1、系统概述

1.1、系统简介

简要概述系统的基本情况和背景，还包括概要设计中术语的介绍等。

基于小程序实现人民币同其他货币之间的汇率换算及 15 天的汇率变化曲线。

1.2、系统目标

概述概要设计要实现的目标，包括功能目标、性能目标等。

功能目标：

1. 实现用户对当日外汇汇率的实时查看。
2. 为用户提供本币到外币的金额实时换算。
3. 在 GUI 上完成目标 1、2。
4. 进阶目标：绘制某外汇汇率某一时期内的变化曲线。
5. 进阶目标：对各换汇平台汇率进行对比，为用户提供最经济的方案。
6. 进阶目标：在小程序上实现上述功能。

性能目标：

1. 响应时间：在用户操作后响应时间不超过 300ms。
2. 系统数据量：十五日以内的各平台的汇率数据。

1.3、系统运行环境

包括对硬件平台、操作系统、数据库系统、编程平台、网络协议等的描述。

软件环境

分类	名称	版本	语种
操作系统	Windows 10	家庭中文版	简体中文
操作系统	iOS/Android	/	简体中文
应用平台	微信小程序	/	简体中文
数据库平台	SQlite	3.28.0	英文
运行环境	Python	3.7 以上	Python

硬件环境

设备名称	设备要求
Andriod 手机	安装微信
iPhone	安装微信
PC	装有 python3.7 以上版本及 SQlite 数据库

1.4、开发环境

列举进行系统分析、程序设计和程序开发时要使用的工程工具和开发语言。应描述每一工具软件的名称、版本等。

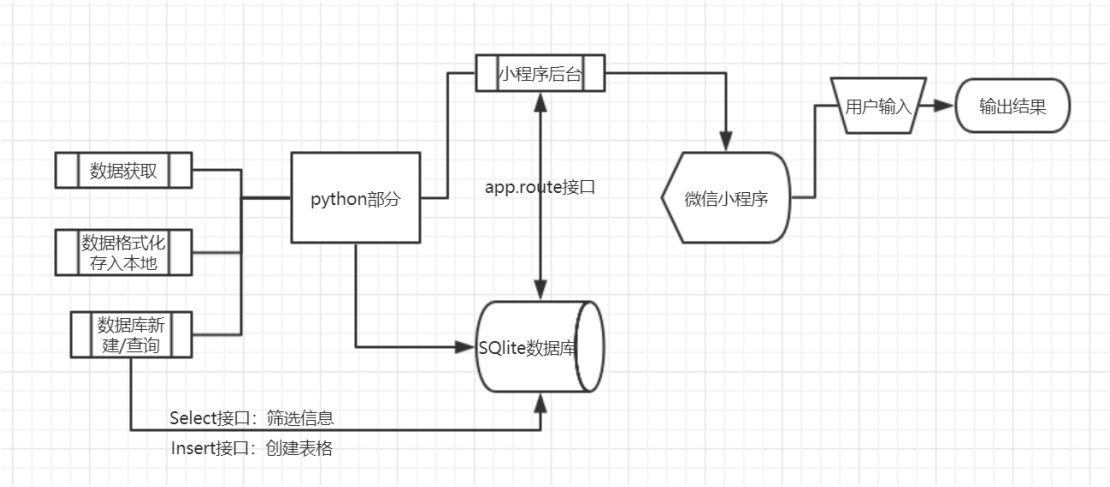
分类	名称	版本	语种
操作系统	Windows 10	家庭中文版	简体中文
开发平台	PyCharm	COMMUNITY 2019.1.1	英文/python
开发平台	Anaconda（Spyder）	1.9.2	英文/python

开发平台	微信开发者工具	1.02.1803210	中文/JavaScript/WXML/WXSS
数据库平台	SQlite	3.28.0	英文

2、总体结构设计

2.1、软件结构

按照不同功能进行整体结构层次的划分，并使各层功能相对独立。以图形方式给出软件系统的子系统（或软件包）划分，模块划分，子系统间、模块间关系等，并用接口来描述各模块之间的调用关系，给出各模块之间的松散耦合关系。



2.2、设计思想

结合上图阐述软件的基本设计思想和理念。

将整个系统划分为三个部分，python 程序，微信小程序和 sqlite 数据库。这种划分主要是由每个部分对系统的分工来划分的，后台部分使用 python，用户端用小程序的形式，此系统数据量并不大，使用 SQlite 这种轻量数据库比较方便。各个部分下再继续划分模块，是为了系统的可扩展性和复用性。为后续系统功能的增加留下充足条件。

