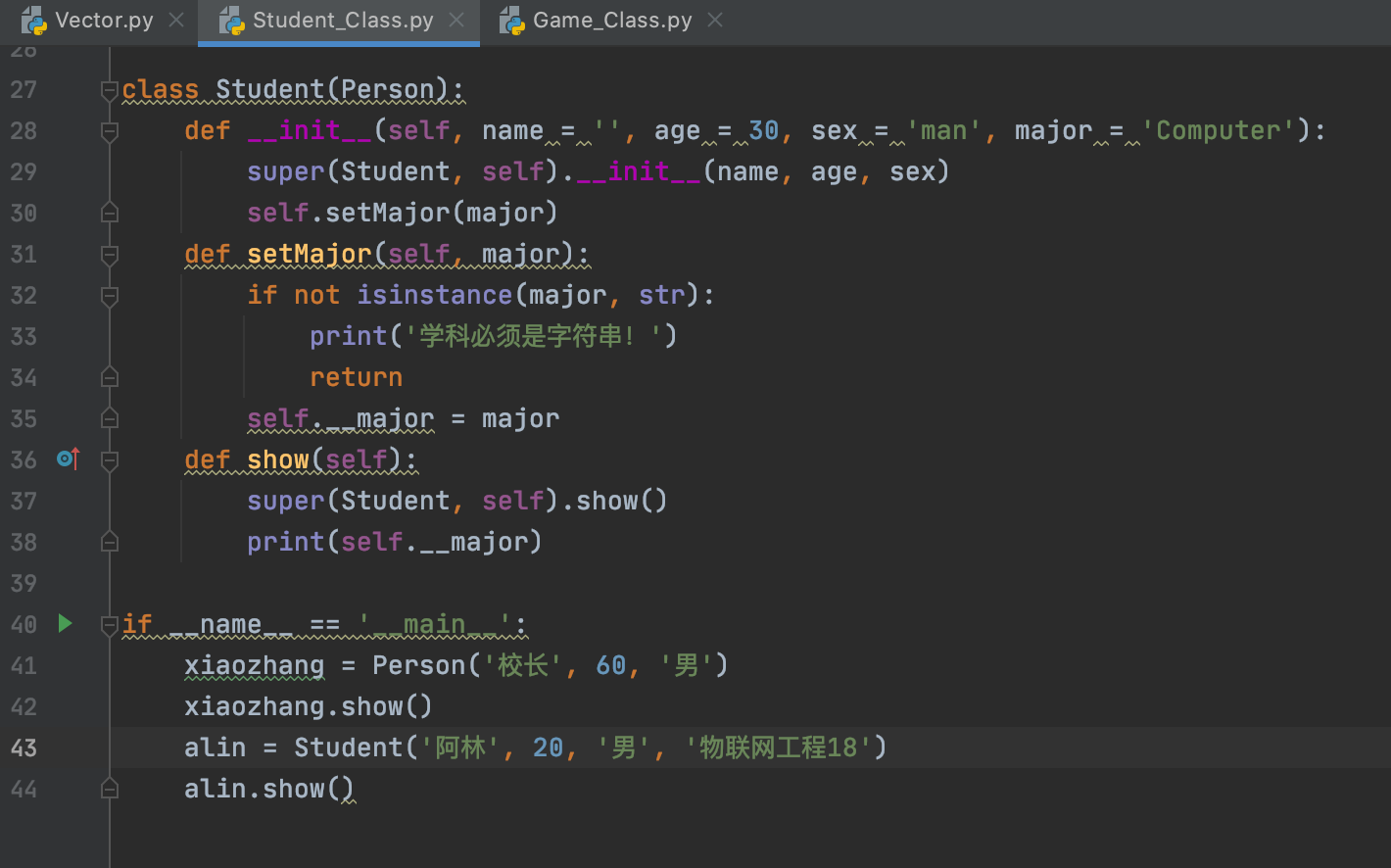
v1.length()

