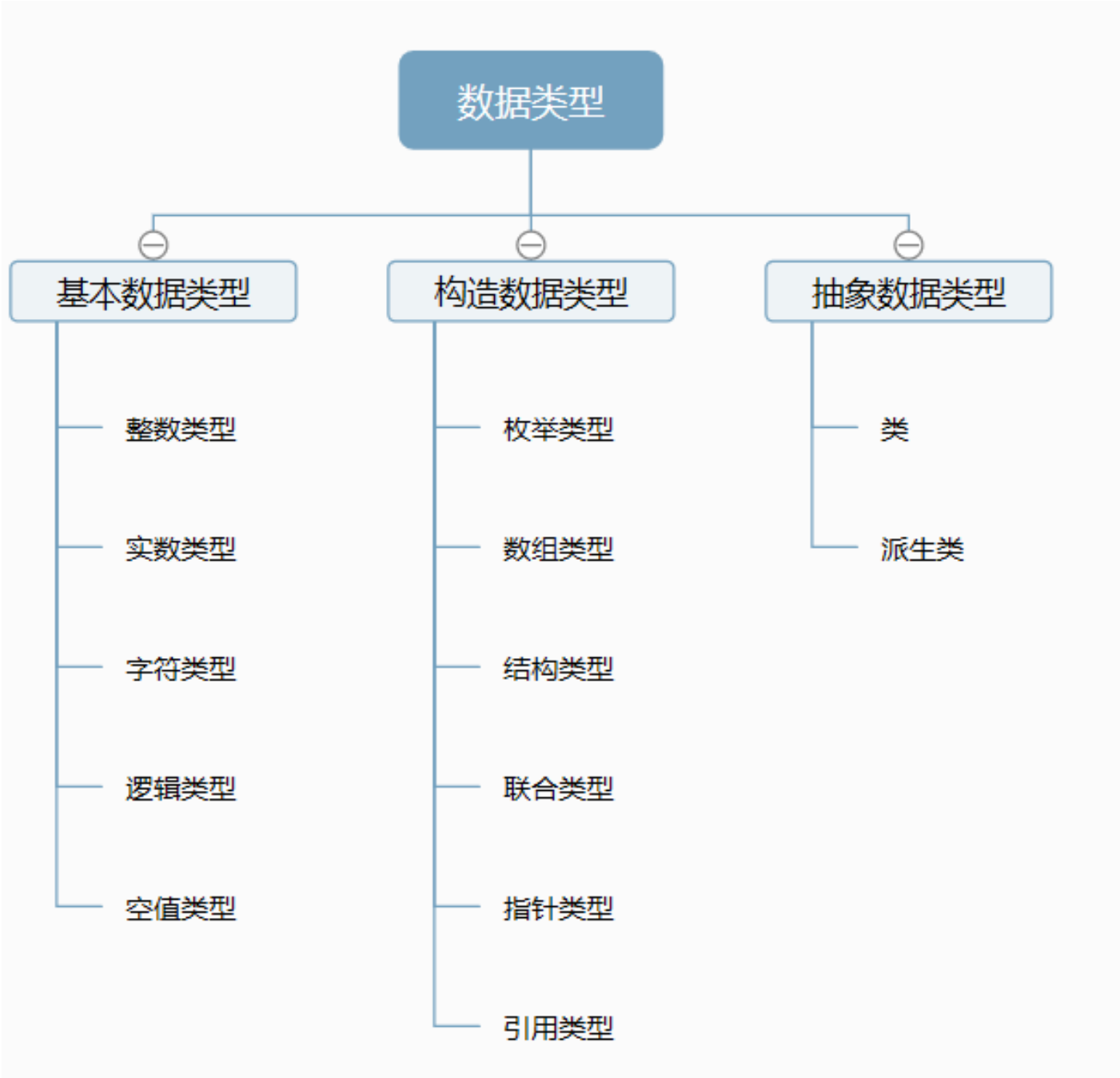
金融软件工程-第八次作业

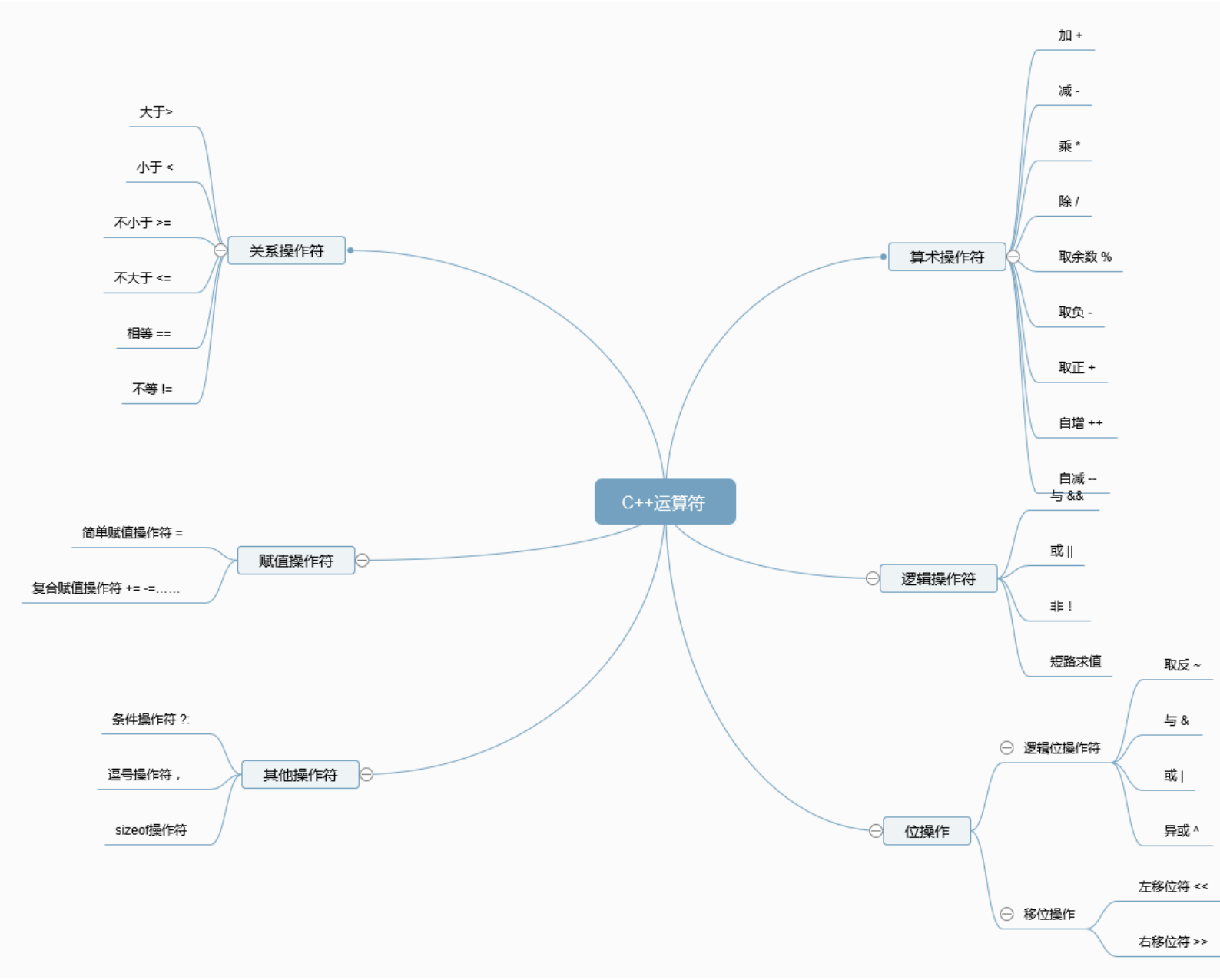
161220056 敬舒舒

一、C++程序设计语言的基本成分

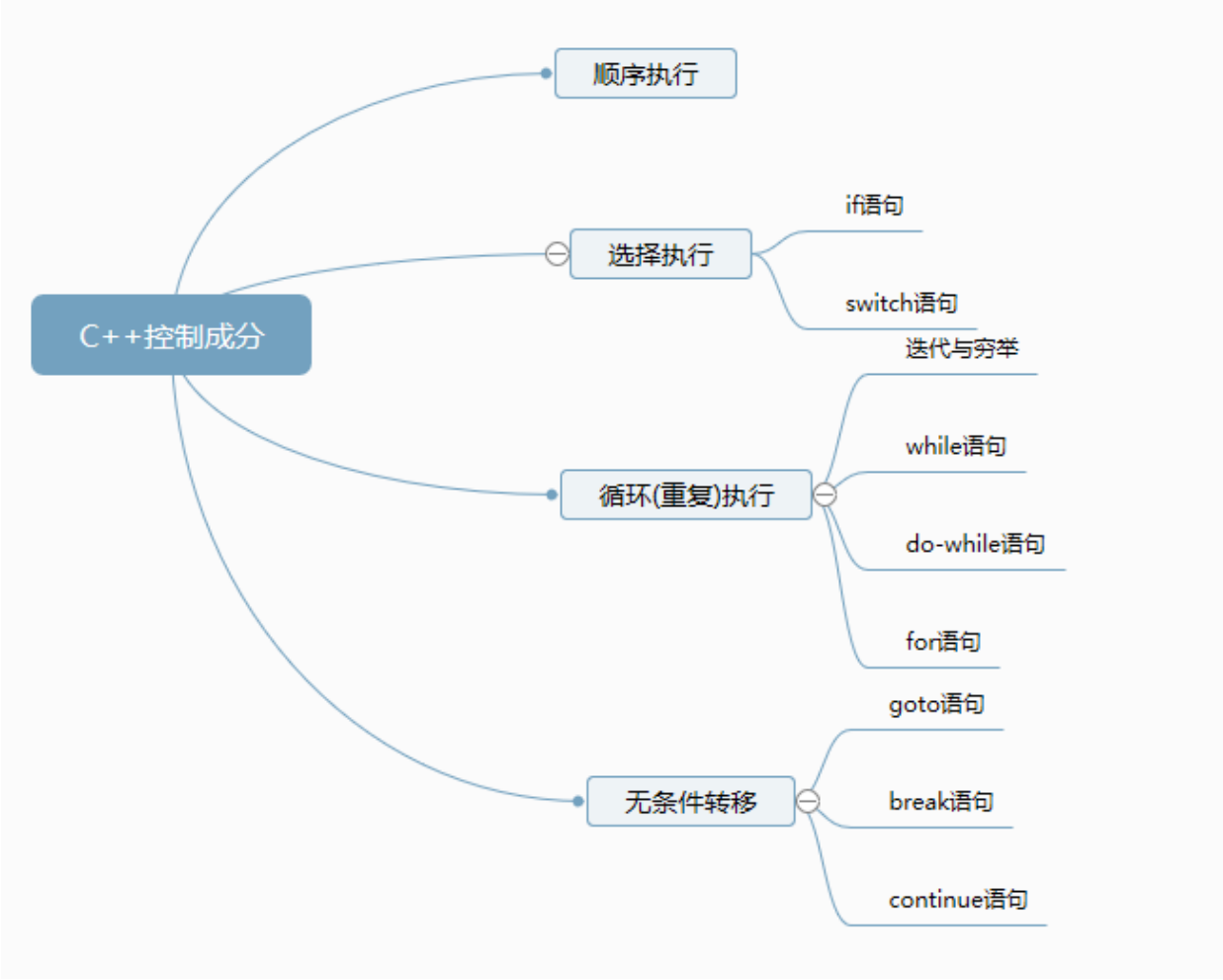
（1）C++程序设计语言的**数据成分**：



（2）C++程序设计语言的**运算成分**



（3）C++程序设计语言的**控制成分**



（4）C++程序设计语言的**传输成分**

C 风格标准输入：scanf　　标准输出流：std::cout

