基础

1. 变量名不能以数字开头
2. 函数，类等都叫作方法
3. 注释用#
4. 注意str int float函数  
   str()将数字转化为字符串，输出时经常使用  
   int()将字符串转化为数字，但是只能用于整型

float()将字符串转化为数字，可以用于小数

1. 两个函数可以一起用，比如a.strip().title()
2. upper（） lower（）
3. 指定列表中的元素数字不能超过范围，比如  
   x=[‘a’,’b’,’c’]  
   print(x[2])  
   而insert（）函数的指定数字可以超过范围  
   x.insert（100,’d’）  
   del x[y]函数中指定的数字y也不能超过范围（del是语句，不是方法）  
   pop（）函数没有指定数字是默认最后一个，指定了数字时不能超过范围

remove（）函数也可以删去列表中的值，不过记得直接指定，类似于关键字实参

1. sort（）函数首字母列表排序

sort（reverse=True）函数（其中reverse=True为实参）首字母反序

永久性的

1. sorted（）临时的排序，用法：  
   print（sorted（x））
2. reverse（）按照原来列表的顺序反序排列而不是字母表顺序  
   x.reverse()  
   print(x)
3. len(x)输出列表长度，即元素个数  
   对象为字典时，输出键值对的个数
4. for 循环：  
   for i in x:  
    print(i)  
   遍历列表
5. range(20)=range(0,20)默认开始为0

range（1，7）只包含1，2，3，4，5，6

range（1，7，2）指定步长为2,也不包含最后一个（这里是7）  
range中的是数，不是字符串，是5不是‘5’

1. list（）转化为列表

如：x=list（range（1，6））

1. min（x）列表x中的最小值  
   max（）最大值

sum（）求和

1. 列表解析：  
   x=[ k\*\*2 for k in range(1,11)]
2. 切片：

print（x[1:4]）包含这里的1但是不包含这里的4，注意1代表第二个，输出列表中的相应元素  
print（x[2:]）表示第三个到最后一个  
print（x[-3:]）表示倒数第三个到最后一个  
print（x[:3]）表示第一个到第第四个但是不包含第四个（3表示第四个。。。）

反正就是包含前一个数字不包含后一个数字

1. 遍历切片：  
   for k in x[:3]:
2. 复制列表时用切片  
   y=x[:]而不是y=x  
   后者看起来只是将x赋给y，实际上两者关联，在新的y中加元素x中也会出现
3. 元组类似列表，但是不可修改，依然可以用遍历等方法来访问元组，而且用（）不用[ ]
4. if用来判断时，等于==，不等于！=
5. 判断时注意大小写，用函数的方法可以利于判断而且不会改变原来变量的值，例如：  
   a=’bwm’
6. title()==’Bwm’将会返回True
7. 检查特定的元素在不在列表中：  
   if a in x:  
   if a not in x:
8. if 为条件语句，注意什么时候会跳出去  
   如果判断一个模块时，可以用if-elif-else语句  
   如果多个模块，则用多个if语句
9. x是一个列表  
   if x：  
   如果x为空，则if判断为False，如果至少含有一个元素，则判断为True
10. 字典{ }注意有键值对，如：

x={‘color’:’green’,’points’:5}

1. 访问键值对：  
   print(x[‘color’])  
   添加键值对：  
   x[‘a’]=’b’  
   修改同上，但是前提是原字典中要有相应的键值对

删除键值对：  
del x[‘points’]

1. 遍历字典：  
   for a,b in x.items():  
   a,b是两个变量名，可以随便取（不能丢了.items()）
2. 仅仅访问键：

for a in x.keys():  
仅仅访问值：  
for b in x.values():

1. 字典一般来说是没有顺序的，只有对应关系,按顺序遍历的方法如下：  
   for a in sorted(x.keys())：
2. set ()函数用于剔除重复元素：  
   for a in set(x.values()):
3. 有关嵌套：  
   列表中可以有字典，访问举例如下：  
   for k in x[0:3]:

if k[‘color’] == ‘green’:  
 k[‘color’] == ’yellow’