**（2）运行结果（截图）：**



**2、继承已定义的Person类生成Student类，编写新的函数用来设置学生专业，然后生成该类对象并显示信息。**

**（1）程序代码：**



import types

class Person(object):

def \_\_init\_\_(self, name = '', age = 20, sex = 'man'):

self.setName(name)

self.setAge(age)

self.setSex(sex)

def setName(self, name):

if not isinstance(name,str):

print('姓名必须为字符串！')

return

self.\_\_name = name

def setAge(self, age):

if not isinstance(age,int):

print('年龄必须是整数！')

return

self.\_\_age = age

def setSex(self, sex):

if sex != '男' and sex != '女':

print('性别必须是男性或女性！')

return

self.\_\_sex = sex

def show(self):

print(self.\_\_name)

print(self.\_\_age)

print(self.\_\_sex)

class Student(Person):

def \_\_init\_\_(self, name = '', age = 30, sex = 'man', major = 'Computer'):

super(Student, self).\_\_init\_\_(name, age, sex)

self.setMajor(major)

def setMajor(self, major):

if not isinstance(major, str):

print('学科必须是字符串！')

return

self.\_\_major = major

def show(self):

super(Student, self).show()

print(self.\_\_major)

if \_\_name\_\_ == '\_\_main\_\_':

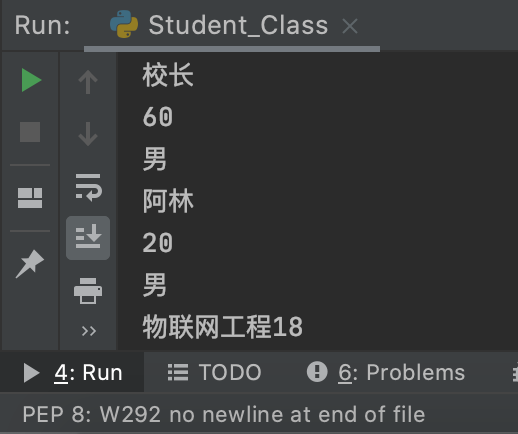
xiaozhang = Person('校长', 60, '男')

xiaozhang.show()

alin = Student('阿林', 20, '男', '物联网工程18')

alin.show()

**（2）运行结果（截图）：**



**3、设计一个Game类，要求如下：**

**（1）属性**

**定义一个属性top\_score记录游戏的历史最高分**

**定义一个属性player\_name记录当前游戏玩家姓名**

**（2）方法**

**show\_help显示游戏帮助信息**

**show\_top\_score显示历史最高分（最高分数可以用随机函数产生一个0-100之间的随机整数表示）**

**show\_game开始当前玩家的游戏**

**（1）程序代码：**



class Game(object):

top\_score = 0

def \_\_init\_\_(self, player\_name):

self.player\_name = player\_name

@staticmethod

def show\_help():

print('【游戏帮助信息】\n')

@classmethod

def show\_top\_score(cls):

print('分数为：%d'%cls.top\_score,'\n')

def start\_game(self):

print('%s:游戏开始啦!'%self.player\_name)

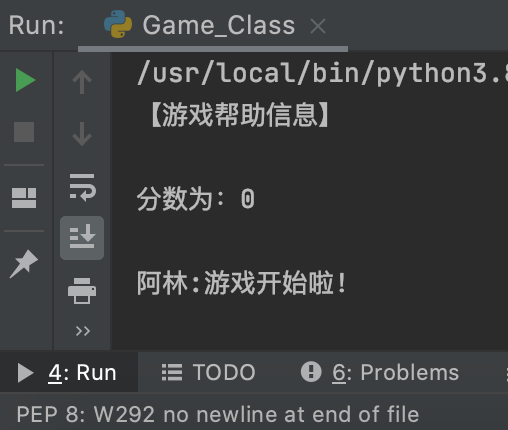
Game.show\_help()

Game.show\_top\_score()

player1 = Game('阿林')