C 风格标准输出：print　　文件输入流：ifstream

C 风格文件输入：fscanf　　文件输出流：ofstream

C 风格文件输出：fprinf　　标准错误流：std:::cerr

标准输入流：std::cin　　标准日志流：std::clog

二、C++语言特性

（1）C++语言设计特性：

类和对象、继承、多态、虚函数和 RTTI、函数重载、引用变量、泛型编程、处理错误条件的异常机制、管理函数类和变量名的名称空间等

（2）C++语言工程特性：

优点：

设计为静态类型，高效且可移植的多用途程序设计语言

C++无需复杂的程序设计环境

C++语言灵活，具有较好的可读性和可移植性

C++计算效率较高

缺点：

C++语言本身复杂，编译系统受到 C++复杂性的影响难以编写

由于本身的复杂性，复杂的 C++程序的正确性难以保证

（3）C++语言应用特性

效率、科学计算、网络软件、分布式应用、操作系统、设备驱动系统、移动设备、嵌入式系统、教育与科研、部分行业应用、其他应用。

三、Python 开源 Flask 框架-示例代码注释分析

（1）**工程简介**：在 GitHub 上有 flask 框架的开源，其中给出了一个博客系统的实现实例，展示了 flask 框架的使用方法。由于在课程个人项目中我使用了 flask 这一 web 开发框架，故选择分析这份示例代码的注释，以更好的理解这一框架并修改自身程序。

