Python自动化笔记

**8月16日**

**数据类型的作业都是用来存储数据的；**

**字符串、元祖是有序不可变类型，列表是可变类型**

**变量名重复使用，以最后一次赋值为准**

**8月16日**

1. site-packages放插件的文件夹
2. 关键字查看：

**import** keyword  
print(keyword.kwlist)

1. 变量名和函数名用\_，类名用驼峰式FirstName
2. 检查类型print(type())
3. //整除，%取余数（取余只能整数除以整数，若除数比被除数大，直接除数就是余数，若除数比被除数小，被除数就除以除数直到剩下的数比除数小，则这个数就是余数，而且注意余数的符号要与被除数的符号一致），模运算，\*\*几次方运算，幂运算

**8月19日**

1. 转移字符



1. 不同的数据类型是不能进行运算的，要一起运算要进行类型的转换，才能正常输出。把数据类型词直接加在前面str(33323134)
2. 函数是个动作，动词；变量名、类名是名词

**8月21日**

1. float做运算要小心，可能不相等
2. Decimal浮点型运算高精度，后面加字符串数据类型

用Decimal需要输入from decimal import Decimal

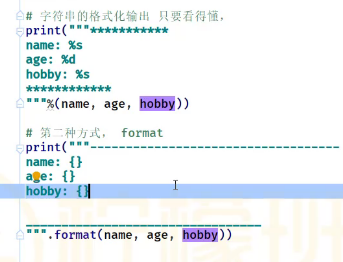
1. 步长，先取开始位置，再加步长，一直取到结束位置。
2. [::-1]倒序
3. A[1:1000]超出范围，全取切片

A[1000]索引超出范围，报错

[0:4:-2}这个情况不打印（步长为正数，从左到右；负数，从右到左）

[2:-2:2]正常打印，从左到右

1. 字符串格式化输出 .format 空花括号，可以加索引

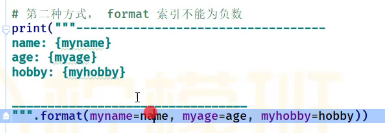




第二种方法{ }（花括号）写format( )里的索引，可以改变数据位置；不能为负数。

先取format括号里的索引再取变量的索引，层层取

关键字插旗子，方便识别是否写错

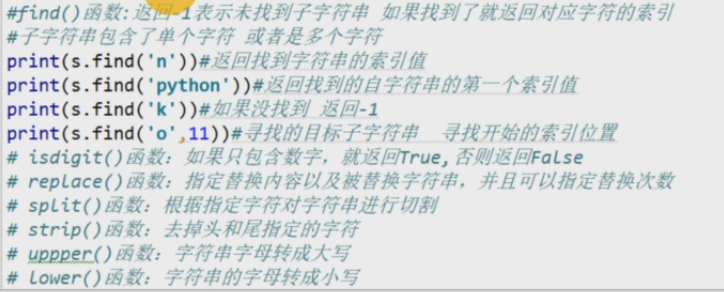


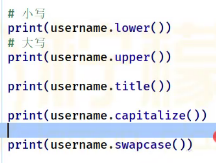
Age:{:.2f}可以精确小数点

f表达式格式化输出。花括号中写变量，可以加变量索引，可以使用函数



1. 字符串的内置函数





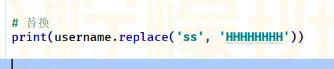
查找元素



统计字符出现的次数



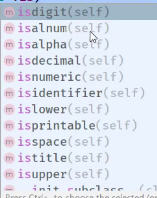
替换指定内容，先写old再写new，可以指定替换次数，按顺位替换，会生成新的字符串



判断是不是正整数，返回bool



扩展



字符串拼接，()可以加列表、元祖等，数量类型只能是字符串；

字符串拼接到，中





剔除去掉，返回新字符串



切割字符串，可以指定切割次数，以字符串中含有的字符作为切隔符，字符直接被去掉用英文逗号表示。返回的值是列表，这是字符串的内置函数

Split()，可以用来input接收进行切割成列表

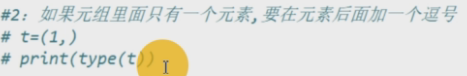


1. input()得到的数据类型永远都是字符串str
2. 取索引和切片的时候，输出必须加字符串标识符

**8月23日**

1. 元祖,()表示，可存储多种数据类型，只是一个容器

元祖里只有一个元素，在要元素后面加一个逗号(1,)