字典中也可以有列表，但是一般处在‘值‘的位置，如：  
 for k in x[‘toppings’]:  
 print(k)

字典中套字典：

for a,b in c.items():

print('Username'+a)

full=b['first']+' '+b['last']

location=b['location']

print('\tFull name:'+full.title())

print('\tLocation:'+location.title())

1. input（）函数：需要一个变量来接受输入值  
   input（）输入数字时输入的类型是字符串而不是数字
2. %取余  
   //取整
3. 用while循环时注意在循环体中加入a+=1等来控制循环次数
4. 循环条件通常使用：  
   a=True（布尔表达式）

while a：

或者while True：加break

1. continue 忽略循环体中余下的代码，并返回到循环的开头
2. x为列表，while x表示x为不为空时进行循环
3. while ‘a’ in x：如果列表x中有元素‘a’（注意引号）
4. name=input(your name)  
   response=input(your favorite food)

x[name]=response (x是空的字典)

1. 定义函数用def a(x，y):x，y为形式参数，调用时需要传递实参a(x,y)注意传递顺序
2. 文档字符串的注释，描述函数是干嘛的’’’…’’’  
   调用时用help（a）a为函数名
3. 用关键字实参传递信息时不用考虑顺序，但是记得形参的名字要相同：  
   def a(x):  
     
   a(x=’hello’)
4. 默认值在定义函数时就要设置好，传递实参时将忽略默认值(一般放在后面)。
5. 返回值return  
   调用函数时要用一个变量接收，如：  
   def a():  
    return   
   x=a()
6. 将默认值设为空字符串，能让实参变为可以选择的，联系if语句
7. def a(\*b):创建一个空元组b，将所有输入值都存入其中，可以在不确定输入几个值时使用。

def a(\*\*b):创建一个空字典b，   
def build\_profile(first, last, \*\*user\_info):

profile={}

profile['first\_name'] = first

profile['last\_name'] = last

for key, value in user\_info.items():

profile[key] = value

return profile

user\_profile = build\_profile('albert', 'einstein', location='princeton',

field='physics')

print(user\_profile)

1. 导入模块  
   a.py  
   def b():

import a（a是文件名）  
a.b(……)

或者  
 from a import b，c，d (这里b，c，d为函数名，不要 加括号！)

起别名：  
from a import b as k

导入所有函数：  
from a import \*

1. 给形参指定默认值时，等号两边不要有空格。  
   函数调用中的关键字实参也要这样
2. 类的开头第一个字母要大写
3. 给属性设定默认值：  
   self.a=b a为某个属性，b为默认值，其中a不需要在初始化时指定形式参数
4. def \_\_init\_\_(self,……):必不可少
5. 创建子类时，父类必须包含在当前文件当中，且位于子类前面，在括号内要指定父类的名称

def \_\_init\_\_(self, make, model, year):   
 """初始化父类的属性"""

super().\_\_init\_\_(make, model, year)

这里的super() 是一个特殊函数，帮助Python将父类和子类关联起来。这行代码让Python调用ElectricCar 的父类的方法\_\_init\_\_() ，让ElectricCar 实例包含父类的所 有属性。父类也称为超类 (superclass)，名称super因此而得名。

1. 导入类：  
   import a：  
   调用时：  
   k=a.A(……)  
   from a import A,B  
   a是文件名，A，B是类的名称注意大小写

导入所有类：  
from a import\*  
将实例用做属性：  
class Electriccar(Car):  
 def \_\_init\_\_(self,make,model,year):  
 super().\_\_init\_\_(make,model,year)  
 self.battery=Battery()

1. python标准库  
   from collections import OrderedDict(这是一个类)  
   a=OrderedDict（） 返回一个空的有序字典  
   from random import randint  
   x=randint(1,6) 返回一个1到6之间的整数(包括1和6)
2. with open('a.txt') as b:

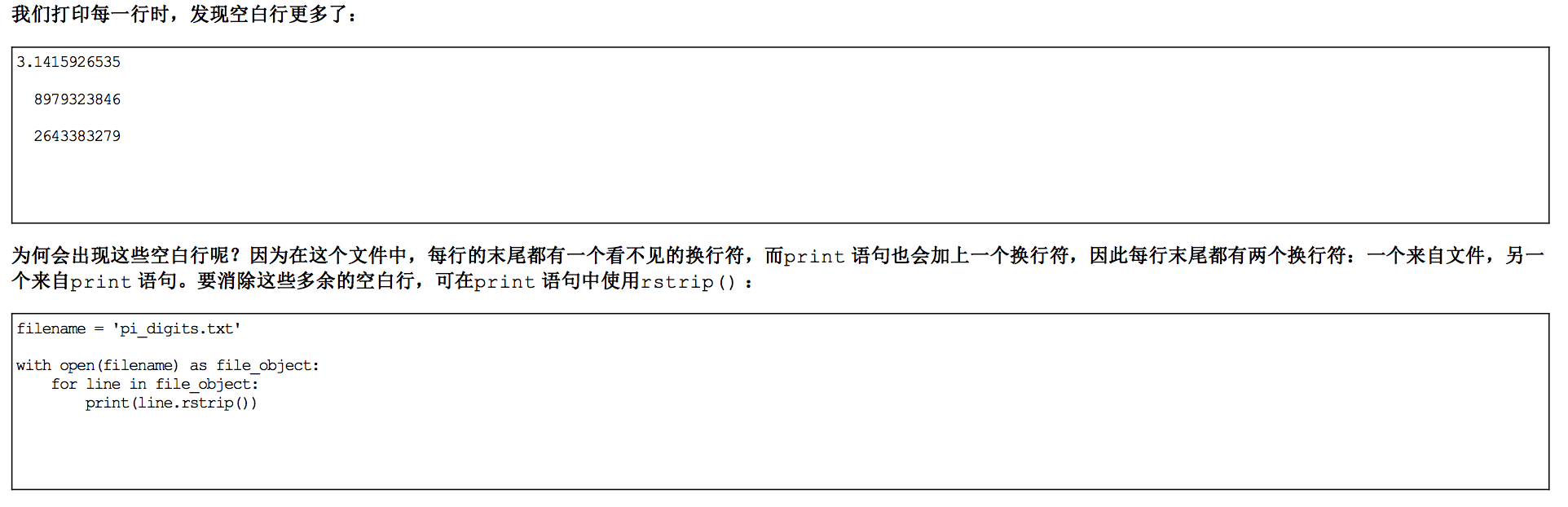
contents = b.read()   
print(contents)   
open后的括号里是文件名（全称包括格式）  
文件必须和py文件在同一级目录中  
open函数返回一个文件对象，储存在变量b中  
关键字with 在不再需要访问文件后将其关闭。  
read读取内容，将其转化成一个长长的字符串储存在变量中

去掉末尾的空行：

因为read（）到达文件末尾时返回一个空字符串，而它显示出来就是一个空行  
用rstrip（）去除

1. macOS系统用／／／来指定文件位置  
   windows系统用[\\\来指定](file:///C:\来指定)

寻找时同级往下找



1. readlines()读取每一行的内容，并返回一个列表
2. 字符串中的切片，长度

e=‘3.14159265358’

print(e[:8])