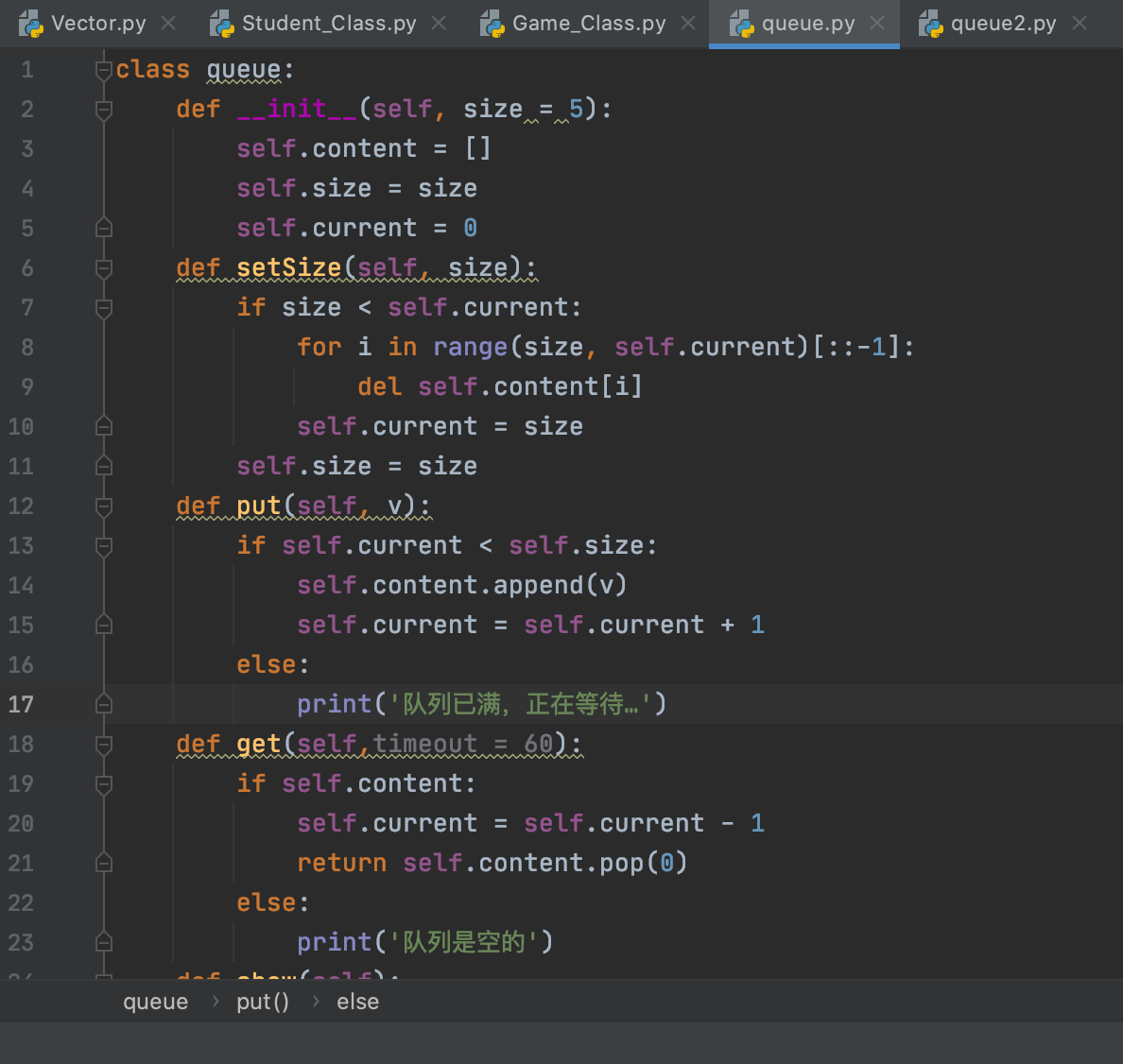
player1.start\_game()

