**Python基础**

Python 优点

语法简单

跨平台

可扩展

开放源码

累库丰富

python 用途

python 脚本

python 爬虫

python 数据分析

python GUI编程

python 人工智能

Python 历史

1990 第一个python版本

2000 python2版本

2012 2.7.2..X版本

Python解析器

cPython

JPython

iPython

Pypy

Python 数据类型

整型 浮点型 字符型 布尔类型

检测数据类型 type()

显式类型转换

序列：成员都是有序的排列，并且可通过下标偏移量访问到的数据类型

字符串：字符串也是一种数据类型，编码问题。

列表：list是一种有序的集合，可以随时添加和删除其中的元素。

元组：元组tuple 元素不可改变 因此项目中 尽量使用tuple。

字典：使用键-值（key-value）存储，具有极快的查找速度。

集合：集合和字典类似，也是一组key的集合，但不存储value。

Python 变量

1 变量名只能是字母、数字、下划线

2 变量名以数字开头，不轻易以"\_"开头

3 "\_"代表一些特殊变量

4 变量名要有意义，严格区分大小写

5 变量命名方式：驼峰命名法 - reactWidth、ReactWidth；

下划线命名 react\_width

Python 运算符

算术运算符：+ - \* / % , \*\*(幂运算) //(向下取整)

比较运算符：== != >= <= > <

逻辑运算符：and or not (一边为假即为假 两边为真才为真)

赋值运算符：+= -+ /= \*/ %/ \*\*= //=

成员运算符：in, not in (测试某个元素是否在列表或元素中)

三目运算符：ret = 'true' if 1==1 else 'false'

Python 条件判断

if <条件判断1>:

<执行1>

elif <条件判断2>:

<执行2>

elif <条件判断3>:

<执行3>

else:

<执行4>

Python 循环

循环语句：for while

跳出/退出循环：continue break

Python文件内建函数

open('文件路径','打开方式','编码格式')

read() 读取

readline() 读取一行

seek() 文件内移动

write() 写入

close() 关闭文件

tell() 光标此时处于什么位置

Python 变量与参数

可变长参数（\*）：def func(first,\*other)

全局变量（global）

Python 迭代器

可以直接作用于for循环的对象统称为可迭代对象。

Python生成器

使用了 yield 的函数被称为生成器，每次遇到 yield 时函数会暂停并保存当

前所有的运行信息，返回 yield 的值,并在下一次执行 next() 方法时从当前

位置继续运行。

Python 模块

模块就是代码量变得相当大后，将需要重复使用的代码有组织的放在一起。

这部分代码段可以附加到现有程序中。

附加进来的过程叫 导入import。

1 导入标准模块

2 导入自定义模块

3 第三方模块

pip3 install module

pip3 uninstall module

pip3 list 查看已安装模块

Python 文件操作

本目录：os.path.abspath(".")

当前目录的上级目录：os.path.abspath("..")

检测目录存不存在：os.path.exists("/XX/XX")

判断是不是文件：os.path.isfile("/XX")

判断是不是目录：os.path.isdir("/XX")

返回指定文件夹包含的文件或文件夹：os.listdir(".")

路径组合：os.path.join("/XX/XX","YY/YY ")

模糊搜索：

for line in files:

if (not line.endswith(" ")) and (" " in line) :

…

如何按照文件类型归类到各自的文件夹中：

步骤1 怎么移动文件？

os模块 筛选出移动的文件

shutil模块 对文件进行移动 打包 处理 压缩 解压缩 针对文件的增删改查

shutil.move完成移动文件

步骤2 归类的规则是什么？

自动创建以文件后缀名为文件名的文件夹 将符合的文件归类于此

如何创建文件夹：

os.makedirs(path+folder\_name)

如何浏览各个子文件中的文件：

浏览多个文件夹，可以先将这些文件夹名字放到一个list里

如何移动文件夹中的文件：

shutil.move(file,path)

如何删除子文件夹：

os.remove()

如何压缩文件：

shutil.make\_archive(zip\_name,"zip",img\_path)

如何解压文件：

shutil.unpack\_archive(file,target\_path)

