[选取日期]

|  |
| --- |
| Python学习笔记 | Wang Jingqi |



|  |  |
| --- | --- |
| MyOwn | PYTHON学习心得 |

[1. 基础 3](#_Toc129156802)

[1.1. 安装python 3](#_Toc129156803)

[1.2. 学习资源 3](#_Toc129156804)

[1.3. Git 3](#_Toc129156805)

[1.4. 缩进 3](#_Toc129156806)

[1.5. 数据结构 3](#_Toc129156807)

[1.6. 函数 5](#_Toc129156808)

[1.7. 控制流 6](#_Toc129156809)

[1.8. 模块 8](#_Toc129156810)

[2. 应用 11](#_Toc129156811)

[2.1. datetime 11](#_Toc129156812)

[2.2. Random 11](#_Toc129156813)

[2.3. CSV 12](#_Toc129156814)

[2.4. sqlite3 13](#_Toc129156815)

[2.5. 安装pip 17](#_Toc129156816)

[2.6. Class 17](#_Toc129156817)

[2.7. Schedule定时方法 18](#_Toc129156818)

[2.8. 收发邮件 18](#_Toc129156819)

[2.9. 分段计税 20](#_Toc129156820)

[2.10. CSV/RE模块应用：部件查询程序 20](#_Toc129156821)

[2.11. Excel应用 21](#_Toc129156822)

[2.12. Plotly/Plotly express 28](#_Toc129156823)

# 基础

## 安装python

访问http: //python.org/downloads/，并单击你要安装的Python版本。

Python --version

如果不能执行简单命令python ，你就需要告诉Windows去哪里查找Python解释器。要确定Python解释器的位置，请打开C盘，并在其中查找名称以Python打头的文件夹（要找到这样的文件夹，你可能需要在Windows资源管理器中的搜索栏中输入单词python）。打开这个文件夹，并查找名称为python（全部小写）的文件。右击这个文件并选择“属性”，你将在“位置：”右边看到它的路径。

打开控制面板并单击“系统和安全”，再单击“系统”。单击“高级系统设置”，在打开的窗口中单击按钮“环境变量”。

在“系统变量”部分，找到并单击变量Path，再单击按钮“编辑”。在出现的对话框中，单击“变量值”，并使用右箭头键滚到最右边。千万不要覆盖变量原来的值，如果你不小心这样做了，单击“取消”按钮，再重复前面的步骤。在变量值的末尾添加一个分号，再添加文件python.exe的路径。

在控制台中输入如下命令，察看python3.5：

Py --version

## 学习资源

http: //stackoverf low.com/

http: //docs.djangoproject.com

http: //reddi t.com/r/learnpython/

http: //reddi t.com/r/learnpython/

## Git

要在Windows系统中安装Gi t，请访问http: //msysgi t.gi thub.io/ ，并单击Download。

## 缩进

放置在一起的语句必须拥有相同的缩进:block

Python始终对block使用 缩进

使用 4个空格来缩进

## 数据结构

### 数据类型

支持数据类型强制转换

a='123'

b=int(a)

list=['01','02','03']

c=list[0]

print(int(c))

list=['1.1','2.2','3.3']

d=float(list[0])

### 列表list:保存一系列有序项目的集合，一张清单，排列顺序可以变化;表达形式：[]

shoplist=['apple','mango','carrot','banana']

print("i have ",len(shoplist),'items to buy')

print('these items are:',end=' ')

for item in shoplist:

print(item,end=' ')

shoplist.append('rice')

shoplist.sort()

shoplist.reverse()

print(shoplist)

shoplist.sort()

print(shoplist)

olditem=shoplist[0]

del shoplist[0]

print("item to del:",olditem)

print(shoplist)

print(new\_zoo[0])

print(new\_zoo[2][2])

len(shoplist)#输出list里的序列数目

zoo=('python','elephant','penguin')

print(len(zoo))

print(zoo)

new\_zoo='mokey','camel','mule',zoo

print(len(new\_zoo))

print(new\_zoo)

### 元组tuple:将多个对象保存在一起，近似看作列表，类似于字符串，不可改变。表达形式：()

a=("abc",20,100)

b=(("abc",20,101),( "abc",20,102),( "abc",20,103))

#按照下标访问元组

print(a[0])

print(a[1])

print(a[2])

print(b[0])

print(b[0][0])

print(b[2][2])

### 元组和列表的转换

可使用list tuple关键字相互转换

a=(1,2,3,4,5)

b=[1,2,3,4,5]

c=list(a)

d=tuple(b)

### 字典dictionary:类似地址簿，有keys（唯一）和values（不可改变的对象）联系在一起。表达形式：{key1:value1,key2:value2};不可被排序。

ab={'sw':'sw@126.com','dk':'dk@126.com','ab':'ab@22.com','cd':'cd@11.com'}

print(ab)

len(ab)#判断长度

print(ab['sw'])

del ab['cd']

print(ab)

for name,address in ab.items():

print("contact {} at {}".format(name,address))

ab['gb']='gb@12.com'

if 'gb' in ab:

print('gb address is:',ab['gb'])

### 集合set

### 运算符

a=True

b=False

print(a or b)

print(a and b)

## 函数

### 内置函数

#### Format()

按照优先顺序替换：print(“{0} is {1}”.format(“jack”,”mark”))

#### Print()

打印

#### Range()

生产序列

for i in range(20):

... print(i)

For I in range(1,20):

Print(i)

list(range(0, 30, 5))

#### Hash()

a=hash(“a”)

print(a)

print(hash(“WJQ”))

### 运算符

+/-/\*//

\*\*：乘方

//：整除

%：取余数

<<:左移 >>:右移

&：按位与 |:按位或

## 控制流

If <condition>:

<Block1>

elif <contition1>:

<block2>

Else:

<block3>

### While

number=23

running=True

while running:

guess=int(input("enter an integer:"))

if guess==number:

print("cong!you got it")

running=False

elif guess<number:

print("a little lower: ")

else:

print("a little higher: ")

else:

print("the while loop is over")

### For:

For x in range(1,5): C++: for (x=0;x<5;x++)

Print(x)

------:1,2,3,4

For x in list(range(5)):

Print(x)

---:0,1,2,3,4

### Break

while True:

s=input("enter sth: ")

if s=='q':

break

print("length of the str is ",len(s))

print("q is input and done")

### Continue: 跳过当前循环块中的剩余语句，并继续该循环的下一次迭 代。

while True:

s=input("enter sth: ")

if s=='q':

break

if len(s)<3:

print("too small")

continue

print("length of the str is ",len(s))

print("q is input and done")

### Def:定义函数

函数定义格式

def function():

print("function1() is printed")

function()

#-----------------

def Max\_ab(a,b):

if a>b:

print('a>b')

elif a==b:

print('a==b')

else:

print("a<b")

Max\_ab(1,3)

函数可使用关键字参数

### 递归函数Resursion

num=10

list1=[]

'''

def func(number):

number=number-1

print("digui:"+str(number))

if number>0:

func(number)

else:

print("number <0")

func(num)

'''

#-----------------------------将列表作为传递参数

def funclist(number,list):

number=number-1

list.append(number)

print("resursion\_"+str(number)+" list:"+str(list))

if number>0:

funclist(number,list)

else:

print("number <0：resursion terminated")

funclist(num,list1)

def function(a,b=5,c=10):

print("a: ",a," b:",b," c:",c)

function(1,3,5)

function(1,c=24)

function(c=30,a=100)

函数可使用可变参数

def total(a=5,\*numbers,\*\*phonebook):

print('a',a)

for single\_item in numbers:

print('single\_item',single\_item)

for first\_part,second\_part in phonebook.items():

print(first\_part,":",second\_part)

print(total(10,1,2,3,4,5,6,7,8,jack=1123,john=2231,inge=1560))

变量的范围

函数体内的变量位局部变量

若需将函数体内的变量声明为全局变量，需在变量前声明:

Global x

DocStrings:文档字符串

def max(x,y):

'''print the maximum of the two numbers.

or other texts'''

if x>y:

return x

elif x==y:

return x

else:

return y

print("max one is ",max(20,3))

print(max.\_\_doc\_\_)

