第二天课知识点总结

一. macdown入门介绍：

1. Markdown是什么？

Markdown 是一种轻量级的「标记语言」，它的优点很多，目前也被越来越多的写作爱好者，撰稿者广泛使用。看到这里请不要被「标记」、「语言」所迷惑，Markdown 的语法十分简单。常用的标记符号也不超过十个，这种相对于更为复杂的 HTML 标记语言来说，Markdown 可谓是十分轻量的，学习成本也不需要太多，且一旦熟悉这种语法规则，会有一劳永逸的效果。

2.Markdown的用途？

Markdown的语法简洁明了、学习容易，而且功能比纯文本更强，因此有很多人用它写博客。世界上最流行的博客平台WordPress和大型CMS如Joomla、Drupal都能很好的支持Markdown。完全采用Markdown编辑器的博客平台有Ghost和Typecho。用于编写说明文档，并且以“README.MD”的文件名保存在软件的目录下面。

3. 3.Markdown语法介绍:

常用语法

最常见的Markdown格式选项和键盘快捷键[3]  :

|  |  |  |
| --- | --- | --- |
| **输出后的效果** | **Markdown** | **快捷键** |
| Bold | \*\*text\*\* | Ctrl/⌘ + B |
| *Emphasize* | \*text\* | Ctrl/⌘ + I |
| Strike-through | ~~text~~ | Ctrl + Alt + U |
| Link | [title](http://) | Ctrl/⌘ + K |
| Inline Code | `code` | Ctrl/⌘ + Shift + K |
| Image | ![alt](http://) | Ctrl/⌘ + Shift + I |
| List | \* item | Ctrl + L |
| Blockquote | > quote | Ctrl + Q |
| H1 | # Heading |  |
| H2 | ## Heading | Ctrl/⌘ + H |
| H3 | ### Heading | Ctrl/⌘ + H (x2) |

标题

标题能显示出文章的结构。行首插入1-6个 # ，每增加一个 # 表示更深入层次的内容，对应到标题的深度由 1-6 阶。

H1 ##Header

H2 :##Header 2

H3 :###Header 3

H4 :#### Header 4

H5 :##### Header 5

H6 :###### Header 6

文本样式

（带“\*”星号的文本样式，在原版Markdown标准中不存在，但在其大部分衍生标准中被添加）

链接 :[Title](URL)

加粗 :\*\*Bold\*\*

斜体字 :\*Italics\*

\*删除线 :~~text~~

\*高亮 :==text==

段落 : 段落之间空一行

换行符 : 一行结束时输入两个空格

列表 :\* 添加星号成为一个新的列表项。

引用 :> 引用内容

内嵌代码 : `alert('Hello World');`

画水平线 (HR) :--------

图片

使用Markdown将图像插入文章，你需要在Markdown编辑器输入 ![]() 。 这时在预览面板中会自动创建一个图像上传框。你可以从电脑桌面拖放图片(.png, .gif, .jpg)到上传框, 或者点击图片上传框使用标准的图像上传方式。 如果你想通过链接插入网络上已经存在的图片，只要单击图片上传框的左下角的“链接”图标，这时就会呈现图像URL的输入框。想给图片添加一个标题, 你需要做的是将标题文本插图中的方括号，e.g;![This is a title]().

二、Trello

Trello是一个伟大的WebApp，可以给您一个数位工作空间就像使用白板和便利贴一样方便，此外功能齐全到使团队能够把想法实现到真正的项目。这里将帮助您上手。Trello，以及如何用到工作团队。使用介绍请参见文档：http://www.jianshu.com/p/fbf015f4f3f3

