class Table：

Attribute：

self.imax = -1

self.imap = [] 用于iteration

self.name = name 表格名字

self.table = [] 主要内容

self.lenmap = [] 记录字符长度

self.colmap = {} 记录栏目名和对应column index

self.searchmap = None 用于搜寻row是否在表格内，以及row所在的index

self.key=None 记录primary key

self.keymap = None 用于搜索primary key 是否在表格内

Table.spacing = \_destripC (可选择中心对齐和左对齐)

self.see()：

查看Table中是所有的no-static attribute

Table.\_destripC(s, n):

中心对齐

str s: Table中value的str形式

int n: 在对应lenmap在对应col中的值

Table.\_destripL(s, n):

左对齐

str s: Table中value的str形式

int n: 在对应lenmap在对应col中的值

Table.shift():

左对齐变中心对齐，中心对齐变左对齐

def \_\_init\_\_(self, table=None, name=None):

# index for iterator

self.imax = -1

self.imap = []

self.name = name

self.table = []

self.lenmap = []

self.searchmap = None

self.key=None

self.keymap = None

# for quick search

self.colmap = {}

"need to do!"

# colmap should be used in saving!!!, and repr and str!!!

self.rowmap = {}

if table is None:

pass

elif isinstance(table, str):

self.table.append(slist(table, ","))

self.setlenmap()

self.setcolmap()

elif isinstance(table, list):

if isinstance(table[0], list):

for row in table:

self.table.append(Row(row, self))

else:

self.table.append(Row(table, self))

self.setlenmap()

# set colmap

self.setcolmap()

# set rowmap

"need to do!"

else:

raise Exception("table should be 2D-list")

self.setkeymap(s):

设置表格的primary key

str s: primary key 比如 “pk1，pk2…” 或者 “pk1”

self.addkeymap():

增加append进来的row的到keymap中

self.\_updatesearchmap(index):

更新第index 行的内容到searchmap

int index: 需要searchmap update的所在行

item in self

self.\_\_contains\_\_(item):

查看元素是否在表格内：

第一次运行O(n) 需要创建整个searchmap

之后运行 O(1) 在searchmap里搜索内容

list/tuple item: 被查找的元素

self.append(row):

增加row并更新，lenmap和searchmap

list row：加进来的row

len(self)

self.\_\_len\_\_():

返回表格行数（不包括第0行）

self.gethead():

返回第0行的copy

self.setlenmap():

设置self.lenmap (整张表格)

self.updatelenmap(row):

更新某row 到 self.lenmap

self.refreshlempa(self,i, entry):

刷新第i列（col）的string长度到 self.lenmap

self.setcolmap():

设置self.colmap

next(self)

self.\_\_next\_\_():

返回下一个在 self中的下一个value

iter(self)

self.\_\_iter\_\_():

返回一个self的迭代器

以上两个语句可使如下语句运行

for row in self:

[deal(row) for row in self]

self.\_check(self, a):

检查a 是否是self.table中的一个attribute（栏目名）

self.\_checklist(lst):

检查lst中的所有value 是否是self.table中的attribute(栏目名)

self.getlist(self, i, ls):

对应ls中的栏目名，在第i行取得list of value 并返回

int i: 第i行

list ls: 需要拿到的值所对应的栏目名

self.getsearchmap(ls):

用ls中的栏目名返回一个搜索用的dict

list ls: 需要拿到的值所对应的栏目名

return {(v1,v2,…):[n,m,..]}

v1,v2是对应栏目名在对应列中的值

n,m是对应列所在的序号

row2str(self, row):

返回某row的str形式

head2str(self):

返回第0行，栏目名所在行，的str形式

str(self):

\_\_str\_\_(self):

返回self的str形式

self.p:

分行打印出self的str形式，enter键打印下一行，输入exit回车停止打印

self

self.\_\_repr\_\_():

self的默认显示形式

Table.read(name):

读取当前目录下的csv文件，并返回一个table

str name: csv 文件名（不包括”.csv”）

Ex:

如果文件名是”file.csv”

那么name 应该输入 “file”

self.save(name=None):

储存当前的self文件为一个.csv文件

如果self已经有名字，name可以不填

如果self没有名字，需要写名字来指定需要存储的.csv文件名(同上，不包括.csv)

get set delete

def \_\_getitem\_\_(self, key):

if key == '\*':

return Col(self)

return self.table[key]

def \_\_setitem\_\_(self, key, value):