Flask 框架是一个微型的 web 开发框架，体量较小，自由度和可扩展性较好，适合用于中小型项目开发。

本示例代码实现的是一个简单博客系统，包含 4 个程序文件：__init__.py auth.py blog.py d.py，这里不详细介绍每个文件的内容，我们来进行注释分析。

代码中的注释如下：

程序名	代码行	程序语言	序言性注释	功能性注释
__init__.py	7	Python		"""Create and configure an instance of the Flask application."""
__init__.py	10	Python		# a default secret that should be overridden by instance config
__init__.py	12	Python		# store the database in the instance folder
__init__.py	17	Python		# load the instance config, if it exists, when not testing
__init__.py	20	Python		# load the test config if passed in
__init__.py	23	Python		# ensure the instance folder exists
__init__.py	33	Python		# register the database commands
__init__.py	37	Python		# apply the blueprints to the app
__init__.py	42-45	Python		# make url_for('index') == url_for('blog.index') # in another app, you might define a separate main index here

				with
				    # app.route, while giving the blog blueprint a url_prefix, but for
				    # the tutorial the blog will be the main index

blog.py	14	Python		"""Show all the posts, most recent first."""

blog.py	17-19	Python		'SELECT p.id, title, body, created, author_id, username'
				        ' FROM post p JOIN user u ON p.author_id = u.id'
				        ' ORDER BY created DESC'

blog.py	25-35	Python		"""Get a post and its author by id.
				    Checks that the id exists and optionally that the current user is
				    the author.
				    :param id: id of post to get
				    :param check_author: require the current user to be the author
				    :return: the post with author information
				    :raise 404: if a post with the given id doesn't exist
				    :raise 403: if the current user isn't the author
				    """

blog.py	55	Python		"""Create a new post for the current user."""
blog.py	82	Python		"""Update a post if the current user is the author."""

blog.py	110-114	Python		"""Delete a post.
				
				    Ensures that the post exists and that the logged in user is the
				    author of the post.
				    """

auth.py	14	Python		"""View decorator that redirects anonymous users to the login page."""
auth.py	27-28	Python		"""If a user id is stored in the session, load the user object from the database into ``g.user``."""
auth.py	41-45	Python		"""Register a new user.

				Validates that the username is not already taken. Hashes the password for security. """
auth.py	62-63	Python		# the name is available, store it in the database and go to # the login page
auth.py	78	Python		"""Log in a registered user by adding the user id to the session."""
auth.py	94	Python		# store the user id in a new session and return to the index
auth.py	106	Python		"""Clear the current session, including the stored user id."""
db.py	9-12	Python		"""Connect to the application's configured database. The connection is unique for each request and will be reused if this is called again. """
db.py	24-26	Python		"""If this request connected to the database, close the connection. """
db.py	34	Python		"""Clear existing data and create new tables."""
db.py	43	Python		"""Clear existing data and create new tables."""
db.py	50-52	Python		"""Register database functions with the Flask app. This is called by the application factory. """

本项目程序中的注释大部分为功能性注释，对每个程序段的说明非常仔细，但是缺乏序言性注释，但其实整个程序文件的分工非常明确，从其中的文件名和功能性注释我们可以轻松看出每个文件的功能。但是，这样忽略序言性注释的做法依然不可取。

