

**Harbin Institute of Technology, Weihai**

**Python程序设计**

**实验报告**

**学 号： 180400715**

**姓 名： 杨卓宸**

**班 级： 1804007**

**组 号：**

**任课教师： 张小东**

哈尔滨工业大学（威海）计算机科学与技术学院

前 言

《Python语言程序设计》实验是基于限定性自选项目的实践训练，学生将利用《Python语言程序设计》中的基本概念、原理、技术和方法，开展针对实际应用的问题的程序设计及相关语言训练。

通过本实验课程的训练和实践，引导学生熟练掌握python语言程序设计要点，能够针对一些简单的实际问题，进行问题分析、使用规范的方法描述建立应用软件模型，选择适当的数据结构、设计有效算法及程序代码编写。

本实验课程要求学生通过一定的调研，结合实际问题的应用需求来选题，并由任课教师来对学生选题做评定。要求所设计及研发的程序符合软件设计的规律，能够按照规范的格式进行书写，能够完整的展示出软件设计与编写的过程，并能清晰的表达出来。本实践课程主要的教学环节包括：学生按限定的要求自选题目、对题目进行分析、设计、编写代码、实验报告撰写等。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 报告评价等级 | A+ | A | B+ | B | C+ | C |
| D+ | D | E+ | E | F+ | F |

# 实验总体要求

为避免重复与抄袭，python程序设计的实验只规定选题范畴及相关要求，具体的题目由学生依据现实当中的问题自行拟定，选题的难易会影响实验得分。题目尽量避免重复。完全相同的实验报告得0分，不同的重复率扣不同的分数。

所有实验都需要包含八个组成部分：

1. 实验题目

要求：一句简要的话概括或抽象出所做的实验内容

1. 题目功能描述

要求：题目当中有多个模块、类或者函数，每个模块、类或者函数的功能。

1. 设计与代码描述

要求：

1. 设计描述

类，需要符合UML要求的类图，表示出类间关系；

函数需要有流程图表达出设计思路。

1. 代码

根据设计中的描述，按类、函数依次编写代码。

1. 调试分析记录

要求：

列出测试数据，展示测试结果并进行分析。

# 实验一

1. 实验题目

软件工程中基本设计原则在python程序设计中的应用

1. 实验所要遵循的规则

请用自己的语言来解释一下三面个原则，并这些原则应用到了代码设计的哪一段？

1. 单一职责原则（SRP：Single responsibility principle）

单一职责原则的意思是，一个类应该只有一个发生变化的原因。因为当一个类有多个可以发生改变的原因的话，当其中一个改变导致类受到变化后，另一个改变原因的功能可能出现问题，也就是第一个原因和第二个原因被整合归类到了一起，这样程序就会出现原则性错误。

1. 里氏代换原则(Liskov Substitution Principle, LSP)

里氏代换原则的意思是：任何基类可以出现的地方，子类一定可以出现。这个原则的意思其实是，只有当衍生类可以替换掉基类，并且软件功能不受影响时，基类才可以真正意义上被复用。这个原则实际上是对实现抽象化的具体示范。

1. 开闭原则(Open-Closed Principle, OCP)

开闭原则的意思是：一个软件实体，如类，模块，或函数都应该对扩展开放，对修改关闭。这个意思是，当软件进行更新升级的时候，若对原有代码修改，可能会对旧的程序代码引入错误，导致巨大的问题。所以更新时，我们应该通过扩展软件的方法来实现变化。

【实验示例】

(1) abstractChart：绘制统计图的抽象类

(2) PieChart和BarChart：图表类，继承于abstractChart

(3) Record：根据不同要求，调用PieChart和BarChart两个类

Chart=AbstractChart()

setChart(self,chart):

self.Chart=chart

Display():­­­­­­­

chart.display();

由于使用抽象类abstractChart为Record中对象，在Record被使用时，依据传入的实例化对象来最终确定chart的方法是PieChart还是BarChart的。符合上述三定理。

1. 题目功能描述

我设计的这个程序，目的是实现输入iPhone的机型，可将iPhone的草图画出，并且在草图iPhone的屏幕上注明这款iPhone的各个参数。

1. 设计与代码描述

我设计了一个父类iPhone，在此基础上设计了两个子类，一个是经典大边框的iPhone，例如iPhone 4，iPhone 6等等，一个是全面屏iPhone，例如iPhone X，iPhone XS Max。每个子类中有各自的show方法，为了将iPhone草图画出。