同一类型的团队协调工具还有tower，链接：https://tower.im/

使用介绍请参见文档：

<http://www.jianshu.com/p/fbf015f4f3f3>

画图工具：processon 免费在线作图，实时协作 链接：https://www.processon.com/

上边是三款工具的简单介绍，下边是python相关的知识点：

python数据类型

python中有五种内建的数据结构：

列表、元祖、字典、集合、字符串

其中最常用的就是列表、字典和字符串

1.列表介绍：

像字符串类型一样，列表类型也是序列式的数据类型，可以通过下标或者切片操作来访问某一个

或者一块连续的元素。字符串只能由字符组成，而且是不可变的，而列表则是能保留任意数目的

python对象的灵活的容器。

列表不仅可以包括python的标准类型，而且可以用用户定义的对象作为自己的元素。列表可以

包含不同类型的对象，而且要比C或者Python自己的数组类型都要灵活，因为数组类型所有的

元素只能是一种类型。列表可以执行pop,empt,sort,reverse等操作。列表可以添加或者删除元素，还可以跟其他的列表结合或者把一个列表分成几个。可以对单独一个元素或者多个元素执行insert、update或remove等。

序列是Python中最基本的数据结构。序列中的每个元素都分配一个数字-它的位置或者索引，第一个索引是0，第二个索引是1，以此类推。

1)列表的增删改查 (几种存储方式总结)

###数据存储

###变量

name = “taoyake”

###列表存储

user = [‘taoyake’,’26’]

###字典存储

user = {‘name’:’taokey’,’age’:18}

###文件存储

cat user.txt

panda

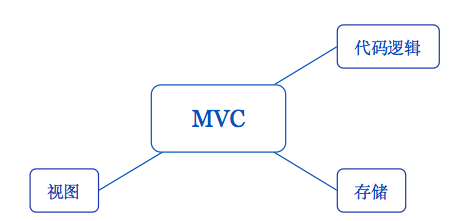
kk

###数据库存储

mysql

重点：任何一个存储，都是增删改查

mvc框架简单介绍:



个人简单理解: MVC就是通过代码逻辑例如流程控制语句(if elif else while for)等，把存储在数据库(list ,dict,mysql)中的数据，用页面展示出来。

1. 定义一个列表：

>>> List = ['abc','abd','aa']

>>> List

['abc', 'abd', 'aa']

>>> type(List)

<type 'list'> #查看List变量的类型

注意：list是python内置的函数，最好不要用list做变量，可以用大小写区分开，最好设置的变量要有意义。

help(list) #列表的帮助，列出所有列表的用法

type(name) #查看变量的类型是列表、字典、元组还是字符串

增加列表元素 append和insert：

>>> List = ['taoyake','26']

>>> List.append('IT')

>>> List

['taoyake', '26', 'IT']

>>> List.insert(2,'Beijing')

>>> List

['taoyake', '26', 'Beijing', 'IT'] #列表中的元素从0开始计算

apped和insert的区别：append直接在列表末尾追加，insert指定位置插入列表中，列表和字典的区别：字典是用哈希的形式用key value表示 字典是无序的，列表是有序

set集合，可以去重复：

>>> List

['taoyake', '26', 'Beijing', 'IT']

>>> List2 = ['taoyake','IT']

>>> set(List+List2)

set(['Beijing', '26', 'taoyake', 'IT'])

len(List) #查，查看列表的元素个数

>>> List = ['20','18','15','50','89']

>>> len(List)

5

max(List) #查，列表中的最大值

>>> List = ['20','18','15','50','89']

>>> max(List)

'89'

min(List) #查，列表中的最小值

>>> List = ['20','18','15','50','89']

>>> min(List)

'15'

List.count(‘apple’) #查，统计列表中某元素的个数

If apple in List #查，搜索apple是否在列表中

List.index(“apple”) #查，输出apple元素在列表中的索引位置

分片

List[0] #分片查询，输出索引为0的元素

List[0:2] #分片查询，输出前三个元素(注意不包括第三个元素)

>>> List = ['20','18','15','50','89']

>>> List[0:2]

['20', '18']

List[:] #分片查询，查处所有的元素

List[1:] #分片查询，输出第二个和后边所有的元素

List[0:-1] #分片查询，除了最后一个都输出

例子：

>>> List = ['20','18','15','50','89']

>>> List[0:-1]

['20', '18', '15', '50']

List.sort() #查询，将列表按首字母排序，数字>大写>小写

List.reverse() #查询，列表顺序倒转

循环打印所有列表中的元素：

>>> List = ['20','18','15','50','89']

>>> for i in List:

... print i

...

