向文婷 1813088

2020/6/15 补充

1. Git操作

查看版本：查看历史记录 git log

查看状态：git status

工作区和暂存区：git add将文件添加到暂存区，git commit将暂存区的内容全部提交到当前分支。

版本回退：若修改后想退回到以前版本，命令git reset – hard HEAD^回退到上一个版本。Git reset –hard 版本号，回退到指定版本。

撤销修改：只在工作区修改，未添加到暂存区，git checkout –文件名，将该文件在工作区的所有修改全部撤销。添加到了缓存区git reset HEAD 文件名，将缓存区的修改回退到工作区。

删除文件：rm 文件名，将工作区没有的文件删除。Git rm 文件名，从版本库删除该文件，再rm 工作区的文件，再重新提交。

1. GitHub

分支：创建分支，切换到新分支 git checkout -b 分支名，创建并切换，git branch 查看当前分支，合并指定分支到当前分支 git merge 分支名。切换分支也可用 git switch -c 分支名。删除分支 git branch -d 分支名。

当发生冲突时，Git不能自动合并分支，必须先手动解决冲突再提交。试图合并时失败，会提醒哪些存在冲突。git log –graph可查看分支合并图。

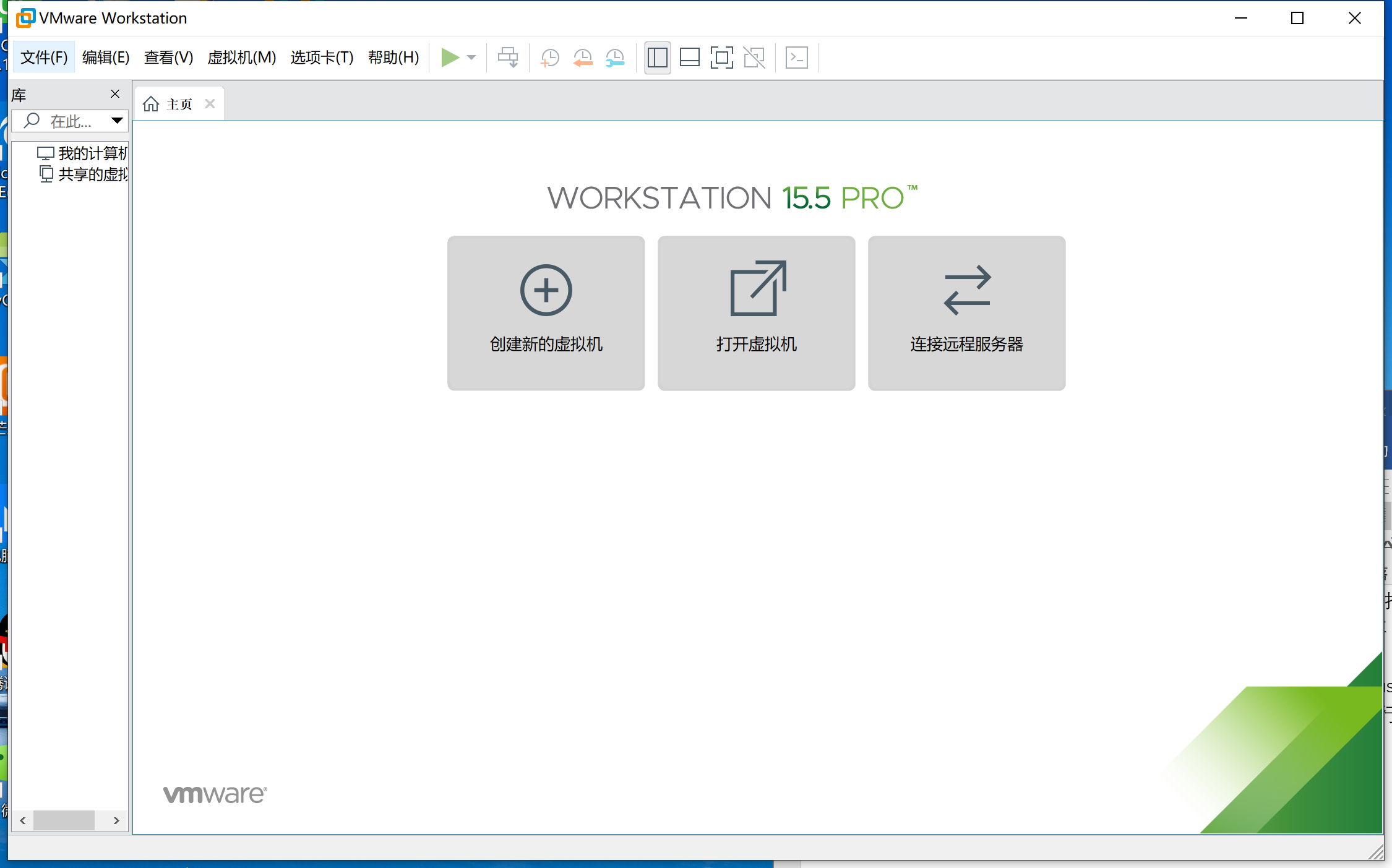
标签：发布一个版本时，通常先在版本库中打一个标签tag。方便取出被打标签的版本。打标签，先切换到需要打标签分支，再git tag 标签名；或git tag 标签名 版本号。可用git tag查看所有标签。git tag -d 标签名。推送标签git push origin 标签名。远端删除标签，先本地删除，再git push origin : refs/tags/标签名。

SourceTree：Git图形界面工具。下载，运行。添加本地Git库。方便提交、分支、推送等操作。

2020/6/16

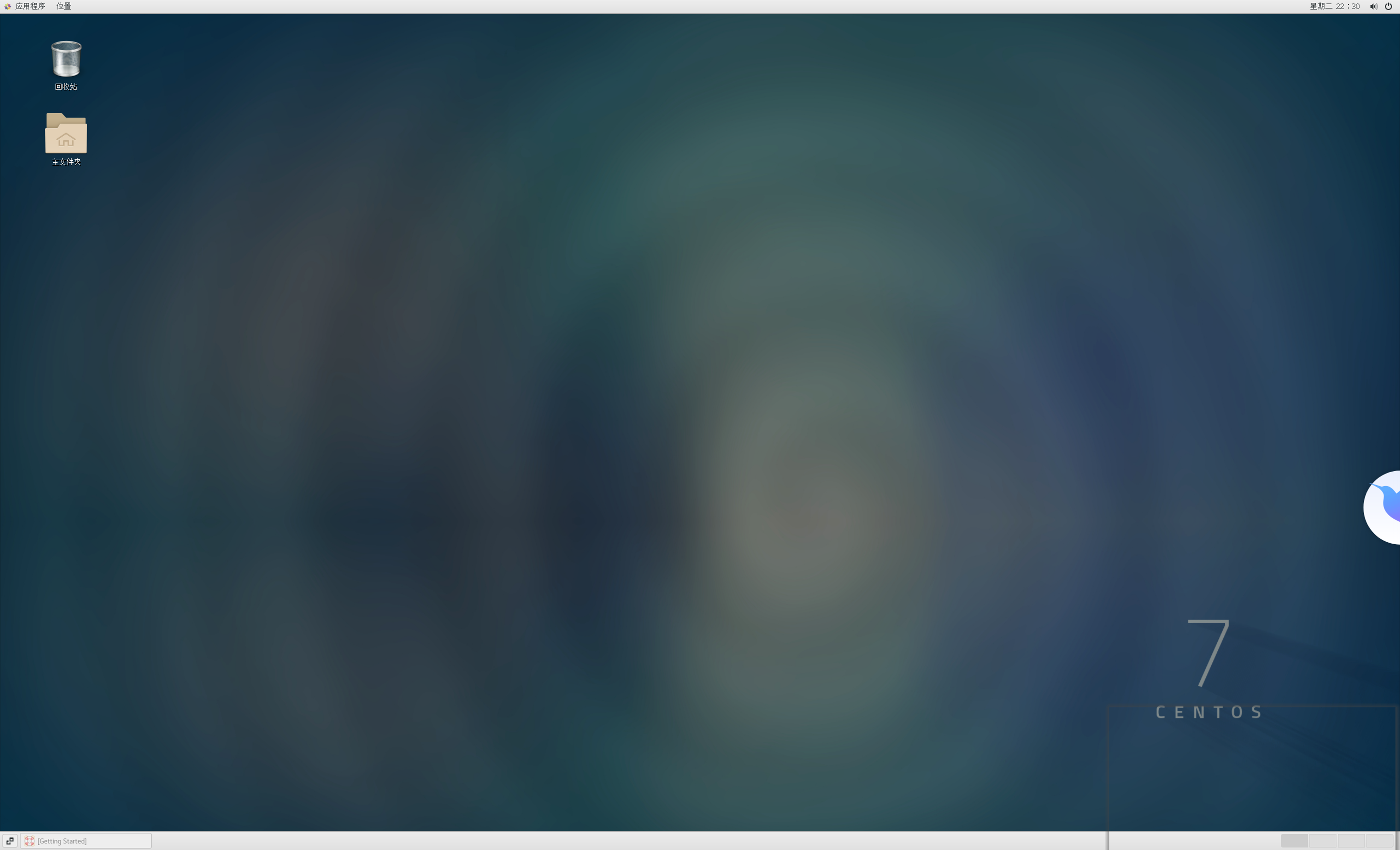
1. 安装虚拟机：

下载安装VMware



下载CentOS 7镜像

打开VM，创建新的虚拟机。



1. Linux：

基于POSIX和UNIX的多用户、多任务、支持多线程和多cpu的操作系统。

Ls列出目录及文件，ls -p查看文件属性、所属用户和组。文件类型；属主权限；属组权限；其他用户权限。d,目录文件；-文件；l链接文档；b可供储存的接口设备；c端口设备。

更改文件属性，chgrp：更改文件属组；chown：更改文件属主，也可以同时更改文件属组；chmod：更改文件9个属性。

cd：切换目录

pwd：显示目前的目录

mkdir：创建一个新的目录

rmdir：删除一个空的目录

cp: 复制文件或目录

rm: 移除文件或目录

mv: 移动文件与目录，或修改文件与目录的名称

用户和用户组管理：

useradd: 添加新的用户账号，useradd 选项 用户名

选项：-c comment 指定一段注释性描述。

-d 目录 指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。

-g 用户组 指定用户所属的用户组。

-G 用户组，用户组 指定用户所属的附加组。

-s Shell文件 指定用户的登录Shell。

-u 用户号 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。

userdel：删除账号 userdel 选项 用户名 -r把用户主目录一起删除。

usermod: 修改账号， -l改为新用户名。

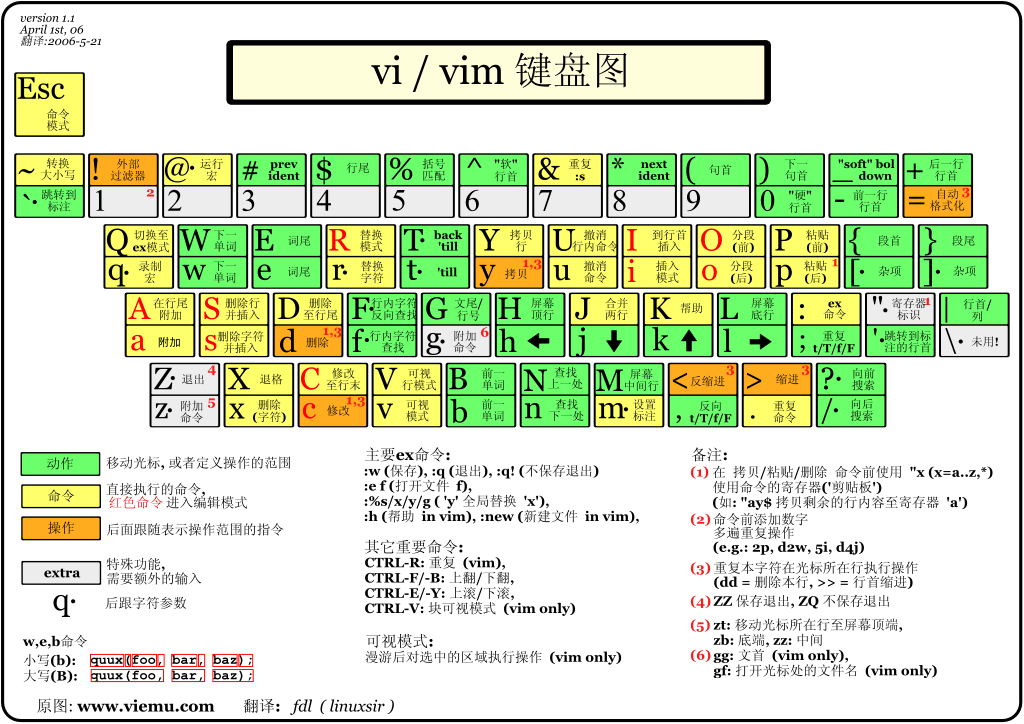
groupadd：添加新的用户组。-g 指定新用户组的组标识号。

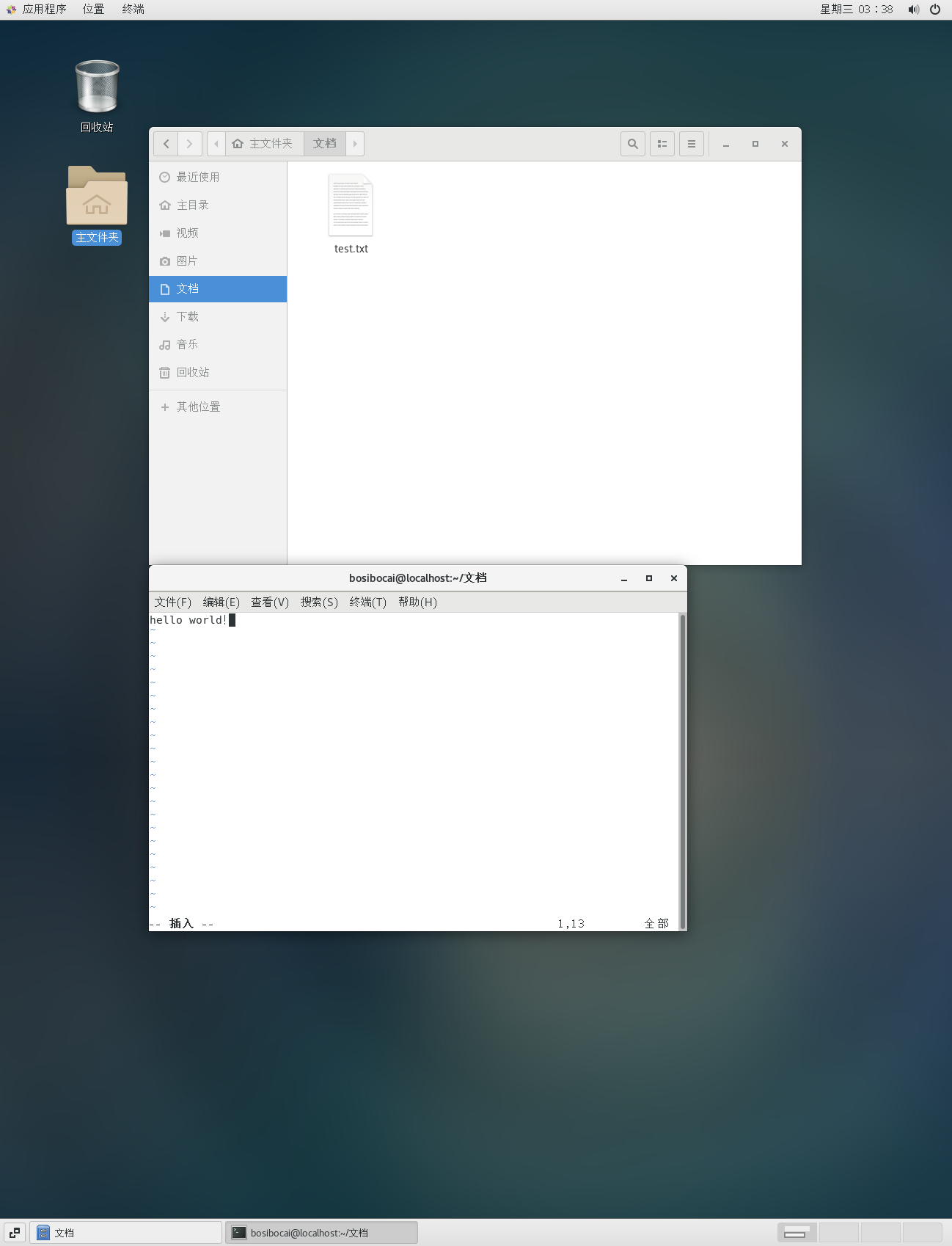
groupdel：删除用户组。

groupmod：修改用户组属性。

1. Vi/Vim编辑器:

Vi三个模式，插入模式、命令模式、底线命令模式。





1. rpm：

rpm软件包的管理工具。

选项：

-a：查询所有套件；

-b<完成阶段><套件档>+或-t <完成阶段><套件档>+：设置包装套件的完成阶段，并指定套件档的文件名称；

-c：只列出组态配置文件，本参数需配合"-l"参数使用；

-d：只列出文本文件，本参数需配合"-l"参数使用；

-e<套件档>或--erase<套件档>：删除指定的套件；

-f<文件>+：查询拥有指定文件的套件；

-h或--hash：套件安装时列出标记；

-i：显示套件的相关信息；

-i<套件档>或--install<套件档>：安装指定的套件档；

-l：显示套件的文件列表；

-p<套件档>+：查询指定的RPM套件档；

-q：使用询问模式，当遇到任何问题时，rpm指令会先询问用户；

-R：显示套件的关联性信息；

-s：显示文件状态，本参数需配合"-l"参数使用；

-U<套件档>或--upgrade<套件档>：升级指定的套件档；

-v：显示指令执行过程；

-vv：详细显示指令执行过程，便于排错。

-ql：查询

1. yum：

是一个在Fedora和RedHat以及SUSE中的Shell前端软件包管理器。提供了查找、安装、删除某一个、一组甚至全部软件包的命令。

yum 可选 操作 操作对象

常用命令：

列出所有可更新的软件清单命令：yum check-update

更新所有软件命令：yum update

仅安装指定的软件命令：yum install <package\_name>

仅更新指定的软件命令：yum update <package\_name>

列出所有可安裝的软件清单命令：yum list

删除软件包命令：yum remove <package\_name>

查找软件包 命令：yum search <keyword>

清除缓存命令:

yum clean packages: 清除缓存目录下的软件包

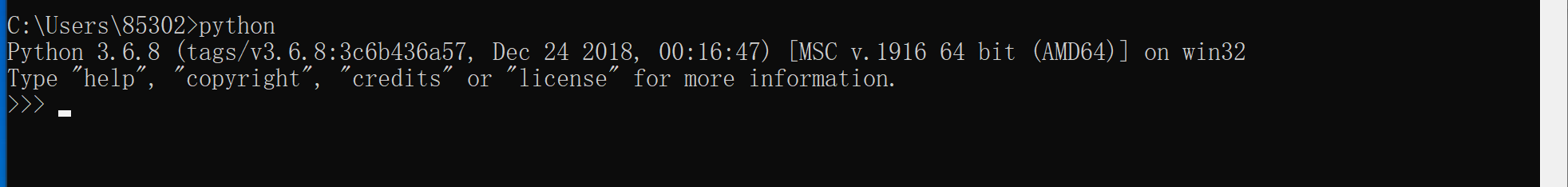
yum clean headers: 清除缓存目录下的 headers

yum clean oldheaders: 清除缓存目录下旧的 headers

yum clean, yum clean all (= yum clean packages; yum clean oldheaders) ：清除缓存目录下的软件包及旧的headers

2020/6/17

1. 下载安装python



1. Python

代码块不使用{}控制类，函数以及其他逻辑。用缩进来写模块。\可将一句语句分为多行。[]、()、{}内也为一条语句。

标识符，\_开头为不可直接访问的类属性 \_\_开头为类的私有成员 \_init\_结尾代表类的构造函数。

变量和数据类型：

变量赋值不需要类型声明。在使用前必须赋值。

五种标准数据类型，Numbers,String,List,Tuple元组,Dictionary字典。del语句删除引用。

Number包括int,long,float,complex。String从左向右从0开始，从右向左从-1开始。截取s[头下标:尾下标]。List标识符[]，切片，l[头下标:尾下标]，复制列表=list[:]而不使用切片时则会指向一个list，添加元素，append()添加到列尾，insert(索引，值)添加到列表中；删除元素del list[索引]，pop()删除列为元素，pop(索引)弹出任意位置元素；sort()永久性排序，sorted()临时排序，如sorted(list)为排序后，list仍为原顺序；reverser()反转list；len()长度，对数值列表range(开始,结束)但不包括结束。min(),max(),sum()进行统计计算。Tuple标识符()，元组不可修改元素，但可重新赋值，相当于只读列表。Dictionary，{}标记，list为有序对象的集合，字典为无序对象的集合。通过键来存取，’key’:’value’,添加/修改 dictionary[‘key’]=’value’，删除del dictionary[‘key’]。类型转化。

条件语句：if 条件判断：，and、or检查多个条件，某元素是否存在在列表中，if 元素 not in/in list，if-elif-else结构。对于列表 if list:若list不为空else：list为空

for语句：for 变量 in 范围： 缩进控制开始和结束 遍历字典 for key,value in dictionary.items()：，遍历字典所有键for key in dictionary.keys()：，遍历字典所有值for key in dictionary.values()：,嵌套{{,,},{,,},{,,},}和字典存储字典，存储list，key=list

while语句：while 条件：break退出循环，continue继续循环。While value in list:list.remove()。

用户输入：input()等待用户输入返回字符串，int()返回数字。

函数：def构造一个函数，形参实参，可以关键字值对传值，def describe\_pet(animal\_type, pet\_name):，describe\_pet(animal\_type='hamster', pet\_name='harry')，默认值def describe\_pet(pet\_name, animal\_type='dog'):，也可返回字典，禁止函数修改列表可以传切片，参数\*trup ，将所有参数存入trup元组中因此可以传递任意数量的参数，参数\*\*trup,未知传入参数类型时。匿名函数 lambda创造 只含有 一个语句 函数名=lambda 参数1，参数2：语句。导入模块，import 模块名 或 from 模块import 函数。as取别名，from 模块 import函数as别名。

类：创建类，如dog类，class dog(): def \_init\_()初始化函数。继承 class bigdog子类(dog父类):，super()父类构造函数，重写父类方法。导入模块，from模块import类，python标准库。

文件和异常：读取文件，with open(‘绝对/相对路径文件名’) as file\_object: contents=file\_object.read() print(contents) 会比原文件多一排空行，若想删除最后一行改为print(contents.rstrip())。逐行阅读for line in file\_object:。使用文件的内容，lines =file\_object.readlines() for line in lines:。写入文件，写入空文件open之后，file\_object.wirte(“”)，添加内容到文件，open时在文件后传入参数‘a’。异常，try-except代码块，else代码块。

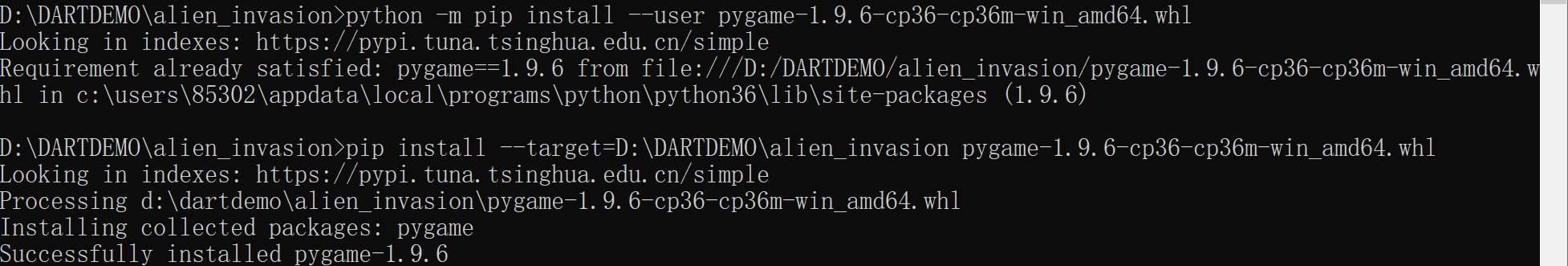
2020/6/18

外星人入侵：

在游戏《外星人入侵》中，玩家控制着一艘最初出现在屏幕底部中央的飞船。玩家可以使用箭头键左右移动飞船，还可使用空格键进行射击。游戏开始时，一群外星人出现在天空中，他们在屏幕中向下移动。玩家的任务是射杀这些外星人。玩家将所有外星人都消灭干净后，将出现一群新的外星人，他们移动的速度更快。只要有外星人撞到了玩家的飞船或到达了屏幕底部，玩家就损失一艘飞船。玩家损失三艘飞船后，游戏结束。

新建文件夹alien\_invasion

下载安装Pygame



1. 武装飞船：将创建一艘可左右移动的飞船，这艘飞船在用户按空格键时能够开火。

创建一个空的Pygame窗口。监听各种事件。设置背景窗口大小。

添加飞船图像。新建文件夹image，将飞船图像文件保存到这个文件夹中，绘制图形。

重构game\_functions，存储运行函数。管理事件check\_events（），更新屏幕update\_screen()

驾驶飞船：响应按键；允许不断挪动；左右移动根据移动标记调整飞船的位置；调整飞船的速度每次while循环时飞船最多移动的像素；限制飞船的活动范围；

射击：按空格时发射子弹，向上穿行，抵达上边缘消失。子弹类；将子弹存储到编组中；发射子弹，响应空格键按下；删除已消失子弹；限制子弹数量

1. 外星人：

创建外星人图像。

创建一群外星人，一行容纳多少个，创建多行。

移动外星人群，控制速度，添加方向，检查是否到达边缘。

射杀外星人，判断子弹是否击中外星人。

生成新的外星人群。

提高子弹速度。

结束游戏，判断外星人是否与飞船碰撞或是否到达屏幕底部。发生时游戏结束。

1. 计分：

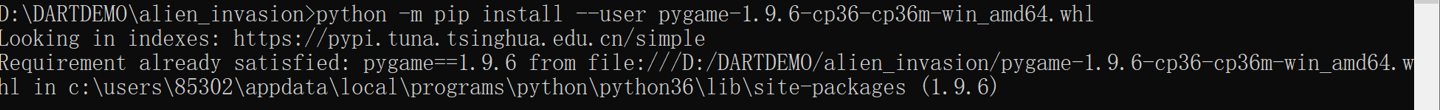
添加play按键，按下时游戏开始，并且按键消失。Button类，绘制按钮。

等级提高，速度增加，每消灭一群外星人后提高等级，加快速度。

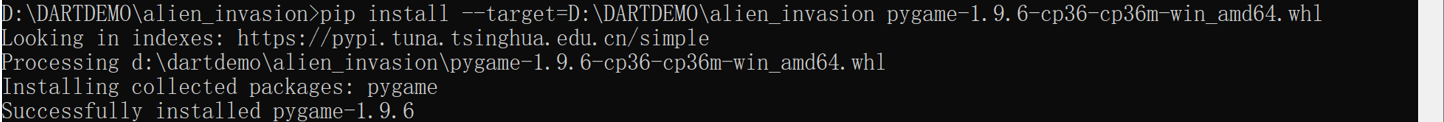
计分，跟踪玩家得分，显示最高得分、当前等级和飞船数。创建记分牌，显示在右上角，每次开始重置为0。每消灭一个外星人计分，随着等级的提高，每消灭一个外星人加分更多。记录所有玩家最高得分，有玩家超过时更新，在上方中间。显示等级，在右上角。显示属于飞船数，在左上角，绘制出来。

1. 问题与解决

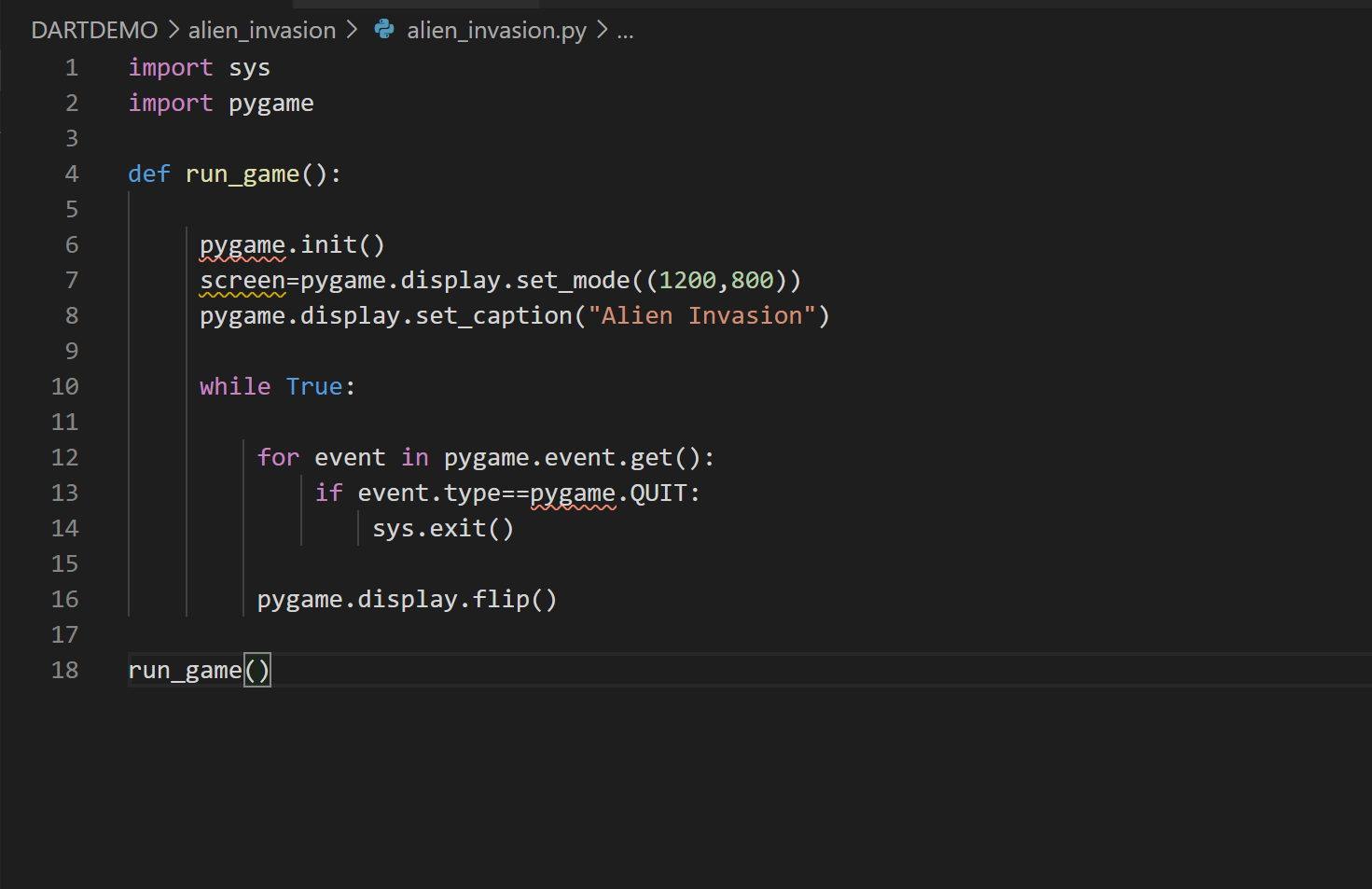
1、在安装pygame时，先打开项目文件



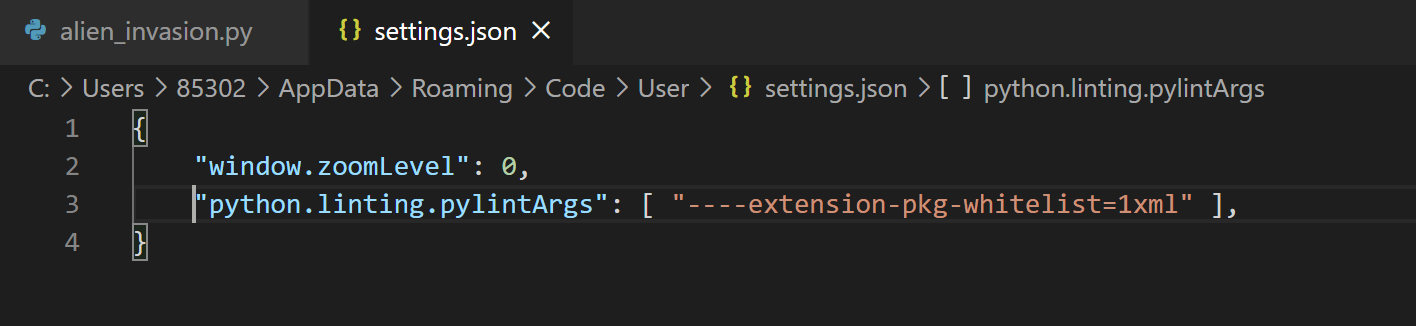
报错，应该指定安装路径



2、在VScord+python导入pygame模板时报错



它与Visual Studio Code按字面意思获取代码的事实有关，因为你无法导入pygame.init（），它认为它不是一个正确的模块。 要解决此问题，请打开settings.json（方法1：按ctrl+shift+p，搜索"setting"，选择“首选项：打开设置(json)”。方法2：文件->首选项->设置->搜索栏搜索"python"->点击“在setting.json中编辑”。）并粘贴"python.linting.pylintArgs": [ "----extension-pkg-whitelist=1xml" ],