## 模块

编写一个以.py为扩展名的文件，可以重复定义函数

引用已被编译的C语言模块

import sys

print("the commmand line arguments are:")

for i in sys.argv:

print(i)

print("\n\n the pythonpath is:",sys.path,"\n")

### From…import语句

引用一个已被编写保存的.py文档中的函数

Module\_demo.py:

def say\_hi():

print("say hi:this say\_hi() module was referred")

\_\_version\_\_='0.1'

import module\_demo

module\_demo.say\_hi()

print(module\_demo.\_\_version\_\_)

### 包packages

### 内置函数

### Math模块

该模块提供了对C 标准定义的数学函数的访问。

这些函数不适用于复数；如果你需要计算复数，请使用*cmath* 模块中的同名函数。将支持计算复数的函

数区分开的目的，来自于大多数开发者并不愿意像数学家一样需要学习复数的概念。得到一个异常而不是一个复数结果使得开发者能够更早地监测到传递给这些函数的参数中包含复数，进而调查其产生的原因。

该模块提供了以下函数。除非另有明确说明，否则所有返回值均为浮点数。

import math

import cmath

a=2

b=3

c=4

d=10.5

print("math.ceil(d): ",math.ceil(d))

print(math.fabs(-20))

print(math.factorial(10))

print(math.floor(20.5))

#print(math.isqrt(100))

print(math.modf(2.5))

print(math.exp(2))

print(math.log10(100))

print(math.pow(2,10))

print(math.sqrt(100))

math.pi

print(math.pi\*2)

File类

#file

f=open("tax.py",'w')

#w

#r

#x

#a

#b

#t

#+

f.write("something")

for i in range(1,10):

f.write(str(i))

#f.write(char(10))

f.close()

f=open("tax.py",'r')

f.read(10)

f.read()

f.close()

#--------------------------

f=open("file1.py",'w')

for i in range(20):

t="something\_"+str(i)+'\n'

f.write(t)

f.close()

f=open("file1.py",'r')

for i in range(20):

print(f.readline())

f.close()

for line in open("file1.py"):

print(line,end='')

open("file1.py").read()#diaplay all contents

print(open("file1.py").read())#display contents with ‘r’

# 应用

## 安装pip

访问https: //bootstrap.pypa.io/get-pip.py，如果出现对话框，请选择保存文件；如果get-pip.py的代码出现在浏览器中，请将这些代码复制并粘贴到文本编辑器中，再将文件保存为get-pip.py。将get-pip.py保存到计算机中后，你需要以管理员身份运行它，因为pip将在你的系统中安装新包。

Pip镜像地址：

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple module\_name/#例如： Django，wxpython

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple some-package

pip install -i http://pypi.douban.com/simple --trusted-host pypi.douban.com packagename # packagename是要下载的包的名字

pip install -i http://e.pypi.python.org --trusted-host e.pypi.python.org --upgrade pip # 升级pip

在python 3.5中使用pip:

Py –m pip install somepackage

例如，安装wxpython:

Py –m pip install wxpython

### 查看可升级的包

pip list -o

升级脚本:

Pip install somepackage –U # somepakage: package name

安装特定版本：

Pip install somepackage==5.15.4 #安装5.15.4版本的somepackage

### 查看包内的内容

Dir(somepackage)

Dir(csv)

Dir(math)

Dir(math.sin)

## file

### 常规用法

#file

f=open("tax.py",'w')

#参数：

#w

#r

#x

#a

#b

#t

#+

f.write("something")

for i in range(1,10):

f.write(str(i))

#f.write(char(10))

f.close()

f=open("tax.py",'r')

f.read(10)#读取10个字符

f.read()#全部读取

for row in f:

print(row)

for i in range(3):

print(str(i) + ': ' + f.readline(), end='')# 要读取文件中的所有行，并以列表的方式返回它

f.close()

#是迭代其中的行，而不是每个字符

with open(filename) as f:

while True:

line = f.readline()

if not line: break

process(line)

import sys

text=sys.stdin.read("tax.py")

work

### 读每行入列表

f=open("tax.py",'r')

#f.read(10)

#f.read()

content=['']

for row in f:

content.append(row)

print("all appended")

print(content)

for i in range(10):

print("line\_"+str(i)+": "+content[i],end='')

f.close()

### 读取文件并将内容转为数字写入另一个文件

f=open("file\_data.txt",'r')

#f.read(10)

#f.read()

content=[]

for row in f:

content.append(float(row))#将每行内容读取并转化为float类型，添加到列表中

print("all appended")

print(content)

sum=0

for i in range(len(content)):

sum=sum+content[i]

print("sum: "+str(sum))

w=open("txt.bin",'w')

for i in range(len(content)):

w.write(str(content[i])+'\n')#逐行写入文件txt.bin

f.close()

w.close()

'''

import sys

text=sys.stdin.read("tax.py")

#file

f=open("tax.py",'w')

'r' 读取模式（默认值）

'w' 写入模式

'x' 独占写入模式

'a' 附加模式

'b' 二进制模式（与其他模式结合使用）

't' 文本模式（默认值，与其他模式结合使用）

'+' 读写模式（与其他模式结合使用）

f.write("something")

for i in range(1,10):

f.write(str(i))

#f.write(char(10))

f.close()

'''

## StringIO

## datetime

*datetime* 对象是包含来自*date* 对象和*time* 对象的所有信息的单一对象。

与*date* 对象一样，*datetime* 假定当前的格列高利历向前后两个方向无限延伸；与*time* 对象一样，*datetime* 假定每一天恰好有3600\*24 秒。

import datetime

d1=datetime.date(2023,2,20)

y=d1.year

m=d1.month

d=d1.day

print(d1)

print(d1.weekday())

print(d1.isoweekday())

d1.replace(year=2013,month=12,day=2)

print("original date: ",d1)

print("altered date: ",d1.replace(year=2013,month=12,day=2))

print(d1.toordinal())

y1=int(input("input year:"))

m1=int(input("input month:"))

day1=int(input("input day:"))

d2=datetime.date(y1,m1,day1,hour1,minute1,second1)

print(d2)

print(d2.isoformat())

print(d2.ctime())

print("d1:",d1)

print("d2:",d2)

print(d2-d1)

print(abs(d2-d1))

hour1=int(input("input hour:"))

minute1=int(input("input minute:"))

second1=int(input("input second:"))

t1=datetime.time(hour1,minute1,second1)

print(t1)

## Random

该模块实现了各种分布的伪随机数生成器。

import random

print(random.seed())

print(random.random())#返回[0.0, 1.0) 范围内的下一个随机浮点数

print(random.uniform(2,5))#生成一个2到5范围内的数字

print(random.randbytes(20))#生成n 个随机字节。

print(random.randrange(20))#生成20以内的一个整数数字

print(random.randrange(2000,2023,2))

print(random.randrange(20,28,-2))

print(random.randint(a, b))#返回随机整数N 满足a <= N <= b。相当于randrange(a, b+1)。

print(random.getrandbits(k))#返回带有k位随机的Python 整数,即2的k次方为最大数

for i in range(1,20):

print(i,": ",random.randrange(i))

for i in range(1,20):

print(random.randrange(20,28,2))

random.gauss(mu, sigma)#高斯分布。mu 是平均值，sigma 是标准差。

random.normalvariate(mu, sigma)#正态分布。mu 是平均值，sigma 是标准差。

random.lognormvariate(mu, sigma)#对数正态分布。如果你采用这个分布的自然对数，你将得到一个正态分布，平均值为mu 和标准差为sigma 。mu 可以是任何值，sigma 必须大于零。

## timeit

timeit模块可以测试某个python语句或者函数的运行时间

import timeit

def test():

for i in range(10):

print(str(i))

#使用方法1：

t1=timeit.Timer("test()","from \_\_main\_\_ import test")

print("test()函数的执行时间：",t1.timeit(number=1000), "milliseconds")

t1.timeit(number=1000)

#使用方法2：

timeit.timeit('"-".join(str(n) for n in range(10000))', number=10000)#打印执行number次数的语句的时间（单位是秒）

## CSV

是python内嵌的模块,*csv* 模块中的*reader* 类和*writer* 类可用于读写序列化的数据。也可使用*DictReader* 类和*DictWriter* 类以字典的形式读写数据。