不加括号，直接加逗号也是元祖

A = 1,2,3

1. 元祖解包，就是通过索引取值

a,b = 2,3 a,b = (2,3) 函数传参加\*

比如：

a = (3,4), 放到函数 sum(\*a) 就是表示 sum(3,4)  
把外面的括号去掉了。



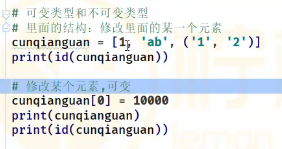
1. 列表，可变类型，[]表示，和索引有区别前面没东西

可做增删操作

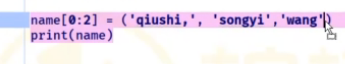
1. 列表修改元素

可变类型和不可变类型，可以去修改里面的某一个元素

列表里的任何整体元素都可被修改



注意：给两个索引位置，传三个值也可以修改。占的坑位总会被替换掉



5、列表添加元素

在列表最后添加一个元素，最多一个

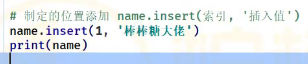


在列表最后添加多个元素，只能添加在后面，可以添加元祖、列表

必须要将多个值整合成一个整体

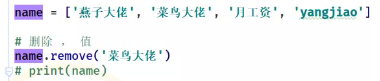


指定位置添加元素，只能一个一个插入。第一个是索引，第二个是要插入的值

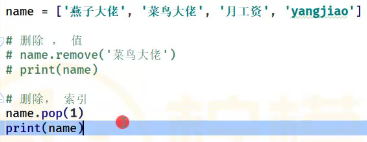


6、列表删除元素

根据值删除元素，只能输入值，删除找到的第一个

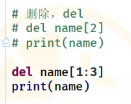


根据索引删除元素，不输入索引默认删除最后一个元素



根据del关键字删除，一次性可删除多个

Del可以在内存中删除数据



删除整个列表的元素



7、列表的内置函数

List.Index获取索引值 str.index检测B字符串是否包含A字符串

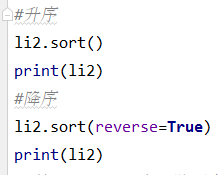
Count统计

倒序



排序，同一种数据类型才能排序





复制列表，ID不一致，切片复制也不一致





1. 字典，{‘key’:’value’,’key2’:’value2’}

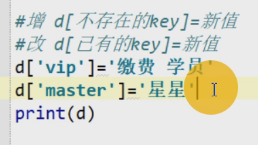
把key值作为输入的值，value作为得出的值 比较好

通过key来取值，保证key的唯一性，字典是无序的

扩展内容：key必须是不可变类型，如列表不能作为key值

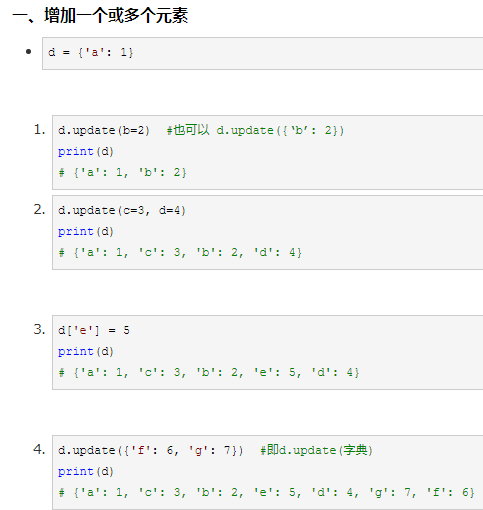
1. 字典的增和改，变量[key]=新value

只能添加单个key



添加和修改多个key键值对

Update()可以添加和修改多个key键值对，()是已有的key则是修改，()是没有的key则是新增。



1. 字典的删除



为什么不方便删除多个元素，无序性。存的内存地址不是连续性。

1. 字典的内置函数

获取所有的key，打印出来类似列表，但不是列表



获取所有的values



获取所有的键值对，元祖存储



获取某个key的值。Get可以指定一个没有的key的默认值





