## 1、环境配置

pycharm：已激活

Anaconda 5.2（默认安装numpy,pandas,scipy 下载地址：<https://repo.anaconda.com/archive/>）

Python 3.6.8:已安装

Django 2.0.5

①记得换清华源 -c <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/> ② 打开Anaconda Prompt 或者 anaconda 安装在C盘-管理员打开cmd

（# 永久配置清华conda镜像

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/

conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/）

## 2、思考题

### 2.1 输入浮点，输出int

要点：① 浮点数相等的判断。因为浮点数的存储方式，所以判断相等使用math.isClose()方法or比较差值是否在一定精确度内。

②在python 3.6.8中doctest使用doctest.testmethod()方法。

疑问：① 为什么测试案例中 3.4,和7.8对应的答案那么大

实现：①对于变量x，寻找一个数divisor，使得x\*divisor是一个小数位为0的数字great，

②使用辗转相除法求得great和divisor公约数 c

③ great和divisor分别除以公约数 c，得到的结果即为满足条件的最小数值对。

### 2.2 集合的交集和差集

容器存储使用set，理由是set是一个无序不重复元素集，并有交集、差集等运算符

集合a， b

交集：两个集合中都有的元素。 a&b

差集：从一个集合中剔除出现在另一个集合中的元素。 a-b

### 2.3 日期操作

Python3 日期和时间-菜鸟教程：<https://www.runoob.com/python3/python3-date-time.html>

Pickle 详解： <https://www.cnblogs.com/cangqinglang/p/12563788.html>

输入当前时间，输出上个月的1号日期

输入UTC时间，输出Local时间

输入当前时间，以类似“20180101235513”的字符串格式输出当前时间

输入类似的“20180101235513”的字符串，输出这个字符串所代表的时间

将一个时间值序列化存储一个文件里，再将其用程序读出来（思考为什么要序列化存储）

答：序列化的目标是为了节省时间、精力。①可以通过将程序中运行的对象信息保存到文件中，到达永久存储的目的；通过反序列化操作，可以从文件读取上一次程序保存的对象。例如在模型训练过程中，对原始数据需要做预处理（清洗脏数据、划分测试训练集等），如果不对中间产生的对象信息序列化的话，意味着下次运行时，需要重新运行预处理的代码，既废时间又废精力。②数据结构复杂，数据库等存取不方便。

接上题，将一组这样的时间压缩序列化存储（并了解hadoop上数据的压缩格式有哪些，这些格式有什么对比）

要点：先压缩，再序列化。hadoop数据的压缩格式，并对比

答： ①压缩序列化存储的目标:省空间

关键语句

对象->文件（序列化+压缩）

fill = gzip.open(filename, 'wb')

pickle.dump(data, fill)

fill.close()

文件->对象（解压+反序列化）

fill = gzip.open(filename, 'rb')

while True:

try:

data = pickle.load(data, fill)

except EOFError:

pass

fill.close()

pickle.dump（obj，file，protocol = None，\*，fix\_imports = True ）

fix\_imports为true且protocol小于3，可保存为python2可读的pickle数据流

protocol 整数，指定pickler使用协议版本，支持协议是0-HIGHEST\_PROTOCOL。默认为DEFAULT\_PROTOCOL。若指定为负数，则选择HIGHEST\_PROTOCOL。