草图的绘画，我采用的是turtle库，计算好每款iPhone的长宽大小按比例输出，并且利用printer将参数在画布上打印出。

我利用了字典的比对，来调用每款iPhone的字典参数。

下面是代码：

iPhone.py

#!/usr/bin/python

# -\*- coding:utf-8 -\*-

import time

import turtle

def round\_rectangle(length,high,cor\_angle,cor\_rad):

    for i in range(2):

        turtle.fd(high)

        turtle.circle(cor\_rad,cor\_angle)

        turtle.fd(length)

        turtle.circle(cor\_rad,cor\_angle)

class iPhone:

    def \_\_init\_\_(self):

        self.name='iPhone'

        self.brand='Apple'

        self.system='iOS'

    def show(self):

        print(self.name+'\n'+self.brand+'\n'+self.system)

class iPhone\_bezel\_less(iPhone):

    def \_\_init\_\_(self,a):

        super(iPhone\_bezel\_less,self).\_\_init\_\_()

        self.name+=' '+a['name']

        self.screensize=a['screensize']

        self.color=a['color']

        self.year=a['year']

        self.processor=a['processor']

    def show(self):

    #画布

        turtle.setup(500,800)

        pythonsize=2

        turtle.pensize(pythonsize)#画笔宽度

        turtle.speed(20)

        turtle.seth(90)#启动时运行角度

    #最外边框

        turtle.pencolor("black")

        turtle.penup()

        turtle.goto(151,-231)

        turtle.pendown()

        round\_rectangle(242,532,90,30)

    #填充

        turtle.penup()

        turtle.goto(150,-230)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("black")

        round\_rectangle(240,530,90,30)

        turtle.end\_fill()

    #屏幕

        turtle.pencolor("black")

        turtle.penup()

        turtle.goto(140,-225)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("#ffffff")

        round\_rectangle(230,520,90,25)

        turtle.end\_fill()

    #听筒

        turtle.penup()

        turtle.goto(75,310)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("black")

        round\_rectangle(130,10,90,10)

        turtle.end\_fill()

        turtle.hideturtle()

    #输出文字

        printer = turtle.Turtle()

        printer.hideturtle()

        printer.goto(0,0)

        printer.penup()

        printer.color("black")

        printer.write(self.name+"\n\n\n\n\n",align="center", font=("Consolas",16,"bold"))

        printer.write(self.brand+'\n\n\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write(self.system+'\n\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write('in '+str(self.year)+'\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write(str(self.screensize)+' inches Display\n', align="center", font=("Consolas", 16, "bold"))

        printer.write(self.processor+' Processor',align="center", font=("Consolas",16,"bold"))

        printer.goto(0,-20)

        printer.write(self.color,align="center", font=("Consolas",16,"bold"))

        time.sleep(10)

class iPhone\_classic(iPhone):

    def \_\_init\_\_(self,a):

        super(iPhone\_classic,self).\_\_init\_\_()

        self.name+=' '+a['name']

        self.screensize=a['screensize']

        self.color=a['color']

        self.year=a['year']

        self.processor=a['processor']

    def show(self):

    #画布

        turtle.setup(500,800)

        pythonsize=2

        turtle.pensize(pythonsize)#画笔宽度

        turtle.speed(20)

        turtle.seth(90)#启动时运行角度

    #最外边框

        turtle.pencolor("#8E8e8e")

        turtle.penup()

        turtle.goto(152,-202)

        turtle.pendown()

        round\_rectangle(244,484,90,30)

    #填充

        turtle.penup()

        turtle.goto(150,-200)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("#ffffff")

        round\_rectangle(240,480,90,30)

        turtle.end\_fill()

    #屏幕

        turtle.pencolor("black")

        turtle.penup()

        turtle.goto(135,-150)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("black")

        round\_rectangle(270,380,90,0)

        turtle.end\_fill()

    #听筒

        turtle.penup()

        turtle.goto(30,265)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("#474747")

        round\_rectangle(60,4,90,1)

        turtle.end\_fill()

    #感光器

        turtle.penup()

        turtle.goto(5,290)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("#3c3c3c")

        turtle.circle(4,360)

        turtle.end\_fill()

    #摄像头

        turtle.penup()

        turtle.goto(-50,265)

        turtle.pendown()

        turtle.begin\_fill()

        turtle.color("#3c3c3c")

        turtle.circle(6,360)

        turtle.end\_fill()