常规的使用方法如下：

import csv

c1=open("2.csv")

read=csv.reader(c1)#csv文件的每一行都读取为一个由字符串组成的列表

content=list(read)

content

#获取单行内容

For I in range(20):

Content[i]

#获取所有行内容：

for row in content:

print(str(row[0])+str(row[1])+str(row[2]))

#获取特定行、列的内容

Content[0][0]

Content[21][2]

#使用for循环获取单行内容，避免一次性读取大容量内容占用内存

import csv

file=open(“2.csv”)

read=csv.reader(file)

for row in read:

print(str(row))

print("line"+str(read.line\_num)+":"+str(row))#row是一个turple,可以使用下标row[0]读取

#写入CSV文件

csv\_w=open("3.csv",'w',newline='')#使用newline可去除写入的隔行写入现象

wc=csv.writer(csv\_w)

wc.writerow(['date','follow','unfollow'])

list1=[content[0][0],content[0][1],content[0][2]]

for i in range(20):

list=[content[i][0],content[i][1],content[i][2]]

print(list)

wc.writerow(list)

csv\_w.close()

## RE

内置RE库，用于查询字符串。

import re

str\_des="abcdedfghijk"

str\_obj\_1="hij"

str\_obj\_2="lmn"

#使用re.search()搜索，调用group()后结果返回为一个字符串

#re.search():扫描整个 string 寻找第一个匹配的位置， 并返回一个相应的 匹配对象。如果没有匹配，就返回 None ；

#re.group():返回一个或者多个匹配的子组。如果只有一个参数，结果就是一个字符串，如果有多个参数，结果就是一个元组（每个参数对应一个项），

str\_res\_1=re.search(str\_obj\_1,str\_des)

str\_result\_1=str\_res\_1.group()

print(str\_result\_1)

#若未查询到，则返回None:

str\_res\_2=re.search(str\_obj\_1,str\_des)

if str\_res\_2!=None:

str\_result\_2=str\_res\_2.group()

print(str\_result\_2)

else:

print("found none")

#------------re.match():仅查询目标字符串中的起始字符是否包含，除起始字符外的位置都不能匹配并返回None.如果 string 的 开始位置 能够找到这个正则样式的任意个匹配，就返回一个相应的 匹配对象。如果不匹配，就返回 None

str\_obj\_3="abc"

str\_res\_3=re.match(str\_obj\_3,str\_des)

if str\_res\_3!=None:

str\_result\_3=str\_res\_3.group()

print(str\_result\_3)

else:

print("none matched")

#re.match()仅查询起始字符是否匹配，若匹配则返回查询到的字符串结果，否则返回None

print(re.match("cd","cdedf").group())#返回"cd"

print(re.match("cd","acdedf"))#返回None

print(re.fullmatch("cd","cdedff"))#re.fullmatch():查询两个字符是否完全匹配，否则返回None

print(re.fullmatch("cd","cd").group())#返回"cd"

print(re.sub("cd","c\_d","cdefg"))#:查询"cd"并使用"c\_d"完成替代

re.purge()#清除正则表达式的缓存CSV

## sqlite3

SQLite 是一个C 语言库，它可以提供一种轻量级的基于磁盘的数据库，这种数据库不需要独立的服务器进程，也允许需要使用一种非标准的SQL 查询语言来访问它。一些应用程序可以使用SQLite 作为内部数据存储。可以用它来创建一个应用程序原型，然后再迁移到更大的数据库，比如PostgreSQL 或Oracle。

SQLite，是一款轻型的数据库，是遵守ACID的关系型数据库管理系统，它包含在一 个相对小的C库中。它的功能特点有： 1. ACID事务 2. 零配置 – 无需安装和管理配置 3.储存在单一磁盘文件中的一个完整的数据库 4.数据库文件可以在不同字节顺序的机器间自由的共享 5.支持数据库大小至2TB 6. 足够小, 大致13万行C代码, 4.43M 7. 比一些流行的数据库在大部分普通数据库操作要快 8. 简单, 轻松的API 9. 包含TCL绑定, 同时通过Wrapper支持其他语言的绑定 10. 良好注释的源代码, 并且有着90%以上的测试覆盖率 11. 独立: 没有额外依赖 12. 源码完全的开源, 你可以用于任何用途, 包括出售它 13. 支持多种开发语言，C, C++, PHP, Perl, Java, C#,Python, Ruby等

### 实例程序1

import sqlite3

dbName="sql.db"

table\_name="table\_code0"

con=sqlite3.connect(dbName)

cur=con.cursor()

#initiate table:

#sql='''CREATE TABLE'''+table\_name+'''(\_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT,price REAL,weight REAL)'''

#wrong: sql = '''CREATE TABLE '''+table\_name +''' (\_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT,price REAL,weight REAL)'''

#---------there MUST be a space after"CREATE TABLE":

#right: sql = '''CREATE TABLE '''+table\_name+''' (\_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT,price REAL,weight REAL)'''

'''

sql="CREATE TABLE movie(title,year,score)"

cur.execute(sql)

con.commit()

'''

#-----------------insert records

sql\_insert="INSERT INTO movie VALUES('title\_0',1975,8.2),('title\_1',2023,8.5)"

for i in range(10):

cur.execute(sql\_insert)

con.commit()

#---------------insert many records:

data = [

("Monty Python Live at the Hollywood Bowl", 1982, 7.9),

("Monty Python's The Meaning of Life", 1983, 7.5),

("Monty Python's Life of Brian", 1979, 8.0),

]

cur.executemany("INSERT INTO movie VALUES(?,?,?)",data)

con.commit()

#----------------review data:

re=cur.execute("SELECT score FROM movie")

re.fetchall()

re=cur.execute("SELECT title,year,score FROM movie")

re.fetchall()

res = cur.execute("SELECT name FROM sqlite\_master")

res.fetchone()

sql="SELECT title FROM movie WHERE title='title\_1'"

re=cur.execute(sql)

re.fetchall()

sql="SELECT title,year,score FROM movie ORDER BY score DESC"

re=cur.execute(sql)

re.fetchall()

#---------------close the database

cur.close()

con.close()

### 实例程序2

import sqlite3

import random

dbName="sql.db"

table\_name="table\_code0"

con=sqlite3.connect(dbName)

cur=con.cursor()

for i in range(100):

a="title\_"+str(i)

b=random.randrange(2000,2023,2)

c=random.randrange(5,9)

record=(a,b,c)

sql\_insert="INSERT INTO movie VALUES"+str(record)

cur.execute(str(sql\_insert))

con.commit()

re=cur.execute("SELECT title,year,score FROM movie")

rec=re.fetchall()

rec\_one=re.fetchone();If row\_factory is None, return the next row que. result set as a tuple. Else, pass it to the rowfacto. and return its result. Return None if no more data is available.

rec\_many=re.fetchmany();Return the next set of rows of a que. result as a list. Return an empty list if no more rows areavailable.

#close the database

#cur.close()

#con.close()

### 实例程序3

import sqlite3

import random

dbName="sql\_1.db"

table\_name="movie"

con=sqlite3.connect(dbName)

cur=con.cursor()

sql="CREATE TABLE table\_name(title,year,score)"

cur.execute(sql)

con.commit()

#增加一列：

sql\_add\_column="ALTER TABLE movie ADD COLUMN note "

cur.execute(sql\_add\_column)

#增加表格内记录：

sql\_insert="INSERT INTO movie VALUES('title\_0',1975,8.2,10),('title\_1',2023,8.5,20)"

for i in range(10):

cur.execute(sql\_insert)

#查询所有表格内容：

re=cur.execute("SELECT \* from movie")

rec=re.fetchall()

print(rec)

#获取每行的数据并返回一个列表

#row 是逐行读取rec列表中的行数据

re=cur.execute("SELECT \* from movie")

rec=re.fetchall()

for row in rec:

print(rec)

print("title: "+row[0])

print("year: "+str(row[1]))

print("score: "+str(row[2]))

print("note: "+str(row[3]))

#获取表格名称：

sql\_table="SELECT name FROM sqlite\_master WHERE type='table'"

re=cur.execute(sql\_table)

table=re.fetchall()

print(table)

print(table[0])

