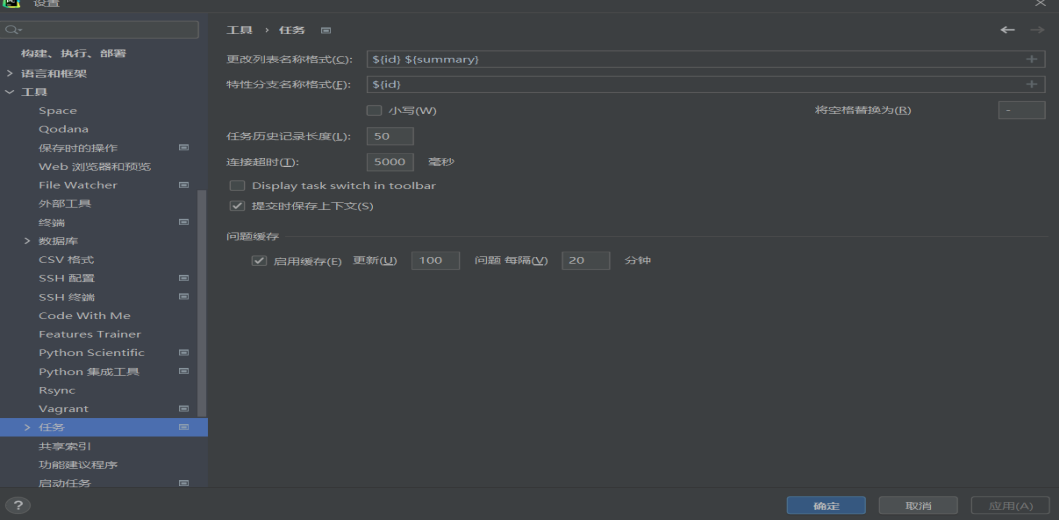
Python笔记：

1. 常用配置
2. pycharm关闭项目缓慢->取消勾选问题缓存 启用缓存



1. pycharm整合git

dir/a 显示全部文件 dir/a:h显示隐藏文件 dir/a:-h显示非隐藏文件

1. Python快速入门

Ctrl+Alt+L——代码格式化

1.卸载Python（控制面板中程序和功能，卸载python，会同时将环境变量卸载干净）

2.安装Python和Pycharm环境

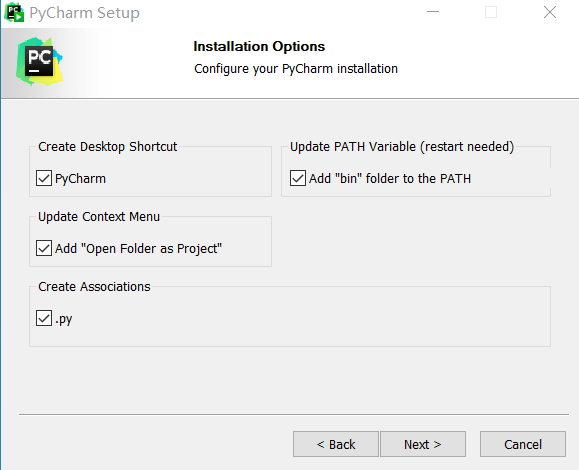
安装Python3.8.10：

1）.选择客制化+Add Python to PATH



Optional Features、Advanced Options两页全选

2）.安装PyCharm（软件管家）



3.Python都能干啥？

Web开发、运维开发、机器学习、深度学习、计算机视觉与自然语言处理、数据挖掘、Spark大数据系列

4.Python数值运算(+-\*/%)

几种常用类型 int/float/str/bool

1).求幂运算\*\*

2\*\*3//==8

2).赋值（自动类型推导机制）

tang=3

print(type(tang))// <class 'int'>

3).类型转换

tang=1.5

tang2=int(tang)//强制类型转换

4).基本数值操作

A.取绝对值

print(abs(-15.6))//15.6

B.四舍五入

print(round(15.4))//15

print(round(15.5))//16

C.取最小值/最大值

print(min(2,3,4,5))//2

print(max(2,3,4,5))//4

D.科学计数法

print(1.3e-5)//1.3\*10-5

print(1.3e5)//130000

E.16进制

print(0xFF)//255

5.字符串操作

A.+字符串拼接

tang='hello'+'python'

print(tang)// hellopython

B.\*字符串复制

tang='hello'\*3

print(tang)// hellohellohello

C.字符串个数

tang='hello'

print(len(tang))//5

D.字符串切分（返回list类型）

tang='1 2 3 4 5'

print(tang.split())//默认以空格进行切分 ['1', '2', '3', '4', '5']

tang='1,2,3,4,5'

print(tang.split(","))//['1', '2', '3', '4', '5']