    #Home键

        turtle.pencolor("#9d9d9d")#（#3000440）

        turtle.penup()

        turtle.goto(25,-185)

        turtle.pendown()

        turtle.circle(25,360)

    #Home键方框

        turtle.pencolor("#9d9d9d")#（#3000440）

        turtle.penup()

        turtle.goto(10,-190)

        turtle.pendown()

        round\_rectangle(9, 9, 90, 5)

        turtle.hideturtle()

    #输出文字

        printer = turtle.Turtle()

        printer.hideturtle()

        printer.goto(0,0)

        printer.penup()

        printer.color("#ffffff")

        printer.write(self.name+"\n\n\n\n\n",align="center", font=("Consolas",16,"bold"))

        printer.write(self.brand+'\n\n\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write(self.system+'\n\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write('in '+str(self.year)+'\n\n',align="center", font=("Consolas",16,"bold"))

        printer.write(str(self.screensize)+' inches Display\n', align="center", font=("Consolas", 16, "bold"))

        printer.write(self.processor+' Processor',align="center", font=("Consolas",16,"bold"))

        printer.goto(0,-20)

        printer.write(self.color,align="center", font=("Consolas",16,"bold"))

        time.sleep(10)

iPhone = {'name':'','screensize':3.5,'color':'Silver','year':2007,'processor':'A1'}

iPhone3g = {'name':'3G','screensize':3.5,'color':'Black','year':2008,'processor':'A2'}

iPhone3gs = {'name':'3GS','screensize':3.5,'color':'Black','year':2009,'processor':'A3'}

iPhone4 = {'name':'4','screensize':3.5,'color':'Black','year':2010,'processor':'A4'}

iPhone4s = {'name':'4S','screensize':3.5,'color':'White','year':2011,'processor':'A5'}

iPhone5 = {'name':'5','screensize':4,'color':'Silver','year':2012,'processor':'A6'}

iPhone5s = {'name':'5S','screensize':4,'color':'Gold','year':2013,'processor':'A7'}

iPhone5c = {'name':'5C','screensize':4,'color':'Blue','year':2013,'processor':'A6'}

iPhone6 = {'name':'6','screensize':4.7,'color':'Space Grey','year':2014,'processor':'A8'}

iPhone6p = {'name':'6 Plus','screensize':5.5,'color':'Gold','year':2014,'processor':'A8'}

iPhone6s = {'name':'6S','screensize':4.7,'color':'Rose Gold','year':2015,'processor':'A9'}

iPhone6sp = {'name':'6S Plus','screensize':5.5,'color':'Rose Gold','year':2015,'processor':'A9'}

iPhonese = {'name':'SE','screensize':4,'color':'Pink Gold','year':2016,'processor':'A9'}

iPhone7 = {'name':'7','screensize':4.7,'color':'Bright Black','year':2016,'processor':'A10'}

iPhone7p = {'name':'7 Plus','screensize':5.5,'color':'Bright Black','year':2016,'processor':'A10'}

iPhone8 = {'name':'8','screensize':4.7,'color':'Glass Black','year':2017,'processor':'A11'}

iPhone8p = {'name':'8 Plus','screensize':5.5,'color':'Glass Black','year':2017,'processor':'A11'}

iPhonex = {'name':'X','screensize':5.8,'color':'White','year':2017,'processor':'A11'}

iPhonexr = {'name':'Xr','screensize':6.1,'color':'Coral','year':2018,'processor':'A12'}

iPhonexs = {'name':'XS','screensize':5.8,'color':'Gold','year':2018,'processor':'A12'}

iPhonexsm = {'name':'XS Max','screensize':6.5,'color':'Gold','year':2018,'processor':'A12'}

L={'iPhone':iphone,'3g':iPhone3g,'3gs':iPhone3gs,'4':iPhone4,'4s':iPhone4s,'5':iPhone5,'5s':iPhone5s,'5c':iPhone5c,'6':iPhone6,'6p':iPhone6p,'se':iPhonese,'7':iPhone7,'7p':iPhone7p,'8':iPhone8p,'x':iPhonex,'xr':iPhonexr,'xs':iPhonexs,'xsm':iPhonexsm,'6s':iPhone6s,'6sp':iPhone6sp}

i=input("Please input the iPhone's name:")

if i=='x' or i=='xr' or i=='xs' or i=='xsm':

    a=iPhone\_bezel\_less(L[i])

    a.show()

else:

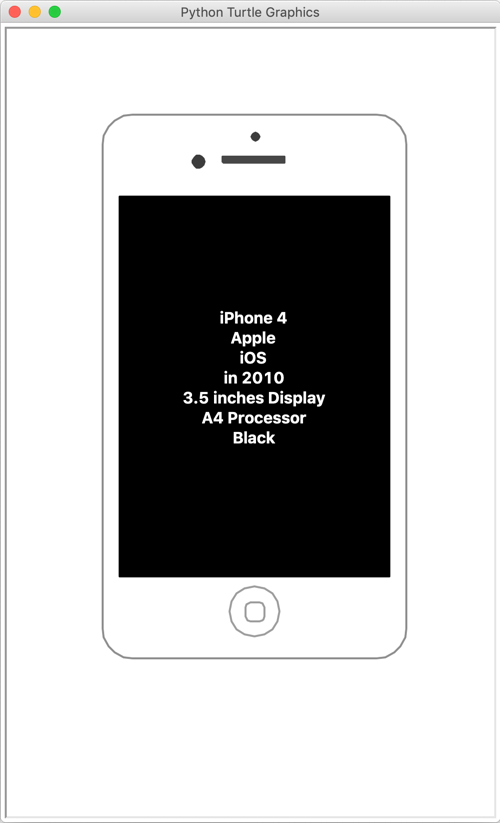
    a=iPhone\_classic(L[i])

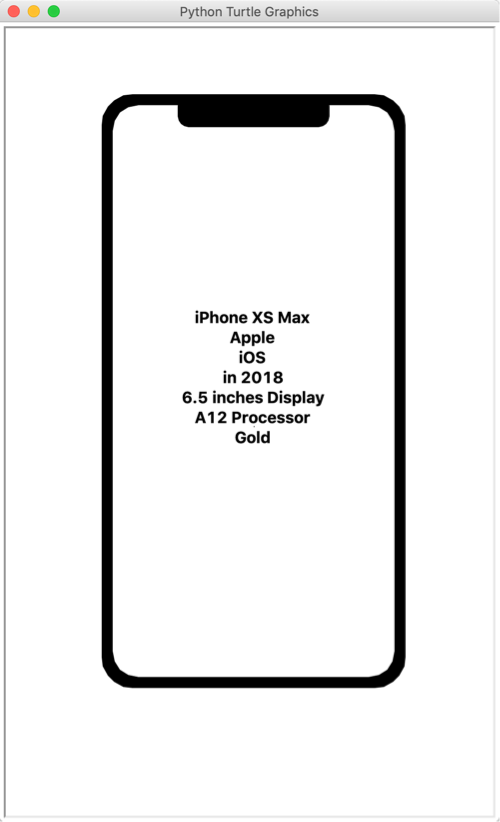
    a.show()

1. 调试分析记录

由于对字典和类的方法的不熟练，一开始总是出现SyntaxError: invalid syntax这样的问题，

对iPhone数据的测量计算是非常麻烦的，每款都要调试很多遍才可以得到最完美的样子。





# 实验二

1. 实验题目

编写基于TCP的网络通信程序

1. 题目要求

将服务器端程序和客户端程序封装成类，在带有图形界面的应用程序里进行信息交流。

1. 客户端可以向服务器端发送信息，显示已发送的信息和服务器返回的信息
2. 服务器可以显示客户端发来的信息
3. 题目功能描述

我设计的这个程序，目的是通过TCP协议的网络通信，建立聊天服务器和聊天客户端。其实质上相当于一个网络聊天室，可以多个客户端登录，可以实时通信。

1. 设计与代码描述

聊天服务器调用了sqlite3库，用于建立数据库。

Server的处理过程看似代码很多很复杂，实际上并不复杂，只需以列表判断一下data格式并且将用户名和密码存储即可。

Send message和get message也并不复杂，只需按特定格式存储并传输即可。

(1)db.py

import sqlite3

def create\_db(): # 创建数据库

conn = sqlite3.connect('test.db')

c = conn.cursor()

c.execute('''CREATE TABLE USER

(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

NAME TEXT NOT NULL,

PASSWORD TEXT NOT NULL);''')

conn.commit()

conn.close()

def register(username, password): # 注册

conn = sqlite3.connect('test.db')

c = conn.cursor()

print("Opened database successfully");