3、模块设计

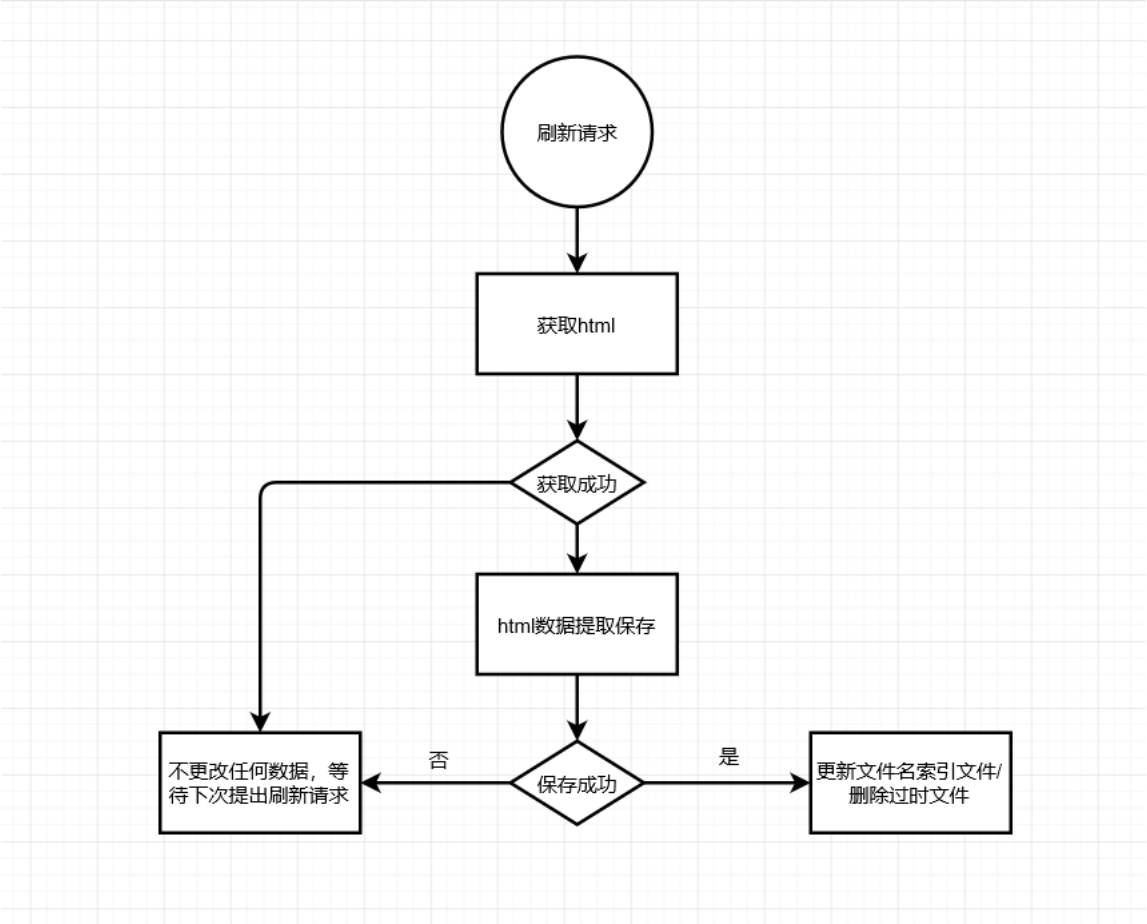
Python 部分

在此以列表形式说明各模块的名称、功能、接口等信息。

3.1、模块 1——数据获取

详细描述各功能模块的功能、接口及内部实现。

3.1.1、功能描述



有条理地叙述该模块功能，也可画功能流程图。

3.1.2、接口描述

描述与其他功能模块的每个接口的名称、输入信息、输出信息、异常处理等。

接口名称	输入信息	输出信息	异常处理
Path	None	Path	Try 新数据且可存储 Except 使用老数据
List	Path	List(New)	Try 删除原数据并更新目录 Except 报错使用老数据

3.1.3、数据结构描述

每天数据的存储可认为使用了堆栈的结构每日数据被压栈，指针在某一日内始终指向某一存储空间，当日每次存取数据都进行 pop 原数据入栈新数据操作。

20170429
20170428
20190427

名称	数据结构	元素类型	功能
list	二维数组	Tag	存储网页上的标签信息
ilst	一维数组	String	将标签中的内容提取出来
list2	一维数组	Float	存储部分货币的折

			算价并转换成 float 型
data	一维数组	Tuple	存储货币 id 和折算价
infoDict	字典	String	将每种货币及其汇卖价，钞卖价，汇买价，钞买价，折算价以键值对的形式进行存储
SQlite	Database	Table	存储数据到数据库