文件解压缩img.zip到img文件夹中，执行解压缩命令需等2s，

解压完成后删除压缩包，步骤如下：

1 找到后缀名为.zip的文件

2 找到.zip文件后对其解压缩

3 删除压缩包

4 多个.zip文件循环执行

**wxpy（微信个人号 API）**

wxpy 在 itchat 的基础上，通过大量接口优化提升了模块的易用性，并进行

丰富的功能扩展。

通过机器人对象 Bot 的 chats(), friends()，groups(), mps() 方法,

可分别获取到当前机器人的 所有聊天对象、好友、群聊，以及公众号列表。

**Python爬虫**

BeautifulSoup

BeautifulSoup 是一个可以从HTML或XML文件中提取数据的Python库 。

第一步 BeautifulSoup。

Soup = BeautifulSoup(html,parse)

第一个参数是网页内容，第二个参数是解析器

解析器有5种：html5.parse() BeautifulSoup自带的解析器 不需安装

html5lib 需要安装 pip3 install html5lib

lxml 也需要安装，pip3 install lxml 解析速度比较快

第二步 描述要爬取的元素在哪里 并解析成你想要的格式。

Soup.select()

第三步 从标签中得到你想要的信息。

css selector body > div.content > div:nth-child(1) > img

xpath /html/body/div[3]/div[1]/img

请求响应模式

浏览器请求数据的过程：请求(request)-响应(response)模式

浏览器发送 http 或 https 请求给服务器

服务器接收到请求，查找有没有请求的资源，若有返回200

请求真实服务器的网站 需要模块request -- pip3 install requests

爬虫所需安装库

beautifulsoup4: 可以从HTML或XML文件中提取数据的Python库

通过解析文档为用户提供需要抓取的数据

requests: requests是Python实现的最简单易用的HTTP客户端库

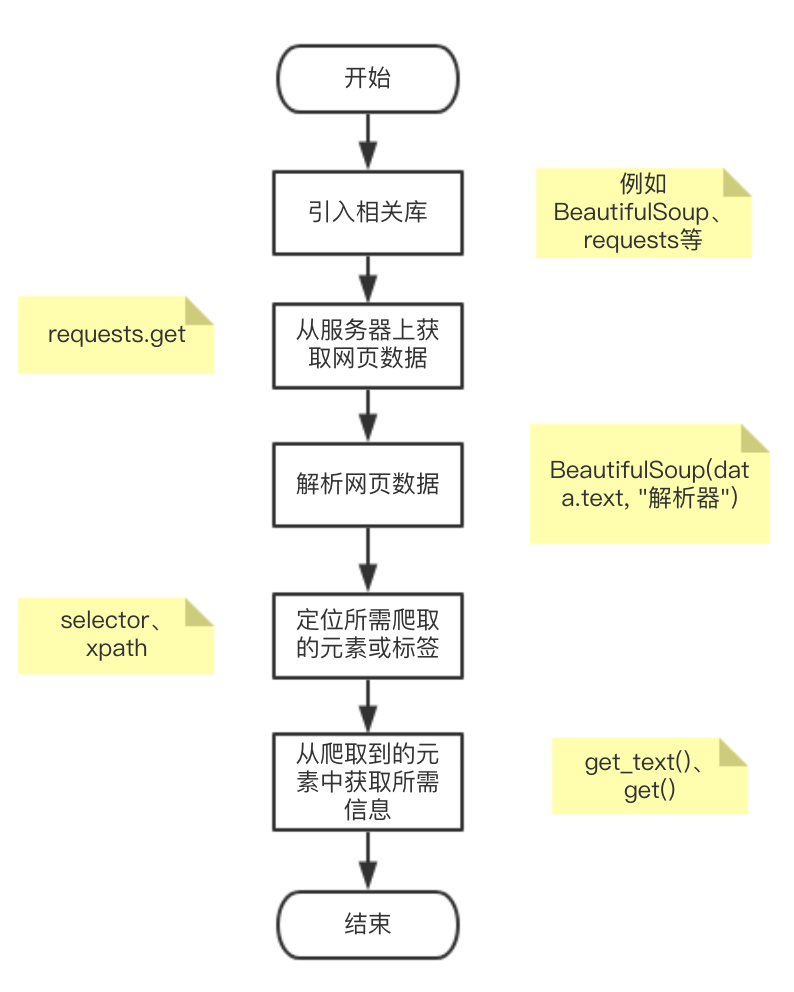
基于urllib编写

html5lib: 是Python的一个解析库 可以像Web浏览器一样解析html页面

lxml: 是Python的一个解析库 支持HTML和XML的解析

支持XPath解析方式

数据爬取流程



**MongoDB**

1 MongoDB简介

MongoDB是一个基于分布式文件存储的数据库。将数据存储为一个文档，

数据结构由键值(key=>value)对组成。类似于JSON对象。

2 MongoDB安装与配置

3 MongoDB基本增删改查

首先 启动服务器端: sudo mongod

然后 启动应用端: sudo mongo

show dbs:显示数据库列表

show collections：显示当前数据库中的集合

use db\_name：切换当前数据库

db.x.find()：对当前数据库中的x集合进行查找 若无条件则列出所有

db.createCollection("x"):创建名为x的集合

db.dropDatabase():删除当前使用数据库 (首先use database)