协议0与gzip组合效果较优

②hadoop数据的压缩格式

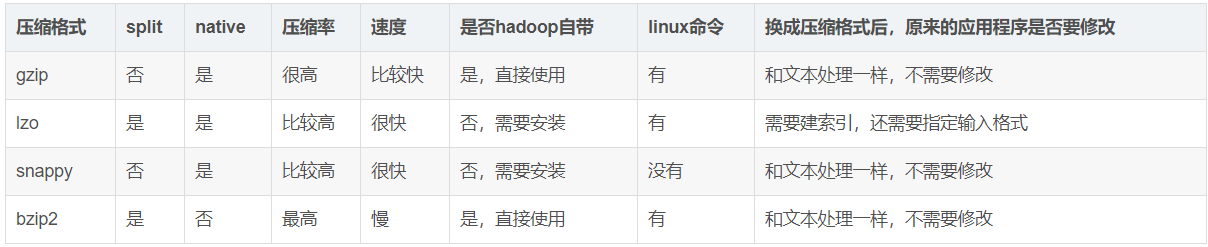
Hadoop 常用存储与压缩格式

<https://www.bbsmax.com/A/o75NKbNdW3/>

<https://blog.csdn.net/lifuxiangcaohui/article/details/52623660>

Hadoop和Spark的异同 <https://zhuanlan.zhihu.com/p/32800581>

hadoop支持的压缩格式:zlib/gzip/bzip2/lzo/lz4/snappy



### 2.4 常用python写法

常用python写法：

将字符串倒序输出

s = "123456789"

print(s[::-1])

交换x,y两个变量的值

a,b = b,a

下面代码第6行的赋值顺序

1 def fib(n):

2 """Compute the nth Fibonacci number, for n >= 2."""

3 pred, curr = 0, 1 # Fibonacci numbers 1 and 2

4 k = 2 # Which Fib number is curr?

5 while k < n:

6 pred, curr = curr, pred + curr

7 k = k + 1

8 return curr

9

10 result = fib(8)

pred = 0, 1, 1, 2, 3, 5, 8

curr = 1, 1, 2, 3, 5, 8, 13

## 3、 可选项

### 3.1 了解闭包原理，以及程序的内存结构，函数的入栈出栈等概念

闭包：

函数作为第一类对象（First-Class Object）

闭包构成条件：

1、函数的内部定义了另一个函数.外部函数和内部函数

2、内部函数里运用了外部函数的临时变量

3、外部函数的返回值是内部函数的引用

闭包使得局部变量在函数外被访问成为可能.

注意：① 闭包无法修改外部函数的局部变量

② 闭包无法直接访问外部函数的局部变量，除非加nonlocal关键字

③ python循环中不包含域的概念

④ 返回闭包中不要引用任何循环变量，或者后续会发生变化的变量。

应用场景：

　 1、装饰器 2、实现单利模式

闭包源码解读

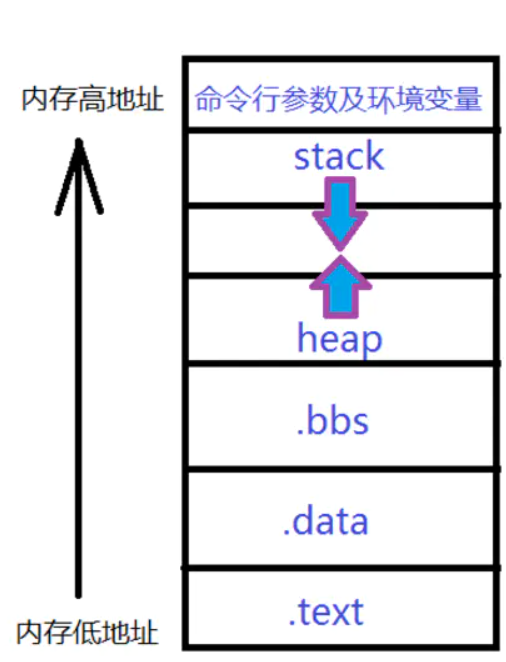
<http://wklken.me/posts/2015/09/04/python-source-closure.html>

闭包概念：

<https://www.cnblogs.com/yssjun/p/9887239.html>

<https://foofish.net/python-closure.html>

程序内存结构：部分见4.3 部分



 text段：

代码段（code segment/text segment）通常是指用来存放程序执行代码的一块内存区域。在代码段中，也有可能包含一些只读的常数变量，例如字符串常量等。

 data段：  
数据段（data segment）通常用来存放程序中已初始化的全局变量数据段属于静态内存分配。

 bss段：  
bss（Block Started by Symbol） 通常用来存放程序中未初始化的全局变量。bss段属于静态内存分配。

 堆（heap）：  
堆用于存放动态变量，它的大小并不固定，可动态扩张或缩减。  
主要由程序员手工分配：  
　　当进程调用malloc等函数分配内存时，新分配的内存就被动态添加到堆上；  
　　当利用free等函数释放内存时，被释放的内存从堆中被剔除。