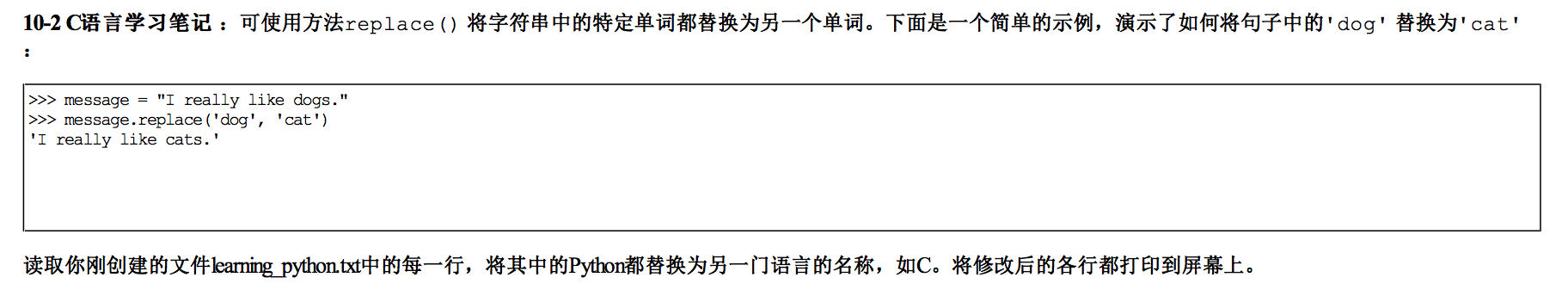
输出为3.141592

len(e)  
输出为13

1. 判断一个字符串b是否包含另一个字符串a：  
   if a in b:
2. 3种读文件的方法：  
   1.read函数+print语句

2.for 语句+变量名

3.readlines语句



1. 读取模式‘r’  
   写入模式‘w’  
   附加模式‘a’

读取和写入模式‘r+’

1. w时如果没有文件，系统创建  
    如果有，返回文件对象时将清空(亲测有效)
2. a时不会清空文件，加在后面
3. try:  
   except 错误类型：  
   else：  
   常见错误类型：  
   ZeroDivisionError  
   FileNotFoundError

TypeError

1. 方法split（）以括号后的内容拆分生成一个列表
2. 失败时一声不吭：pass
3. count（）方法：特定单词在字符串中出现了多少次
4. 存储数据用json模块：  
   json.dump(a,b)