db.collection.drop():删除当前聚集集合

cd /usr/local/mongodb/bin

mongoexport -d database -c collection -o xx.json 导出成json文件

mongoexport -d database -c collection --type=csv -f key1,key2 -o xx.csv 导出成csv文件

mongoimport -d database -c collection --file xx.json 导入json文件

mongoimport -d database -c collection --type=csv --file xx.csv

导入csv文件

4 Python操作MongoDB

import pymongo

from pymongo import MongoClient

host = "localhost"

port = 27017

client = MongoClient(host,port)

test = client["test"]

sheet = test["sheet"]

# 插入数据

for index in range(1001):

print(index)

data = {

"name": "name"+str(index),

"age": index

}

sheet.insert\_one(data)

# 查看数据

for item in sheet.find():

print(item)

**Django Web开发**

1 Django简介与安装

Django是一个开放源代码的Web应用框架，由Python写成。

采用了MVC的框架模式，即模型M，视图V和控制器C。

[Django 准备工作-第二课.pdf](Django%20准备工作-第二课.pdf)

2 Django的MTV模式

MVC model view controller

MTV model template view

用户输入url，经过url.py处理请求转到view层。

去model挂载数据（跟数据库进行交互）。

进入template层，返回模板。

3 用Django开发第一个网站

4 Get与Post

POST和GET是HTTP协议定义的与服务器交互的方法。

GET一般用于获取/查询资源信息，而POST一般用于更新资源信息。

另外，还有PUT和DELETE方法。

POST和GET都可以与服务器完成查，改，增，删操作。

5 服务器对话

python3 manage.py runserver

6 分页器

views.py

from django.core.paginator import Paginator

limit = 10

pagenator = Paginator(room\_info, limit)

index = 1

while True:

index += 1

page = request.GET.get("page",index)

loaded = pagenator.page(page)

context = {

"RoomInfo": loaded

}

return render(request,'index.html',context)