E.list类型组合成字符串

print(" ".join(tang))//tang为list类型['1', '2', '3', '4', '5']🡺1 2 3 4 5 6

F.字符串替换

tang="hello python"

print(tang.replace("python","world"))//hello world

G.字符串转换为大写/小写

tang="hello python"

print(tang.upper())//HELLO PYTHON

tang="HELLO PYTHON"

print(tang.lower())//hello python

H. 移除字符串头/尾/头尾指定的字符/字符序列（lstrip(),rstrip(),strip()）

tang=" hello python "

print(tang.strip())//hello python

tang="———————hello python————————"

print(tang.strip("—"))//hello python

I.格式化字符串

print("{}的价格是{}元".format("衬衫","$9.15"))//衬衫的价格是$9.15元

print("{1} {2} {0}".format("lin", "ze", "yuan"))// ze yuan lin

print("{a}的价格是{b}元".format(b="$9.15",a="衬衫"))//衬衫的价格是$9.15元

print("%s的价格是%s%f元"%("衬衫", "$", 9.15))// 衬衫的价格是$9.15元

J.判断子串

string="tangyudi"

print("tang" in string)// True

6.list结构

1).list创建

通过[]来创建一个list结构，list可以存储任何类型数据，且没有一个长度限制

tang=[]/list()

print(type(tang))// <class 'list'>

tang=[1,'tangyudi',3.5]/list([1,'tangyudi',3.5])

print(type(tang))// <class 'list'>

2).list操作

A.list长度

tang=[1,'tangyudi',3.5]

print(len(tang))// 3

B.list合并操作

a=[123,456]

b=['tang','yudi']

print(a+b)// [123, 456, 'tang', 'yudi']

C.list复制

a=[123,456]

print(a\*3)// [123, 456, 123, 456, 123, 456]

D.list索引

I.根据索引赋值/修改

a=[123,456]

print(a[0],a[1],a[-1],a[0:])// 123 456 456 [123, 456]（切片是前闭后开的）

a=[0,1,2,3,4,5,6,7,8,9]

print(a[::2], a[::-1], a[::-2])// [0, 2, 4, 6, 8] [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] [9, 7, 5, 3, 1]



a=[123,456]

a[0]=10000

print(a)// [10000, 456]

a=[0,1,2,3,4,5,6,7,8,9]

a[2:6]=[1,2,3,4]

print(a)// [0, 1, 1, 2, 3, 4, 6, 7, 8, 9]（a[:]=;全替换）

II.根据索引删除

a=[0,1,2,3,4,5,6,7,8,9]

del a[0]

print(a)// [1, 2, 3, 4, 5, 6, 7, 8, 9]

III.判断某个元素是否在列表list中

a=[0,1,2,3,4,5,6,7,8,9]

print(8 in a)// True

print(10 in a)// False

print(10 not in a)// True

7.dict结构

1). dict创建

通过{}来创建一个dict结构,是一种key-value结构,无顺序的结构

tang={}/dist()

print(type(tang))// <class 'dict'>

tang=dict([('name','linzeyuan'),('age',18),('height',178.15)])//列表+元组赋值

print(tang)// {'name': 'linzeyuan', 'age': 18, 'height': 178.15}

2).dict操作

A.dict索引key

I.key索引赋值/修改

tang={}

tang['name']="linzeyuan"

tang['age']=18

tang['height']=178.15

print(tang)// {'name': 'linzeyuan', 'age': 18, 'height': 178.15}

II.key索引取值

tang={'name':'linzeyuan','age':18,'height':178.15}

print(tang['name'])// linzeyuan

print(tang.get('name'))// linzeyuan

tang=dict([('name','linzeyuan'),('age',18),('height',178.15)])

print(tang.get('weight',70.28))//70.28(如果没有key值“weight”,就返回70.28)

III.key索引弹出

tang={'name':'linzeyuan','age':18,'height':178.15}

print(tang)//{'name':'linzeyuan','age':18,'height':178.15}

tang.pop('height')

print(tang)//{'name':'linzeyuan', 'age': 18}

IV.key索引删除

tang={'name':'linzeyuan','age':18,'height':178.15}

print(tang)

del tang['height']

print(tang)