20

18

15

50

89

小练习：列表元素去重

file\_list = [1,2,3,3,4,5,5,2,8,8]

new\_list = []

for i in file\_list:

if not i in new\_list:

new\_list.append(i)

print new\_list

list列表切片练习：

cat 06.py

log = '124.238.248.52 - - [30/Jul/2017:15:22:42 +0800] "POST /crontab/collect HTTP/1.1" 200 151 "-" "Python-urllib/2.6" "-" "0.166" "10.3.0.136:5000" "200" "0.106"'

new\_log = []

a = [0,3,6,8]

for i in a:

new\_log.append(log.split(" ")[i])

print new\_log

L = ','.join(new\_log) #”分隔符”.join (列表)—把列表转换成字符串

print L

python 06.py

['124.238.248.52', '[30/Jul/2017:15:22:42', '/crontab/collect', '200']

124.238.248.52,[30/Jul/2017:15:22:42,/crontab/collect,200

#字符串.split(‘分隔符’)－把字符串变成列表

获取列表下标：

log = ['124.238.248.52','[30/Jul/2017:15:22:42 +0800]','POST /crontab/collect HTTP/1.1']

for i,element in enumerate(log):

... print "%s --> %s" %(i,element)

...

0 --> 124.238.248.52

1 --> [30/Jul/2017:15:22:42 +0800]

2 --> POST /crontab/collect HTTP/1.1

python数据类型中，数字0，空字符串，None,空字典，空列表，空元素的布尔值都是false,常用于if判断

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

if "":

print "abd"

else:

print "1234"

List = []

if List:

print "真的"

else:

print "假的"

Dict = {}

if Dict:

print "真的"

else:

print "假的"

**python数据类型—字典：**

字典是python语言中唯一的映射类型。映射类型对象里哈希值(键,key)和指向的对象(值，value)是一对多的关系。通常被认为是可变的哈希表。一个字典对象是可变的，它是一个容器类型，能存储任意个数的python对象，其中也包括其他容器类型

字典类型和序列类型容器类(列表、元组)的区别  
是存储和访问数据的方式不同。序列类型只用数字类型的键(从序列的开始起按数值顺序索引)。映射类型可以用其他对象类型做键，一般最常见的是用字符串做键。和序列类型的键不同，影像类型的键直接或间接地和存储的数据值相关联。但因为在映射类型中，我们不再用“序列化排序”的键，所以映像类型  
中的数据是无序排列的。

注意：字典内部可以套用列表，字典内部可以是字典，添加元素字典的存储不分顺序

定义一个字典：

>>> Dict = {'name':'taoyak','age':'21','job':'sa'}

>>> Dict

{'job': 'sa', 'age': '21', 'name': 'taoyak'}

获取字典所有的key

>>> Dict.keys()

['job', 'age', 'name']

根据key取值：

>>> Dict['name']

'taoyak'

获取所有的value值

>>> Dict.values()

['sa', '21', 'taoyak']

通过字典的key修改values值

>>> Dict['age'] = 26

>>> Dict['age']

26

字典中可以存储列表，例如：

>>> Dict['score'] = [78,85,83]

>>> Dict

{'job': 'sa', 'age': 26, 'score': [78, 85, 83], 'name': 'taoyak'}

字典中可以存储字典，例如：

>>> Dict['gf'] = {'name':'coco','age':25,'job':'model'}

>>> Dict

{'job': 'sa', 'gf': {'job': 'model', 'age': 25, 'name': 'coco'}, 'age': 26, 'score': [78, 85, 83], 'name': 'taoyak'}

>>> Dict['gf']['job']

'model'

>>> Dict['score'][1]

85

删除字典：

>>> Dict

{'job': 'sa', 'gf': {'job': 'model', 'age': 25, 'name': 'coco'}, 'age': 26, 'score': [78, 85, 83], 'name': 'taoyak'}