#获取表格内所有信息：

sql\_all="SELECT \* from sqlite\_master"

re=cur.execute(sql\_all)

rec=re.fetchall()

print(rec)

#删除表格

sql\_drop\_table="drop table if exists movie"

re=cur.execute(sql\_drop\_table)

rec=re.fetchall()

print(rec)

con.commit()

#更新表格内记录

sql\_update="update movie set year=2000 where title='title\_1'"

cur.execute(sql\_update)

con.commit()

#删除表格内记录

sql\_del="delete from movie where title='title\_0'"

cur.execute(sql\_del)

con.commit()

### 条件筛选

a=str(input("input TS or Crimpint tool('q' to quit): "))

sql="select \* from parts where Description like '%Washer%'"

sql="select \* from tools\_WAM\_E3 where TS like '"+"%"+a+"%'"+" or "+"DESCRIPTION like'"+"%"+a+"%'"+" or "+"CRIMPING\_TOOL like'"+"%"+a+"%'"+" or "+"TOOL\_OUT like'"+"%"+a+"%'"

re=cur.execute(sql)

rec=re.fetchall()

print("there r totally "+str(len(rec))+" items found")

for i in range(len(rec)):

print(e[i])

### 读取CSV并写入数据库

import csv

import sqlite3

#------open and read a csv file--------#

csv\_sql=open("某咖啡公司销售数据.csv")

read=csv.reader(csv\_sql)

content=list(read)

#逐行打印CSC表格内容：

print("CSC表格内容：\n")

for row in content:

print(row)

#------initiate a database

dbName="coffee.db"

tbName="coffee\_sale"

con=sqlite3.connect(dbName)

cur=con.cursor()

#----创建表格tnName

#sqltable="CREATE TABLE csv\_sql(item,name,description)"

#sql\_table="CREATE TABLE "+tbName+"('订单日期', '市场类别', '区域', '产品类别', '产品名称', '预计销售成本', '预计毛利', '预计利润', '预计销售额', '销售成本', '存货', '毛利', '利润额', '销售额')"

#cur.execute(sql\_table)

#con.commit()

#----查询是否创建

sql="select \* from coffee\_sale"

r=cur.execute(sql)

rec=r.fetchall()

rec

#------将CSV文件的内容插入数据库

print("将CSV文件的内容插入数据库:\n")

#-----方法1（适用列数较少）

for i in range(read.line\_num):

#a=content[i][0]

#b=content[i][1]

#c=content[i][2]

#d=(a,b,c)

e=(content[i][0],content[i][1],content[i][2])

sql\_insert="INSERT INTO csv\_sql VALUES"+str(e)

cur.execute(sql\_insert)

con.commit()

#-----方法2（将CSV读取返回的列表转化为元组）

for i in range(len(content)):

sql\_insert="insert into coffee\_sale values "+str(tuple(content[i]))

cur.execute(sql\_insert)

con.commit()

#------方法2（将部分内容插入数据库）

for i in range(1,178):

sql\_insert="insert into coffee\_sale values "+str(tuple(content[i]))

cur.execute(sql\_insert)

con.commit()

#------方法2（将某一行内容插入数据库）

sql\_insert="insert into coffee\_sale values "+str(tuple(content[1]))

cur.execute(sql\_insert)

con.commit()

re=cur.execute("SELECT \* from coffee\_sale order by '订单日期'")

rec=re.fetchall()

print(rec)

#------------查询或删除内容

sql="select \* from coffee\_sale order by '订单日期'"

sql="select \* from coffee\_sale where 订单日期='2010/1/1 0:00'"

sql="delete from coffee\_sale where 订单日期='2010/1/1 0:00'"

re=cur.execute(sql)

for i in range(10):

re.fetchone()

rec=re.fetchall()

print(rec)

#查询数据库中所有表格内容：

count=0#下标计数

for row in rec:

#print("line"+str(read.line\_num)+":"+str(row))#逐行打印

#print("row\_"+str(count)+": "+str(row))#逐行打印

#count=count+1

print("row\_"+str(count)+": "+str(row[0])+"...."+str(row[1])+"...."+str(row[2]))#逐行打印

count=count+1

print("there r "+str(count)+" items in table")#获取数据库表格中的记录数目：

#获取数据库表格中的记录数目：

len(rec)

len(content)

#------关闭CSV文件

csv\_sql.close()

#------关闭数据库

cur.close()

con.close()

'''-----------

#删除所有记录：

sql\_del="delete from csv\_sql where item='\*'"

for i in range(10):

cur.execute(sql\_del)

#删除表格：

sql\_drop\_table="drop table if exists csv\_sql"

re=cur.execute(sql\_drop\_table)

con.commit()

'''

### 读取数据库并写入CSV

import csv

import sqlite3

#------open and read a sqlite database--------#

dbName="coffee.db"

tbName="coffee\_sale"

con=sqlite3.connect(dbName)

cur=con.cursor()

#----读取数据库

sql="select \* from "+tbName

r=cur.execute(sql)

rec=r.fetchall()

#------initiate and open a csv file--------#

csv\_sql=open("coffee\_sale\_csv\_1.csv",'w',newline='')

wc=csv.writer(csv\_sql)

#------将数据库记录插入CSV文件

print("将数据库记录插入CSV文件\n")

for i in range(len(rec)):

wc.writerow(rec[i])

csv\_sql.close()#关闭CSV文件

con.close()#关闭数据库文件

## tkinter

Tkinter是Python的标准GUI库。Python使用Tkinter可以快速地创建GUI应用程序。当然常用的GUI库还有PyQt5，我们只需要知道这两个常用的即可，如果你真的想学习的话。由于Tkinter属于Python标准库，就不需要使用pip安装，直接导入使用即可。

import tkinter

from tkinter import \*

from tkinter import messagebox

root=Tk()## 创建窗口：实例化一个窗口对象。

root.geometry("600x400")#传入的是"宽x高",号是小写的英文字母x,调整窗口的大小；

root.geometry("600x400+100+200")#100,200表示的是窗口顶点，距离电脑左上角的坐标

root.title("my title")

#-----------添加标签控件，并定位

label=Label(root)

#label = Label(root,text="签名",font=("宋体",25),fg="red")

label.grid()

#------------添加输入框，并定位

entry=Entry(root,font=("宋体",25),fg='red')

'''第一个参数传入的就是实例化的那个root窗口对象；第二个参数可写可不写，指的是我们输入的

字体的字体样式和字体大小；第三个参数同样可写可不写，表示的是我们输入的字体的颜色。'''

entry.grid()#只有调用grid()方法，定位后，才会真正显示这个输入框.

#entry.grid(row=0,column=1)#row=0,column=1表示我们将输入框控件，放在第1行第2列的位

#-----------添加点击按钮

button=Button(root,text='签名设计',font=('宋体',25),fg='blue')

button.grid(row=1,column=1)

def function():

print("print sth")

button\_fuc=Button(root,text='button\_func',font=('宋体',25),fg='blue',command=fuction)

button\_fuc.grid(row2,column=1)

root.mainloop()## 显示窗口

## Turtle

### 基本配置

import turtle

t=turtle.Turtle()

#------------移动方向

t.right(dist)#t.rt(d)

t.forward(dist)#t.fd(d) 从当前位置向前距离

t.left(dist)#t.lt(d) 从当前位置向左偏转角度

t.backward(dist)#t.bk(d)

turtle.bgpic("079 Summer Games.png") # ⽤图⽚做背景

t.circle(radius)

t.circle(radius,arc)# 语法格式：circle(半径,⻆度)

t.dot(radius,color)

t.hideturtle()#隐藏画笔箭头

t.setx()#设置海⻳的x 坐标， y 坐标不变

t.setheading()#seth()设置朝向

t.home()

t.goto(0,0)

t.setpos(a,b)#设置画笔位置为(a,b)坐标位置

#--------------判断当前位置

a=t.pos()

if t.pos()>=(100,20)#直角坐标区域分布

### 改变设置

turtle.bgclolor('blue')#'red''green',or 输入相应的十六进制码即可，如# 2 8 5 0 7 8

turtle.title("my title")

t.shapsize(0,1,2)#拉伸长度(Stretch length),拉伸宽度(Stretch width) 轮廓宽度(Outline width)