V.key索引更新

tang1={'name':'linzeyuan','age':18,'height':178.15}

tang2={'name':'chenhebin','weight':70}

tang1.update(tang2)

print(tang1)// {'name': 'chenhebin', 'age': 18, 'height': 178.15, 'weight': 70}

VI. 判断某个键值是否在字典dict中

tang={'name':'linzeyuan','age':18,'height':178.15}

print('name' in tang)// True

print('weight' in tang)// False

VII.打印字典的所有键/值/键值对

tang={'name':'linzeyuan','age':18,'height':178.15}

print(tang.keys())//dict\_keys(['name', 'age', 'height'])相当于print(tang)

print(tang.values())//dict\_values(['linzeyuan', 18, 178.15])

print(tang.items())//dict\_items([('name', 'linzeyuan'), ('age', 18), ('height', 178.15)])

8.set结构

1).set创建(set集合中元素不能重复，会保留下来唯一的那些元素)

tang=set([123,123,456,789])/{123,123,456,789}

print(tang)// {456, 123, 789}

tang=set()

print(type(tang))// <class 'set'>

2).set操作

A.求并集

a={1,2,3,4}

b={2,3,4,5}

print(a.union(b))

print(a|b)// {1, 2, 3, 4, 5}

B.求交集

a={1,2,3,4}

b={2,3,4,5}

print(a.intersection(b))

print(a&b)// {2, 3, 4}

C.求差集

a={1,2,3,4}

b={2,3,4,5}

print(a.difference(b))

print(a-b)// {1}

D.判断b是不是a的子集

a={1,2,3,4}

b={2,3}

print(b.issubset(a))// True

print(b<=a)// True

E.往set集合中添加元素

b={1,2,3}

b.add(4)

print(b)// {1, 2, 3, 4}

(对比往list列表中添加元素)

a=[1,2,3,4]

a.append(5)

print(a)// [1, 2, 3, 4, 5]

F.更新集合

a={1,2,3,4}

a.update({4,5,6})

print(a)// {1, 2, 3, 4, 5, 6}

9.赋值机制

tang=1000

yudi=tang

print(id(tang))//2383329208336

print(id(yudi))// 2383329208336

print(tang is yudi)//True

（tang,yudi指向内存中的同一块空间

python的内置函数id()，用于返回对象的身份，就是该对象的内存地址）

tang=1000

yudi=123456

print(tang is yudi)//False

tang=15e20

yudi=15e20

print(id(tang))//2353176817264

print(id(yudi))//2353176817264

（赋值机制,在Python中构造一个比较大的数,会分别构建一片内存空间？？？;构造一个比较小的数,会指向同一片内存空间）

10.判断结构

if语句：

tang = 100

if tang > 200:

print("OK")

print("TEST")

tang=[123,456,789]

if 123 in tang:

print('OK')

if-elif-else语句：

tang=50

if tang>200:

print('200')

elif tang<100:

print('100')

else:

print('100-200')

11.函数

I.基本结构

def add\_ab(a,b):

return a+b

add\_ab(3,5)

（TypeError: add\_ab() takes 2 positional arguments but 3 were given add\_ab()只接受两个参数）

II.参数默认值

def add\_ab(a=2, b=0):

return a+b

print(add\_ab())//2

print(add\_ab(1))//1

print(add\_ab(b=1))//3

print(add\_ab(1,1))//2

III.缺省参数

def add\_number(a,\*args):

for i in args:

a += i

return a

print(add\_number(1,2,3,4,5))//15

def print\_dict(\*\*kvargs):

for key, value in kvargs.items():

print("key===>", key, "value===>", value, "\n")

print\_dict(x=2,y=3,z=4)//

key===> x value===> 2

key===> y value===> 3

key===> z value===> 4

IV.返回两个参数

def min\_max\_num(ls):

a=0

b=0

//a,b=0,0

a=min(ls)

b=max(ls)

return a,b

ls=[0,1,2,3,4,5,6,7,8,9]

a,b=min\_max\_num(ls)

print('a===>',a,'b===>',b)// a===> 0 b===> 9

12.模块与包

tang.py文件

tang\_v = 10

def tang\_add(tang\_list):

tang\_sum = 0

for i in range(len(tang\_list)):

tang\_sum += tang\_list[i]

return tang\_sum

tang\_list=[1,2,3,4,5]

print(tang\_add(tang\_list))