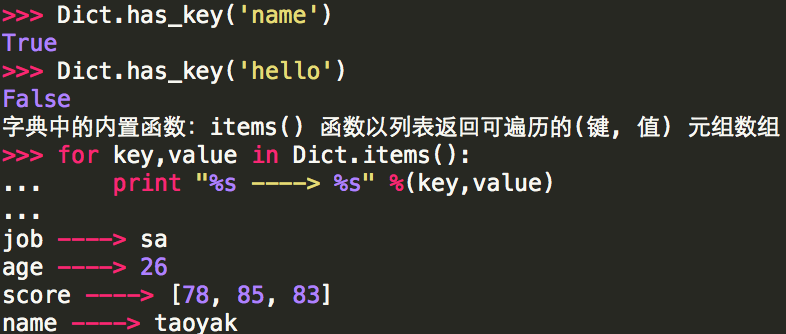
>>> Dict.pop('gf')

{'job': 'model', 'age': 25, 'name': 'coco'}

>>> Dict

{'job': 'sa', 'age': 26, 'score': [78, 85, 83], 'name': 'taoyak'}

字典中的内置函数：has\_key 如果字典中有指定的元素，就返回(True)真,没有就返回(False)假。例如：



判断一个元素是否在字典中，两种方式：

1．用has\_key判断

2. dict.get(key, default=None) ,给字典一个默认的值例如：

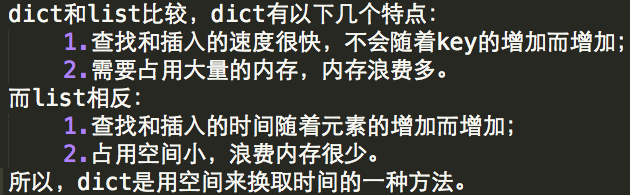
>>> Dict.get('hello','apple')

'apple'

思考：为什么字典查询的速度要比列表快呢？

因为字典的实现原理和查字典是一样的。假设字典包含了一万个汉字，我们要查某一个字，第一种方式是把字典从第一页往后翻，知道找到为止。这种方式就是列表中查找元素的方法，列表越大，查找的速度就越慢。

第二种方法是先在字典的索引表里查找找个字对应的页码，然后直接翻到该页，找到这个字，无论找那个字，这种查询速度都非常快，不会随字典大小改变速度。



小练习，一个列表中有字典，字典中有用户名和密码,

1.判断登录用户是否在列表中

2.如果用户存在，则判断密码是否正确

3．如果用户密码都正确， 打印该用户所有信息。

python 大练习.py

请输入您的用户名: kk

{'passwd': 'abcdef', 'age': 19, 'job': 'cto', 'name': 'kk'}

welcome kk,your job is cto,age is 19

bogon:day2 playcrab$ python 大练习.py

请输入您的用户名: aa

用户不存在

大练习.py

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

users = [{'age': 18, 'job': 'coo', 'name': 'wd', 'passwd': '12323'},

{'age': 19, 'job': 'cto', 'name': 'kk', 'passwd': 'abcdef'},

{'age': 20, 'job': 'cio', 'name': 'pc', 'passwd': 'ABC'}]

username = raw\_input("请输入您的用户名: ")

tmp = {}

for i, user in enumerate(users):

tmp[user['name']] = i

if username in tmp:

print users[tmp[username]]

info = users[tmp[username]]

print "welcome %(name)s,your job is %(job)s,age is %(age)s " % info

else:

print "用户不存在"