3.1.4、实现思路

给出实现该模块的基本思路，包括对模块内部结构、算法、编程方法等的初步设想。

- 内部结构：

try 获取

try 保存 flag=1 以时间为文件名保存当前时刻的数据

except 报错，本次数据获取进程终止使用老数据，flag==0

except

if flag==1

try 删除原数据并更新目录

except 报错使用老数据

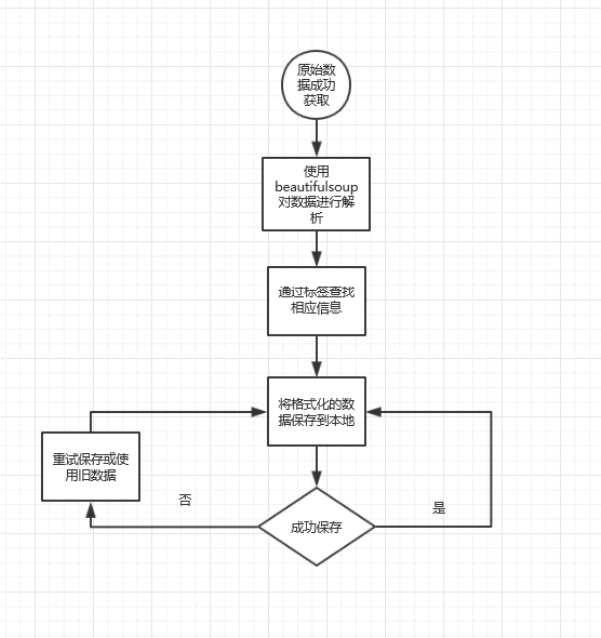
3.1.5、其他

选写另外一些有必要叙述的部分，如模块的表现形式（以何种形式运行，如服务程序、应用程序、数据库脚本等等）、有无人机交互界面（有则给出其类型和设计原则）等。

是数据库数据的获取模块

3.2、模块 2 ——将数据进行格式化处理，存入表格

3.2.1、功能描述



3.2.2、接口描述

接口名称	输入信息	输出信息	异常处理
GetHTML	url	HTML	中止连接并返回错误
FindFilepath	文件路径	打开文件	如果打不开用旧数据
FindData	获取所有数据		同上
DataStuct	将获取的所有数据	结构化提取所需的数据	同上
SelectToSQ	在 data 数组存储的货币信息	SQlite 中的新建 Table	使用原本存储的 table

3.2.3、数据结构描述

每日数据：循环队列，长度为 15，每日在队尾更新一次，获得新数据并删除旧数据

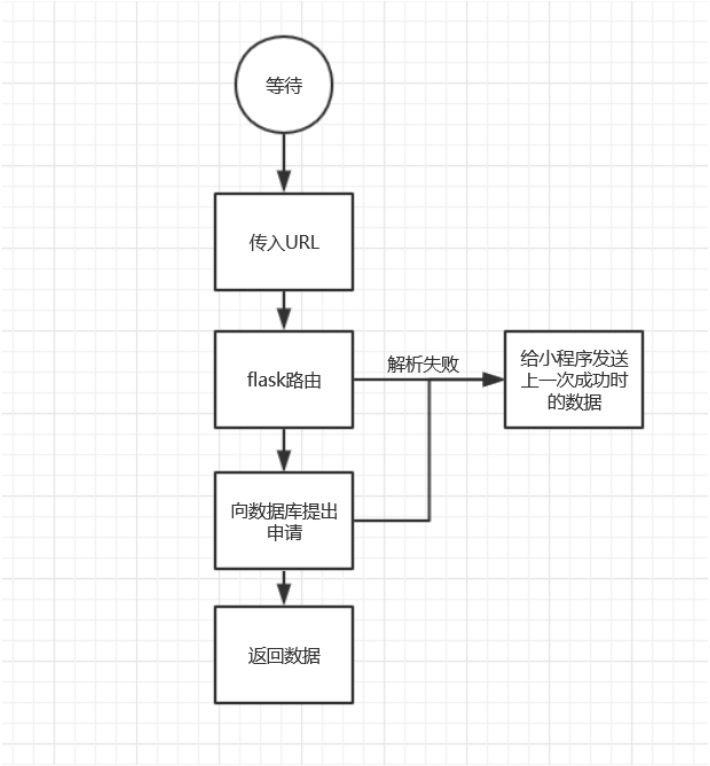
当日数据：以结构体方式存储。结构为：货币名称，汇买价，钞买价，汇卖价，钞卖价，折算价