c.execute("INSERT INTO USER (NAME,PASSWORD) VALUES (\'{}\',\'{})".format(username,password));

conn.commit()

print("Records created successfully username is {username}".format(username=username));

conn.close()

return True

def login(username, password): # 登录

conn = sqlite3.connect('test.db')

c = conn.cursor()

answer = c.execute("SELECT \* FROM USER WHERE NAME = '%s' AND PASSWORD = '%s" %(username,password))

if answer :

return True

return False

# create\_db()

(2)server.py

#!/usr/bin/python

import db

chat = {'all':{'message':[]}}

class Server(asyncio.Protocol):

    def connection\_made(self, transport):

        self.transport = transport

        self.address = transport.get\_extra\_info('peername')

        self.data = b''

        self.item = self.address[0]+":"+str(self.address[1])

        chat[self.item] = {'name': '', 'message':[]}

        print('Accepted connection from {}'.format(self.address))

    def data\_received(self, data):

        self.data += data

        # print(chat[self.item]['name'])

        if chat[self.item]['name'] == '':

            if b'login' in self.data:

                x = str(self.data).split(' ')

                # print(x)

                if len(x) == 3 and 'login' in x[0]: #处理登录

                    username = x[1]

                    password = x[2]

                    if(db.login(username,password)):

                        chat[self.item]['name'] = username

                        print(chat[self.item]['name']+' login')

                        self.transport.write(b'login successful')

                    else:

                        self.transport.write(b'username or password not right')

            if b'register' in self.data: #处理注册

                x = str(self.data).split(' ')

                # print(x)

                if len(x) == 3 and 'register' in x[0]:

                    username = x[1]

                    password = x[2]

                    if(db.register(username,password)):

                        chat[self.item]['name'] = username

                        self.transport.write(b'register seccessful')

                    else:

                        self.transport.write(b'register bad')

                else:

                    print('register bad')

        if chat[self.item]['name'] == '' :

            self.transport.write(b'please login ')

        else:

            if self.data.endswith(b'$'):

                if b':' in self.data: # 判断是否私聊

                    siliao = self.data.decode()

                    people = siliao.split(':')

                    # print(people[0])

                    # print(chat)

                    for i in chat.keys():

                        try:

                            if chat[i]['name'] == str(people[0]):

                                message = "{}-->".format(chat[self.item]['name'])+people[1]

                                chat[i]['message'].append(message.encode())

                        except:

                            pass

                else:

                    answer = self.data

                    # chat[self.item]['message'].append(chat[self.item]['name'].encode()+b' : '+answer)

                    # print(chat)

                    chat['all']['message'].append(chat[self.item]['name'].encode()+b' : '+answer)

            else:

                for i in chat['all']['message']:

                    self.transport.write(i)

                for i in chat[self.item]['message']:

                    self.transport.write(i)

                    print(i+b'\n')

        # print(self.data)

        self.data = b''

    def connection\_lost(self, exc):

        if exc:

            print('Client {} error: {}'.format(self.address, exc))

        elif self.data:

            print('Client {} sent {} but then closed'.format(self.address, self.data))

        else:

            print('Client {} closed socket'.format(self.address))

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

    address = ('127.0.0.1', '8889')

    loop = asyncio.get\_event\_loop()

    coro = loop.create\_server(Server, \*address)

    server = loop.run\_until\_complete(coro)

    print('Listening at {}'.format(address))

    try:

        loop.run\_forever()

    finally:

        server.close()

        loop.close()

(3)client.py

from socket import \*

import time

buffsize = 2048

def main():

client\_scoket = socket(AF\_INET,SOCK\_STREAM)

client\_scoket.connect(('127.0.0.1',8889))

print("Welcome to chat")

while True:

text = input('What do you want to do ?')

client\_scoket.send(text.encode())

if text == "q":#退出

print('Good bye')

break

elif text == "g":#得到消息

time.sleep(0.5)

context = client\_scoket.recv(buffsize)

context = context.decode()

try:

context = context.split('$')

for i in context:

print(i)

except:

print(context)

elif text == "s":#发送消息

text = input('($ in end)\nPrivate --> [name]:[message]\n: ')

client\_scoket.send(text.encode())

else:#提醒

try:

x = str(text).split(' ')

if 'login' in x[0] or 'register' in x[0]:

# print(x)

context = client\_scoket.recv(buffsize)

print(context.decode())

else:

print('s --> send message\ng --> get message\nq --> quit\nregister [username] [password]\nlogin [username] [password]')

except:

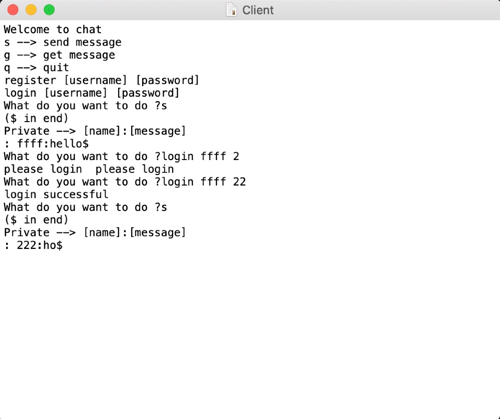
print('s --> send message\ng --> get message\nq --> quit\nregister [username] [password]\nlogin [username] [password]')

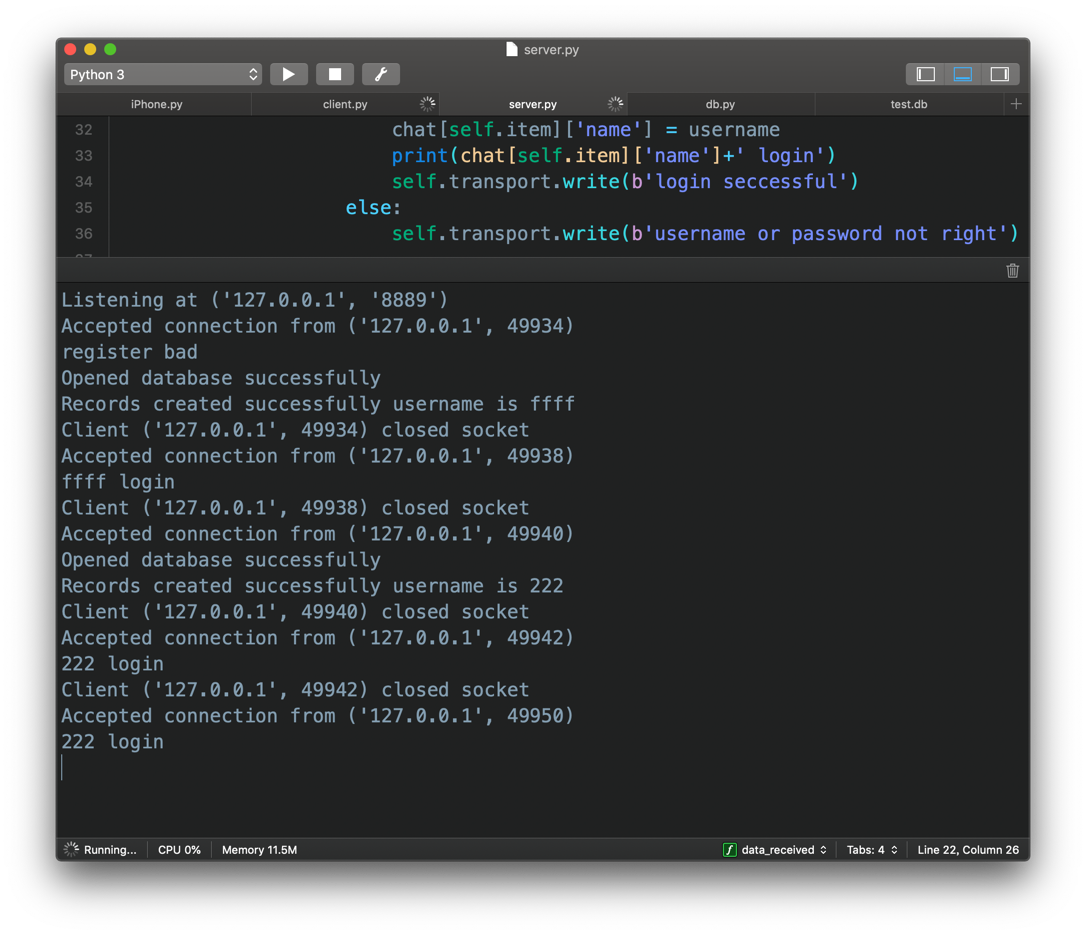
if \_\_name\_\_ == '\_\_main\_\_':

main()

1. 调试分析记录

这个是图形界面的运行窗口。



这个是服务器程序的运行输出