错误消失

2、加载图片时出现了错误Couldn't open images\ship.png

将加载语句改为

self.image=pygame.image.load(r'D:\DARTDEMO\alien\_invasion\images\ship.png').convert\_alpha()

加r和绝对路径即可加载

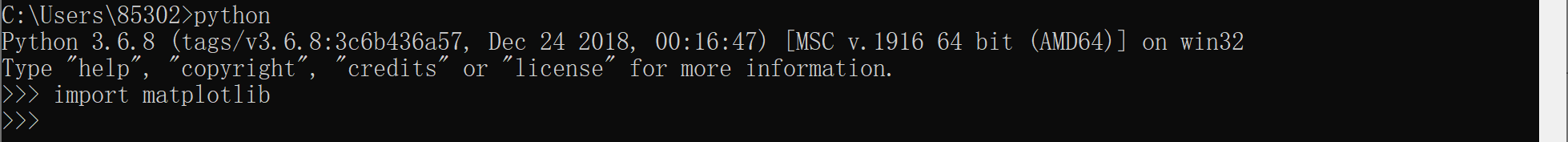
2020/6/19

数据可视化：通过可视化表示来探索数据

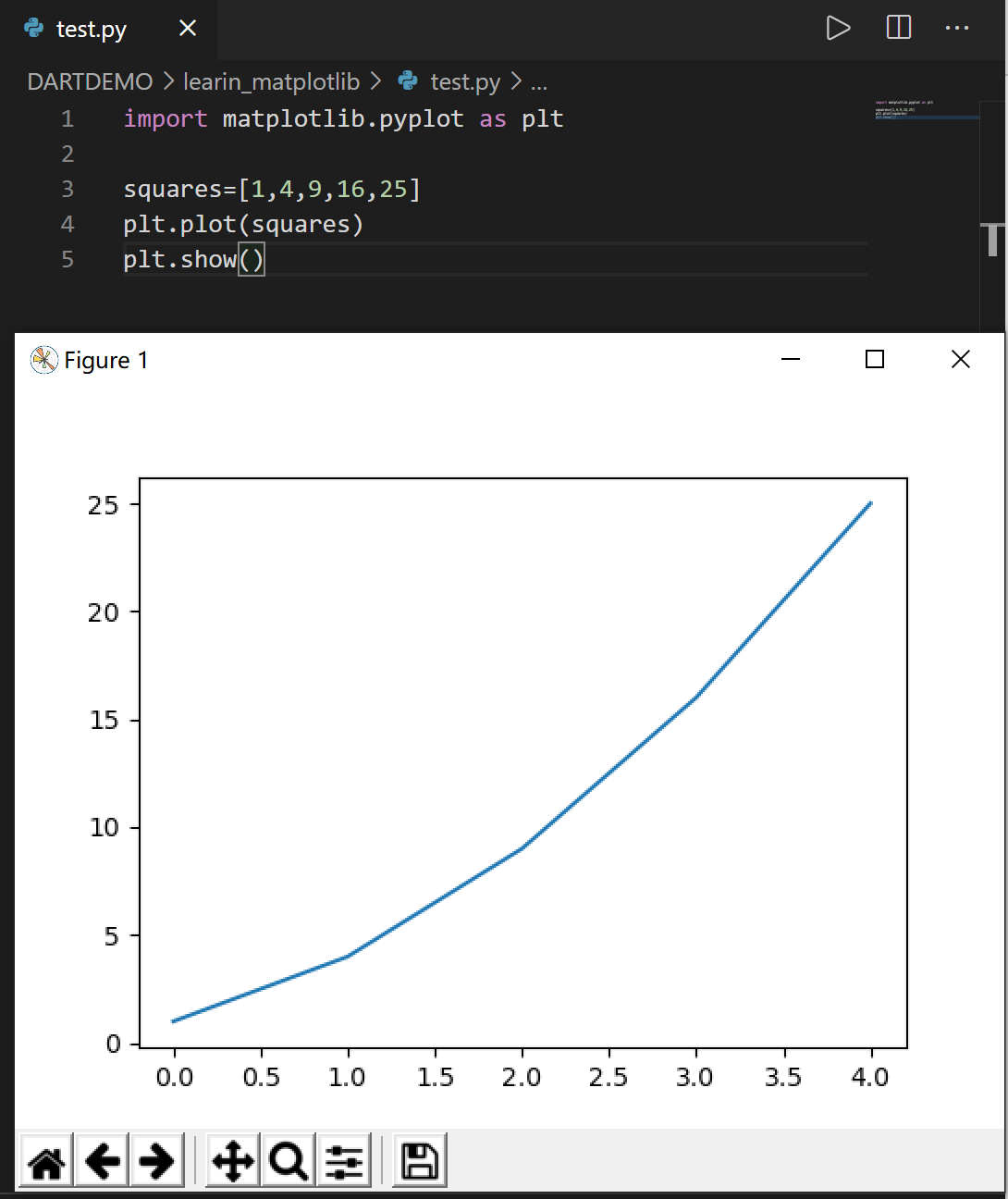
1. 安装matplotlib

进入项目文件夹，运行

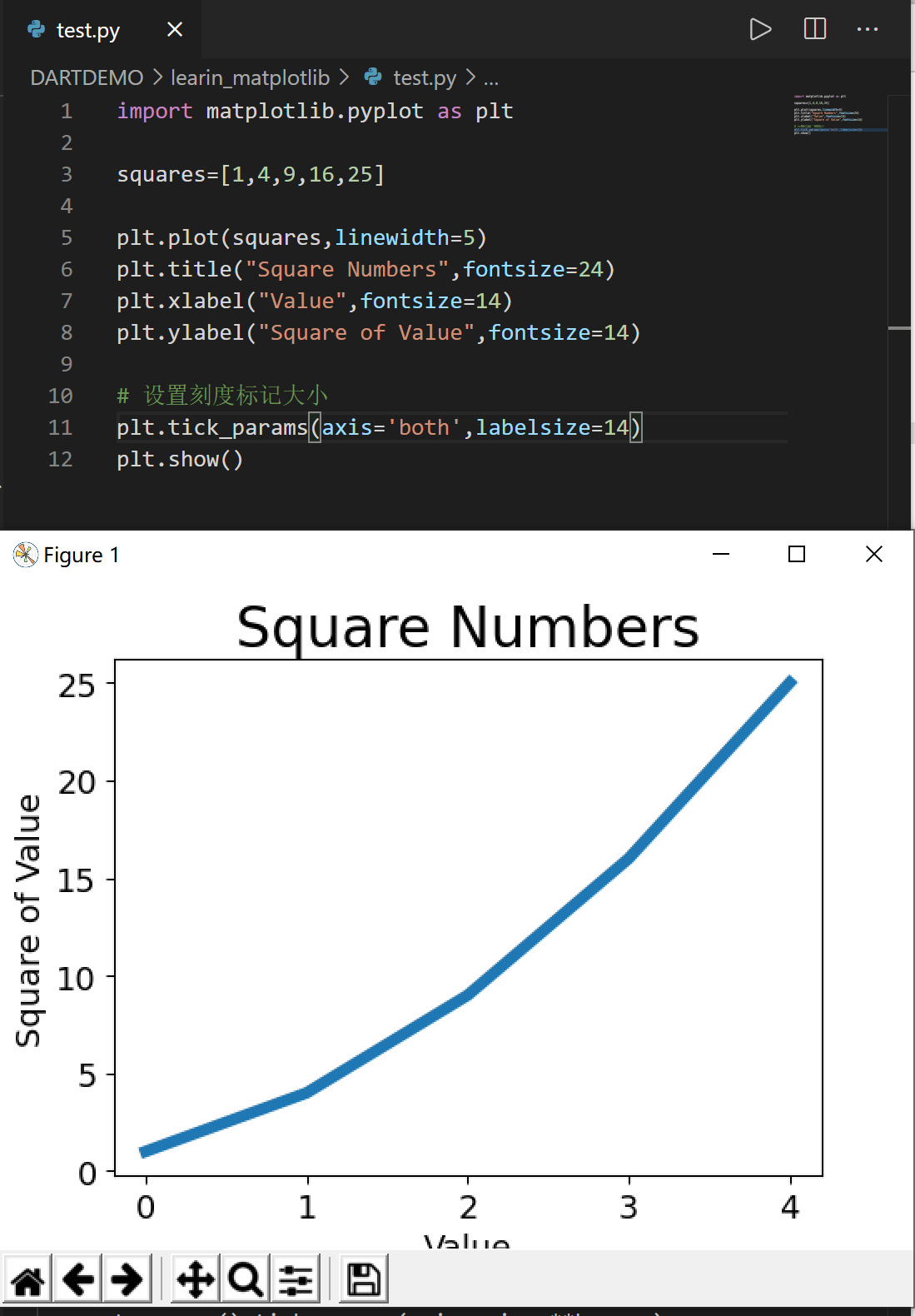
pip install --target=D:\DARTDEMO\learin\_matplotlib matplotlib， 安装matplotlib，安装成功