 栈(stack)：  
存储局部、临时变量，函数调用时，存储函数的返回指针，用于控制函数的调用和返回，在程序块开始时自动分配内存,结束时自动释放内存,主要由编译器自动管理。  
　　在函数被调用时，其参数会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。



进程的内存结构：

<https://www.jianshu.com/p/4e1c751d85f8>

函数的入栈出栈：

<https://blog.csdn.net/varyall/article/details/80517620>

### 3.2 写一个统计函数执行时间的装饰器

装饰器也是基于闭包的一种应用场景，其作用就是为已经存在的对象添加额外的功能。

装饰器：<https://zhuanlan.zhihu.com/p/25648515> 非常细致的装饰器介绍

import logging

import time

def print\_run\_time(func):

def wrapper(\*args, \*\*kwargs):

start = time.clock()

func(\*args, \*\*kwargs)

end = time.clock()

print("{} run time {}".format(func.\_\_name\_\_, end-start))

return wrapper

@print\_run\_time

def foo(name='foo'):

print("i am %s" % name)

print('----')

for i in range(10000000):

pass

foo()

输出：

i am foo

----

foo run time 0.1449036

[Finished in 0.3s]

### 3.3 用objgraph输出如下变量z的内存引用结构：

x = [1, 2, 3]

y = [x, dict(key1=x)]

z = [y, (x, y)]

import objgraph

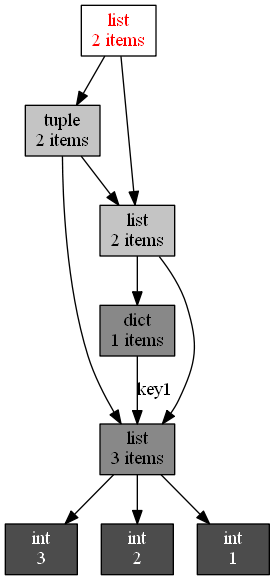
objgraph.show\_refs([z], filename='ref\_topo.png')

提示错误：Image renderer (dot) not found, not doing anything else

情况：Xdot、graphviz已安装

原因：pip3 install graphviz安装的版本为0.14

解决方法：<https://graphviz.gitlab.io/_pages/Download/Download_windows.html> 下载msi格式 安装。安装好后可参考设置生成图片 <https://blog.csdn.net/zhouguoqionghai/article/details/63682774>



### 3.4 了解设计模式中的单例模式

单例模式：

单例是一种设计模式，应用该模式的类只会生成一个实例。

单例模式的主要优点是共享资源和减少资源消耗，主要应用于IO或数据库的线程池,缓存,日志,对话和需共享数据的资源等

单例模式保证了在程序的不同位置都可以且仅可以取到同一个对象实例：如果实例不存在，会创建一个实例；如果已存在就会返回这个实例。因为单例是一个类，所以你也可以为其提供相应的操作方法，以便于对这个实例进行管理。

常用4种单例模式实现方法 <https://zhuanlan.zhihu.com/p/37534850>

<https://www.jianshu.com/p/ec6589e02e2f>

[Python官网Singleton装饰器](https://link.jianshu.com/?t=https://wiki.python.org/moin/PythonDecoratorLibrary#Singleton) <https://wiki.python.org/moin/PythonDecoratorLibrary>

## 4、python基础编程

### 4.1 学习《SICP for python》<http://composingprograms.com/>

1.5.

⑥testing

assert

assert 表达式, "表达式为false时，执行这部分代码"

启动Python解释如何关闭assert：python -O test.py

doctest

（1）测试用例编写: ‘’‘ >>>'''

（2）main() 方法编写 import doctest doctest.testmethod()

1.6

① 函数作为参数

② 函数作为生成方法

③ 函数定义III：嵌套定义（闭包）

④ 函数作为返回值 （函数可以嵌套：f(g(x)) ）

⑤ 示例：牛顿法

⑥ currying: 给定函数f（x，y），可以定义一个函数g，使得g（x）（y）等效于f（x，y）

⑦ Lambda表达式

lambda x : f(g(x)) "A function that takes x and returns f(g(x))"

⑧ 抽象和函数作为第一类对象

⑨ 装饰器

### 4.2 学习datacamp在线课程《Intro to Python for Data Science》<https://www.datacamp.com/courses/intro-to-python-for-data-science>