**HTML基础**

1. 基础语法

1.1. html基础课程介绍

1.1.1. HTML概念

1.1.2. HTML发展历史

1.1.3. HTML语言的特点

1.1.4. 开发工具

1.2. html基础语法

1.2.1. html基本结构

1.2.2. html标签

1.2.3. html元素

1.2.4. html属性

1.2.5. html注释

2. 文档段落

2.1. 文档声明和META标签

2.2. 文字和段落

2.3. 修饰标签

2.4. 特殊符号

3. 列表标签

3.1. 无序列表

3.2. 有序列表

3.3. 定义列表

4. 图像

4.1. 标签介绍

4.2. 属性介绍

4.3. 相对路径和绝对路径

5. 超链接

5.1. 超链接引入

5.2. 空连接 target title属性

5.3. 锚链接 同一页面

5.4. 锚链接 不同页面

5.5. 链接功能拓展

6. html表格

6.1. 基础表格

6.2. 表格操作

6.3. 带标题，表头和结构的表格

6.4. 表格属性

6.4.1. width

6.4.2. align 表格相对周围元素的对齐方式

6.4.3. border

6.4.4. bgcolor

6.4.5. cellpadding

6.4.6. cellspacing

6.4.7. frame 规定外侧边框的哪个部分是可见的

6.4.8. rules 规定内侧边框的哪个部分是可见的

6.5. 表格跨行跨列

6.6. 表格嵌套

6.7. 表格布局

7. html表单

7.1. 表单介绍

7.2. 搭建表单页面结构

7.3. input标签（文本，文件，单选，复选，按钮，图像域和隐藏域）

7.4. select下拉菜单和列表，分组下拉菜单和列表

7.5. textarea 多行文本域

7.6. 表单属性

8. 总结

8.1. 行内元素与块级元素

8.2. html标签嵌套规则

9. 表格布局网页实战（综合项目）

**CSS基础**

1. css简介，语法，选择器

1.1. 课程介绍

1.2. css基础语法

1.2.1. css基础语法

1.2.2. css使用方法1

1.2.3. css使用方法2

1.2.4. css使用方法优先级

1.3. css选择器

1.4. css继承，层叠和优先级

1.5. css应用

2. css文本样式

3. css背景和列表样式

4. css盒模型

5. float浮动

6. css定位（position）

7. css网页布局基础

8. 网页布局实战

**JavaScript基础**

1. JavaScript语法

1.1. JavaScript语言简介

1.2. 基础语法

1.3. 数据类型

1.4. 运算符

2. JavaScript流程控制语句

2.1. 条件

2.2. 分支

2.3. 循环

3. JavaScript函数

3.1. 函数介绍

3.2. 封装

4. JavaScript内置对象

4.1. 属性

4.2. 方法

5. JavaScript DOM基础

6. JavaScript DOM事件

6.1. 键盘事件

6.2. 鼠标事件

7. JavaScript BOM

8. 模块化案列专题

8.1. 轮播

8.2. tab切换

8.3. ...

**jQuery基础**

1. jQuery选择的艺术

1.1. jQuery介绍

1.2. jQuery版本选择

1.3. 选择器

1.4. 筛选器

2. jQuery DOM操作

3. jQuery事件

4. jQuery插件

5. jQuery综合案例

5.1. 弹出层

5.2. 瀑布流

5.3. ...

**html4.01+css2.0**

1. html

1.1. 什么是HTML？超文本标记语言

1.2. 标题，段落，图片，超链接，列表

1.2.1. h1-h6

1.2.2. p

1.2.3. img (src)

1.2.4. a href

1.2.5. ol,ul,li

1.3. 表格

1.3.1. table

1.3.2. 行 tr

1.3.3. 单元格 td，标题单元格th

1.4. 表单

1.4.1. file

1.4.2. text

1.4.3. submit

1.4.4. checkbox

1.4.5. radio

1.4.6. password

1.4.7. reset

1.4.8. email

1.4.9. tel

1.4.10. color

1.4.11. date

1.4.12. ...

2. css

2.1. 选择器

2.1.1. 标签选择器

2.1.2. class 类选择器

2.1.3. ID选择器 不重复

2.1.4. 后代选择器div p

2.1.5. 子选择器 div>p

2.1.6. :active,伪类选择器

2.1.7. \* 通配符选择器

2.2. 各类样式

2.2.1. 文字样式

2.2.1.1. color 文字颜色

2.2.1.2. font-size 文字大小

2.2.1.3. font-weight 文字加粗（100-900，bold bolder）

2.2.1.4. font-family 文字是哪种（微软雅黑）

2.2.1.5. font-style:italic，normal

2.2.1.6. text-decoration: none,underline overline

2.2.1.7. line-height: 行高

2.2.2. 背景

2.2.2.1. background-color：背景颜色

2.2.2.2. background-image：背景图片

2.2.2.3. background-repeat：no-repeat repeat-x,repeat-y

2.2.2.4. background-position:背景图片的位置（雪碧图）

2.2.3. 列表

2.2.3.1. list-style

2.3. 布局方式

2.3.1. 盒模型（标准盒模型）

2.3.1.1. width，height 内容宽高

2.3.1.2. 外边距：margin

2.3.1.3. 内边距：padding

实际占位=width+margin+padding+border

2.3.1.4. border:1px solid black

border-width

border-style

border-color

2.3.2. float 浮动

2.3.2.1. left

2.3.2.2. right

2.3.2.3. clear

2.3.3. position 定位

2.3.3.1. static

2.3.3.2. relative

2.3.3.3. absolute

2.3.3.4. fixed

**AJAX Node.js JSON**

1 AJAX

Ajax：Asynchronous(异步) JavaScript and XML(数据格式)

另一种数据格式JSON（JavaScript对象表示法）

实现AJAX的步骤：

第一步，创建XMLHTTP对象（IE7+）

第二步，调用XMLHTTP request 对象的Open方法

第三步，监听readystatechange事件

第四步，向服务器发送请求

2 Node.js

HTTP是Node.js的网络请求模块

V8y引擎迁移到了服务器端，这个服务器端的V8就叫Node.js

HTTP是Node.js的网络请求模块

Node.js中所有的模块都是用require引入的

3 JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

{

"employees": [

{ "firstName":"Bill" , "lastName":"Gates" },

{ "firstName":"George" , "lastName":"Bush" },

{ "firstName":"Thomas" , "lastName":"Carter" }

]

}