1. 绘制简单折线图：

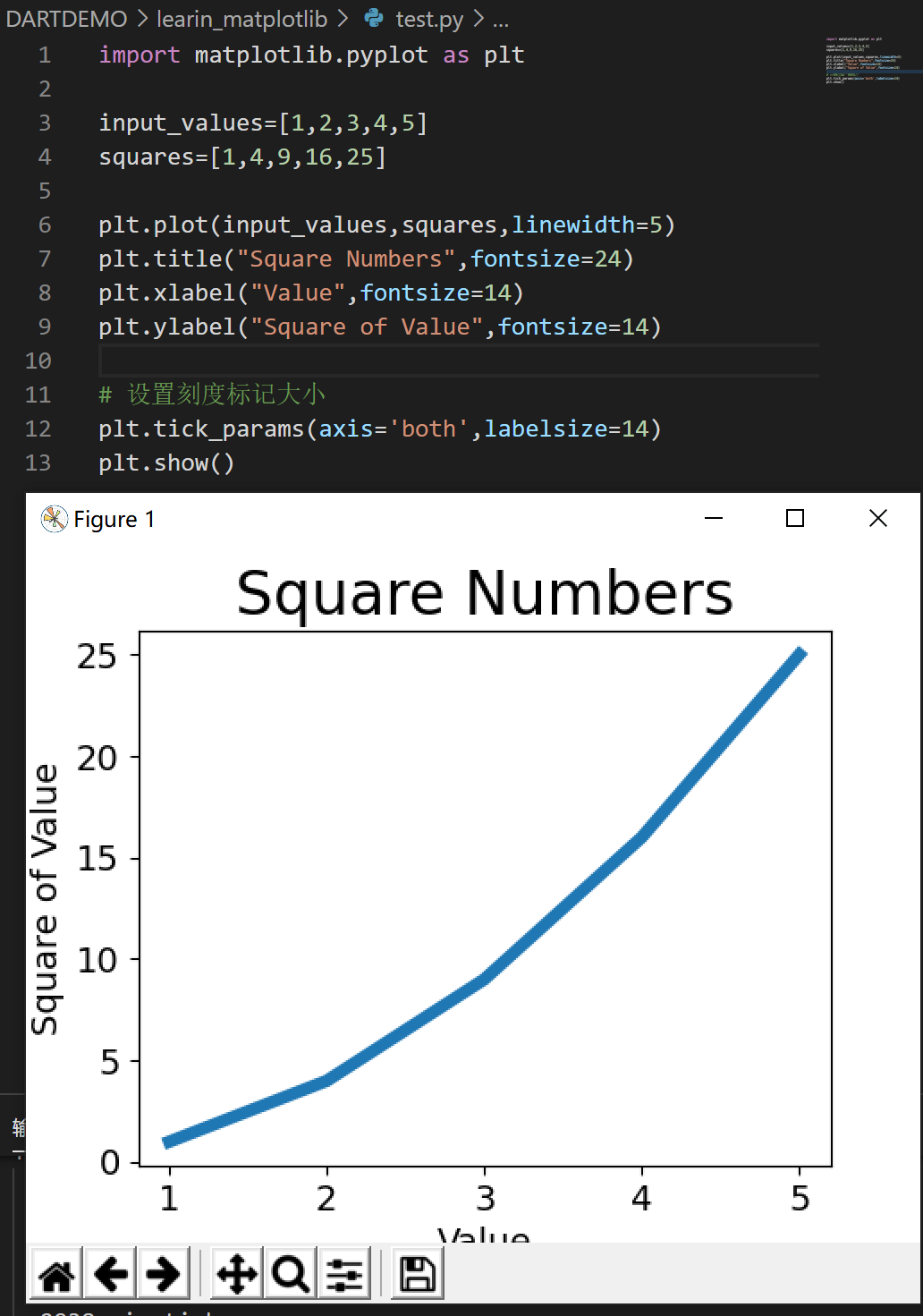


修改标签文字和线条粗细

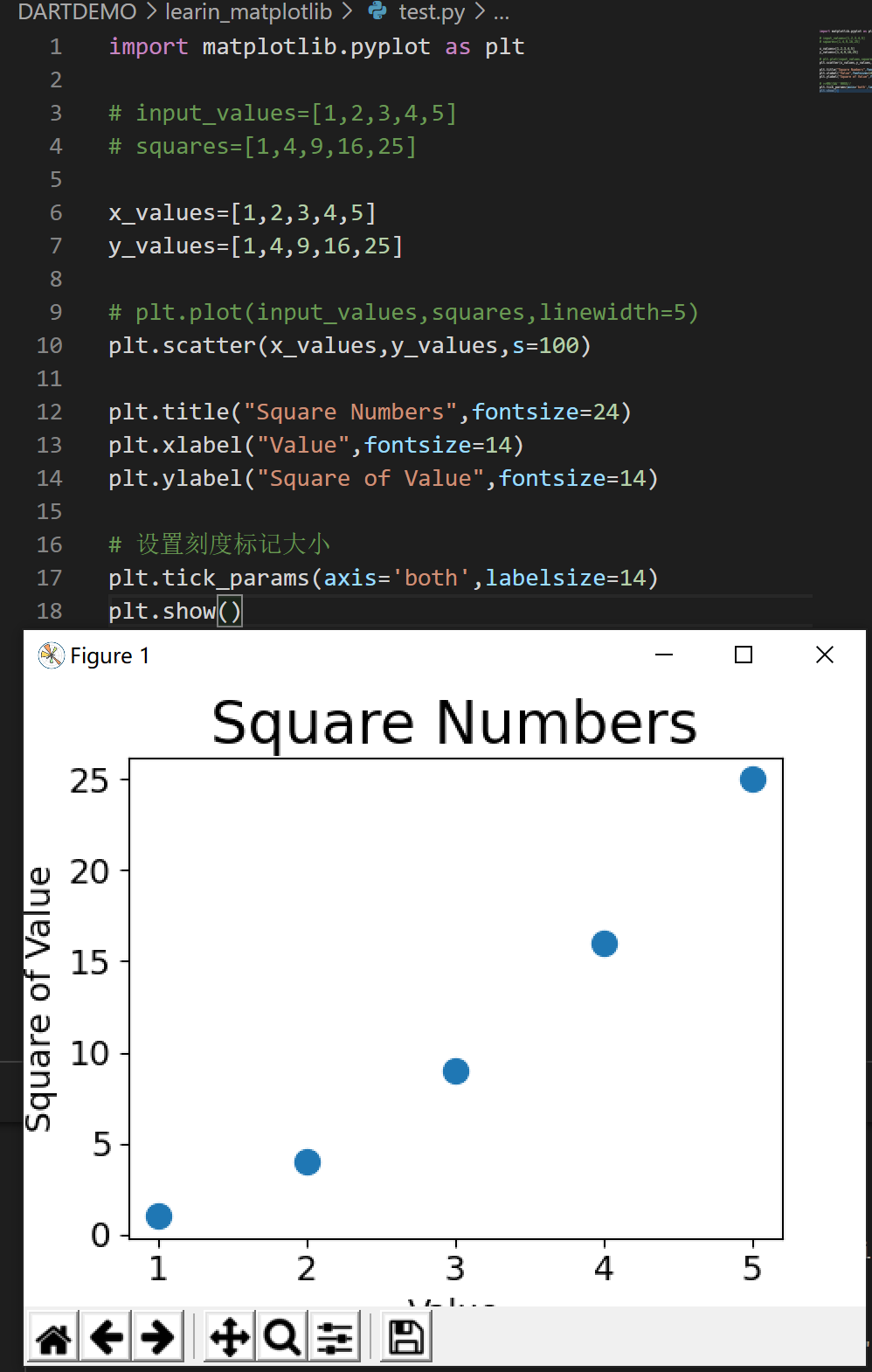


校正图形：

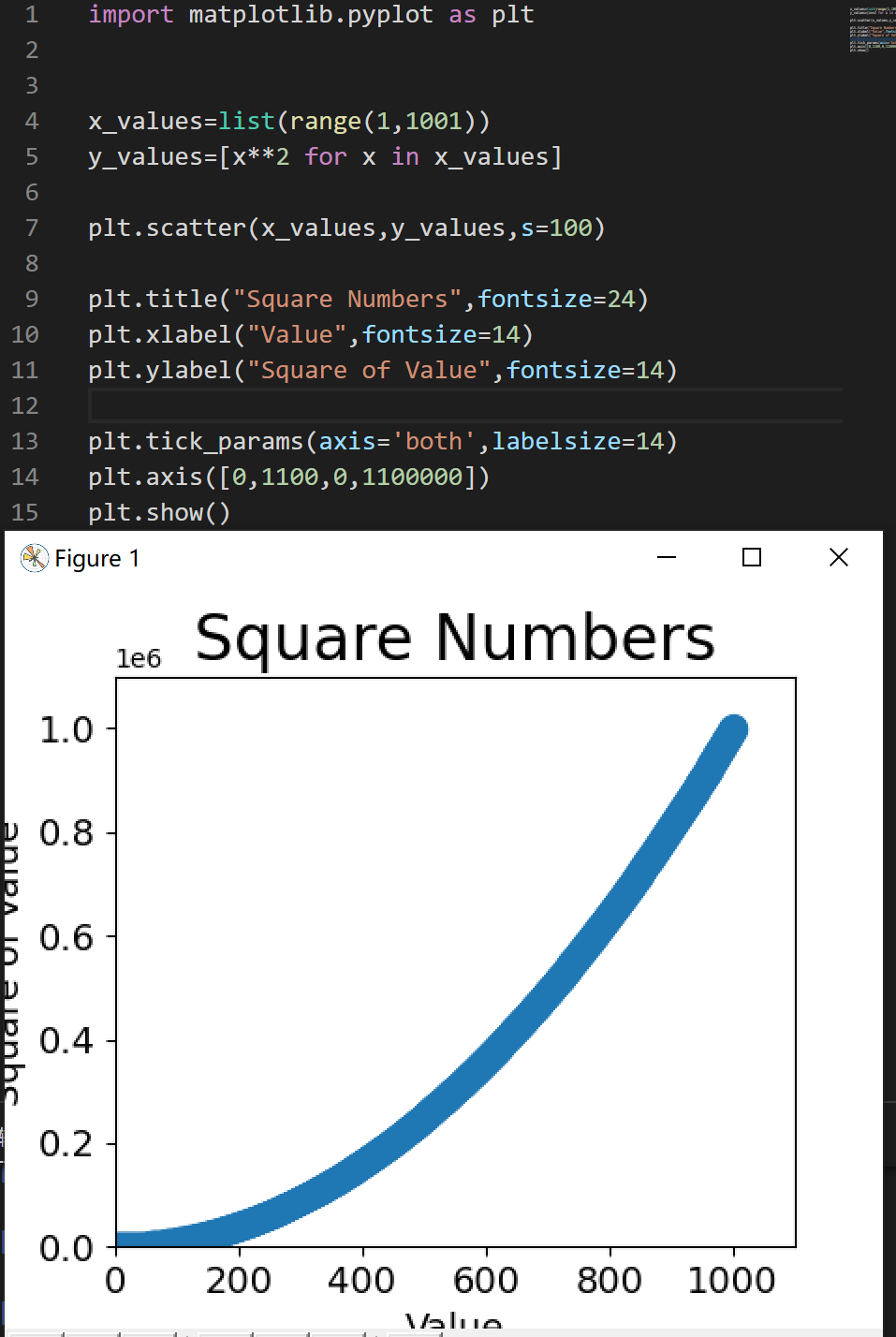
向plot()提供一系列数据时，默认第一个数据点对应x坐标为0，改变默认模式可向plot()提供输入值和输出值



scatter()绘制散点图



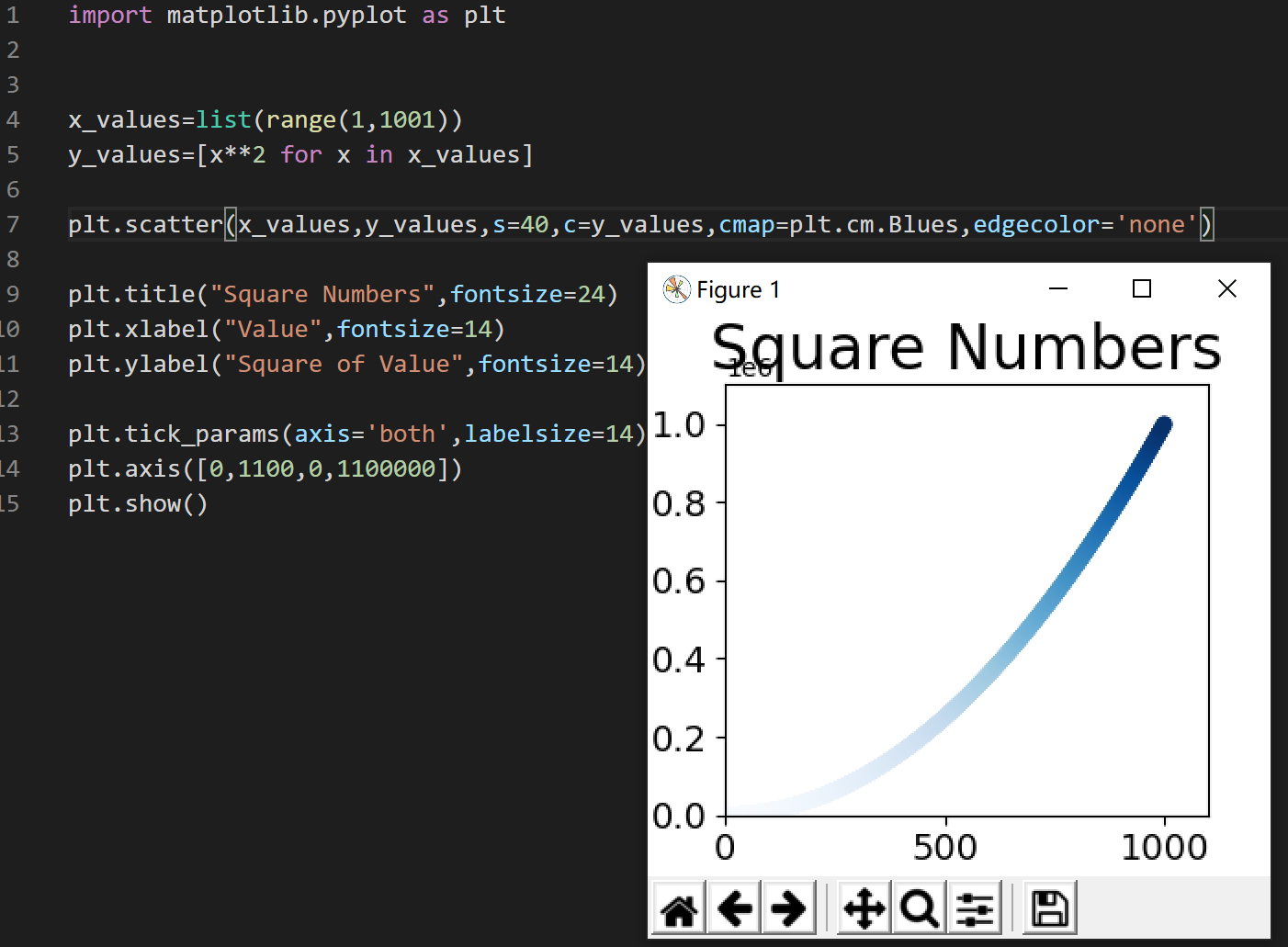
自动计算数据



删除数据点的轮廓，默认为蓝色点黑色轮廓，scatter()传参时传递实参edgecolor='none'，如plt.scatter(x\_values, y\_values, edgecolor='none', s=40)

自定义颜色scatter()传递参数c，如plt.scatter(x\_values, y\_values, c='red', edgecolor='none', s=40)，c也可为RGB模式。

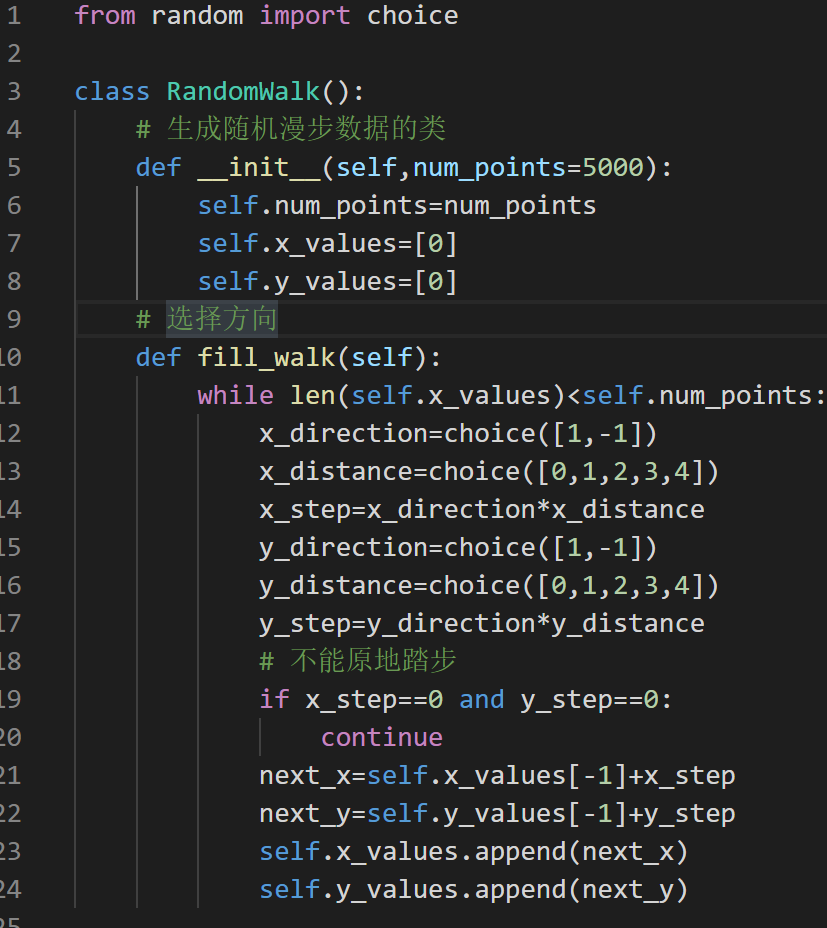
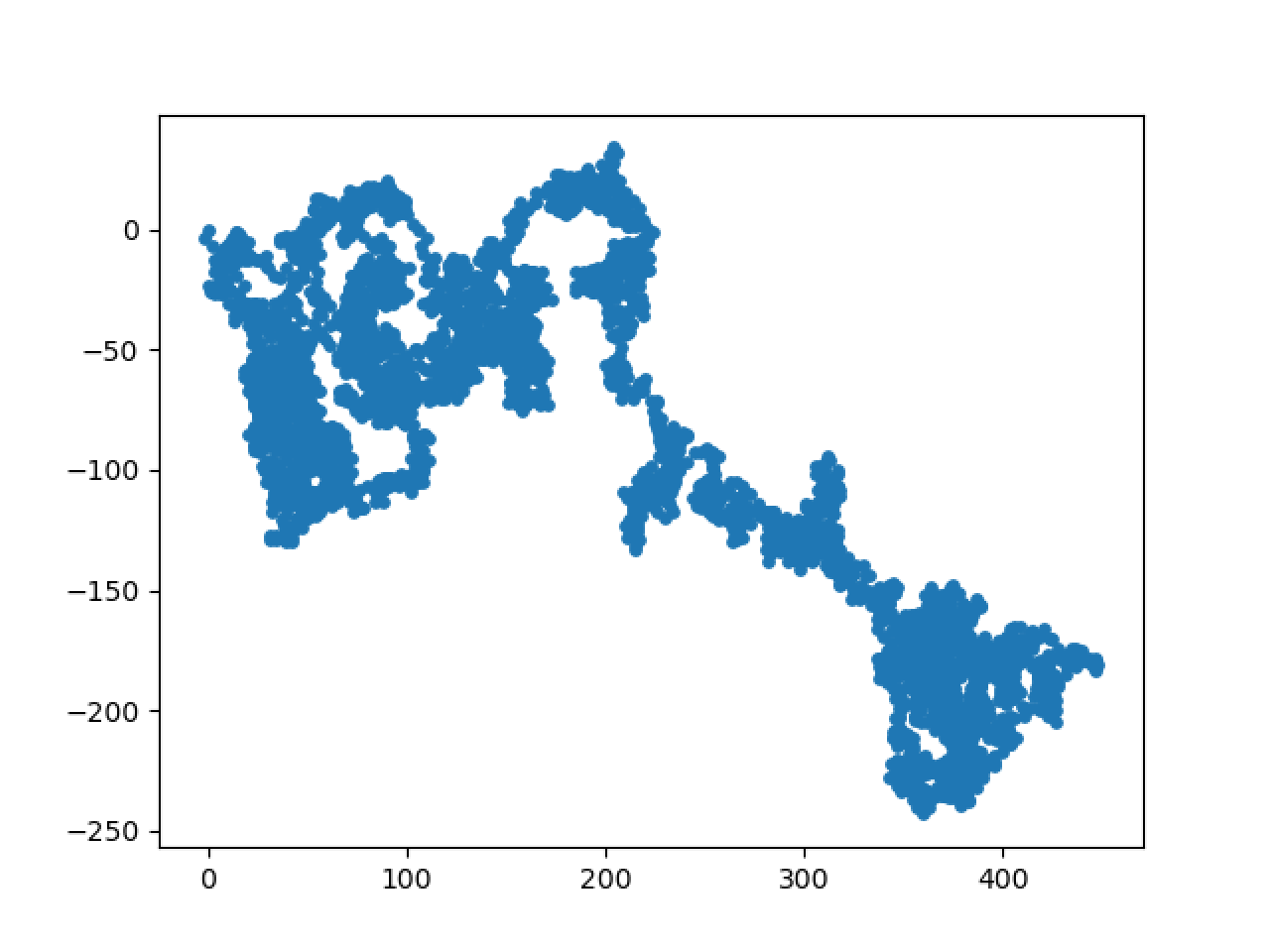
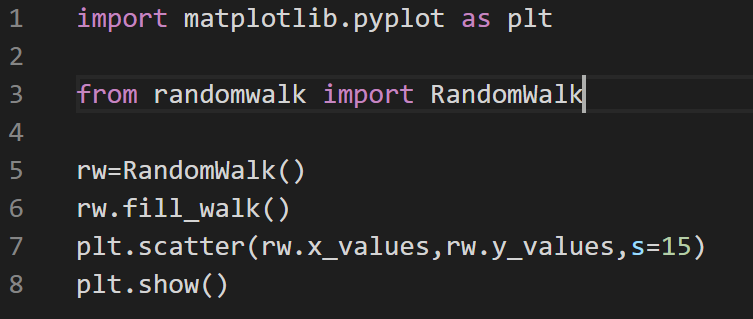
颜色映射，一系列颜色，从开始颜色到结束颜色，方便看出数据规律