### 4.3 “python内存管理”：<http://www.cnblogs.com/vamei/p/3232088.html>

对象的内存使用：

查询对象的内存地址：idx()

16进制输出：hex()

Python只缓存整数和短小字符

如何检查两个引用是否指向同一个对象： if a is b:

如何查看某个对象的引用计数：

from sys import getrefcount

getrefcount(对象名)

为什么getrefcount()的结果，会比期望的多1？ 当使用某个引用作为参数，传递给getrefcount()时，参数实际上创建了一个临时的引用。

对象引用对象:

对象可以引用其它对象

引用减少:

对象引用次数减少 ①deel删除某个引用（del也可以删除容器元素） ② 引用被重定向

垃圾回收:

什么时候自动启动垃圾回收？。Python运行时，分配对象(object allocation)和取消分配对象(object deallocation)次数的差值高于某个阈值时。

如何查看阈值：import ge /n print(gc.get\_threshold()) return (700, 10, 10)，700即垃圾回收启动阈值，10,10与分代回收相关

如何设置阈值：set\_threshold()

如何手动启动垃圾回收：gc.collect()

分代回收:

Python将所有的对象分为0，1，2三代。上一返回的(700, 10, 10)中 10,10 表示每10次0代垃圾回收，才进行1次1代的垃圾回收；而每10次1代的垃圾回收，才1次的2代垃圾回收。

同样使用set\_threshold()设置阈值

孤立的引用环:

Python复制每个对象的引用计数，可以记为gc\_ref

from sys import getrefcount

# 对象的内存使用

a = "Python只缓存整数和短小字符"

b = "Python只缓存整数和短小字符"

if a is b: # 检查两个引用是否指向同一个对象

print(getrefcount(a)) # 创建一个临时引用传递给getrefcount()，所以预期：4 结果：5

# 对象引用对象

z = [a, dict(key1=b)]

import objgraph

objgraph.show\_refs([z], filename='ref\_topo.png') # 需要安装xdot和objgraph

# 引用减少

print("引用减少", getrefcount(a))

del a

print("引用减少", getrefcount(a))

# 垃圾回收和分代回收

import ge

print(gc.get\_threshold())

set\_threshold(800, 15, 10)

### 4.4 调试

2.4.1 python -m pdb

设置断点： break或b 继续执行程序：continue或c 查看当前代码段： list 或 l 进入函数：step 或s 执行代码直到从当前函数返回： return 或r 中止并退出： exit或q 执行下一行： next 或n 打印变量值：p （变量名）

help

2.4.2 pycharm Debug程序

Debug进入调试调试：1、设置断点处2、右键 3、任务栏 进入debug模式

断点调试：设置断点->debug模式

F8 单步调试，F7 进入函数 ，shift + F8 跳出函数，F9直接运行到下一断点处。

<https://zhuanlan.zhihu.com/p/62610785>

## 5、框架学习任务

### 5.1 Git学习任务

了解git的安装机常规使用，包括创建仓库，克隆仓库到本地环境，创建本地分支。分支merge。Push、pull、commit代码提交相关命令。Stash变更缓存操作

git安装教程

<http://wiki.mathartsys.com/wiki/dev/%E6%9C%8D%E5%8A%A1%E5%99%A8%E8%BF%90%E7%BB%B4/Git%E5%9C%A8%E9%A1%B9%E7%9B%AE%E5%BC%80%E5%8F%91%E4%B8%AD%E7%9A%84%E5%AE%89%E8%A3%85>

git学习教程 <https://www.liaoxuefeng.com/wiki/896043488029600>

精通git书籍下载： <https://pan.baidu.com/s/13Hcl06fvsfu9G9fsH7U73Q> 密码： kqmq

#### 任务一

本机安装 git软件，将本地机器作为git服务器，创建一个新的项目仓库prj-learning。并克隆到本地目录，创建分支dev，并在dev分支上将第一周的学习工作任务涉及的代码提交到dev分支，并push代码。将dev分支所有提交merge到master分支，并push。