#----------------Changing the Pen Size

t.pensize(5)

t.forward(100)

#----------------Changing the Turtle and Pen Color

t.shapesize(3,3,3)

t.fillcolor('red')

t.pencolor('green')

t.color('green','red')#第一个颜色参数是画笔的颜色，第二个颜色参数是指填充的颜色

#---------------Filling in an Image

#当您使用t.begin\_fill()时，你是在告诉计算机程序，你将绘制一个需要填充的封闭形状。然后，使用t . e n d \_ f i l l ( )表示你已经完成了形状的创建，然后告诉计算机你可以填充它了。

t.begin\_fill()

t.fd(100)

t.lt(120)

t.fd(100)

t.lt(120)

t.fd(100)

t.end\_fill()

#----------------Changing the Turtle Shape

t.shape("turtle")#'square''triangle''classic'

t.shape("arrow")

t.shape("circle")

#---------------Changing the Pen Speed

t.speed(1)#从0 (最慢的速度)到1 0 (最高的速度)的任何数字 1 最慢， 1 0 快， 0 最快

t.forward(100)

t.speed(10)

t.forward(100)

#---------------Picking the Pen Up and Down

t.penup()#不画

t.pendown()#开始画

t.undo()#你运行的最后一个命令撤消

t.clear()#清理当前的屏幕

t.reset()#屏幕将被清除，海龟的设置将全部恢复到默认参数#会清理当前的屏幕

t.stamp()#即在当前位置印⼀个海⻳的图形.返回⼀个stamp\_id 。

t.clearstamp(stampid)#删除一个特定的戳

c=t.clone()#复制多个海龟：

c = t.clone()

t.color("magenta")

c.color("red")

t.circle(100)

c.circle(60)

### 控制

#----------定时

turtle.ontimer(function,t=1000)#1000ms后执行fuction

#----------事件

screen.onkey(function(), "Up")

screen.listen()

turtle.onkeyrelease(fun, key)#• – a function with – a string: key (e.g. “a”) or key-symbol (e.g. “space”)

turtle.onkeypress(fun, key=None)

turtle.onclick(fun, btn=1, add=None)

turtle.onscreenclick(function(), btn=1, add=None)

#• – number of the mouse-button, defaults to 1 (left mouse button)

#• – True or False – if True, a new binding will be added, otherwise it will replace a

former binding

常用函数总结

#---------------- Using events

onclick()

onrelease()

begin\_poly()

end\_poly()

get\_poly()

clone()

getturtle() | getpen()

getscreen()

setundobuffer()

undobufferentries()

#---------------- Using events

#----------------Animation control

delay()

tracer()

update()

#----------Using screen events

listen()

onkey() | onkeyrelease()

onkeypress()

onclick() | onscreenclick()

ontimer()

mainloop() | done()

#------------Input methods

textinput()

numinput()

setup()

title()

### example

from turtle import \*

color('red', 'yellow')

begin\_fill()

while True:

forward(200)

left(170)

if abs(pos()) < 1:

break

end\_fill()

## Class

### 创建一个类，默认创建了一个与类同名的方法method,有额外的self变量。

class Person:

def sayhi(self):

print("hello,hi")

p=Person()

p.sayhi()

### \_\_init\_\_方法：在类的对象被实例化instantiated时立即运行。

## Schedule定时方法

import schedule

import time

def morning():

print("good morning")

def noon():

print("good noon")

def evening():

print("good evening")

def task1():

print("task1 was initiated")

def task2():

print("task2")

def task3():

print("task3")

evening()

schedule.every().day.at("23:18").do(task1)

schedule.every().day.at("23:19").do(task2)

schedule.every().day.at("23:20").do(task3)

schedule.every(1).minutes.do(morning)

schedule.every(30).seconds.do(noon)

while True:

schedule.run\_pending()

time.sleep(1)

## getpass

可以实现密码的隐藏输入

import getpass

flag=input("input 'q' to quit:")

while flag!='q':

user=getpass.getuser()

password=getpass.getpass()

print("user:"+user)

print("password: "+password)

flag=input("input 'q' to quit:")

## Yagmail收发邮件

POP3服务器: pop.126.com

SMTP服务器: smtp.126.com

IMAP服务器: imap.126.com

import yagmail

yagmail.register("dake124@126.com","XXXXXXX")

yag=yagmail.SMTP(user="dake124@126.com",host="smtp.126.com")

contents=["the 1st test mail"]

#yag.send("dake124@126.com","new 1st test mail",contents)

contents\_01=[

'邮件炸弹搞死你\n'

'炸死你个破邮箱\n'

'<a href="https://www.baidu.com">点击就要命</a>'

'd:\\1.rar'

]

import schedule

import time

def sendmails():

#yag.send("mailbox","subject",content)

yag.send("dake1248@126.com", "mail bombs", contents\_01)

print("send a mail fo fish every 5 seconds")

schedule.every(5).seconds.do(sendmails)

while True:

schedule.run\_pending()

time.sleep(1)

'''

from imbox import Imbox

import keyring

pwd=keyring.get\_password("yagmail", "dake124@126.com")

with Imbox("imap.126.com", "dake124@126.com", pwd, ssl=True) as imbox:

all\_inbox\_messages=imbox.messages()

for uid, message in all\_inbox\_messages:

print(message.subject)

#print(message.body["plain"])

'''

## 分段计税

def tax(income):

if income>10000:

pay=3000\*0.005+2000\*0.01+5000\*0.015+(income-10000)\*0.02

return pay

else:

if income>5000:

pay=3000\*0.005+2000\*0.01+(income-5000)\*0.015

return pay

else:

if income>3000:

pay=3000\*0.005+(income-3000)\*0.01

return pay

else:

pay=income\*0.005

return pay

while True:

inc=float(input("input your income:"))

print("your pay("+str(inc)+") for tax is : "+str(tax(inc)))

## os.path

使用os.path库操作目录

import os

import os.path

os.renames(old, new)#递归重命名目录或文件。

a=os.path.getatime("d:\MISC")#返回path 的最后访问时间。返回值是一个浮点数，为纪元秒数（参见time 模块）。

a=os.path.getmtime("d:\MISC")#返回path 的最后修改时间。返回值是一个浮点数，为纪元秒数（参见time 模块）

a=os.path.getsize(path)#返回path 的大小，以字节为单位

a=os.path.isabs(path)#如果path 是一个绝对路径，则返回True。

os.path.exists(path)#如果path 指向一个已存在的路径或已打开的文件描述符，返回True。

os.path.lexists(path)#如果path 指向一个已存在的路径，返回True。对于失效的符号链接，也返回True。

os.path.isdir(path)#如果path 是现有的目录，则返回True。

os.path.samefile(path1, path2)#如果两个路径都指向相同的文件或目录，则返回True。

print(a)

## CSV/RE模块应用：部件查询程序

应用CSV、RE模块，实现对CSV文件的读取、查询、结果显示的功能。

### REV.0程序：

#----------------------------------------rev.0----------------------------------------------------

#第一个版本，单次查询并返回全部结果

import csv

import re

file=open("3.csv")

read=csv.reader(file)

content=list(read)

#print("提示：(1)退出输入'q' (2)通配符是'.'“）

input1=input("请输入件号或关键字描述(退出输入'q'):")

a=str(input1).upper()

print("\r")

print(content[0][0]+" "+content[0][6]+" "+content[0][1]+" "+content[0][2]+" "+content[0][3]+" "+content[0][5]+" "+content[0][4])

count=0

while input1!="q":

for i in range(89304):

if re.search(a,str(content[i][0]).upper()) or re.search(a,str(content[i][1]).upper()) or re.search(a,str(content[i][2]).upper()) or re.search(a,str(content[i][3]).upper())!=None:

print(content[i][0]+" "+content[i][6]+" "+content[i][1]+" "+content[i][2]+" "+content[i][3]+" "+content[i][5]+" "+content[i][4])

count=count+1

#-----------print the item quantity:

if(count==0):

print("none has been searched")

else:

print(str(count)+" items searched")

#---------repeat the loop:

count=0

input1=input("请输入件号或关键字描述(退出输入'q'):")

a=str(input1).upper()

print(content[0][0]+" "+content[0][6]+" "+content[0][1]+" "+content[0][2]+" "+content[0][3]+" "+content[0][5]+" "+content[0][4])

re.purge()

print("thanks,88")