（2）程序设计风格分析

A. 内部文档

内部文档完善，在 readme 中详细介绍了程序运行环境的配置和 setup 程序

除了完整的项目代码，还包含了测试代码

标识符的命名符合其含义和代表的实体

功能性注释完善

具有严格的缩进、空行、空格结构

B. 数据说明

对于变量的声明和复制都在程序段的开头

在声明和赋值中都按照一定的顺序，符合人的思维方式，并于数据库中的声明顺序一致

在无法找到其他排序依据时，采用首字母排序的方式（比如 import 函数和变量时）

C. 语句构造

当参数列表较长时，将参数单独放置为一行或多行，便于阅读和寻错

避免一行书写多个语句

在运算符和参数间用空格间隔使程序便于阅读

在功能不同的程序段之间使用空行间隔

D. 输入输出

输入为键盘鼠标，输出为电脑屏幕

对输入进行了检测，对不符合要求的输入有着对应的处理，并告知用户错误原因。

四、个人项目代码修改

代码修改前：

```
# -*- coding: utf-8 -*-
"""

Created on Sun Apr  8 20:25:02 2018

@author: Alice_shu
"""
# -*- coding: utf-8 -*-

from flask import Flask, render_template, request, session, redirect, url_for, flash
import database as db
app = Flask(__name__)


@app.route('/signin', methods=['GET'])
def signin_form():
    return render_template('signin.html')

@app.route('/signin', methods=['POST'])
def signin():
    select =  request.form['select']
    if select == 'register':
        return redirect(url_for('register_form'))
    elif select == 'signin':
        error = None
        username = request.form['username']
        password = request.form['password']
        login = db.db_read_login(username,password)
        if login == 1:
```

```
            error = "用户名错误"#'Invalid username'
        elif login == 2:
            error = "密码错误"#'Invalid password'
        else:
            session['username'] = username
            fresh_account(username)
            print(session)
            db.db_record_operation(username,'登陆',0)
            flash('You were logged in')
            return redirect(url_for('user_form_main',username=username))
        return render_template('signin.html', message=error)


def fresh_account(username):
    user_information = db.db_read_information(username)
    session['logged_in'] = True
    session['sex'] = user_information[1]
    session['phone_number'] = user_information[2]
    session['birthday'] = user_information[3]
    session['money'] = user_information[4]
    session['credit'] = user_information[5]
    session['province'] = user_information[6]


#获取注册页面
@app.route('/register', methods=['GET'])
def register_form():
    return render_template('register.html')

#进行注册
@app.route('/register', methods=['POST'])
def register():
    select = request.form['select']
    print(select)
    print(request.form)
    if select == 'back':
        return render_template('signin.html')
    elif select == 'register':
        username = request.form['username']
        password = request.form['password']
        re_password = request.form['re_password']
        phone_number = request.form['phone_number']
        sex = request.form['sex']
        birthday = request.form['birthday']
```

```
        province = request.form['province']
        ……
        ……
```

代码修改后：

```
# -*- coding: utf-8 -*-
"""
This file controls the functions, deal with the requests from webs.

Created on Sun Apr  8 20:25:02 2018

@author: Alice_shu
"""
# -*- coding: utf-8 -*-

#import flask 框架代码，  数据库函数
from flask import Flask, flash, render_template, request, redirect, session, url_for
import database as db
#获取程序实例
app = Flask(__name__)



@app.route('/signin', methods=['GET'])
def signin_form():
    '''显示登录页面'''
    return render_template('signin.html')

@app.route('/signin', methods=['POST'])
def signin():
    '''检测用户填写信息，进行验证登录
        select：用户选择'登录'或注册'
        username：用户名
        password：密码
    '''
    select =  request.form['select']
    if select == 'register':
        #进入注册页面
        return redirect(url_for('register_form'))
    elif select == 'signin':
        error = None
        username = request.form['username']
        password = request.form['password']
        #对用户名密码进行验证 1：用户名不存在 2：密码错误 0：登录成功
        login = db.db_read_login(username,password)
```

```
            if login == 1:
                error = "用户名错误"#'Invalid username'
            elif login == 2:
                error = "密码错误"#'Invalid password'
            else:
                #从数据库读取用户信息
                session['username'] = username
                fresh_account(username)
                #print(session)
                #记录用户登录操作
                db.db_record_operation(username,'登陆',0)
                #跳转至用户主界面
                return redirect(url_for('user_form_main',username=username))
            return render_template('signin.html', message=error)


def fresh_account(username):
    '''从数据库更新用户信息，在每一步用户信息可能改变的操作前更新'''
    user_information = db.db_read_information(username)
    session['logged_in'] = True
    session['sex'] = user_information[1]
    session['phone_number'] = user_information[2]
    session['birthday'] = user_information[3]
    session['money'] = user_information[4]
    session['credit'] = user_information[5]
    session['province'] = user_information[6]



@app.route('/register', methods=['GET'])
def register_form():
    '''进入注册页面'''
    return render_template('register.html')

@app.route('/register', methods=['POST'])
def register():
    '''进行注册，检测用户填写信息，在数据库中生成新用户
        selete：注册 register or  取消 back
        username：用户名              sex：性别
        password：密码                birthday：生日
        re_password：确认密码     province：省份
        phone_number：电话号码
    '''
    #print(request.form)
```

```
select = request.form['select']
if select == 'back':
    #返回登录页面
    return render_template('signin.html')
elif select == 'register':
    username = request.form['username']
    password = request.form['password']
    re_password = request.form['re_password']
    phone_number = request.form['phone_number']
    sex = request.form['sex']
    birthday = request.form['birthday']
    province = request.form['province']
    #print(username,password,re_password,phone_number)
    #对输入进行检测，返回错误原因
    error = None
    if username == "":
        error = '请输入用户名'#'please input username'
    ..................
    ..................
```