a为你想要存储的数据

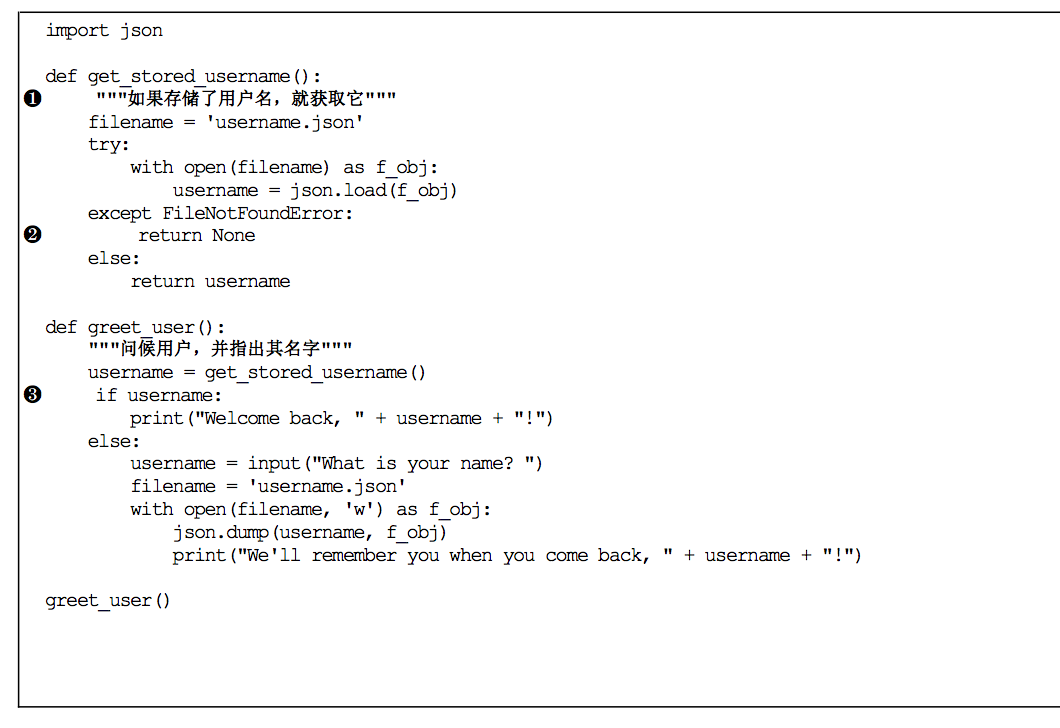
b为接受文件名对象的变量

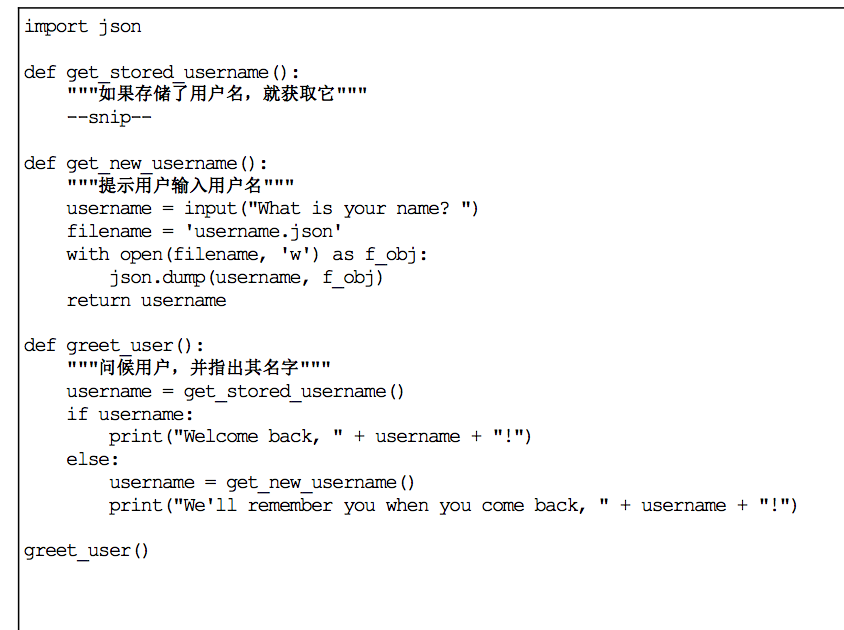
（注意用写入模式‘w’）

a=json.load(b)将相应的内容读取到内存中

b为接受文件名对象的变量

1. 有关重构和存储数据的一个极好例子！





项目

1. import matplotlib.pyplot as a

a.plot(b)

a.show()

plot（）函数画折线图

show（）函数打开matplotlib查看器

b为画图需要传递的参数

1. a.plot(squares,linewidth=5)

参数linewidth决定了plot（）绘制的线条的粗细

1. a.title（‘。。。。。。’，fontsize=14）函数

指定标题

a.xlabel（‘value’，fontsize=14）函数

指定x轴标题

a.ylabel（‘。。。。。。’，fontsize=14）函数

指定y轴标题

fontsize=14

指定总标题，x，y轴标题大小

a.tick\_params(axis=’both’,labelsize=14)

函数设置刻度的样式，labelsize=14将刻度标记的字号设置为14

参数axis的值为'x'、'y'、'both'，分别代表设置X轴、Y轴以及同时设置，默认值为'both'。

1. 若没有提供y的关系，则默认第一个数据点对应的x的坐标值为0
2. 传递x，y的值  
   a.plot(x,y,linewidth=5)

x,y可以为列表

1. scatter()函数画散点图

import matplotlib.pyplot as plt

plt.scatter(2,4, s = 200，edgecolor='yellow')（画出点（2，4））(s表示点的尺寸)（edgecolor表示一个形状的轮廓线的颜色，默认值是'none'）

plt.title("Square Numbers", fontsize=24)

plt.xlabel("Value", fontsize = 14)

plt.ylabel("Square of Value", fontsize = 14)

# Set size of tick labels.

plt.tick\_params(axis = 'both', which = 'major', labelsize = 14)

plt.show()

参数which的值为 'major'、'minor'、'both'，分别代表设置主刻度线、副刻度线以及同时设置，默认值为'major'

1. 绘制一系列点  
   同上给列表即可用a.scatter(x,y,s=200)
2. 自动计算数据

import matplotlib.pyplot as plt

x\_values = list(range(1, 1001))

y\_values = [x\*\*2 for x in x\_values]

plt.scatter(x\_values, y\_values, s = 40)