### REV.1程序：

#------------------------------------------rev.1-----------------------------------------

#单次查询后显示前10条结果

#询问是否全部显示

#询问是否进一步查找

#输入进一步查找的条件

#显示二次查询的全部结果

import csv

import re

file=open("3.csv")

read=csv.reader(file)

content=list(read)

print("Note: 退出输入'q';通配符是'.' \r")

input1=input("请输入件号或关键字描述(退出输入'q'):")

a=str(input1).upper()

list\_temp=[]

#print(content[0][0]+" "+content[0][6]+" "+content[0][1]+" "+content[0][2]+" "+content[0][3]+" "+content[0][5]+" "+content[0][4])

#print("TS "+"合同 "+"描述 "+"尺寸 "+"图纸")

count=0

count\_temp=0

while input1!="q":

for i in range(89304):

if re.search(a,str(content[i][0]).upper()) or re.search(a,str(content[i][1]).upper()) or re.search(a,str(content[i][2]).upper()) or re.search(a,str(content[i][3]).upper())!=None:

list\_temp.append(content[i][0]+" "+content[i][6]+" "+content[i][1]+" "+content[i][2]+" "+content[i][5]+" "+content[i][3])#+" "+content[i][4])

#print(content[i][0]+" "+content[i][6]+" "+content[i][1]+" "+content[i][2]+" "+content[i][5]+" "+content[i][3])#+" "+content[i][4])

count=count+1

#-----------print the item quantity:

if(count==0):

print("none has been found")

else:

#for i in range(0,count):

# print(list\_temp[i])

print("共发现 "+str(count)+" 条信息")

print("TS "+"合同 "+"描述 "+"尺寸 "+"图纸")

if count>10:

print("前10条信息如下：")

for k in range(10):

print(list\_temp[k])

YorN=input("是否全部显示？Y or N?")

if YorN=="Y":

print("TS "+"合同 "+"描述 "+"尺寸 "+"图纸")

for j in range(count):

print(list\_temp[j])

FurInfo=input("是否继续输入额外的关键字查询？若是输入Y: ")

if FurInfo=="Y":

Fuript=input("请输入新的关键字： ")

Fuript\_final=str(Fuript).upper()

for c in range(count):

if re.search(Fuript\_final,str(list\_temp[c]).upper())!=None:

count\_temp=count\_temp+1

print(str(list\_temp[c]))

if count\_temp==0:

print("根据新条件未能查询到")

else:

print("根据新条件共查询到 "+str(count\_temp)+" 条记录")

count\_temp=0

else:

for m in range(count):

print(list\_temp[m])

#---------将检查结果清零：

count=0

list\_temp=[]

re.purge()

#再次输入，重新循环：

input1=input("请输入件号或关键字描述(退出输入'q'):")

a=str(input1).upper()

#print(content[0][0]+" "+content[0][6]+" "+content[0][1]+" "+content[0][2]+" "+content[0][3]+" "+content[0][5]+" "+content[0][4])

print("TS "+"合同 "+"描述 "+"尺寸 "+"图纸")

print("thanks,88")

'''

-----------------------------------------递归算法------------------------------

1.程序主结构为递归算法：

-输出提示信息（使用方法）

-输入查询关键字（第一次）

-调用查询函数

-显示查询结果（查询结果数量和前20条查询结果（若查询结果多余20条，则提示是否全部显示））

-若查询结果大于20条，则开启递归查询过程

2.递归函数：

-根据初始传递的关键字执行查询

-将查询到的结果记录到一个临时list\_temp，将临时list的内容传递到全局list\_all并统计查询数量：

--输出提示信息，询问是否继续查询（'Y' ）

--判断是否查询，若需继续查询则开始如下递归（输入、查询、显示）

--若查询结果数量多于20条，则开始递归查询：

---将搜索的关键字和查询结果的列表作为传递参数

--若查询结果数量少于20条，则显示全部结果和查询数量

-在主程序中定义一个全局性的list\_all,用于存储每一次查询得到的查询结果，并在每次递归函数结束时更新此list（归零）

-在主程序中定义一个全局变量，用于记录查询到的结果数量，并在每次递归函数结束时将其数值归零

'''

## Openpyxl

### Openpyxl应用

from openpyxl import load\_workbook

workbook=load\_workbook(filename="test.xlsx")

print(workbook.sheetnames)#获取sheet名称

sheet=workbook["Sheet1"]

print(sheet)

print(sheet.dimensions)#获取存在内容的表格的范围:A1:H11

cell1=sheet["A1"]

cell2=sheet["C11"]

print(cell1.value,cell2.value)

"""

获取A1/C11的数据

"""

cell3=sheet.cell(row=1,column=1)

cell4=sheet.cell(row=11,column=3)

print(cell3.value,cell4.value)#打印数值

print(cell3,cell4)#打印cell属性（cell 'sheet1'.Al...)

sheet=workbook.active

cell0=sheet["A1"]

cell1=sheet["C11"]

print(cell0.value,cell0.row,cell0.column,cell0.coordinate)#:Name 1 1 A1

print(cell1.value,cell1.row,cell1.column,cell1.coordinate)#:42 11 3 C11

cell=sheet["A:C"]

print(cell)

for i in cell:

for j in i:

print(j.value)#:打印A:C的所有内容，先打印A1：An,再B1:Bn,再C1：Cn

#:按行打印数据

for i in sheet.iter\_rows(min\_row=2,max\_row=5,min\_col=1,max\_col=2):

for j in i:

print(j.value)

#:按列打印数值

for i in sheet.iter\_cols(min\_row=2,max\_row=5,min\_col=1,max\_col=2):

for j in i:

print(j.value)

#:获取所有行

for i in sheet.rows:

print(i)

#获取所有列

for j in sheet.columns:

print(j)

### Openpyxl应用

from openpyxl import load\_workbook

workbook=load\_workbook(filename="test1.xlsx")

sheet=workbook.active

#写入某个cell:

sheet["H11"]="hello111"

sheet["D1"]="height"

workbook.save(filename="test1.xlsx")#:修改H11内容并将文件另存在test1.xlsx

cell0=sheet["H11"]

print(cell0.value)

print(sheet["D1"].value)

#向表格中插入行数据

"""

sheet=workbook.active

data=[

["唐僧","男","180"],

["孙悟空","猴","199"],

["猪八戒","猪","200"],

["沙僧","男","180”],

]

for row1 in data:#row1是列表data中的一行

sheet.append(row1)

workbook.save(filename="test2.xlsx")

"""

#sheet=workbook.active

sheet["E1"]="身高"

workbook.save(filename="test1.xlsx")#:修改H11内容并将文件另存在test1.xlsx

"""

for i in range(2,16):

sheet["D{}".format("i")]=2

#sheet["D

"""

#插入空白行/列，位置由idx指定，数量由amount指定

sheet.insert\_cols(idx=2,amount=2)

sheet.insert\_rows(idx=2,amount=4)

workbook.save(filename="test1.xlsx")#:修改H11内容并将文件另存在test1.xlsx

sheet.delete\_cols(idx=1)

sheet.delete\_rows(idx=1)

workbook.save(filename="test1.xlsx")

### Openpyxl应用

from openpyxl import load\_workbook

workbook=load\_workbook(filename="test1.xlsx")

sheet=workbook.active

#移动格子

sheet.move\_range("C1:D4",rows=2,cols=-1)

#建立新sheet

workbook.create\_sheet("new sheet")

workbook.save(filename="test1.xlsx")

#删除sheet

#workbook.remove(sheet)

workbook.save(filename="test1.xlsx")

#修改sheet名称

sheet.title="new sheet name"

workbook.save(filename="test1.xlsx")

### Openpyxl应用

from openpyxl import Workbook

#新建excel文件

workbook=Workbook()

sheet=workbook.active

sheet.title="new sheet"

workbook.save(filename="D:\Python\Python\_programs\new excel")

## Pandas应用

### Pandas介绍

可以使用pandas对列表数据进行分析。

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

### 数据创建

#传递一个list对象创建Series

s=pd.Series([1,2,3,4,np.nan,5,6])

s

print(s)