3.2.4、实现思路

获取原始数据成功后，由于 python 有 BeautifulSoup 可以对获取的 html 网页进行依据标签的数据处理。通过生成 dom 树等方法，获得所需层次标签中的数据，并将数据进行分割，存入结构数组中，最终获得结构一致的一组数据，将数据存储在硬盘中，以备使用

3.3、模块 3——python 编写小程序后台

3.3.1、功能描述



3.3.2、接口描述

接口名称	输入信息	输出信息	异常处理
app.route	url	字符串	http://flask.pocoo.org/docs/1.0/

3.3.3、数据结构描述

Json 格式，直接用数组保存

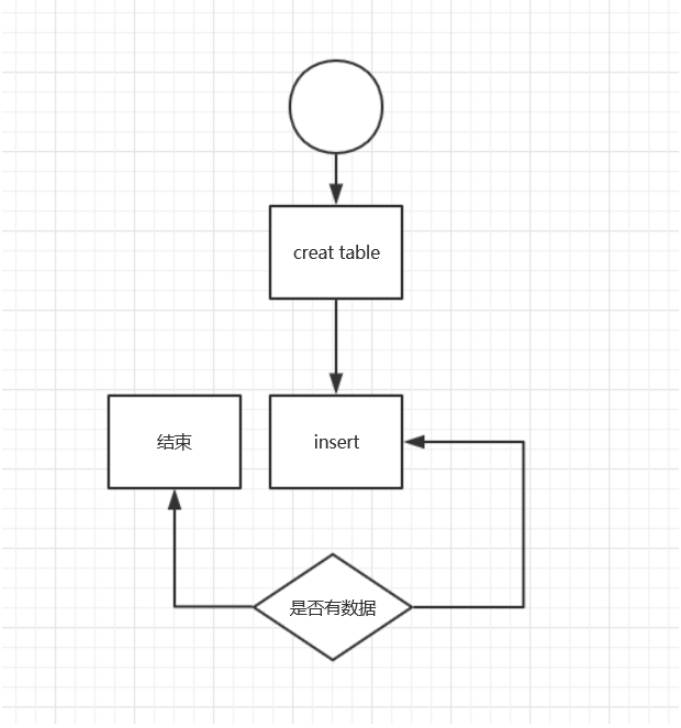
3.3.4、实现思路

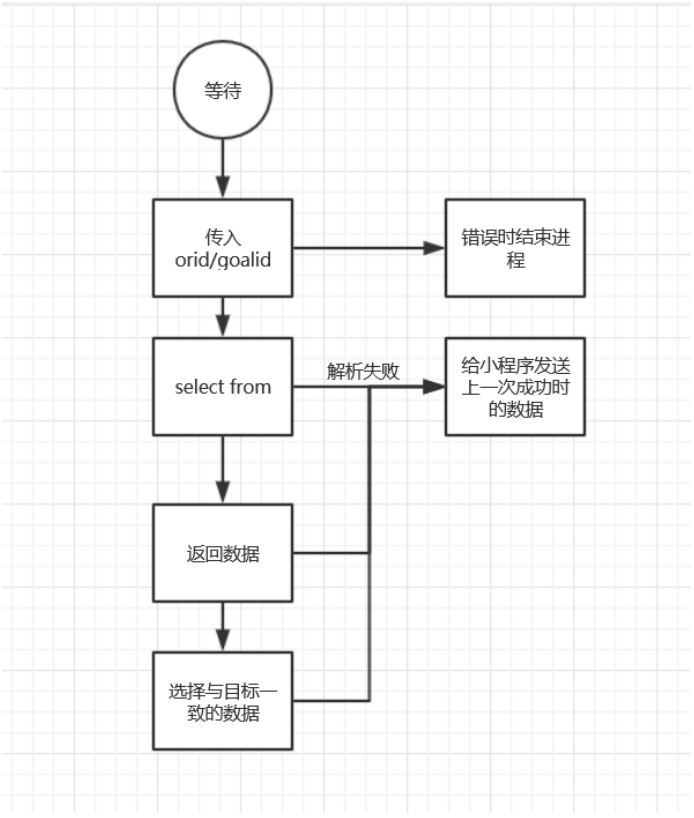
微信小程序生成 URL:'localhost/端口号/原货币 id=a&目标货币 id=b'

将 URL 发送给 python，Python 通过 flask 框架中的 app.route()函数将该 url 转换为可进行操作的数据发送给数据库进行操作。

3.3、模块 4——数据库新建/查询

3.3.1、功能描述





3.3.2、接口描述

接口名称	输入信息	输出信息	异常处理
Select	Id/汇率	对应信息	返回历史值
Insert	表格	成功创建	返回历史有效值
Dic_to_json	字典	Json	返回历史值