**python中文件读写**

1.文件基本读写操作

fo = open(“/tmp/user.txt”) # 打开文件file＝open,默认只读

fo #查看文件句柄

fo.read() #读取，一次全部读完，结果为一个整体的字符串

fo.write('文件路径以及文件名') #写入

fo.close() #关闭文件，关闭后无法打开

2.打开文件的几种模式：

open('path') #默认只读打开

open('path','r+') #读写打开,如果有内容,就会从头覆盖相应字符串的内容

open('path','w') #写入,先删除原文件重新写入,没有文件自己创建

open(‘path’,'w+') #读写,同上

open('path','a') #写入,在文件末尾追加新内容,文件不存在自动创建

open('path','a+') #读写,同上,最常用的

open('path','b') #打开二进制文件,要上述模式结合使用,读取图片

open('path','u') #支持所有的换行符号 \r \n \r\n

例子：

bogon:day2 playcrab$ python test.py

name : taoyake

passwd: 123.com

bogon:day2 playcrab$ cat test.py

F = open('user.txt')

print F.read()

readline 一次之读取一行：

bogon:day2 playcrab$ python test.py

name : taoyake

bogon:day2 playcrab$ cat test.py

F = open('user.txt','r')

print F.readline()

readlines: 把文件一行一行读取出来

bogon:day2 playcrab$ python test.py

name : taoyake

passwd: 123.com

bogon:day2 playcrab$ cat test.py

F = open('user.txt','r')

line = F.readlines()

for i in line:

print i

第三种打开文件的方法：加了容错，而且更加优雅

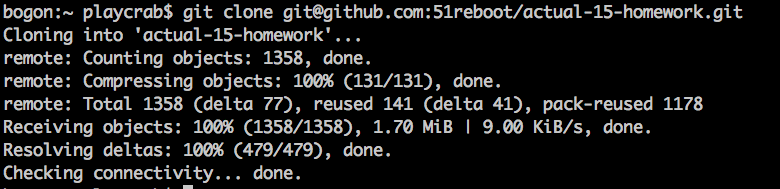
with open(‘user.txt’) as f:

data = f.read()

do something

使用git提交代码简单介绍：

git clone #git clone后边跟仓库地址，克隆到本地一个远端的git仓库

修改好代码之后，添加文件到Git仓库，分两步：

第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；

git add .

第二步，使用命令git commit，完成。

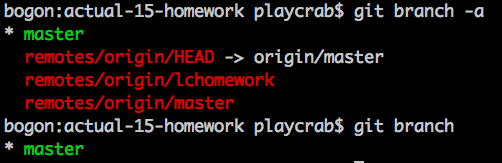
git commit –m "update code"

第三步，上传本地当前分支代码到master主分之。Git仓库可以有多个分支，只有一个master主分支

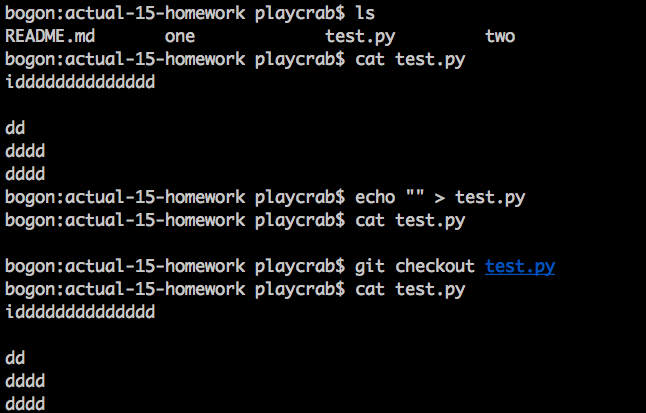
# git push origin master

查看分支：

git branch –a #查看所有分支 git brancha #查看当前分支



撤销未提交的修改：git checkout file #file就是你修改过的但是没有提交的文件



查看git提交历史纪录命令：

## git log



Git必须知道当前版本是哪个版本，在Git中，用HEAD表示当前版本，也就是最新的提交237f38d6521c2ef3b73de7ba3564a676979a7f3a（注意我的提交ID和你的肯定不一样），上一个版本就是HEAD^，上上一个版本就是HEAD^^，当然往上100个版本写100个^比较容易数不过来，所以写成HEAD~100。

git reset --hard HEAD^ #这是线上的作业，我就不执行命令了。

删除git仓库中的文件代码

git rm test.txt

rm 'test.txt'

git commit -m "remove test.txt"

这时候 文件test.txt 才是真的删除掉了。