#传递一个numpy array创建一个DataFrame:

dates=pd.date\_range('20110101',periods=6)

dates

print(dates)

DatetimeIndes(['2013-0101','2013-01-02','2013-03-03','2013-01-04','2013-01-05','2013-01-06',dtype='datetime64[ns]',freq='D')

df=pd.DataFrame(np.random.randn(6,4),index=dates,columns=list('ABCD'))

df

#传递一个能被转换为类似序列结构的字典对象来创建一个DataFrame:

df2=pd.DataFrame({'A':1.,'B'：pd.Timestamp('20130102'),'C':pd.Series(,indes=list(range(4)),dtype='float32'),'D':np.array([3]\*4,dtype=]int32'),'E':pd.Categorical(["test","train","test","train"]),'F':'foo'})

df2

### pandas常规操作

#### 读写CSV文件

import pandas as pd

import numpy as np

import matplotlib

import random

import plotly.express as px

import plotly.io as pio

#读入CSV文件"1.csv"

#a=pd.read\_csv("1.csv")

#print(a)

#b=pd.read\_csv("filename.csv")

#b=pd.read\_excel("filename.xls")

#-----写入文件

#pd.to\_csv("filename.csv",sep=',',index=False)#写入csv：逗号分隔，不带索引

#pd.to\_excel("filename.xls",sheet\_name="sheet1",index=False)#写入excel：指定Sheet名，不带索引

#b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv")#直接读取，index默认

b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv",index\_col="date\_time")#index默认为date\_time

#### 查看记录

b.values#返回除index之外的所有行（每行是一个list)

b.head(10)

b.describe()#统计所有是数字的列（不统计文本的列）

b['holiday']

b['temp'].describe()

b.index

b.set\_index('temp')#设置temp列为index

b.reset\_index(drop=True)## 不保留原索引到DataFrame中

b.columns

b.dtypes#获取列的数据类型

#-----------逐列打印df内容

for row in df:

df[row]

#-----读取dataframe的若干行并将其转化为一个列表list

df1=df[1:10]

df\_2\_list=list(df1)#仅转化第一行

#。。。？？？？？？？如何操作？

#### 条件筛选

df[df.holiday!='None']

df[(df.holiday=='None')&(df.weather\_main!='Thunderstorm')]#--------与-------------

df[(df.holiday=='None')|(df.weather\_main=='Thunderstorm')]#---------或-------------

df[-(df.holiday=='None')&(df.weather\_main!='Thunderstorm')]#---------非-------------

df[df.holiday.isin(['Labor Day','State Fair'])]#-----验证在某个列表范围内

df[-df.holiday.isin(['Labor Day','State Fair'])]#-----验证不在某个列表范围内

df[df.holiday.str.contains('Fair')]#-----包含某个字符（区分大小写）

df[df.holiday.str.contains('Wash')]

b.rename(columns={'temp':'TEMP'})#更改列名

#b['col\_name'].astype(np.float32)#修改列的数据类型

k=['holiday', 'temp', 'snow\_1h', 'rain\_1h', 'clouds\_all', 'weather\_main','weather\_description','traffic\_volume']

b[k]#更改列的顺序

b.weather\_main=='Clouds']#筛选某列的单一数值

df[df.rain\_1h!=0][['temp','holiday','traffic\_volume']]#某选择条件下，选择某几列内容

df[(df.rain\_1h!=0)&(df.holiday!='None')]

df[(df.holiday!='None')]

#### 读写CSV、sqlite3

import sqlite3

import pandas as pd

#从CSV文件读取并写入sqlite3

b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv",index\_col="date\_time")#index默认为date\_time

con=sqlite3.connect("metro.db")#链接数据库metro.db

cur=con.cursor()

cur.execute("drop table if exists Metro")#检查表确保没有存在的表"Metro"

b.to\_sql("Metro",con)#写入数据库

re=cur.execute("select \* from Metro")

re=cur.execute("select \* from Metro where weather\_main like '%thun%'")

re=cur.execute("select \* from Metro where weather\_main like 'thun'")

re=cur.execute("select \* from Metro where weather\_main='Thunderstorm' order by traffic\_volume")

char=input("input what u r looking for: ")

char1=str('%'+char+'%')

sql="select \* from Metro where weather\_main like "+str('char1')

re=cur.execute(sql)

re.fetchone()

#-------------打印全部数据库内容（分别打印每行的列表）

for row in re:

print(row[0]+" "+row[1]+" "+str(row[2])+" "+str(row[3])+" "+str(row[4])+" "+str(row[5]))

for i in range(30):

re.fetchone()#检查读入的前30条记录

#从sqlite3数据库读入pandas

df=pandas.read\_sql("select \* from Metro",con,index\_col='date\_time')#将date\_time作为index读入

df

for i in range(20):

df[i:i+1]#逐行打印df内的内容

df.to\_excel("metro.xls",sheet\_name="metro",index=True)

cur.close()

con.close()

#### 排序

#-----------------------ORDER BY — 排序------------------------------#

默认情况下，Pandas 会使用升序排序。如果要使用降序，请设置 asending=False。

b.sort\_index(axis=1,ascending=False)

b.sort\_index()

b.sort\_values(by='temp')

b.sort\_values(by='date\_time')

b['weather\_description'].describe()

b['weather\_main'].value\_counts()

b['weather\_main'].unique()#检查某一列的唯一值

b['weather\_main'].nunique()#查看某一列唯一值数量

b['col\_name'].idxmax()#求某一列最大值所在的位置;以修改axis参数，当axis=0时表示对行进行操作，当axis=1时表示对列进行操作。

df.corr()#求各列之间的相关性

df[df.holiday!='None'].sort\_values('traffic\_volume')

df[df.holiday!='None'].sort\_values('traffic\_volume',ascending=False)

df.groupby(['holiday','temp']).size()#将holiday和temp分组排列：相同holiday的列出temp

df.groupby(['holiday','temp','traffic\_volume']).size()#相同holiday的列出temp和traffic\_volume

#### 删除替换

#----------------删除-------------------

a=df.drop(df[df.holiday=='Christmas Day'].index)#删除df.holiday中是"Christmas Day"的行

a['holiday'].value\_counts()#检查确认已删除Christmas Day

#---------------替换-------------------

#指定条件替换

df.loc[df.holiday=='Christmas Day','holiday']='CHRISTMAS DAY'

#全局替换

df.replace(to\_replace='CHRISTMAS DAY',value='CHS DAY',inplace=True)

#局部替换

df.replace({'holiday':{'CHS DAY':'Christmas Day','Veterans Day':'Vet Day'}},inplace=True)

#### 数据处理方式操作函数：

1 df.columns = ['a','b','c'] # 重命名列名

2 pd.isnull() # 检查DataFrame对象中的空值，并返回一个Boolean数组

3 pd.notnull() # 检查DataFrame对象中的非空值，并返回一个Boolean数组

4 df.dropna() # 删除所有包含空值的行

5 df.dropna(axis=1) # 删除所有包含空值的列

6 df.dropna(axis=1,thresh=n) # 删除所有小于n个非空值的行

7 df.fillna(x) # 用x替换DataFrame对象中所有的空值

8 s.astype(float) # 将Series中的数据类型更改为float类型

9 s.replace(1,'one') # 用‘one’代替所有等于1的值

10 s.replace([1,3],['one','three']) # 用'one'代替1，用'three'代替3

11 df.rename(columns=lambda x: x + 1) # 批量更改列名

12 df.rename(columns={'old\_name': 'new\_ name'}) # 选择性更改列名

13 df.set\_index('column\_one') # 更改索引列

14 df.rename(index=lambda x: x + 1) # 批量重命名索引

#判断是否是空值

for row in df:

df[row].isnull()#逐列判断是否是空值

c=b['weather\_main'].value\_counts()#返回统计后的一个字典

#将返回的字典拆分未两个list

c\_index=list(c.index)

c\_counts=list(c)

for I in range(len(c)):

print(str(c\_index[i])+str(c\_counts[i]))

c['Clouds']

c['Smoke']

c[0]

c[:5]#前5个统计

#### 绘图

df[df.