# Set chart title, and label axes.

plt.title("Square Numbers", fontsize=24)

plt.xlabel("Value", fontsize = 14)

plt.ylabel("Square of Value", fontsize = 14)

# Set size of tick labels.

plt.tick\_params(axis = 'both', which = 'major', labelsize = 14)

# Set the range for each axis.

plt.axis([0, 1100, 0, 1100000])（框定x，y坐标轴的范围）

plt.show()

1. 自定义颜色  
   参数c=（x，y，z）x，y，z为三原色的成分值，在（0，1）之间（分别为红绿蓝）  
   c=’magenta’为洋红色

plt.scatter(x\_values,y\_values, edgecolor='none',

        c =y\_values, cmap=a.cm.Blues, s = 40)

（注意a是最初as后的名字）

颜色映射（colormap）是一系列颜色，它们从起始颜色渐变到

结束颜色。将参数c设置成了一个y值的列表，并使用参数cmap

说明使用哪个颜色映射（cm表示colormap，Blues表示一组蓝色）

1. 自动保存图表

plt.savefig('squares\_plt.png', bbox\_inches='tight')

把画图保存在与python代码同一个目录下的squares\_plt.png中，

第二个参数bbox\_inches=‘tight’表示图周围的空白边弄小。

注：bbox应该是border box的缩写。

1. 随机漫步

from random import choice

class RandomWalk():

"""A class to generate random walks."""

def \_\_init\_\_(self, num\_points=5000):

"""Initialize attributes of a walk."""

self.num\_points = num\_points

# All walks start at (0, 0).

self.x\_values = [0]

self.y\_values = [0]

def fill\_walk(self):

"""Calculate all the points in the walk."""

# Keep taking steps until the walk reaches the desired length.

while len(self.x\_values) < self.num\_points:

# Decide which direction to go, and how far to go in that direction.

x\_direction = choice([1, -1])

x\_distance = choice([0, 1, 2, 3, 4])

x\_step = x\_direction \* x\_distance

y\_direction = choice([1, -1])

y\_distance = choice([0,1,2,3,4])

y\_step = y\_direction \* y\_distance

# Reject moves that go nowhere.

if x\_step == 0 and y\_step == 0:

continue

# Calculate the next x and y values.

next\_x = self.x\_values[-1] + x\_step

next\_y = self.y\_values[-1] + y\_step

self.x\_values.append(next\_x)

self.y\_values.append(next\_y)

import matplotlib.pyplot as plt

rw=RandomWalk(50000)

rw.fill\_walk()

plt.figure(dpi=128,figsize=(10,6)) （函数figure() 用于指定图表的宽度、高度、分辨率和背景色。你需要给形参figsize 指定一个元组，向matplotlib指出绘图窗口的尺寸，单位为英寸。dpi为分辨率 ）

point\_numbers = list(range(rw.num\_points))

plt.scatter(rw.x\_values, rw.y\_values, c=point\_numbers, cmap=plt.cm.Reds, edgecolor='none', s=1) （随着点的次序变化颜色）

# Emphasize the first and last points.

plt.scatter(0, 0, c='green', edgecolors='none', s=100)

plt.scatter(rw.x\_values[-1], rw.y\_values[-1], c='red', edgecolors='none', s=100)

# Remove the axes.

plt.axes().get\_xaxis().set\_visible(False)

plt.axes().get\_yaxis().set\_visible(False) (隐藏坐标轴)

plt.show()

keep\_running = input("Make another walk? (y/n): ")

if keep\_running == 'n':

break

1. with open(filename) as f:

reader=csv.reader(f)

firstline=next(reader) 返回文件的下一行

for index,column\_header in enumerate(firstline):  
 print(index,column\_header)

enumerate()函数获取每个元素的索引和值

1. x=[]

for row in reader:  
 x.append(row[1]) 第二列加进来

或者high=int(row[1])  
 x.append(high)

绘制折线图表时，将列表x赋给plt.plot()即可

1. 添加日期  
   from datetime import datetime

for row in reader:  
 current\_date=datetime.strptime(row[0],’%Y-%m-%d’)