自动保存图表，将对plt.show()的调用替换为plt.savefig()

1. 随机漫步

生成随机漫步数据，RandomWalk()类

模拟多次随机漫步，只要程序处于活动状态，不断模拟随机漫步，while True:

设置随机漫步图的样式，点着色；重绘起点和终点；隐藏坐标轴；增加点数即为增大num\_points值；调整尺寸设置，设置绘图窗口的尺寸。

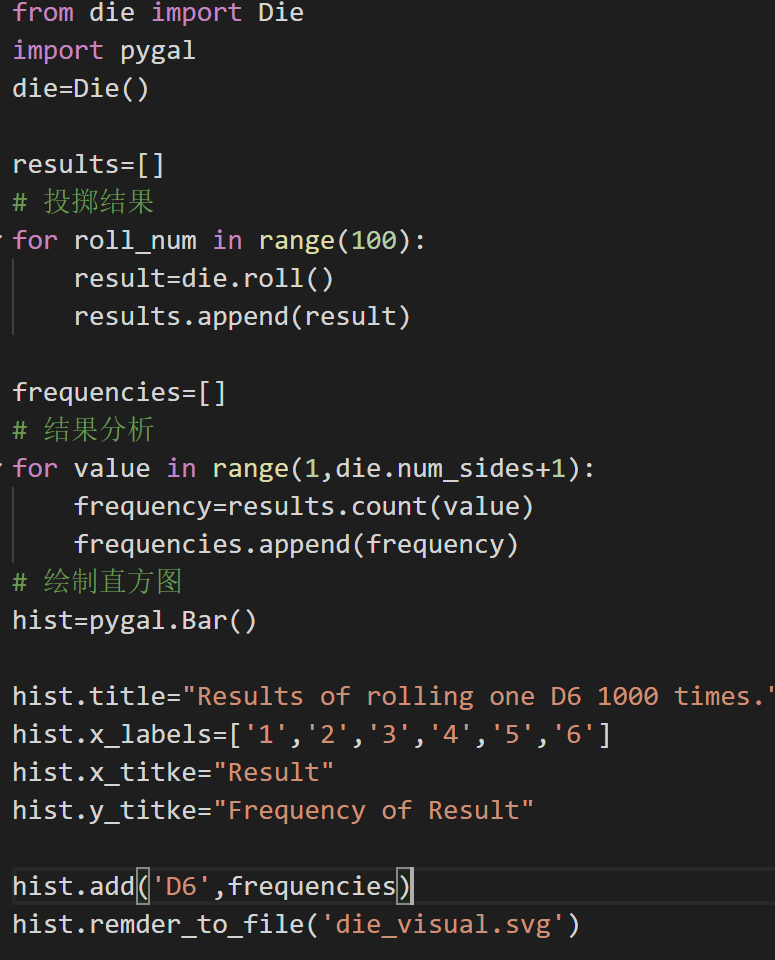
1. 使用Pygal模拟掷骰子

安装Pygal

模拟骰子



掷骰子，分析数据，绘制直方图



还可以同时掷两个骰子，掷两个不同的骰子

1. 下载数据

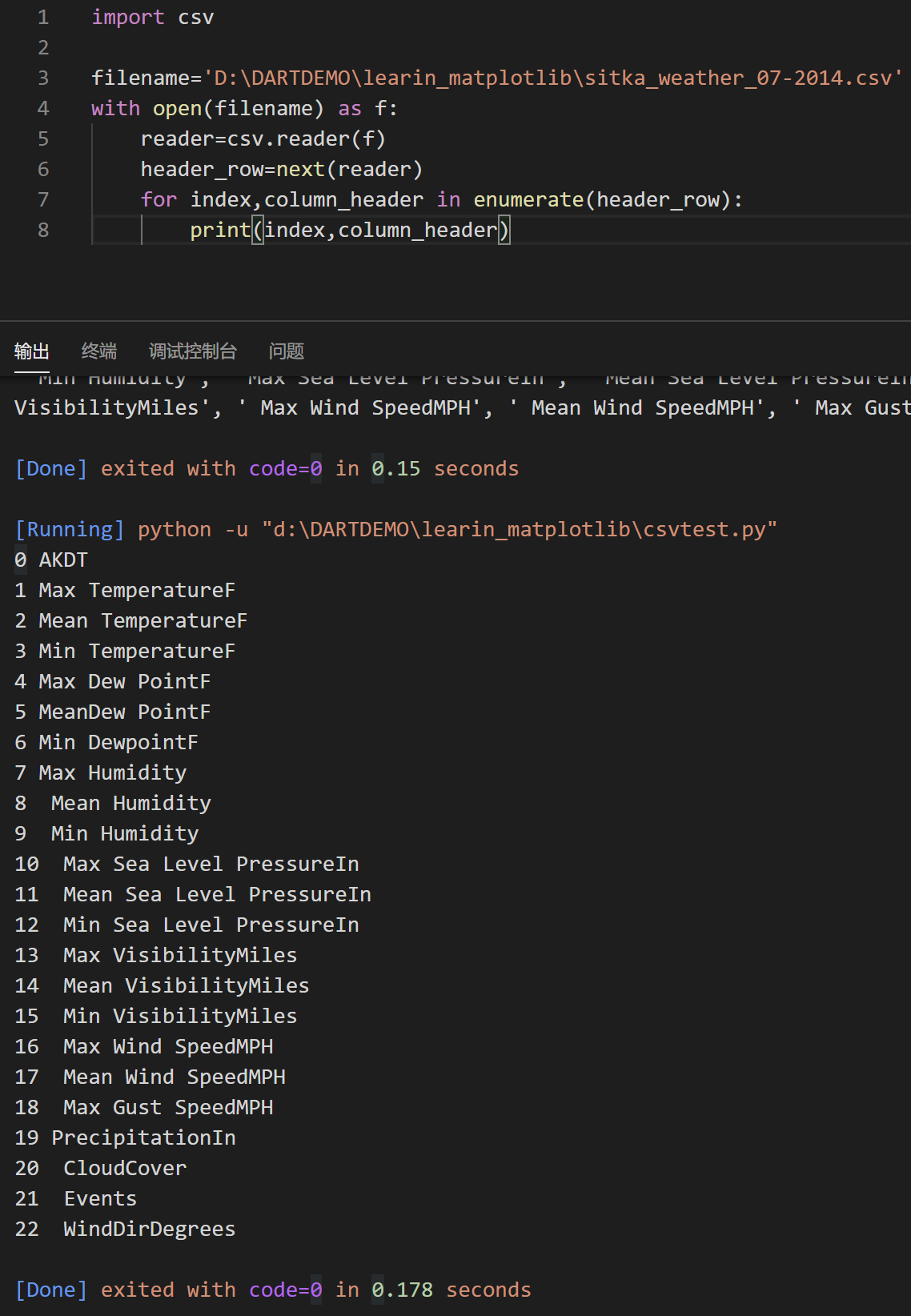
从网上下载数据，并对其进行可视化。两个常见格式存储数据：csv和JSON。

CSV格式：

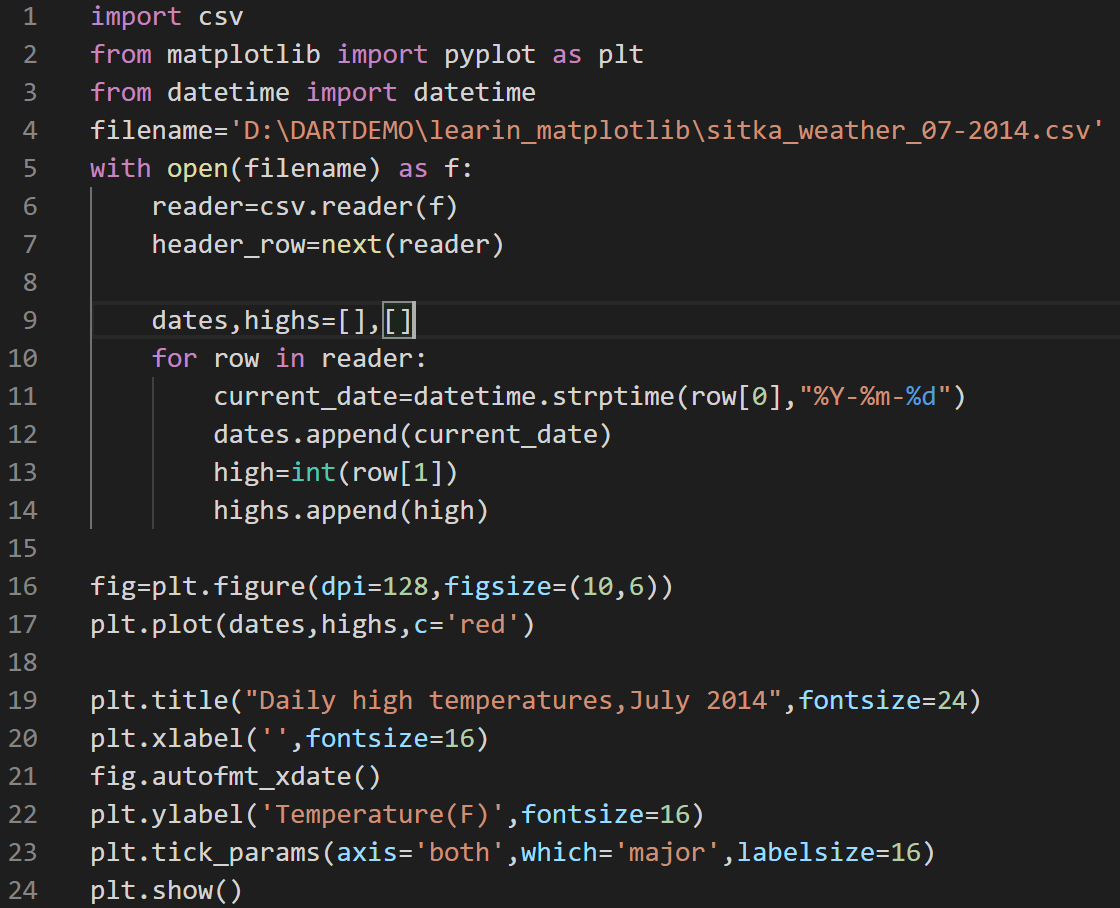
一系列以逗号分隔的值

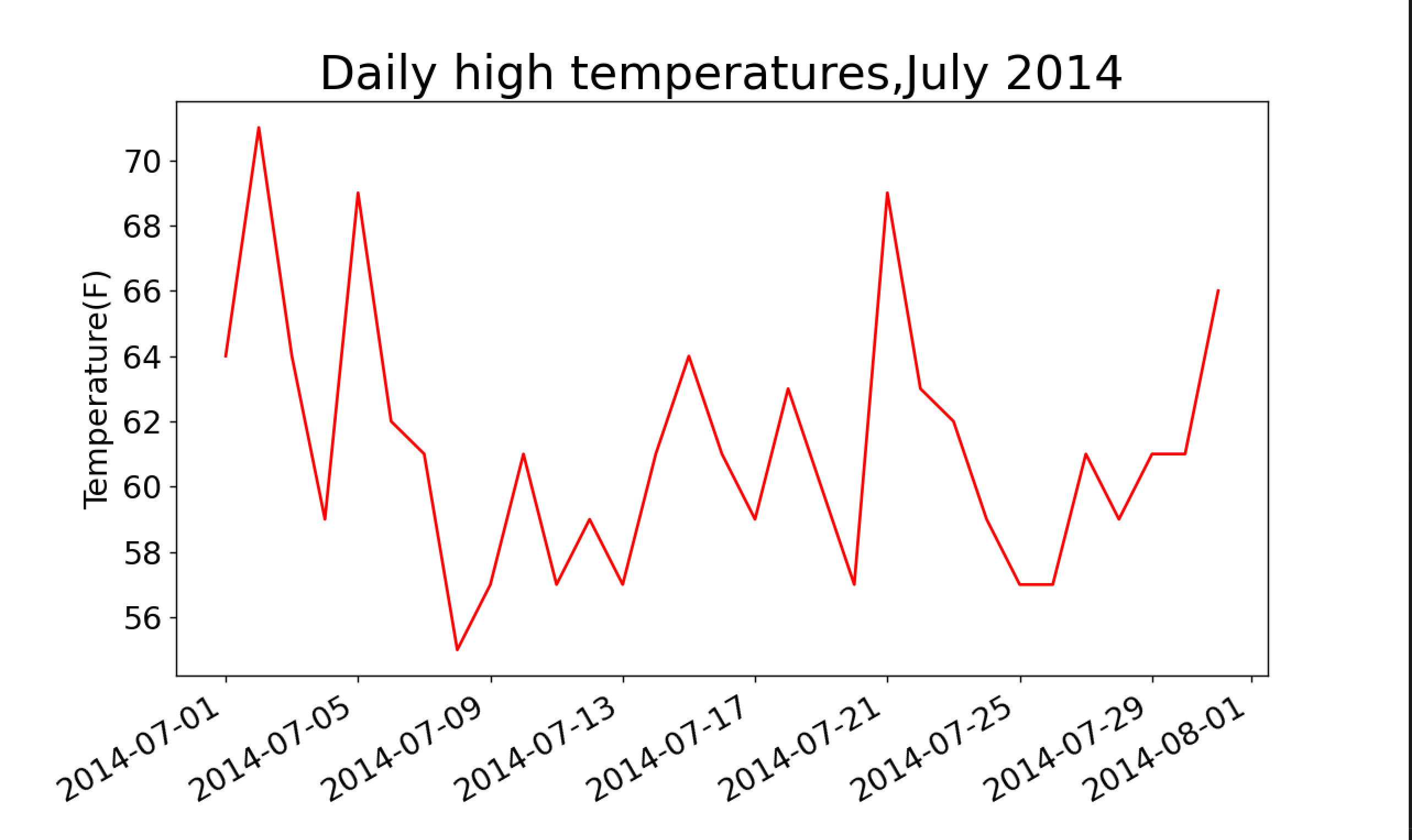
读取csv中的数据

读取每天最高温，将字符串转换为number型，让matplotlib能够读取

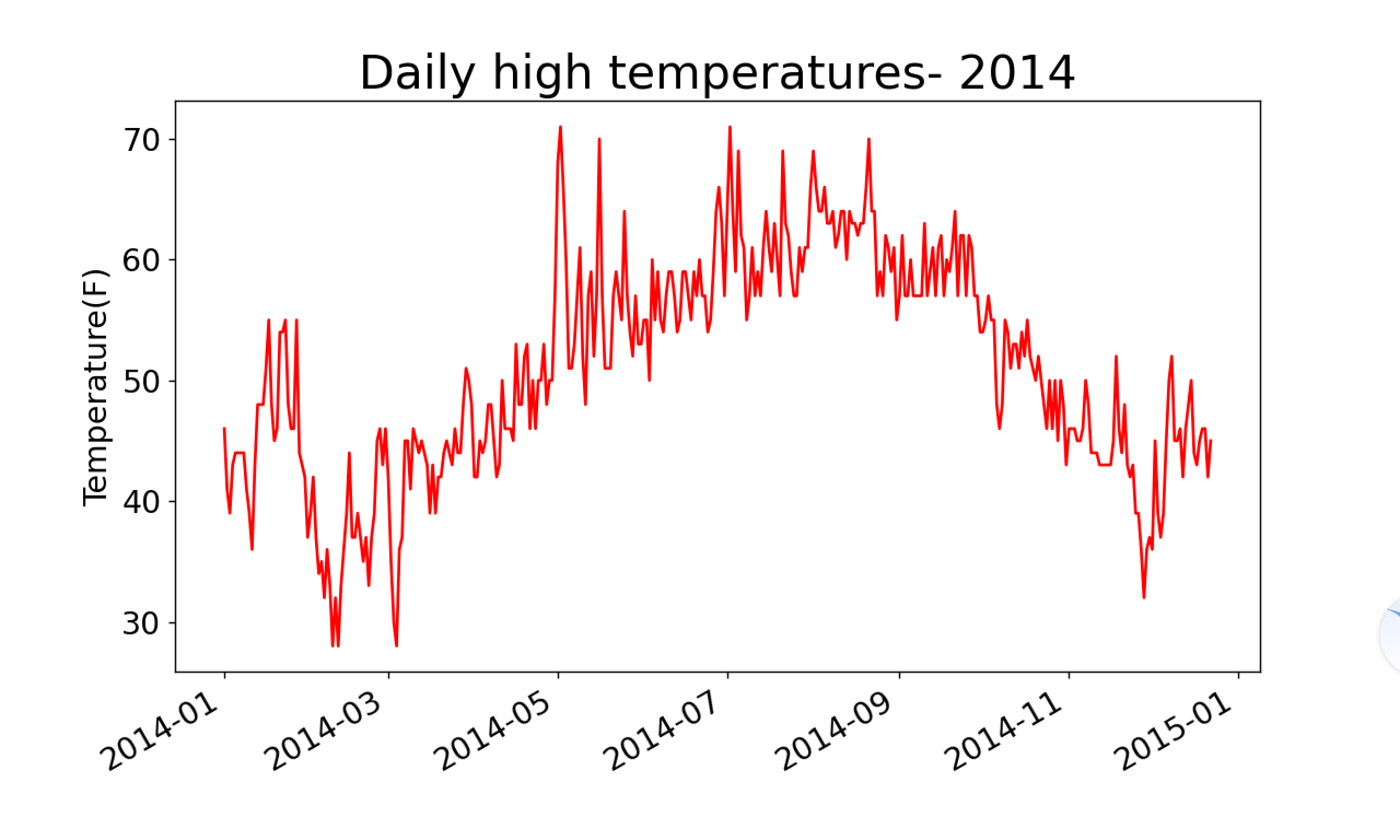


最高气温折线图表示

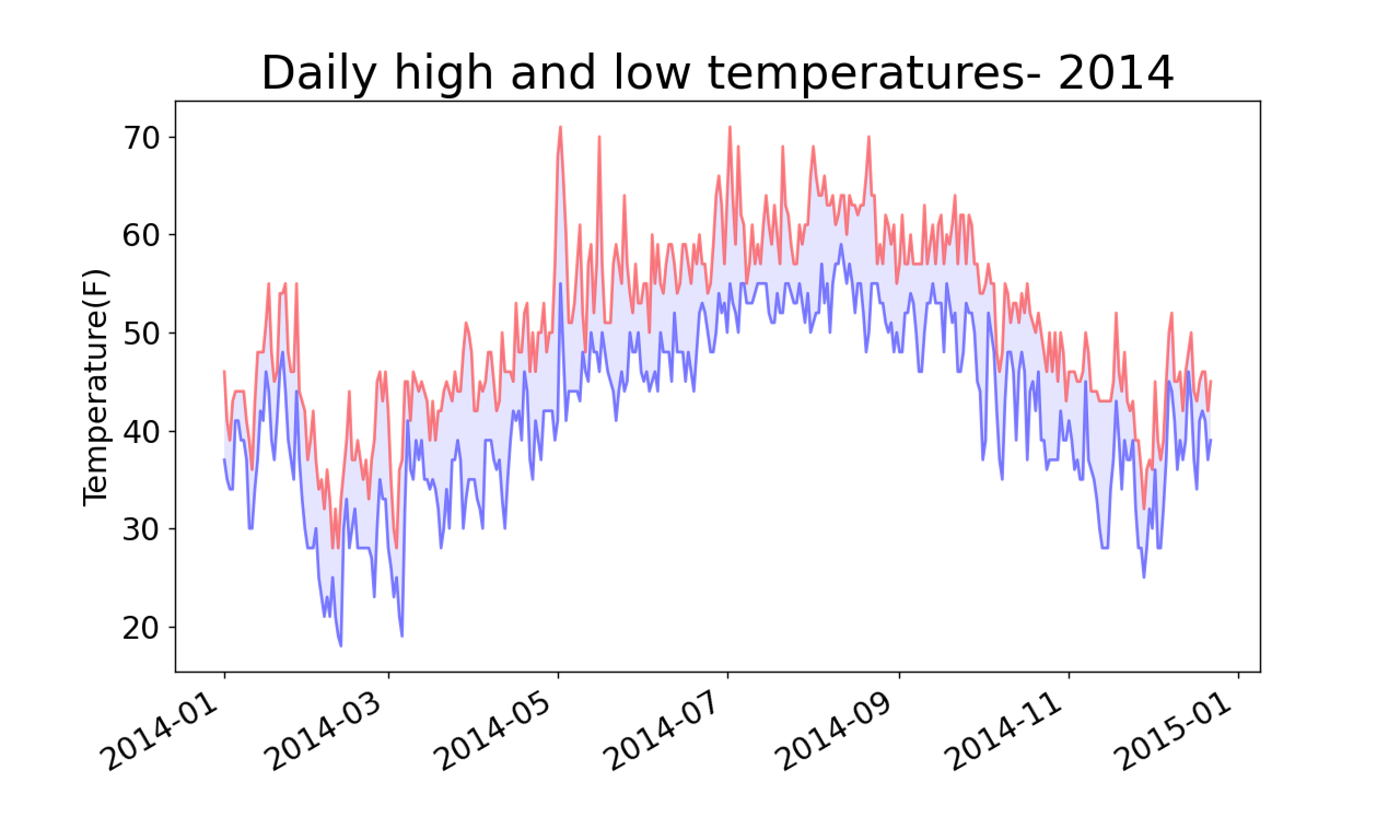




全年



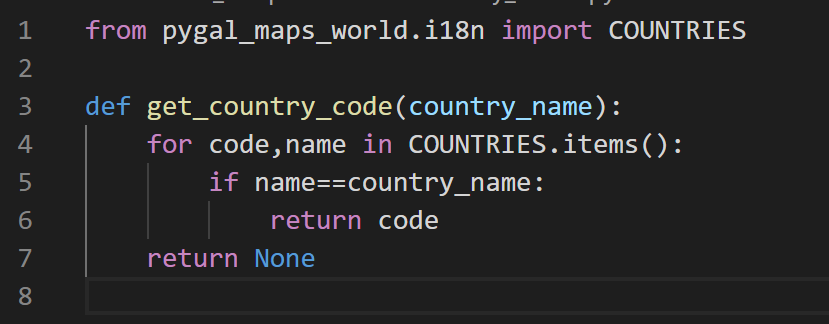
最高温最低温，着色

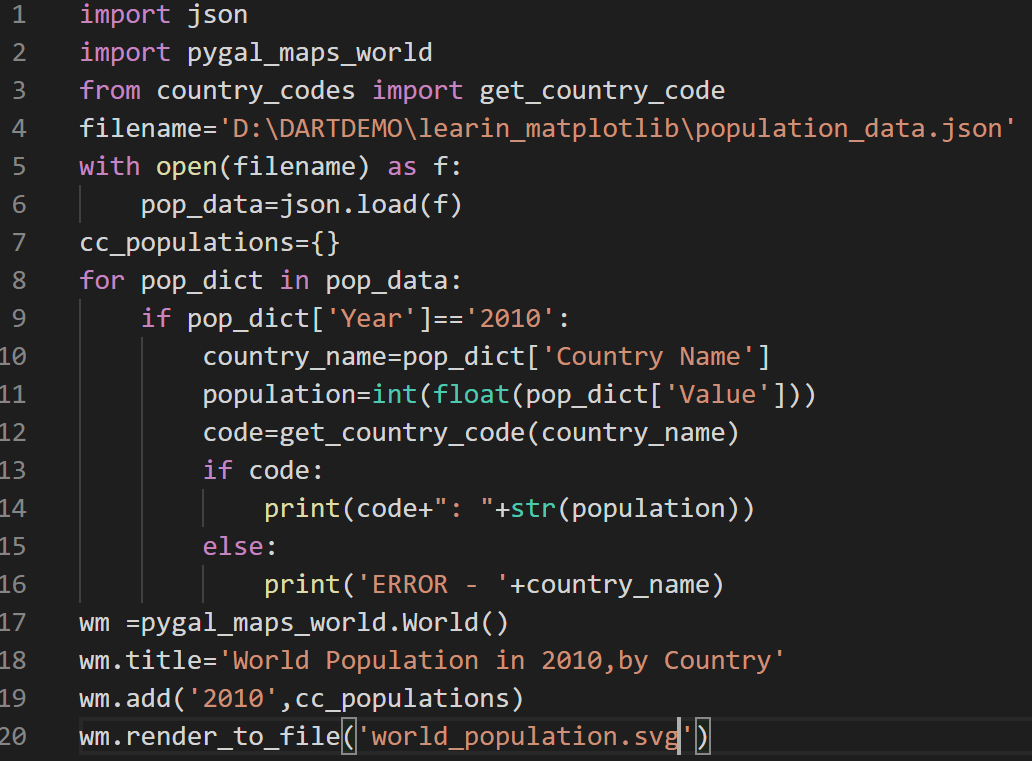


错误检查

JSON格式：

将json中的数据加载到列表中。将字符串转换为数字形式，注意浮点数的转换

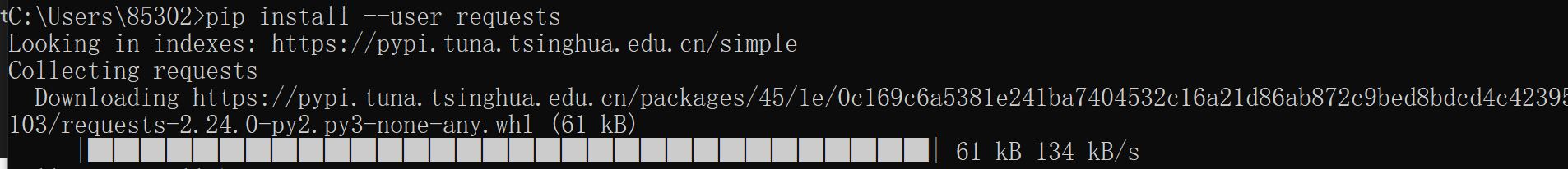




1. 使用API

Web API是网站的一部分，用于与使用非常具体的URL请求特定信息的程序交互。