#c.plot(kind='bar')#绘制条形图

import numpy as np

pd.Series([1,2,3])

pd.Series([1,2,3]).values

np.array([1,2,3])

k=np.array([1,2,3])

k!=2

k1=k[k!=2]#传递不等于2的列表k的剩余元素

#--------pandas处理字符串

clouds=b['weather\_main'].str.contains('Clouds')#查询weather\_main列中包含"Clouds"的列

clouds

#clouds.plot()#绘制图表

### 查看数据

#查看不同列的数据类型：

df2.dtypes

#查看DataFrame中头部和尾部的行：

df.head()

df.tail(3)

#显示索引、列和底层的numpy数据:

df.index

df.colums

df.values

#对数据的快速汇总统计：

df.describe()

#对数据的转置

df.T

#按轴进行排序

df.sort\_index(axis=1,ascending=False)

#按值进行排序

df.sort\_values(by='B')

### 选择数据

使用优化过的pandas代码执行数据选择。

#----------------------选择-------------------

#选择一个单独的列

df['A']

#行选择

df[0:3]

df['20130102':'20130104']

#使用标签获取交叉区域

df.loc[dates[0]]

#使用标签在多个轴上选择

df.loc[:,['A','B']]

#标签切片

df.loc['20130102':'20130104',['A','B']]

#对返回的对象进行维度缩减

df.loc['20130102',['A','B']]

#获取一个标量

df.loc[dates[0],'A']

快速访问一个标量

df.at[dates[0],'A']

#-----------------通过位置选择

#通过传递数值进行位置选择

df.iloc[3] #选择第3行

#通过数值进行切片

df.iloc[3:5,0:2]

#通过制定一个位置的列表

df.iloc[[1,2,4],[0,2]]#选择1/2/4行的0/2列

#对行进行切片

df.iloc[1:3,:]#选择2/2行

#对列进行切片

df.iloc[:,1:3]

#获取特定的值

df.iloc[1,1]

#快速访问标量

df.iat[1,1]

#------------------布尔索引

#使用一个单独列的数值选择数据

df[df.A>0]

#使用where操作选择数据

df[df>0]

#使用isin()方法过滤：

df2=df.copy()

df2['E']=['one','one','two','three','four','three']

print(df2)

df2[df2['E'].isin(['two','four'])]

设置

#设置一个新的列

S1=pd.Series([1,2,3,4,5,6],index=pd.date\_range(‘20130102’,periods=6))

Print(s1)

Df[‘F’]=s1

#通过标签设置新的值

Df.at[dates[0],’A’]=0

#通过位置设置新的值：

Df.iat[0,1]=0

#通过一个numpy数组设置一组新的值：

Df.loc[:,’D’]=np.array([5]\*len(df))

Print(df)

#通过where操作设置新的值：

Df2=df.copy()

Df2[dfx>0]=-df2

Print(df2)

### 读写CSV并执行pandas

import pandas as pd

import numpy as np

#读入CSV文件"1.csv"

#a=pd.read\_csv("1.csv")

#print(a)

#b=pd.read\_csv("filename.csv")

#b=pd.read\_excel("filename.xls")

#-----写入文件

#pd.to\_csv("filename.csv",sep=',',index=False)#写入csv：逗号分隔，不带索引

#pd.to\_excel("filename.xls",sheet\_name="sheet1",index=False)#写入excel：指定Sheet名，不带索引

#b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv")#直接读取，index默认

b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv",index\_col="date\_time")#index默认为date\_time

b.values#返回除index之外的所有行（每行是一个list)

b.head(10)

b.describe()#统计所有是数字的列（不统计文本的列）

b['holiday']

b['temp'].describe()

b.index

b.set\_index('temp')#设置temp列为index

b.reset\_index(drop=True)## 不保留原索引到DataFrame中

b.columns

b.dtypes#获取列的数据类型

b.rename(columns={'temp':'TEMP'})#更改列名

#b['col\_name'].astype(np.float32)#修改列的数据类型

k=['holiday', 'temp', 'snow\_1h', 'rain\_1h', 'clouds\_all', 'weather\_main','weather\_description','traffic\_volume']

b[k]#更改列的顺序

### pandas读写CSV、sqlite3

#------------pandas /sqlite3结合

import slqlite3

import pandas

#从CSV文件读取并写入sqlite3

b=pd.read\_csv("Metro\_Interstate\_Traffic\_Volume.csv",index\_col="date\_time")#index默认为date\_time

con=sqlite3.connect("metro.db")#链接数据库metro.db

cur=con.cursor()

cur.execute("drop table if exists Metro")#检查表确保没有存在的表"Metro"

b.to\_sql("Metro",con)#写入数据库

re=cur.execute("select \* from Metro")

for i in range(10):

re.fetchone()#检查读入的前10条记录

#从sqlite3数据库读入pandas

df=pandas.read\_sql("select \* from Metro",con,index\_col='date\_time')#将date\_time作为index读入

df

df.to\_excel("metro.xls",sheet\_name="metro",index=True)

cur.close()

con.close()

#### 缺失值处理

#### 绘图

使用pandas绘图

Ts=pd.Series(np.random.randn(1000),index=pd.dae\_range(‘1/1/2000’,periods=1000))

Ts=ts.cumsum()

Ts.plot()

对于 DataFrame 来说， plot 是一种将所有列及其标签进行绘制的简便方法：

Df=pd.DataFrame(np.random.randn(1000,4),index=ts.index,columns=[‘A’,’B’,’C’,’D’])

Df=df.cumsum()

Plt.fiture();df.plot();plt.legend(loc=’best’)

#### 导入和保存数据

##### CSV文件

#写入CSV文件

Df.to\_csv(‘foo.csv’)

#从CSC文件读取

Pd.read\_csv(‘foo.csv’)

##### HDF5文件

#写入hdf5文件

Df.to\_hdf(‘foo.h5’,’df’)

#从hdf5文件读取

Pd.read\_hdf(‘foo.h5’,’df’)

##### EXCEL文件

#写入excel文件

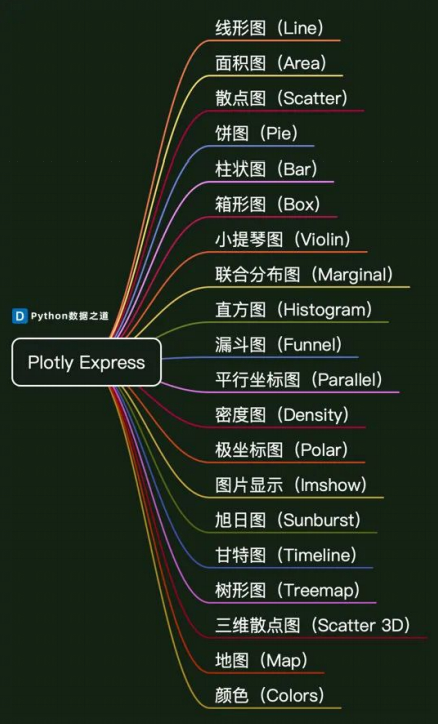
Df.to\_excel(‘foo.xlsx’,sheet\_name=’sheet1’)

#从excel文件读取

Pd.read\_excel(‘foo.xlsx’,’sheet1’,index\_col=none,na\_values=[‘NA’])

## Plotly/Plotly express

Plotly Express是python交互式可视化库plotly的高级组件，可方便进行数据可视化，大多数函数只需要一个函数调用，接受一个整洁的pandas dataframe,并简单描述需制作的图。



### 安装

Pip install plotly

### 导入

Import plotly.express as px

Import plot.io as pio

Import plotly.graph\_objects as go

Import pandas as pd

#### 漏斗图

#Data=dict(

date=pd.date\_range('20110101',periods=6)

Numer=[10000,7000,4000,2000,1000],

Stage=[‘浏览次数’,’关注’,’下载’,’咨询’,’成交’])

Fig=px.funnel(data,x=’number’,y=’stage’)

Fig.show()

另一种漏斗图：

Fig=px.funnel\_area(names=’第一阶段’,’第二’,’第三’,’第四’,’第五’),values=[10000,7000,4000,2000,1000])

Fig.show()

### 显示图片

From skimage import io

Img=io.imread(‘./path…/name.jpg’)

Fig=px.imshow(img)

Fig.show()