# add check

self.table[key] = Row(value)

self.lenmap = [max(self.lenmap[i], len(str(value[i]))) for i in range(len(value))]

def \_\_delitem\_\_(self, key):

# add check and primary key setting

if key == 0:

raise Exception("should not delete 0-th row")

self.table.pop(key)

pass

SQL 语句部分：

self.select(s):

选择attribute返回一个Table

str s: 需要select的栏目名

Ex：如 “a1”

或者 “a1,a2…” 多个用”,” 联接

self.rename(s):

重新命名attribute 在本地修改不返回

str s: 需要rename的栏目名

Ex：如 “a1 = b1”

或者 “a1 = b1, a2 = b2…”

或者 “a1,a2 .. = b1,b2…”

self.getLMR(other):

Table self: self表格

Table other:另一张表格

返回 [list of str: self 中独有的attribute name,

list of str: self 和 other中共有的attribute name,

list of str: other 中独有的 attribute name]

self.\_join(other, on=None, mod="natural"):

Table self: self表格

Table other:另一张表格

str on:

Ex：如 “a1 = b1”

或者 “a1 = b1, a2 = b2…”

或者 “a1,a2 .. = b1,b2…”

mod= "natural","left", "right", "full"

如果on 没有指明，则会根据self table 和 other table 中的attribute用getLMR自动生成对应join栏目，如果指定则会按照指定规则join

self@other

self.\_\_matmul\_\_(other):

等于 self.\_join(other)

self\*other

self.\_\_mul\_\_(other):

返回一个 self cross join other的Table

self.cross(other)

等于 self \* other

self.copy(name =None):

制作一个self的副本copy

如果name省略是副本的名字为None

如果name指定，副本名字为指定名字

self \*\*power

pow(self, power)

self. \_\_pow\_\_(power, modulo=None):

返回power个selves cross join的结果

self.getset():

用表格中的所有行做成一个set，并返回

self | other

self.\_\_or\_\_(other):

union self 和other 生成一个Table并返回

self & other

self.\_\_and\_\_(other):

intersection self 和other 生成一个Table并返回

self^other

self.\_\_xor\_\_(other):

xor self 和other 生成一个Table并返回

self - other

self.\_\_sub\_\_(other):

self 减去 other中的 row 生成一个Table并返回

#增加 union，intersection，和 minus的表示

def orderby(self, s):

def setlib():

s = """

try:

import matplotlib.pyplot as plot

global plt

plt = plot

except Exception:

print("can not import matplotlib!")

try:

from mpl\_interaction import PanAndZoom

global PAZ

PAZ = PanAndZoom

except:

print("please download mpl\_interaction.py")

"""

return exec(s, globals(), locals())

def addrow(self, array):

if (len(array) == len(self.table[0])):

self.table.append(Row(array.copy(), self))

self.lenmap = [max(self.lenmap[i], len(str(array[i]))) for i in range(len(array))]

else:

raise Exception("length not match")

def bar(self, label, value):

Table.setlib()

self.orderby(label)

fig, ax = plt.subplots()

ax.bar([str(v) for v in self.col(label)], self.col(value))

plt.xlabel(label)

plt.ylabel(value)

pan\_zoom = PAZ(fig)

plt.show()

def pie(self, label, value):

Table.setlib()

fig, ax = plt.subplots()

ax.pie(self.col(value), labels=self.col(label), autopct='%1.1f%%')

pan\_zoom = PAZ(fig)

plt.show()

def hist(self, value, low, up, num):

Table.setlib()

fig, ax = plt.subplots()

amount = up - low

block = amount / num

bins = [low + i \* block for i in range(num + 2)]

ax.hist(self.col(value), bins, facecolor='green', edgecolor="yellowgreen")

plt.xlabel(value)

plt.ylabel("number")

pan\_zoom = PAZ(fig)

plt.show()

def plot(self, x, y, line="."):

# add polar coord

# add multiple and lines

# add spline

Table.setlib()

fig, ax = plt.subplots()

ax.plot(self.col(x), self.col(y), line)

plt.xlabel(x)

plt.ylabel(y)

pan\_zoom = PAZ(fig)

plt.show()

def radar(self):

# multiple in one graph

pass

def polar\_radar(self):

pass

def hist2d(self):

pass

def bar2d(self):

pass

def scatter(self):

pass

def classify(self):

# classify the value

pass

def groupby(self, s):

# group table by attributes in s

# form a subtable in groupby dictionary

pass