方式一：main.py文件

import tang

lt= ls=[0,1,2,3,4,5,6,7,8,9]

print(tang.tang\_add(lt))//15 45

方式二：main.py文件

import tang as tg

tg.tang\_v = 100

print(tg.tang\_v)//15 100

方式三：main.py文件

from tang import tang\_v, tang\_add

print(tang\_v)

ls = [0,1,2,3,4,5,6,7,8,9]

print(tang\_add(ls))//15 10 45

方式四：main.py文件

from tang import \*

print(tang\_v)

ls=[0,1,2,3,4,5,6,7,8,9]

print(tang\_add(ls))//15 10 45

补充:

1. 在jupyter中创建/删除.py文件

I.创建tang.py文件

%%writefile tang.py

II.删除tang.py文件

import os

os.remove("tang.py")

III. 取指定文件或目录的绝对路径（完整路径）

import os

print(os.path.abspath("."))

1. for循环

1).for循环输出列表（字典、集合、字符串）

I.列表

ls = ['Paris', 15, 3.1415926, 'Hello world']

for item in ls:

print(item)

II.字典（keys、values、items均为列表）

dc = {'name': 'linzeyuan', 'age': 18, 'height': 178}

for item in dc:

print(item, " ", dc[item])

III.集合

s = {'pig', 'apple', 'orange', 'zoo'}

for item in s:

print(item)// 无顺序 orange apple zoo pig

IV.字符串

string = 'hello world'

for s in string:

print(s, end=' ')// h e l l o w o r l d

2).range限制for循环的循环次数

I.range默认从0开始

for i in range(6):

print(i, end=' ')// 0 1 2 3 4 5

II.range默认前闭后开

for i in range(1, 6):

print(i, end=" ")// 1 2 3 4 5

III.range设置间隔，默认间隔为1

for i in range(1, 6, 2):

print(i, end=" ")// 1 3 5

IV.for...else...(for所有语句正常运行完毕后,才运行else语句)

for n in range(2, 6):

for x in range(2, n):

if n % x == 0:

print( n, 'equals', x, '\*', n/x)

break

else:

print(n, 'is a prime number')

2 is a prime number

3 is a prime number

4 equals 2 \* 2.0

5 is a prime number

13.异常

1).基本语法try-except:

异常的种类有多种ValueError、ZeroDivisionError等

I. ValueError

import math

for i in range(10):

try:

input\_number = input('write a number')

if input\_number == 'q':

break

result = math.log(float(input\_number))

print(result)

except ValueError:

print('ValueError:input must > 0')

break

II. ZeroDivisionError

import math

for i in range(10):

try:

input\_number = input('write a number')

if input\_number == 'q':

break

result =1/math.log(float(input\_number))

print(result)

except ValueError:

print('ValueError:input must > 0')

break

III. Exception

import math

import traceback

for i in range(10):

try:

input\_number = input('write a number')

if input\_number == 'q':

break

result = 1 / math.log(float(input\_number))

print(result)

except Exception as e:

print(e)

traceback.print\_exc()

try…except…except…except 异常发生后，会从except后开始执行

2).自定义异常

class TangError(ValueError):

pass

cur\_list = ['tang', 'yu', 'di']

while True:

cur\_input = input()

if cur\_input not in cur\_list:

raise TangError('Invalid input:%s' % cur\_input)

3).关键字finally

无论是否有异常,都会执行finally后的语句

try:

1/0

finally:

print('finally')

14.文件操作

1).读文件

I.txt.read()返回str

txt = open('./data/tang.txt')

txt\_read = txt.read()

print(type(txt\_read)) // <class 'str'>

print(txt\_read)

txt.close()

II.txt.readlines()返回list

txt = open('./data/tang.txt')

lines = txt.readlines()

print(type(lines))// <class 'list'>

print(lines)

for line in lines:

print("cur\_line:", line)

txt.close()

2).写文件（写完文件要close文件）

I. open('tang\_write.txt', 'w') 覆盖写

txt = open('tang\_write.txt', 'w')

txt.write('jin tian tian qi bu cuo\n')

txt.write('tang yu di')

txt.close()

II.open('tang\_write.txt', 'a') 追加写

txt = open('tang\_write.txt', 'a')

txt.write('123\n')

txt.write('456\n')

txt.close()