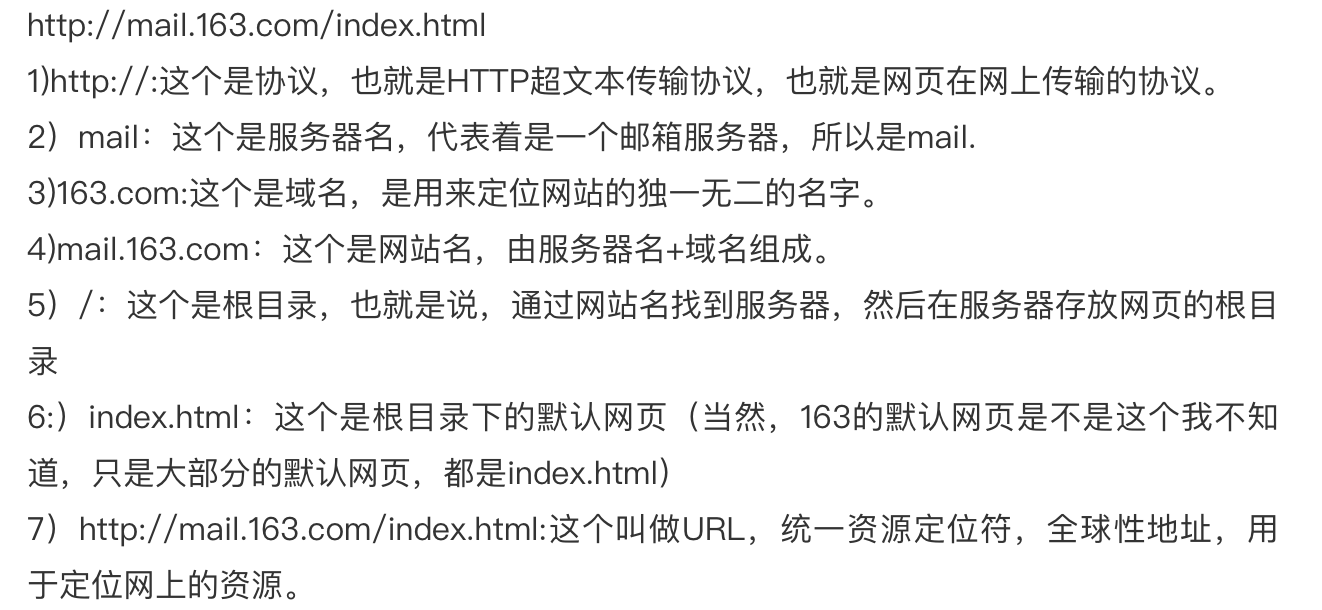
strptime()为方法，需要传递两个实参，第一个是时间，第二个是格式

|  |  |
| --- | --- |
| 实参 | 含义 |
| %A | 星期的名称，如Monday |
| %B | 月份名，如January |
| %m | 用数字表示的月份（01-12） |
| %d | 用数字表示月份中的一天（01-31） |
| %Y | 四位的年份，如2015 |
| %y | 两位的年份，如15 |
| %H | 24小时制的小时数（00-23） |
| %I | 12小时制的小时数（01-12） |
| %p | am或pm |
| %M | 分钟数（00-59） |
| %S | 秒数（00-61） |



fig.autofmt\_xdate()

自动格式化日期，防止其溢出绘图区域（绘制斜的日期标签）

1. alpha表示透明度，在（0，1）之间，0为完全透明
2. 两根线之间区域的颜色  
   plt.fill\_between(x,y1,y2,facecolor=’…’,alpha=…)  
   facecolor为区域颜色
3. 错误处理  
   用try-except-else语句
4. 一些常识。。。应该不考
5. r=requests.get(url)  
   将响应对象储存在变量r中
6. r.status\_code

响应对象包含一个status\_code的属性，它让我们知道请求是否成功了(状态码200表示请求成功)

1. 方法json（）将信息转换成一个字典
2. repo\_dict['owner']['login']  
   联系字典的嵌套