**8月26日**

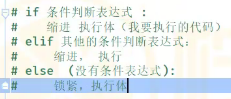
1. 集合

集合的表示 {} 关键字：set

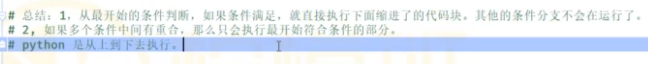
集合重要的特征：没有重复元素，去重；无序

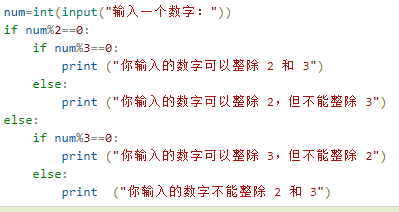
集合的作用：可以给列表去重

1. if elif else条件控制

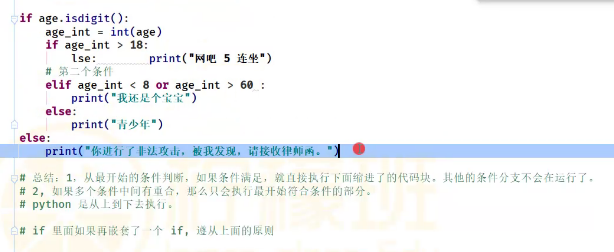


总结：if条件满足则执行if，否则下跳判断elif条件满是否满足，满足执行elif，否则直接执行else。 else下面还可以接if

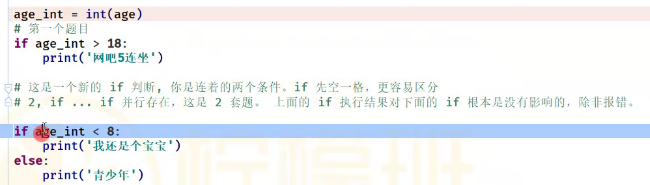


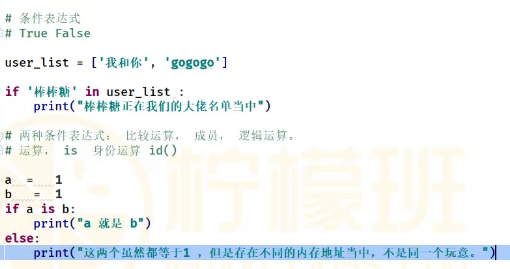


嵌套

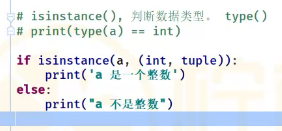


两个if并行存在，遇到冒号(:)要缩进

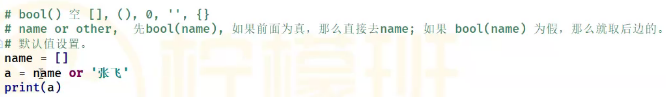


判断条件

判断数据类型 a是一个int或者tuple，都OK

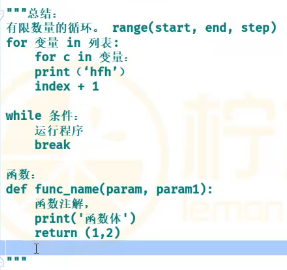


1. 默认值设置



1. if 2，光突突的什么都不写，实际上运行了bool(2)布尔判断
2. 判断数据类型是否为正整数内置函数：isdigit()

**8月28日**



For循环：知道循环次数，知道要循环几次

1. for循环有什么用：遍历：依次访问in后面数据类型里的每一个元素

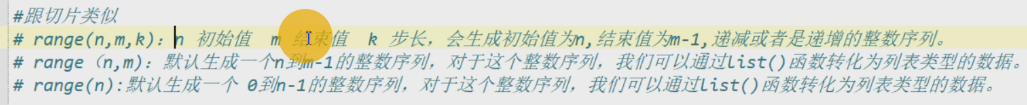
获取每一个元素去进行进一步操作

1. for循环使用规则：for 变量 in 要遍历的数据(列表、元祖、字典、字符串)

循环体。。。

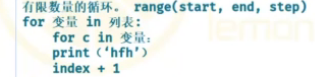
把要’遍历的数据’的每个元素赋值给’变量’

1. 原理：会自动添加索引，索引+1
2. range类，(起始位置,终止位置,步长)，不用手动写很多元素，等于把数据存储到列表中；遍历数字序列可以用；range(len())还可以遍历序列的索引



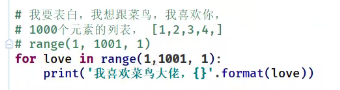
1. 不要再列表 for循环的时候去进行列表的删除、添加、修改操作；如果要就去维护新的备份
2. For循环嵌套逻辑：先执行第一个，然后把第二个所有循环执行完以后，再次执行第一个，然后再执行第二个所有循环……（自己的理解）

2、先执行里面的，再执行外面的



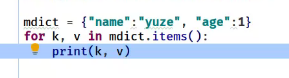






For循环字典，打印的是key值，需要在变量名后面+values。

打印key+values



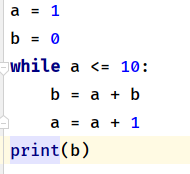
1. while循环：不知道循环次数，知道条件结果，用while；知道的话用if

While 判断条件:

条件满足要循环的代码（跟if的判断条件一致）

引入的变量，让语句跳出死循环（此限制条件要放到最下方，用来限制while循环，其他要得到的值应该放到限制条件的上方）

Print() 这输出是把上面的循环语句得到变量中得到的值输出出来，而不是每次循环一次就输出一次值

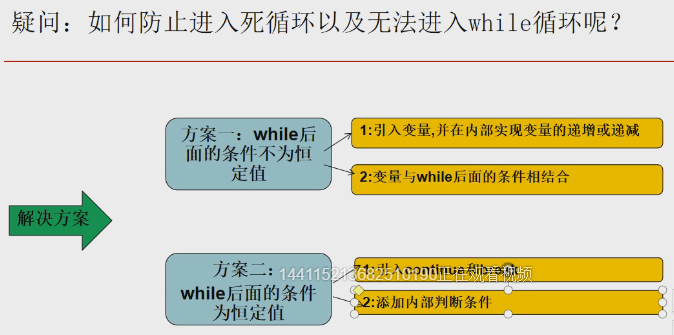




执行逻辑：先判断条件，再根据条件的值决定是否执行下面的代码块；执行完毕之后会再次判断，再执行。

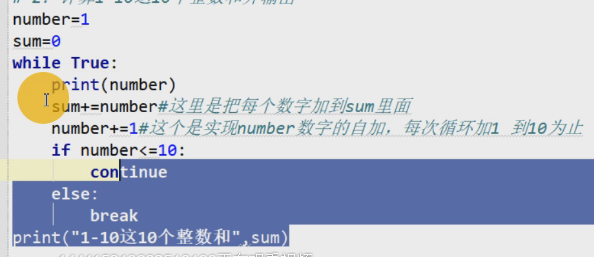
值：True 死循环；False 不会执行while下面的代码块；打印空白

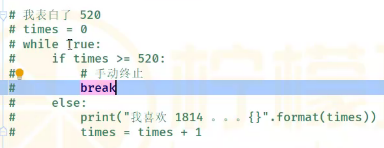
防止进入死循环：



如何选择：1、知道循环次数时，推荐选择方案一，使用方案一总要设置一个变量来限制循环次数；

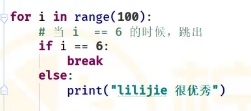
2、不知道循环次数时，只要达到特定条件（只知道判断条件），就终止循环，推荐选择方案二



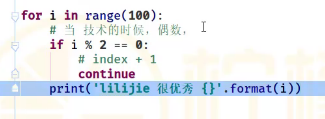




1. 终止循环：break（跳出去）

打印6次

1. 手工跳到下一个循环：continue



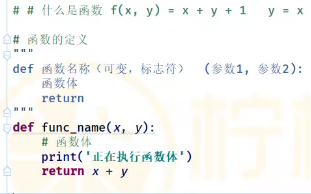
总结：break 语句用于退出循环，continue 语句可以终止当前次循环。

1. 函数

函数之间的参数调用，用return返回值 然后调用此返回值，要么就把变量定义成全局变量

函数用变量接收

函数的定义



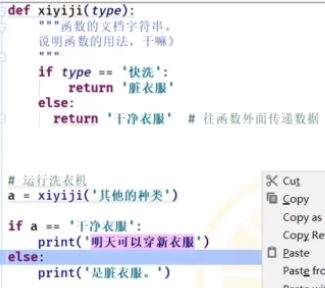
函数的作用：存储一段可执行的代码程序，可直接调用函数

一般函数都要带参数



Return作用：1、输出（拉粑粑）返回值给函数外面的程序，获取值；直接得出结果

1. 函数体遇到return就终止，函数体需要写到return的上面



判读return返回值是什么，用变量接收再打印出来



形式参数：变量，在函数定义的时候的参数，占坑，代替具体的数字的标记

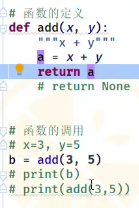
本质上来说是一个局部变量

实际参数：函数定义的时候占坑的实际值（常量）

1. 形式参数有多少个，实际参数就有多少个，a,b=2,3

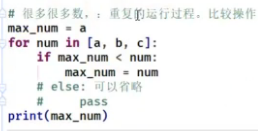
位置参数：位置要一一对应

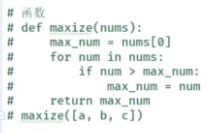
函数的调用



**09月02日**

套路一：多个元素比较求max/min，两个元素比较后可以直接max/min，这是赋值给另一个变量，那么这个变了代表max/min，这个值是不变的，然后进行多次比较。





套路二：一组数据同时用两次就要用两个for循环

1. 写函数要带参数，要加return返回值

关键字参数：可以改变参数的执行顺序；更加具有可读性；调用的时候a=3



关键字参数一定要放在位置参数的后面；写关键字参数的时候位置参数还是一一对应的

默认参数：1、函数定义的时候给的默认值；定义的时候a=3

1. 函数调用时，优先去实际参数，没有实际参数，就取默认参数
2. 默认参数要放到位置参数后面

动态或不定长参数（一个\*号）：\*args接收剩下的，不是必传的位置参数

加了星号 **\*** 的参数会以元组(tuple)的形式导入——类似元祖解包

\*要放到所有位置参数后面

形式参数未定义不定长参数时，实际参数也可以传入不定长参数

关键字参数的不定长参数（两个\*\*号）：\*\*keywords接受剩下的关键字参数

加了两个星号 **\*\*** 的参数会以字典的形式导入

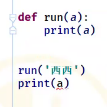
Pass：直接通过，什么都不执行，代码中都可以写

变量作用域

局部变量：函数体里定义的变量

#局部变量存储在locals()中

局部变量不能在函数体外使用



全局变量：函数体外面的，文件层级的变量；可以直接在函数体里使用，如a=3，a就是全局变量；也可以在函数体外面使用

全局变量存储在globals()中

如果局部变量和全局变量同名，优先局部变量

修改变量：

1. 函数体里面不能修改全局变量，但是函数体中写global a 就可以在函数体中修改全局变量，使用global就是指明了这是全局变量。不要轻易使用global

函数体里声明了使用全局变量，定义函数参数时，就不要添加此参数了

1. 函数体外面也不能修改局部变量

Def a(b,c):