3).文件操作与异常处理

txt = open('tang\_write.txt', 'w')

try:

for i in range(100):

10 / (i - 50)

txt.write(str(i)+'\n')

except Exception:

print("error", i)

finally:

txt.close()//文件tang\_write.txt写入0-49

4).with open（）as f无需关闭文件

with open('tang\_write.txt', 'a') as f:

f.write('hello world\n')

15.类:面向对象

self——类本身中的实例

1).类的创建

class People:

# 静态变量 类中所有实例共享

number = 100

# 构造函数 self类本身中的实例

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display(self):

print('number = :', People.number)

def display\_name(self):

print(self.name)

打印信息：

p1 = People('tang yudi', 18)

p2 = People('python', 40)

print(p1.name)// tang yudi

print(p2.name)// python

p1.display()//number = : 100

p2.display()//number = : 100

People.number = 1000

p1.display()//number = : 1000

p2.display()//number = : 1000

静态变量只能通过类名.属性名进行修改，无法通过实例.属性名进行修改。

del p1.name

print(p1.name)

AttributeError: 'People' object has no attribute 'name'

2).类的一些常见操作

print(hasattr(p1, 'name'))//true

print(getattr(p1,'name'))// tang yudi

setattr(p1, 'name', 'lin zeyuan')

print(getattr(p1, 'name'))//lin zeyuan

delattr(p1, 'name')//===del p1.name

print(People.\_\_doc\_\_)// None

print(People.\_\_name\_\_)// People

print(People.\_\_module\_\_)// \_\_main\_\_

print(People.\_\_bases\_\_)// (<class 'object'>,)

print(People.\_\_dict\_\_)// {'\_\_module\_\_': '\_\_main\_\_', 'number': 100, '\_\_init\_\_': <function People.\_\_init\_\_ at 0x0000021C59CBA700>, 'display': <function People.display at 0x0000021C59DD1430>, 'display\_name': <function People.display\_name at 0x0000021C59DD1A60>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'People' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'People' objects>, '\_\_doc\_\_': None}

3).类的继承

I.继承父类的属性/方法：

class Parent:

number = 100

def \_\_init\_\_(self):

print('调用父类构造函数')

def parentM(self):

print('调用父类方法')

def setAttr(self, attr):

Parent.parentAttr = attr

def getAttr(self):

print('父类属性:', Parent.parentAttr)

class Child(Parent):

def \_\_init\_\_(self):

print('调用子类构造方法')

def childM(self):

print('调用子类方法')

c = Child()//调用子类构造方法

c.childM()//调用子类方法

c.parentM()调用父类方法

c.setAttr(100)

c.getAttr()//父类属性: 100

print(Parent.parentAttr)//100

print(Child.parentAttr)//100

II.子类重写父类方法:

class Parent:

number = 100

def \_\_init\_\_(self):

print('调用父类构造函数')

def newM(self):

print('父类要被重写的方法')

class Child(Parent):

def \_\_init\_\_(self):

print('调用子类构造方法')

def newM(self):

print('子类将父类改掉了')

c = Child()//调用子类构造方法

c.newM()//子类将父类改掉了

1. Python常用工具包实例（Pandas、Matplotlib）

1.Pandas数据分析处理库

通常通过numpy进行数据的计算,通过pandas进行数据的读取和处理,pandas封装在numpy基础之上,在数据层面上处理起来更方便。

CSV文件操作

I.head(n)可以读取前n条数据,默认是前5条数据

import pandas as pd

df = pd.read\_csv('./data/food\_info.csv')

NDB\_No Shrt\_Desc ... FA\_Poly\_(g) Cholestrl\_(mg)

0 1001 BUTTER WITH SALT ... 3.043 215.0

1 1002 BUTTER WHIPPED WITH SALT ... 3.012 219.0

2 1003 BUTTER OIL ANHYDROUS ... 3.694 256.0

3 1004 CHEESE BLUE ... 0.800 75.0

4 1005 CHEESE BRICK ... 0.784 94.0

Pandas中的核心结构<class 'pandas.core.frame.DataFrame'> —— 表

II.info()返回当前的信息

df = pd.read\_csv('./data/food\_info.csv')

print(df.info())

print(type(df.info()))//

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8618 entries, 0 to 8617

Data columns (total 36 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 NDB\_No 8618 non-null int64