1. git软件 已安装
2. 使用github上的账号作为服务器 ，并创建一个新的项目仓库prj-learning
3. 克隆项目到本地：git clone https://github.com/yeams/prj-learning local-prj-learning
4. 创建并进入分支dev： git checkout –b dev查看当前分支：git branch
5. 提交代码到dev分支，push代码到分支中
   1. 添加到暂存区：一次加一个：git add filename 一次性加全部： git add –all
   2. 添加到分支：git commit –m “XXX”
6. 合并dev分支到master，并push：
   1. 合并指定分支到当前分支： git merge dev 删除分支：git banch –d dev
   2. Push代码到远程仓库：
      1. git push -u origin master
      2. git push -u

要点：

1. git add 只是将 文件 从 工作区 –> 暂存区
2. git commit将 文件 从 暂存区 –> 分支
3. git switch命令(2.23版本以后有的) 可替代 git checkout
4. git 更新：2.17.1之前用git update 2.17.1之后用git update-git-for-windows

问题一：Please make sure you have the correct access rights and the repository exists.

解决方法：git 关联远程仓库 <https://blog.csdn.net/u012702547/article/details/78954832>

### 5.2 Docker 学习任务

学习docker的安装，docker命令的使用，包括docker启动、停止、重启、删除、进入/退出docker环境、docker环境变量参数配置；学习docker compse的使用，包括compose文件的编写，使用compose启动、停止、重启、删除docker，设置docker参数；学习docker镜像的构建，包括docker file的编写，构建自定义镜像

学习参考资料

<http://c.biancheng.net/view/3118.html>

自己电脑安装vmware虚拟机，安装ubuntu。在虚拟机中安装docker服务。

#### 任务一

使用docker命令启动mysql服务，并将mysql root密码设置为learning，mysql的数据目录映射为Ubuntu的/opt/data目录，日志目录映射为Ubuntu的/opt/log目录。并在mysql中创建learning数据库，字符编码utf8mb4

#### 任务二

编写mysql-docker-compose.yml文件，设置与1中相同的mysql启动配置，通过compose文件启动mysql，并能访问到mysql

### 5.3 Python学习任务——Django

django官方教程

<https://docs.djangoproject.com/zh-hans/2.1/>

1）熟悉setting.py urls.py model.py view.py等关键模块

2）熟悉middleware应用

3）熟悉django权限

django-tastypie官方文档

1）modelresource父类函数调用流程，每个函数的作用

2）理解什么样的业务用什么样的函数解决（bulidfilter, get\_object\_list, dispatch\_list, dehydrate, hydrate)

3）Authentication和Authorization

pdb调式程序

logging掌握（formatters, handlers, Logging Level, TimedRotatingFileHandler, MailHandler, logging.exception）

正则表达式的应用

nginx+uwsgi部署应用

#### 任务一

在mysql的learning数据库中，模拟一张商品销量表，，商品ID（模拟1000个商品）、商品卖出时间（建议2014-01-01 00:00:00:00到2015-03-01 00:00:00:00，间隔random(0,10)秒递增）、卖出价格（建议random(20-30)次变一下成交记录价格，价格变动可以随机在random(100-150)）  
用python模拟生成数据，数据总量要求达到百万级

#### 任务二

用django写api实现以下查询:  
查询出某个商品ID的一段时间内（比如：2014-01-01 00:00:00:00到2014-12-01 00:00:00:00）的最高价格、最低价格、平均价格  
查询出某个商品ID每月的销量，销量的峰值在哪个月，销量的低谷在哪个月  
最后用 django + tastypie(restful api) +postman可以查询上述接口

### 5.4 grpc学习任务

掌握GRPC在python中的及安装配置，使用。包括接口文件的编写，代码的生成，pyhton中的grpc server启动，grpc方法的调用。

学习教程

<http://doc.oschina.net/grpc>

#### 任务一

将学习任务3中的django查询过程中的modelresource方法定义为gprc，创建一个新的django项目，直接使用tastypie的接口通过grpc调用上述modelresource方法，实现相同的查询功能

实现参考

<http://gitlab.mathartsys.com/mas-prd/dmp-team/mas-prd-dmp-2nd/issues/416>

## X、需要进一步深入理解

* 3.1 闭包、程序的内存结构、函数的入栈出栈
* 3.2 装饰器
* 3.3 单例模式