d = b \* c

Return d

09月04日

套路三：判断条件如果全是==，将判断条件中的值定义成字典 return直接输出结果+key值变量

1. 可变类型不要作为默认参数：列表、字典等
2. eval 1+2 可以得到3

模块（可以引入函数、类、变量；引入的都是顶格写的内容

1、Import+包.模块名（pycharm上文件里路径），模块名+函数名（参数）；多用于导入python内置的模块或第三方库

2、import+包.模块名 as 别名，别名+函数名（参数）

3、from+包.模块名+import+函数名，函数名（参数）；

From+包+import+模块，模块.函数名（参数）

From+包+import+模块 as 别名，别名.函数名（参数）

4、As 使用场景：1、模块名过长 2、多个模块下有同名的函数名

5、from+包.模块+import \* 引用包下面模块下面所有的方法

\_\_name\_\_

\_\_main\_\_：主程序，程序的入口，你运行的哪个文件，哪个文件就叫\_\_main\_\_

If \_\_name\_\_ == ‘\_\_main\_\_’:用于调试

Q3: 在module2 中定义一个 func\_name 会存在什么问题？  
# 此问题从两个维度考虑——导包方式、func\_name定义位置  
# （1）若导包方式只导入到模块，没有导入到对应函数，那么调用函数方式为 模块名.函数，此种情况不论func\_name在当前模块中定义  
# 在哪里执行的都是在module1的func\_name函数  
# （2）若导报方式导入到函数名，那么调用函数的方式，直接写函数名就可以：  
# 此时，若module2中新定义的‘func\_name’在函数调用前，那么调用的是当前module2中新定义的func\_name函数  
# 若module2中新定义的func\_name在函数调用后，那么调用的是module1中定义的func\_name函数

列表转为字典的方法：



With open 。。。 as。。。:

代码

打开文件，执行完代码后，自动关闭文件

**9月6日**

路径处理：

Import os

1. 获取文件夹路径（你在哪里运行的文件，就获取这个文件的文件夹）

Os.getcwd()

2、获取文件绝对路径，用的非常多

os.path.abspath(\_\_file\_\_)

3、获取文件的文件夹路径 dirname是变量

Os.path.dirname(os.path.abspath(\_\_file\_\_))参数可赋值给变量

4、路径拼接

Os.path.join()

5、创建文件夹，只能一个一个创建

Os.mkdir(path)

创建多层目录

os.makedirs(path)

6、判断路径是否存在，结合if使用

Os.path.exists() 返回布尔值

7、判断是否为文件夹

Os.path.isdir()

8、判断是否为文件

Os.path.isfile()

文件操作（读写）

1. 打开文件同时可创建文件，encoding编码格式，mode打开文件的模式（‘w’写，’r’读默认参数）

open(path,encoding=’utf-8’) 必传encoding参数

Mode为’w’会覆盖以前写入的内容

Mode为’a’，写入内容不会被覆盖，是追加模式

Mode为’x’，创作模式，需要传入未被创建的文件名

r+ w+ a+ x+ 读写 写读 第一个操作是什么就写什么

没有传绝对路径，就在当前路径下创建打开文件

打开文件后一定要记得关闭，忘记关闭时（自动关闭）：





With open() as a

A.read()

1. read()读取文件的内容

Print(文件名.Readlines()）读取文件每一行，并且以列表格式打印到控制台，结合for循环可把每行内容打印到控制台

3、write(‘啊啊’) 写文件

4、close() 关闭文件

异常处理

Try:

需要执行的代码

Except:（错误类型，写入错误代码报错的错误提醒）

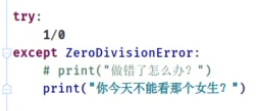
exception表示所有的错误类型，建议不要使用

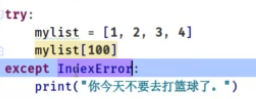
做错之后执行的代码

坏处：不好定位问题

Finally：无论有没有异常 都会运行下面的代码

自定义异常：raise + 异常名称



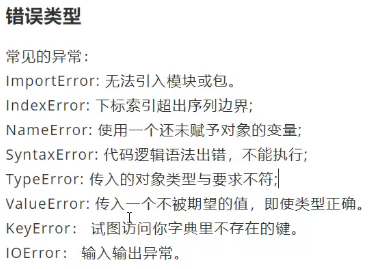


具体的错误类型对应具体的错误提醒

Try遇到错误代码就终止分支，分支里所有的代码都不会执行，直接跳到except

可以写多个except，第一个满足条件就执行第一个，不满足条件就执行下一个except，依次类推

错误类型 syntaxError 不能捕获



内置函数

1、enumerate() 括号带变量名，可同时查询元素的索引和元素值

2、sum() 求和

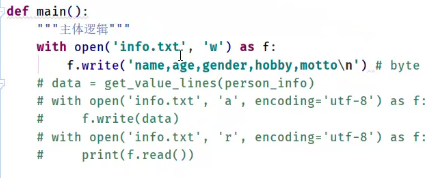
3、len() 判断长度

4、max() 求最大值 min()求最小值

09月09日

PEP8格式优化

1. 打开文件操作一步，再打开文件操作一步，建议一步一步去操作



Debug：

1. step over 单步执行
2. Run to curs
3. Evaluate 写表达式 查看变量 还可以操作

4、进入调用体 debug查看 step into

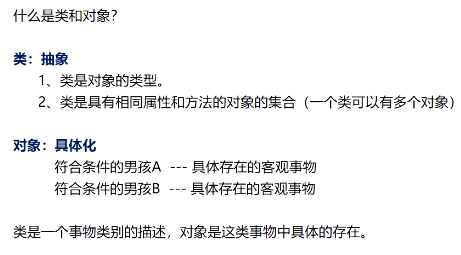
字典解包



类和对象

类变量、实例变量（类属性、实例属性） \_\_init\_\_初始化

Self\cls我的



实例化对象：变量名=类名() —— 实例化了一个对象

每个对象都有类下面的属性和方法

类的属性，所有对象全都通用

类：

共同点 组成类。 类名驼峰

每个类 特征（形容词） 行为（名词）

Class 类名:

获取类属性 类变量值

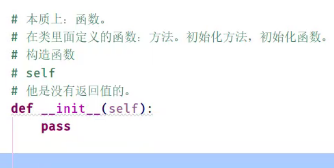


类里面定义的函数叫方法 \_\_init\_\_(魔法函数)是初始化方法，是类下面实例化对象的初始化属性，然后在行为(增删改查)中可以调用对象的初始化属性

在实例化的时候自动调用\_\_init\_\_(self,name) 变量名 = 类名(name) 自动调用魔法函数\_\_init\_\_ 其余自己定义的函数变量.函数名()

Self(我的）:对象实例本身 在类外面用变量 实例化都要self

Init函数没有返回值

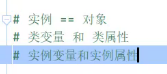


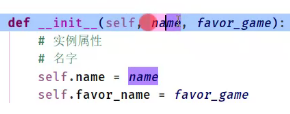


对象：

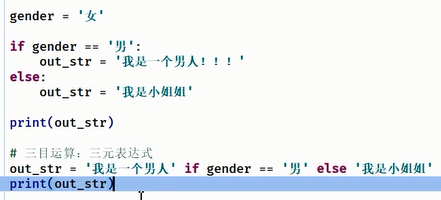
类下面一个具体的个体

实例属性：实例变量 实例对象

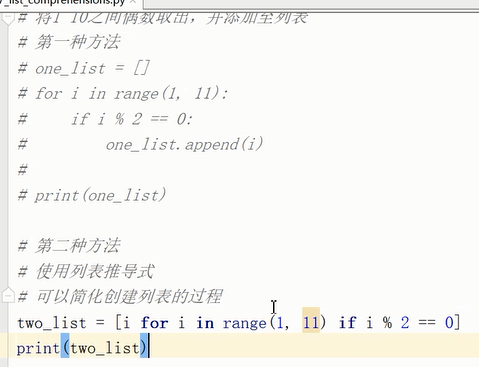
（同一回事）



三目运算



列表推导式 for循环与if在列表里结合使用



9月11日

类的变量和类变量不是一个概念，类变量包含了类变量和实例变量

对象的实例化就是把类（带参数） \_\_init\_\_ 赋值给一个变量



类外面调用类变量，对象都具有类属性

1、类名.类变量名

2、实例变量名（对象）.类变量名

类外面调用实例属性

1、实例变量名（对象）.实例属性名

实例方法用类变量，可以不定义参数变量名

类方法（很少用） cls:类本身

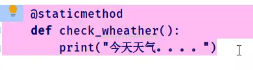
@classmethod（声明这是类方法，不声明就被当成实例方法用了）

Def a(cls):

Print(cls.类属性)