1 Shrt\_Desc 8618 non-null object

2 Water\_(g) 8612 non-null float64

3 Energ\_Kcal 8618 non-null int64

III.index[类属性]

df = pd.read\_csv('./data/food\_info.csv')

print(df.index)

RangeIndex(start=0, stop=8618, step=1)

IV.columns[类属性]

df = pd.read\_csv('./data/food\_info.csv')

print(df.columns)

Index(['NDB\_No', 'Shrt\_Desc', 'Water\_(g)', 'Energ\_Kcal','Protein\_(g)',

'Lipid\_Tot\_(g)', 'Ash\_(g)', 'Carbohydrt\_(g)', 'Fiber\_TD\_(g)',

'Sugar\_Tot\_(g)', 'Calcium\_(mg)', 'Iron\_(mg)', 'Magnesium\_(mg)',

'Phosphorus\_(mg)', 'Potassium\_(mg)', 'Sodium\_(mg)','Zinc\_(mg)',

'Copper\_(mg)', 'Manganese\_(mg)', 'Selenium\_(mcg)','Vit\_C\_(mg)',

'Thiamin\_(mg)', 'Riboflavin\_(mg)', 'Niacin\_(mg)','Vit\_B6\_(mg)',

'Vit\_B12\_(mcg)', 'Vit\_A\_IU', 'Vit\_A\_RAE', 'Vit\_E\_(mg)', 'Vit\_D\_mcg','Vit\_D\_IU', 'Vit\_K\_(mcg)', 'FA\_Sat\_(g)', 'FA\_Mono\_(g)', 'FA\_Poly\_(g)','Cholestrl\_(mg)'],dtype='object')

V.dtypes[@property]（被 @property 装饰的方法是获取属性值的方法）

df = pd.read\_csv('./data/food\_info.csv')

print(df.dtypes)

NDB\_No int64

Shrt\_Desc object

Water\_(g) float64

Energ\_Kcal int64

Protein\_(g) float64

VI.values[类属性]

df = pd.read\_csv('./data/food\_info.csv')

print(df.values)

[[1001 'BUTTER WITH SALT' 15.87 ... 21.021 3.043 215.0]

[1002 'BUTTER WHIPPED WITH SALT' 15.87 ... 23.426 3.012 219.0]

[1003 'BUTTER OIL ANHYDROUS' 0.24 ... 28.732 3.694 256.0]

...

[90480 'SYRUP CANE' 26.0 ... 0.0 0.0 0.0]

[90560 'SNAIL RAW' 79.2 ... 0.259 0.252 50.0]

[93600 'TURTLE GREEN RAW' 78.5 ... 0.088 0.17 50.0]]

2.Pandas基本操作

1).创建一个DataFrame结构

data = {'country': ['aaa', 'bbb', 'ccc'], 'population': [10, 12, 14]}

df\_data = pd.DataFrame(data)

print(df\_data)

country population

0 aaa 10

1 bbb 12

2 ccc 14

2).取指定的数据

I.打印指定列

df = pd.read\_csv('./data/food\_info.csv')

print(df['Water\_(g)'])

0 15.87

1 15.87

2 0.24

3 42.41

4 41.11

...

Pandas中的核心结构Series——DataFrame中的一行或一列

df = pd.read\_csv('./data/food\_info.csv')

water = df['Water\_(g)']

print(type(water)) <class 'pandas.core.series.Series'>

print(water[:5])

0 15.87

1 15.87

2 0.24

3 42.41

4 41.11

Series的基本操作

print(water.index)// RangeIndex(start=0, stop=8618, step=1)

print(water.values[:5])// [15.87 15.87 0.24 42.41 41.11]

II.打印指定行、列的数据（先指定列再指定行）

df = pd.read\_csv('./data/food\_info.csv')

print(df['Water\_(g)'][1000])//68.65

III.将指定列设置为索引

df = pd.read\_csv('./data/food\_info.csv')

df = df.set\_index('Shrt\_Desc')

print(df.head())

IV.运算操作

加减乘除

water = water + 10//指定列+10

统计指标、平均值、最大/小值

water = df['Water\_(g)'][:5]

print(water.describe())

print(water.mean())

print(water.max())

print(water.min())

count 5.00000

mean 23.10000

std 18.19589//标准差

min 0.24000

25% 15.87000//四分位

50% 15.87000

75% 41.11000

max 42.41000

Name: Water\_(g), dtype: float64

23.099999999999998

42.41

0.24