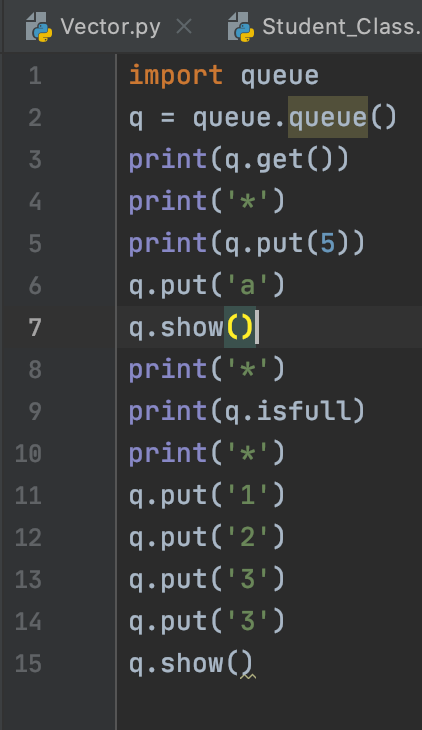
**（2）运行结果（截图）：**



**4、编写程序，实现自定义类，模拟队列结构。要求实现入队、出队以及修改队列大学和判断队列是否为空、是否为满的功能；同时要求在入队时如果队列已满，则等待指定时间；出队时如果队列已空则等待指定时间等辅助功能。**

**（1）程序代码：**





class queue:

def \_\_init\_\_(self, size = 5):

self.content = []

self.size = size

self.current = 0

def setSize(self, size):

if size < self.current:

for i in range(size, self.current)[::-1]:

del self.content[i]

self.current = size

self.size = size

def put(self, v):

if self.current < self.size:

self.content.append(v)

self.current = self.current + 1

else:

print('队列已满，正在等待…')

def get(self,timeout = 60):

if self.content:

self.current = self.current - 1

return self.content.pop(0)

else:

print('队列是空的')

def show(self):

if self.content:

print(self.content)

else:

print('队列是空的')

def empty(self):

self.content = []

self.current = 0

def isempty(self):

if not self.content:

return True

else:

return False

def isfull(self):

return self.current == self.size

if \_\_name\_\_ == '\_\_main\_\_':

print('使用模块')

import queue

q = queue.queue()

print(q.get())

print('\*')

print(q.put(5))

q.put('a')

q.show()

print('\*')

print(q.isfull)

print('\*')

q.put('1')

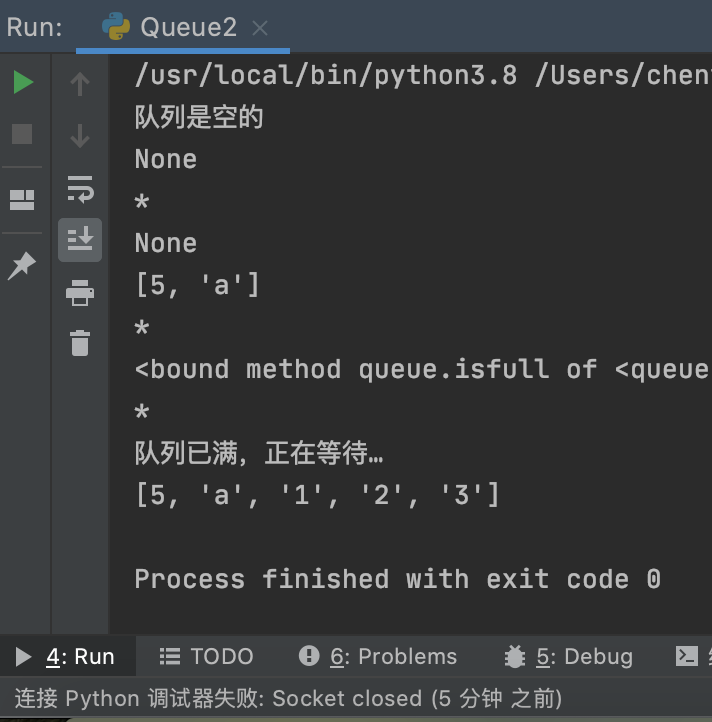
q.put('2')

q.put('3')

q.put('3')

q.show()

**（2）运行结果（截图）：**



1. **实验总结：**

**（本次实验的知识点小结；实验体会；未解决的问题；改进的方法）**

通过本次试验，基本掌握了Python的面向对象程序设计的方法、技巧和内容，实践了好几个题目项目，中间也遇到了问题，但是最终通过纠错还是成功完成对应的内容，代码都能正常运行，要求效果都能实现，获益良多。