静态方法

它和类、对象没有关系，但是有关联（独立），它和普通函数没有区别，实现非此类特有的功能。它没有必要调用类属性，要调用类属性直接定义成类方法



实例方法中共同的参数，可以初始化成实例属性。

初始化参数写在类括号里，实例参数写在实例方法括号里。

9月16日

类的继承

1. 子类括号带父类的类名即继承，可以用父类中的所有方法属性
2. 函数名同名会覆盖，上面的函数，不要定义重名函数

超继承

函数同名情况下，如果都想用，就在子类函数中加上super().子类函数(父类同名函数的参数)

Super()不会调用\_\_init\_\_函数



父类有\_\_init\_\_需要传参，再调用实例方法，Mobile类名



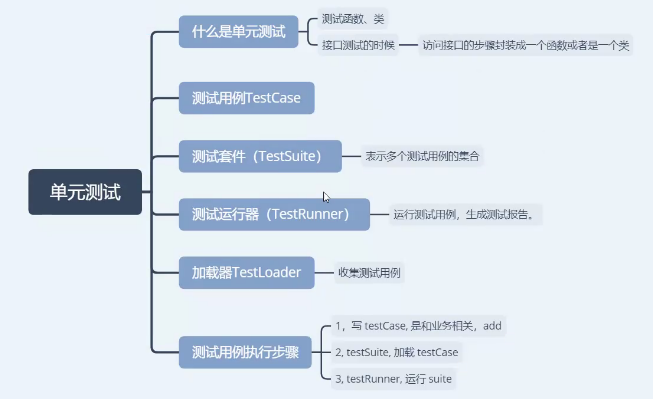
多重继承

在括号里多写几个类名

重写，从左至右的类名优先调用类里面的方法



单元测试



断言 assert 判断条件

Unittest py自带的单元测试框架



Expected预期结果

9月18日

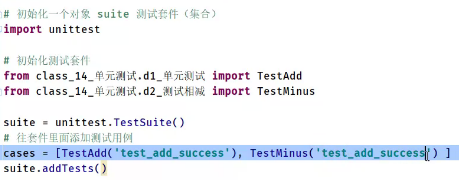
1. Unittest.main() python+文件名 执行此测试用例文件
2. 右键执行测试用例文件
3. if name == main 右键运行

assertTrue(条件表达式) assertEqual(a,b)逗号

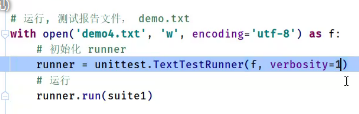
.通过 F不通过

Python -m unittest 模块.测试类名.测试方法 执行单个测试

将测试方法作为参数添加



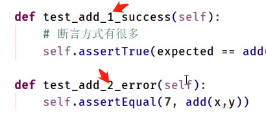
打开文件生成测试报告



verbosity表示详细程度

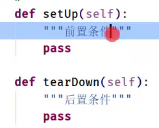
Stream几乎是文件 descriptions描述

方法加数字调节执行顺序

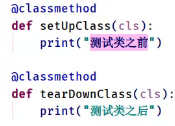


前置条件（执行用例之前自动运行）、后置条件（执行用例之后自动运行）

每执行一次测试方法就执行条件



类条件只执行一次



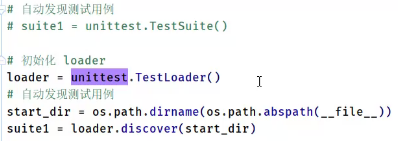
添加测试用例

1. 一个一个方法添加
2. 类添加
3. 模块添加

TestLoader 用来加载测试用例的



自动发现测试用例



根据时间生产测试报告名



9月20日

<>表示对象

Python操作excel，excel操作

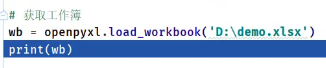
导入openpyxl，操作xlsx

Xlrd（读）、xlwt（写），操作xls

Tablib，都可以操作（复杂）

步骤：

1. 打开excel，获取工作薄



1. 打开（获取）表单，sheet 所有表单或索引取单表单



根据名字获取，第二种方法类似于字典的key值

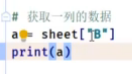


1. 获取单元格信息 一个对象代表一个单元格的数据

获取一个单元格信息，字符串



获取一列数据，(<B1>, <B2>)