3.3.3、数据结构描述

Id	Value

3.3.4、实现思路

输入：建立表格后重复 insert 语句

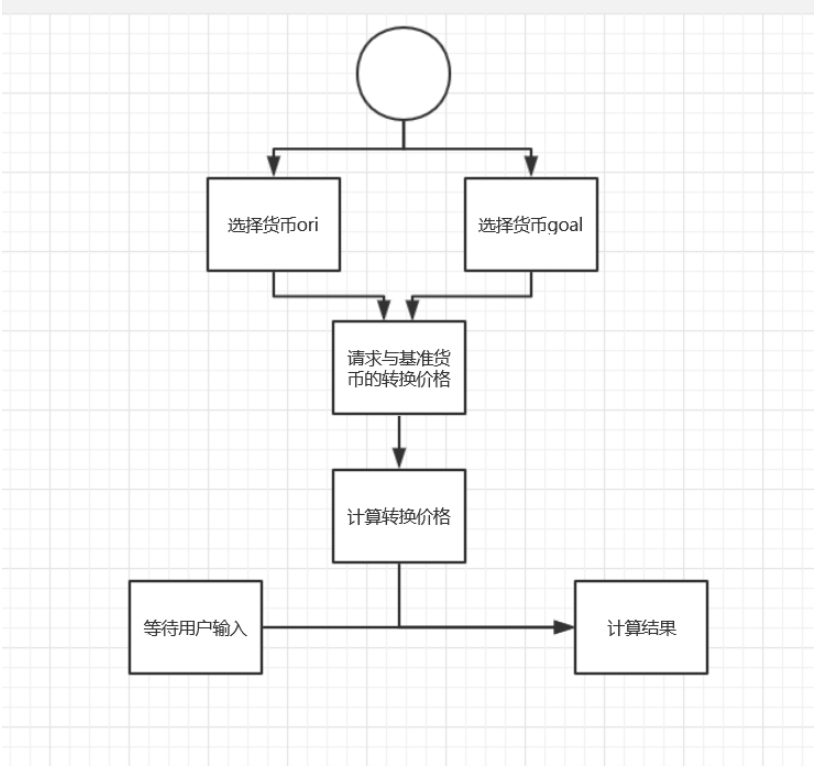
查询：查询全部数据，

将 id 值与解析所得 id 作比较，将改 id 下的数据存入对应的 money 项

微信小程序部分

3.4、模块 4 ——微信小程序

3.4.1、功能描述



3.4.2、接口描述

Work():function{}

3.4.3、数据结构描述

json 对象：

{key：value}

3.4.4、实现思路

选择货币类型：使用 multipicker

显示当前货币之间的转换金额
输入待计算的原货币金额
提交进行计算

4、数据库与数据结构设计

4.1、 数据库及数据表

4.2、 数据结构设计

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。
文件地址：堆栈，高度为 1，每次使用时均 pop 过时地址
每日数据：循环队列，长度为 15，每日在队尾更新一次，获得新数据并删除旧数据
当日数据：以结构体方式存储。结构为：货币名称，汇买价，钞买价，汇卖价，钞卖价，折算价

4.3、 数据存储设计

访问方法：程序通过寻找地址直接取用
每次取用整组数据，对整组数据进行搜索，最终将匹配的到的数据在程序中保存或处理。
其余数据不再参与程序运行的后续过程。

绝大部分数据存储在电脑硬盘中，由程序按期删除
由于是公众信息，不涉及保密问题

5、接口设计

4.1、外部接口

描述需要访问的外部接口的类型、控制方式。
地址接口

```
GetURL
{
    A=GetURL();
}

CREATEURL
{
    B=CreateURL();
```

}

4.2、内部接口

描述与其他模块及子系统的接口。

数据存储

{

CreateFilename

SaveData

SaveFilepath

}

数据提取

{

FindFilepath

FindData

DataStuct

}

数据修改

{

Uploaddata

Changepath

}

6、其他设计

内存管理：每次均覆盖程序上次运行获取的数据。在遇到各类存取障碍时均使用近期数据平均值/某一较近历史时期平均值进行数据处理。

存储单元：仅存储过去半个月内的数据，对超期数据作删除处理。
使用了 sqlite3 进行数据的管理与存储，使得整个数据库轻量且稳定。

系统稳定性：使用 flask 框架，进行 web 开发，可以保证应用的稳定性