安装requests



处理API响应，requests.get(url)

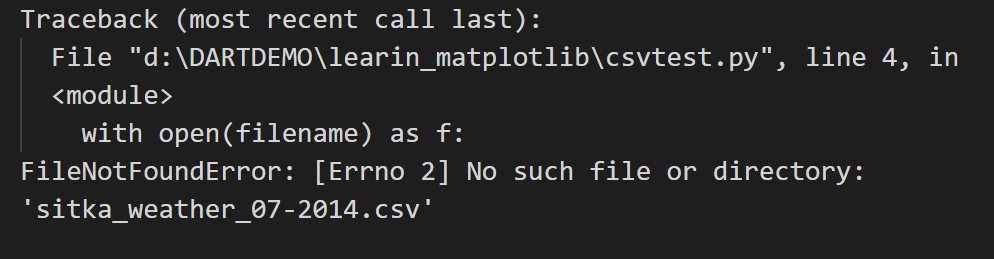
处理响应字典，API调用返回的信息存储到字典中，处理字典中的数据

监视API的速度限制，在浏览器中输入<https://api.github.com/rate_limit>查看

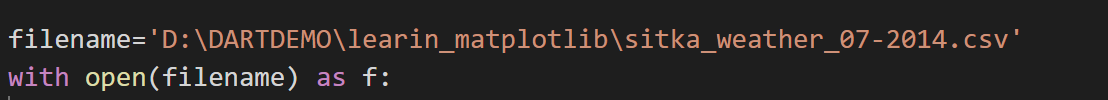
使用pygal可视化仓库，呈现GitHub上python项目受欢迎的程度。

1. 问题和解决方法
2. 运行打开csv文件时，出错



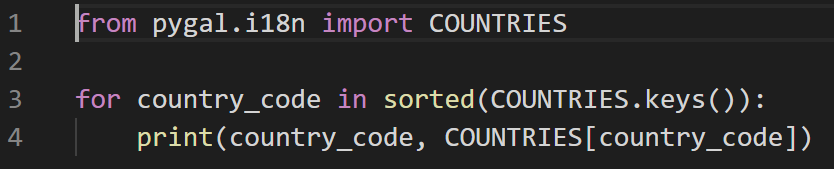


改成

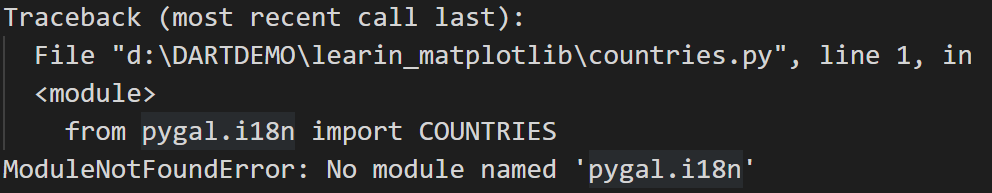


成功运行

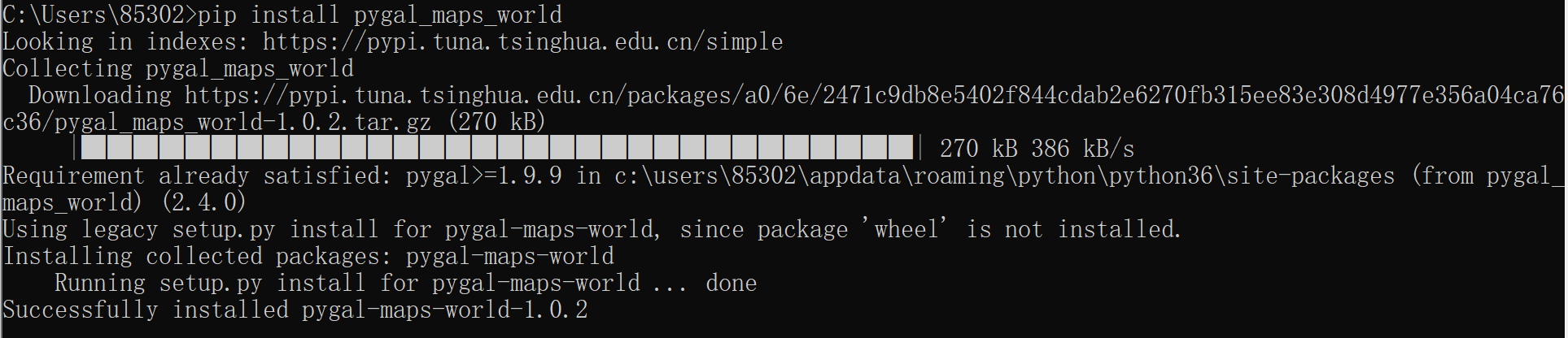
1. Pygal.i18n



报错

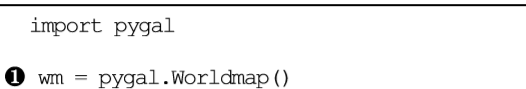


事实上，pygal.i18n已经不存在了，改为了pygal\_maps\_world，需单独通过pip下载



对应的在制作世界地图时

将



改为



2020/6/22

1. Hadoop

Hadoop，一个分布式系统基础架构。实现了分布式文件系统HDFS，存储海量数据集。MapReduce对这些数据集运行分布式计算。

Hadoop最底部是HDFS，上一层为MapReduce引擎。

Hadoop得以在大数据处理应用中广泛应用得益于其自身在数据提取、变形和加载(ETL)方面上的天然优势。Hadoop的分布式架构，将大数据处理引擎尽可能的靠近存储，对例如像ETL这样的批处理操作相对合适，因为类似这样操作的批处理结果可以直接走向存储。Hadoop的MapReduce功能实现了将单个任务打碎，并将碎片任务(Map)发送到多个节点上，之后再以单个数据集的形式加载(Reduce)到数据仓库里

1. 安装
   1. 下载jdk1.8，Linux，64位jdk-8u144-linux-x64.tar.gz。下载Hadoop
   2. 更换主机名，关闭防火墙，时间设置。
   3. 将jdk，hadoop解压包放到home目录下，并新建java，hadoop文件夹

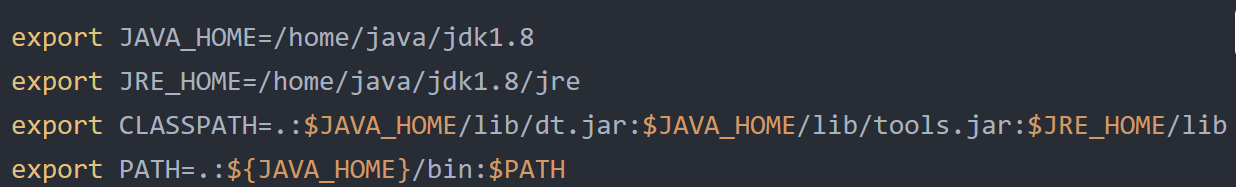
解压文件tar -xvf jdk-8u144-linux-x64.tar.gz

tar -xvf hadoop-3.2.0.tar.gz

并移动到文件夹java，hadoop下，并将文件夹重命名为jdk1.8和hadoop2.8

* 1. jdk环境配置 java -version 查看是否安装了jdk。

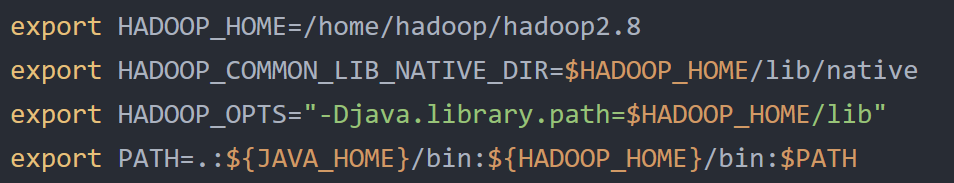
profile文件更改 vim /etc/profile 整体配置文件



source /etc/profile配置生效

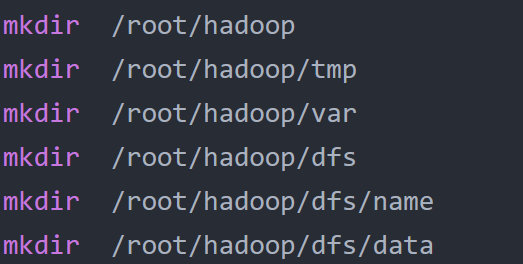
* 1. Hadoop环境配置

Profile文件修改 vim /etc/profile 整体配置文件



source /etc/profile配置生效

新建文件夹，在现在的root目录下建立文件夹



切换到 /home/hadoop/hadoop3.2/etc/hadoop/ 目录下

修改core-site.xml

vim core-site.xml



修改hadoop-env.sh export JAVA\_HOME=/home/java/jdk1.8

修改hdfs-site.xml

修改mapred-site.xml

* 1. 启动Hadoop

初始化，在目录/home/Hadoop/hadoop3.2/bin目录下输入 ./Hadoop namenode -format

启动HDFS

启动YARN

1. 运行wordcount

统计单词出现次数。

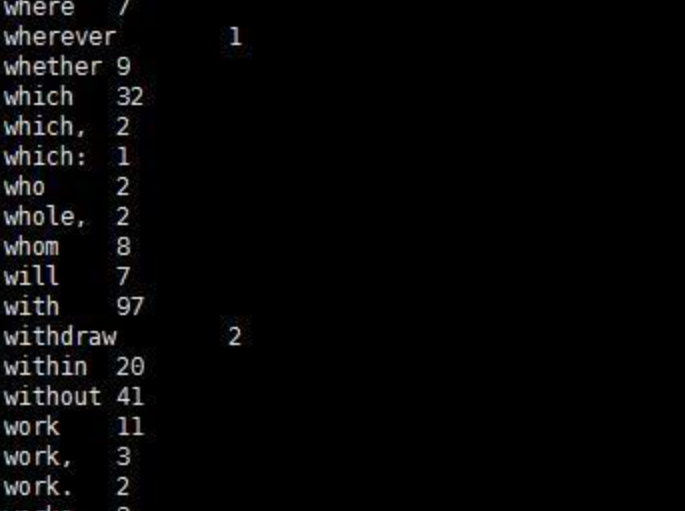
启动hadoop，进入hadoop安装目录cd /home/hadoop/hadoop/ 。查看目录情况$ ll，以文件LICENSE.txt为输入源文件。

在hadoop 目录下，输入命令:$hadoop fs -mkdir /input 创建/input目录。

将输入源文件放入input目录下，$hadoop fs -put LICENSE.txt /input

将/input通过wordcount统计每个单词出现次数。$hadoop jar /home/hadoop/hadoop/share/mapreduce/hadoop-mapreduce-examples-2.7.3.jar wordcount /input /output

打开文件part-r-00000查看结果



1. 问题与解决方法

在虚拟机上安装Hadoop比较困难，主要参考网上教程。

2020/6/23

了解MapReduce和HDFS

Hadoop四大组件：HDFS：分布式存储系统；MapReduce：分布式计算系统；YARN： hadoop 的资源调度系统；Common： 以上三大组件的底层支撑组件，主要提供基础工具包和 RPC 框架等

（一）MapReduce

Mapreduce 是一个分布式运算程序的编程框架，是用户开发“基于 hadoop的数据分析应用”的核心框架。MapReduce是一种编程模型，用于大规模数据集的并行运算。

Mapreduce 核心功能是将用户编写的业务逻辑代码和自带默认组件整合成一个完整的分布式运算程序，并发运行在一个 hadoop 集群上。

将单机版程序扩展到集群来分布式运行，将极大增加程序的复杂度和开发难度。引入 MapReduce 框架后，开发人员可以将绝大部分工作集中在业务逻辑的开发上，而将分布式计算中的复杂性交由框架来处理。

MapReduce 程序的业务编码分为两个大部分，一部分配置程序的运行信息，一部分编写该 MapReduce 程序的业务逻辑，并且业务逻辑的 map 阶段和 reduce 阶段的代码分别继承 Mapper 类和 Reducer 类。

* 1. 用户编写的程序分成三个部分： Mapper， Reducer， Driver(提交运行 MR 程序的客户端)。
  2. Mapper 的输出数据是 KV 对的形式（ KV 的类型可自定义）。
  3. Mapper 中的业务逻辑写在 map()方法中。
  4. map()方法（ maptask 进程）对每一个<K,V>调用一次。
  5. Reducer 的输入数据类型对应 Mapper 的输出数据类型，也是 KV。
  6. Reducer 的业务逻辑写在 reduce()方法中。
  7. Reducetask 进程对每一组相同 k 的<k,v>组调用一次 reduce()方法。
  8. 用户自定义的 Mapper 和 Reducer 都要继承各自的父类。
  9. 整个程序需要一个 Drvier 来进行提交，提交的是一个描述了各种必要信息的 job 对象。

一个完整的 mapreduce 程序在分布式运行时有两类实例进程：

(1) MRAppMaster：负责整个程序的过程调度及状态协调 (该进程在yarn节点上)

(2) Yarnchild：负责 map 阶段的整个数据处理流程。负责 reduce 阶段的整个数据处理流程。

以上两个阶段 maptask 和 reducetask 的进程都是 yarnchild，并不是说这 maptask 和 reducetask 就跑在同一个 yarnchild 进行里（Yarnchild进程在运行该命令的节点上）

maptask 的并行度决定 map 阶段的任务处理并发度，进而影响到整个 job 的处理速度。

1. HDFS

存储非常大的文件，采用流式的数据访问方式，运行于商业硬件上。

* 1. Blocks

物理磁盘中有块的概念，磁盘的物理Block是磁盘操作最小的单元，读写操作均以Block为最小单元，一般为512 Byte。文件系统在物理Block之上抽象了另一层概念，文件系统Block物理磁盘Block的整数倍。通常为几KB。Hadoop提供的df、fsck这类运维工具都是在文件系统的Block级别上进行操作。

HDFS的Block块比一般单机文件系统大得多，默认为128M。HDFS的文件被拆分成block-sized的chunk，chunk作为独立单元存储。比Block小的文件不会占用整个Block，只会占据实际大小。例如， 如果一个文件大小为1M，则在HDFS中只会占用1M的空间，而不是128M。

Block抽象的好处

block的拆分使得单个文件大小可以大于整个磁盘的容量，构成文件的Block可以分布在整个集群，理论上，单个文件可以占据集群中所有机器的磁盘。

Block的抽象也简化了存储系统，对于Block,无需关注其权限，所有者等内容（这些内容都在文件级别上进行控制）。

Block作为容错和高可用机制中的副本单元，即以Block为单位进行复制。

* 1. Namenode & Datanode

整个HDFS集群由Namenode和Datanode构成master-worker（主从）模式。Namenode负责构建命名空间，管理文件的元数据等，而Datanode负责实际存储数据，负责读写工作。

Namenode：

Namenode存放文件系统树及所有文件、目录的元数据。元数据持久化为2种形式：

namespcae image

edit log

争对单点故障的2种解决形式

1）备份持久化元数据

将文件系统的元数据同时写到多个文件系统， 例如同时将元数据写到本地文件系统及NFS。这些备份操作都是同步的、原子的。

2）Secondary Namenode

Secondary节点定期合并主Namenode的namespace image和edit log， 避免edit log过大，通过创建检查点checkpoint来合并。它会维护一个合并后的namespace image副本， 可用于在Namenode完全崩溃时恢复数据。

Datanode：

数据节点负责存储和提取Block，读写请求可能来自namenode，也可能直接来自客户端。数据节点周期性向Namenode汇报自己节点上所存储的Block相关信息。

1. 问题和解决

网络参考博客 https://blog.csdn.net/sjmz30071360/article/details/79877846

2020/6/24

1. Hadoop的I/O操作
   1. 数据完整性：

HDFS会对写入的所有数据计算校验和，并在读取数据时验证校验和。datanode负责在收到数据后存储该数据及其校验和之前对数据进行验证。datanode的管线中最后一个datanode执行校验。HDFS存储着每一个数据块的复本，因此可以通过数据复本来修复锁坏的数据块。可以用hadoop的命令fs -checksum来检查一个文件的校验和，可用于检查HDFS中两个文件是否具有相同的内容。

* 1. 文件压缩

通用的压缩工具是gzip，是否可切分列表示对应的压缩算法是否支持切分，也就是说，是否可以搜索数据流的任意位置并进一步往下读取数据。可切分压缩格式尤其适合MapReduce。

* 1. 序列化

将结构化对象转化为字节流以便在网络上传输或写到磁盘进行永久存储的过程，反序列化是指将字节流转回结构化对象的逆过程。序列化用于分布式数据处理的两大领域：进程间通信和永久存储。

Hadoop使用的是自己的序列化格式Writable，它紧凑、速度快，但不太容易被Java以外的语言进行扩展和使用。因为Writable是Hadoop的核心(大多数MapReduce程序都会为键和值类型使用它)

* 1. Writable封装器

Writable类对所有Java基本类型提供封装，除char类型(可以存储在IntWritable中)，所有的封装包含get()和set()方法用于读取或存储封装的值。 定长格式编码很适合数值在整个值域中分布非常均匀的情况，例如哈希函数。但大多数数值变量的分布都不均匀，一般而言变长格式会更节省空间，变长编码的另一个优势是可以在VIntWritable和VLongWritable转换，所以选择变长格式之后，便有增长的空间，不必一开始就用8字节的long表示。

* 1. Text类型

是针对UTF-8序列的Writable类。一般认为是String类的Writable等价。但String和Text还是有很本质的区别的，比如长度。String的长度是其所含的char编码单元的个数，但Text对象的长度却是其UTF-8编码的字节数。另外，String类的indexof()方法返回char编码单元中的索引位置，Text类的find()方法则返回字节偏移量。

与String相比，Text 的另一个区别在于它是可变的(与所有Hadoop Writable接口实现类似)，可以通过调用其中一个set()方法来重用Text实例。Text并不用有足够的操作String类的API，所以多数情况下要将Text 对象转换成String对象，这一转换可以通过toString()方法来实现，new Text("Hadoop").toString()

NullWritable是Writable的特殊类型，它的序列化长度为0，既不从数据流读取数据，也不向数据流写入数据，它充当占位符。

1. MapReduce应用开发

Api的配置、配置开发环境、编写单元测试、本地运行测试数据

MapReduce的工作流：

JobControl的实例表示一个作业的运行图，可以加入作业的配置，告知JobControl实例作业之间的依赖关系。在一个线程中JobControl将按照依赖顺序执行这些作业。如果一个作业失败，JobControl将不执行与之有依赖关系的后续作业。（JobControl在客户端运行并提交作业）

MapReduce的工作机制，建立在一个YARN的系统上。设置mapreduce.framework.name属性值选择执行的框架。local表示本地的作业运行器，classic表示经典的MapReduce框架（使用一个jobtracker和多个tasktracker），yarn表示新的框架。不同的执行框架表示运行MapReduce程序的不同途径。

YARN：

YARN将Jobtracker的职能划分为多个独立的实体。 Jobtracker负责作业调度和任务进度监视，YARN将这两种角色划分为两个独立的守护进程：管理集群上资源使用的资源管理器和管理集群上运行任务生命周期的应用管理器。基本思路是：应用管理器与资源管理器协商集群的计算资源：容器，在这些容器上运行特定应用程序的进程。容器由集群节点上运行的节点管理器监视。

MapReduce 的输入输出模型

MapReduce 中，reduce函数的输入类型必须与map 函数的输出类型一致，例如map: (k1,v1)->list(k2,v2) reduce:(k2,list(v2))->list(k3,v3)

MapReduce 中的常用设置：

输入数据类型由输入格式(InputFormat)设置

map 输出的key 类型通过setMapOutputKeyClass 设置

MapReduce 输入格式：

MapReduce 处理的输入文件一般存储在HDFS上,这些输入文件的格式多种多样，比如基于行的日志文件、二进制文件等等，而且有些文件非常之大。因此，MapReduce 定义了一些接口来读取这些文件数据。

InputFormat 接口

InputFormat 能够从一个job中得到一个split 集合，再为这个split集合配上一个合适的RecordReader来读取每个split 中的数据。

MapReduce 输出格式：

Hadoop 默认情况下只有一个Reduce，即输出只有一个文件。输出文件的个数有Reduce的个数一致。

OutputFormat 接口：

OutputFormat 主要用于描述数据的格式，它能够将用户提供的key/value 对写入到特定格式的文件中。

Combiner ：

Hadoop 框架使用Mapper 将数据处理成一个<key,value> 键值对，再在网络空间中对其进行整理(shuffile)，然后使用Reducer 处理数据并进行最终输出。

在MapReduce 框架中，Combiner 是为了避免map任务和Reduce 任务之间的无效数据传输而设置的。Hadoop 允许用户针对map任务的输出指定一个合并函数，减少传输到Reducer中的数据量。通过削减mapper 的输出数量，来减少网络带宽和Reducer 上面的负载。

可以将Combiner操作看作是在每个单独的节点上先做一次Reducer操作，其输入和输出的参数和Reducer 是一样的。

Partitioner:

在进行MapReduce 计算时，有时候需要将最终的输出数据分到不同的文件中，比如按照省份、性别等等来进行划分。如果想要得到多个文件，就意味着有同等数量的Reducer 任务在运行。Reducer任务的数据来自于Mapper 任务，也就是说Mapper 任务要划分数据，将不同的数据分配给给不同的Reducer 运行，Mapper 任务划分数据的过程称为Partition,负责划分数据的类为Partitioner。

MapReduce 默认的Partitioner 是HashPartitioner。一般Patitioner 先计算key 的散列值，然后对Reducer 个数进行取模运算。这种方式能随机地将整个key 空间均匀地分配给每个Reducer,同时也能确保不同Mapper 产生的key 能被分配到同一个Reducer。

自定义RecordReader：

RecordReader 表示以怎样的方式从分片中读取一条记录，每读取一条记录都要调用一次RecordReader 类，系统默认的RocordReader 是 LineRecorderReader,它是TextInputFormat 对应的RecordReader。SequenceFileInputFormat 对应的RecordReader 是SequenceFileRecordReader。

LineRecordReader 以每行的偏移量作为读入Map 的Key,每行读入的内容作为Map 的Value。

有时候我们需要自定义RecordReader以我们需要的方式去读取记录，比如以文件名或者行号去读取记录。

自定义RecordReader 的实现步骤：

1、继承抽象类RecordReader，实现RecordReader 的一个实例。

2、实现自定义InputFormat 类，并重写其中的CreateRecordReader()方法，返回值是自定义的RecordReader实例。

3、配置job.setInputFormatClass() 为自定义的InputFormat 实例。

2020/6/25

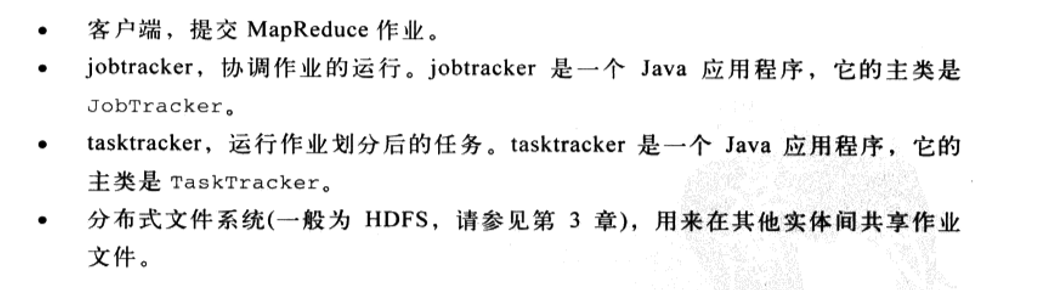
1. mapreduce工作原理
2. 运行mapreduce作业

用JobClient.runJob(conf)运行一个mapreduce作业。

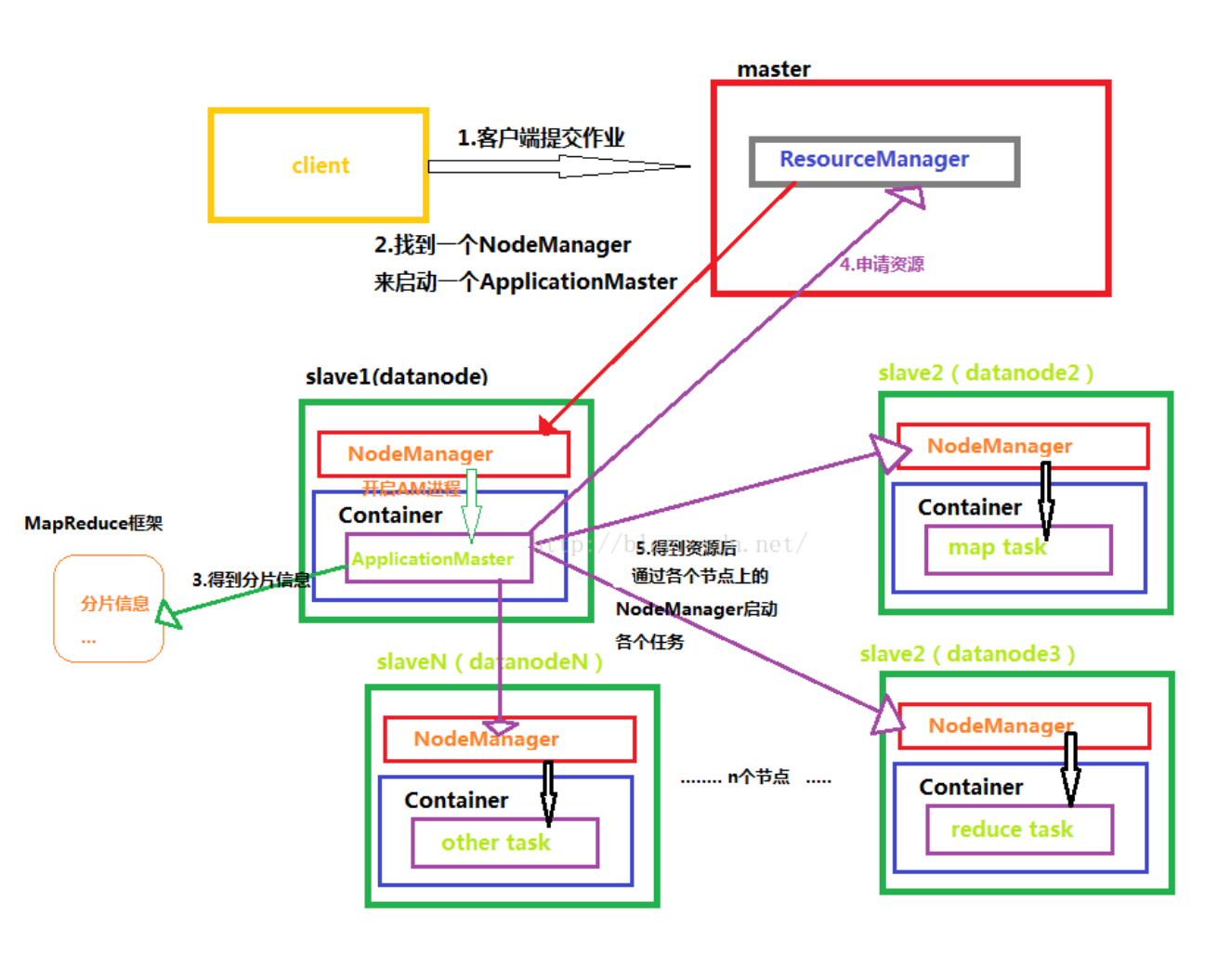


具体过程。

在最上层有4个独立的实体。



* 1. 在客户端启动一个作业。
  2. 向JobTracker请求一个Job ID。
  3. 将运行作业所需要的资源文件复制到HDFS上，包括MapReduce程序打包的JAR文件、配置文件和客户端计算所得的输入划分信息。这些文件都存放在JobTracker专门为该作业创建的文件夹中。文件夹名为该作业的Job ID。JAR文件默认会有10个副本（mapred.submit.replication属性控制）；输入划分信息告诉了JobTracker应该为这个作业启动多少个map任务等信息。
  4. JobTracker接收到作业后，将其放在一个作业队列里，等待作业调度器对其进行调度（这里是不是很像微机中的进程调度呢，呵呵），当作业调度器根据自己的调度算法调度到该作业时，会根据输入划分信息为每个划分创建一个map任务，并将map任务分配给TaskTracker执行。对于map和reduce任务，TaskTracker根据主机核的数量和内存的大小有固定数量的map槽和reduce槽。这里需要强调的是：map任务不是随随便便地分配给某个TaskTracker的，这里有个概念叫：数据本地化（Data-Local）。意思是：将map任务分配给含有该map处理的数据块的TaskTracker上，同时将程序JAR包复制到该TaskTracker上来运行，这叫“运算移动，数据不移动”。而分配reduce任务时并不考虑数据本地化。
  5. TaskTracker每隔一段时间会给JobTracker发送一个心跳，告诉JobTracker它依然在运行，同时心跳中还携带着很多的信息，比如当前map任务完成的进度等信息。当JobTracker收到作业的最后一个任务完成信息时，便把该作业设置成“成功”。当JobClient查询状态时，它将得知任务已完成，便显示一条消息给用户。



1. 失败

1、任务失败

2、Tasktracker失败

3、Jobtracker失败

1. 作业的调度

Hadoop的mapreduce有多种调度器可选，默认的是原始的基于队列的FIFO调度器，还有一种叫公平调度器的多用户调度器Fair Scheduler。该调度器目的是让每个用户公平的共享集群。

1. Shuffle和排序

输入分片（input split）：在进行map计算之前，mapreduce会根据输入文件计算输入分片（input split），每个输入分片（input split）针对一个map任务，输入分片（input split）存储的并非数据本身，而是一个分片长度和一个记录数据的位置的数组，输入分片（input split）往往和hdfs的block（块）关系很密切。

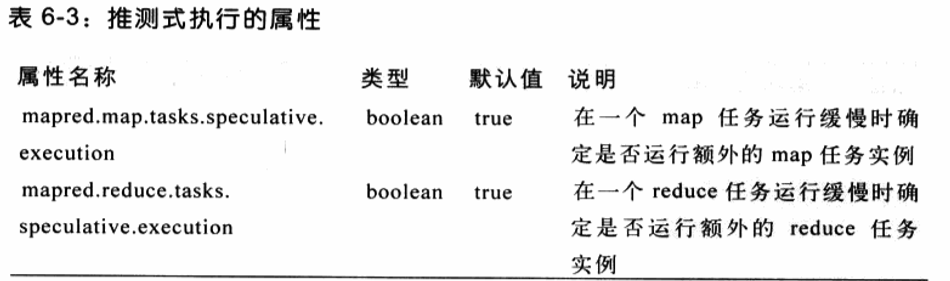
Map端：map函数开始产生输入结果时，利用缓冲写入内存，并处于效率原因预先进行排序。每个map任务都有一个环形内存缓存区。

Reduce端：处理过程的reduce端。Tasktracker需要它为分区文件运行reduce任务。而且，reduce任务需要为其特定分区文件从集群上若干个map任务的map输出。复制阶段，reduce任务有少量复制线程，能够并行地取得map输出。

配置的调整：调整shuffle以提高mapreduce性能。

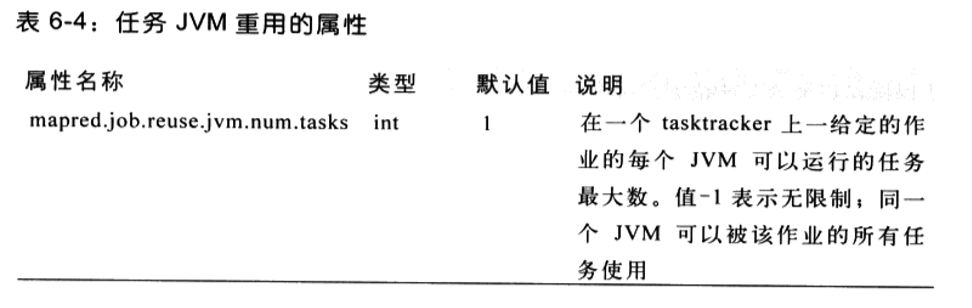
1. 任务的执行

推测式执行，mapreduce模型将作业分割成任务，并行运行任务，使对运行缓慢的任务很敏感。当一个任务运行比预期慢时，它会进行检测，启动另一个相同的任务作为备份。



任务jvm重用

启动jvm重用后，任务并没有同时运行在一个jvm中。Jvm顺序运行任务。

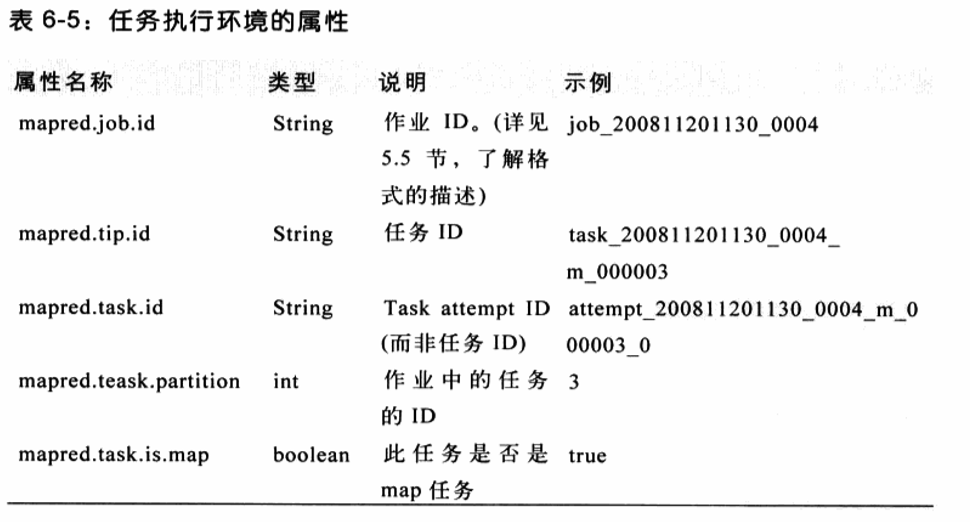


跳过坏记录

跳过损坏、不同格式的记录和缺失的文字。处理错误记录的最佳位置是mapper和reducer代码，忽略或抛出一个异常来取消作业。Bug若存在于第三方的库中无法修改它，使用skipping模式来自动跳过错误记录。

任务执行环境

Hadoop为任务提供运行环境的信息，通过为mapper或reducer提供一个configure方法实现。



1. mapreduce的类型与格式

MapReduce的类型：

map函数的输入的键值对类型(K1,V1)不同于输出类型(K2,V2)，但reduce函数的输入类型必须与map函数的输出类型相同，但reduce函数的输出类型(K3,V3)可以不同于输入类型。如果使用combiner函数，它与reduce函数形式相同，不同之处是它的输入类型是中间的键-值对类型(K2,V2)。combiner函数与reduce函数通常是一样的，在这种情况下，K3与K2类型相同，V3与V2类型相同。patition函数对中间结果的键-值对(K2和V2)进行处理，并且返回一个分区索引。实际上，分区由键单独决定(值被忽略)。

分区数等于作业的reduce任务个数。键的哈希码被转换成一个非负整数，它由哈希值和最大的整型值做一次按位与操作，然后用分区数进行取模操作，来决定该记录属于哪个分区索引。

默认情况下，只有一个reducer，也就是只有一个分区，但在实际的应用中，通常会把reducer数设置为较大的数，否则由于所有的中间数据都会放到一个reduce任务中，作业处理极其低效。目标reducer保持在每个运行5分钟左右，且产生至少一个HDFS块的输出比较合适。

输入格式：

一个输入分片就是一个由单个map操作来处理的输入块，每一个map操作只处理一个输入分片。每个分片被划分为若干个记录，每条记录就是一个键值对，map一个接一个的处理记录。在数据库的场景下，一个输入分片可以对应于一个表上的若干行，而每一条记录对应到一行。(DBInputFormat的输入格式用于从关系数据库中读取数据)

输入分片在Java中表示为InputSplit接口，包含一个以字节为单位的长度和一组存储位置(即一组主机名)，分片并不包含数据本身，而是指向数据的引用，存储位置供MapReduce系统使用以便将map任务尽量放在分片数据附近，而分片大小用来排序分片，以便优先处理最大的分片，从而最小化作业运行时间。InputFomat创建InputSplit，负责创建输入分片并将它们分割成记录。RecordReader像是记录上的迭代器，map任务用一个RecordReader来生成记录的键-值对，然后再传递给map函数。在使用不管是mapper和reducer接口中，迭代器会被反复使用，所以在调用迭代器之间，一定要复制任何需要保留的任何对象。

MapReduce处理数据的最佳速度最好与数据在集群中的传输速度相同，而处理小文件将增加运行作业而必需的寻址次数。在HDFS中集群中存储大量的小文件会浪费namenode的内存。一个可以减小大量小文件的方法是使用顺序文件将这些小文件合并成一个或多个大文件，可以将文件名作为键，文件名作为值，如果HDFS中已经有大量的小文件，CombineFileInputFormat方法比较适合(将多个小文件打包到一个分片中以便每个mapper可以处理更多的数据)