获取一行数据，(<A1>, <B1>, <C1>) 元祖嵌套对象

获取多行数据，((<A1>, <B1>), (<A2>, <B2>)) 元祖嵌套元祖再嵌套对象

一个嵌套的元祖代表一行或一列的数据

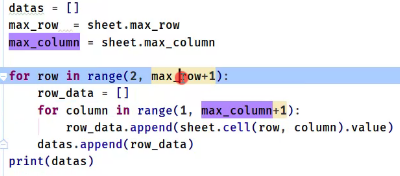


获取所有行的数据 剔除标题。两种方法

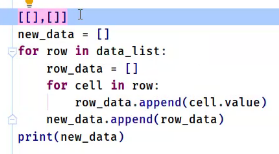


Data表示生成器，把生成器数据转换成列表data\_list

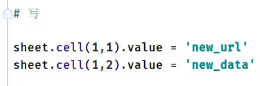
获取最大行数和最大列数，再获取所有行和列数据



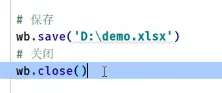
提取单元格信息



写操作



保存、关闭



9月23日

If...raise ValueError 主动抛出异常

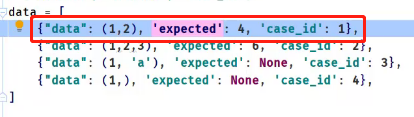
Unittest 一条测试用例对应一个测试方法，然后再优化

运行unittest，run的时候

If \_\_name\_\_ == ‘\_\_main\_\_’”

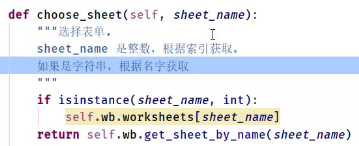
Unittest.main()

函数体传元祖、列表，直接解包 前面加\*



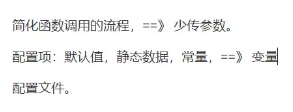


Isinstance(参数, 数据类型）是否成立



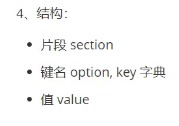
配置文件：

配置项和配置文件的作用

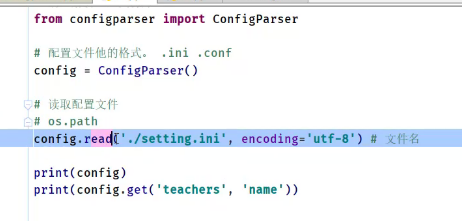


相对于项目来说常量 全部大写的标识符

配置文件的结构



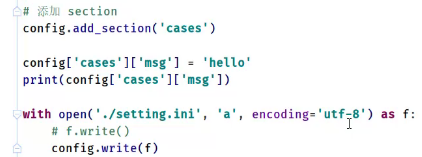
获取配置文件的信息





前面加Eval可以获取到想要的数据类型，写进去的是什么类型得到的就是什么类型，脱衣服

写入section



9月25日

日志 log

步骤：

初始化收集器  
设置收集器日志级别  
初始化日志处理器  
设置处理器日志级别  
处理器添加到收集器  
设置输出格式  
处理器关联输出格式

Import logging



完整的用例：





Logger = logging.Logger(‘aaa’, level=)可以初始化和设置级别一步到位

收集器只会收集定义的级别以上的错误级别

#初始化文件处理器之后定义处理器收集错误级别

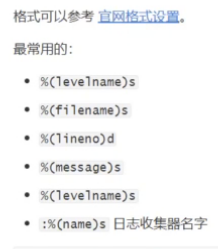
Concole\_handler.setLevel(logging.ERROR) 控制台显示日志，调试级别设置低

File\_handler.setLevel(logging.ERROR) 文件里显示日志，正式的级别设置高

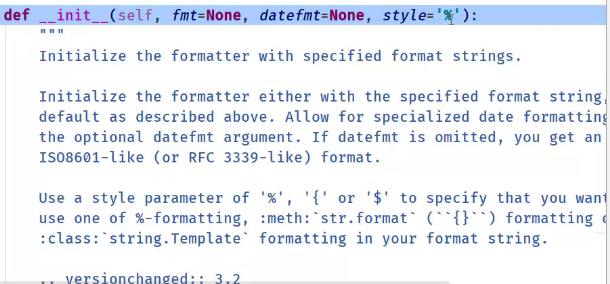
处理器就是表示在哪里显示log信息

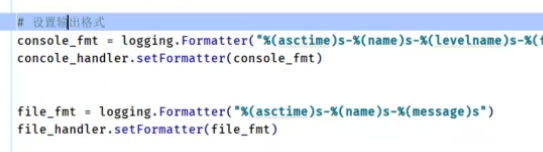
展示出来的是处理器设置的错误级别，日志收集器设置的错误级别只会被收集起来，不一定会展示出来

设置格式









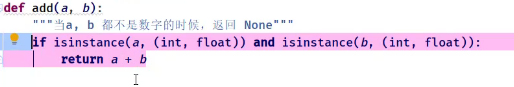
9月28日

DDT

搭配unittest起来使用

10月9日

Isinstance，判断a和b是不是整型或浮点型



10月11日

接口

什么是接口：连接了两个不同的事物、系统，可以进行数据传输；公开访问的函数才能是接口；类也可以是接口

UI：user interface用户接口，用户界面

API：application programming interface，应用程序可编程接口

客户端：前端，提需求，主动请求；不是固定的

服务端：后端，提供服务，被动接受

接口协议：双方约定的规则

HTTP：请求（请求首行、请求头、请求体）和响应，

请求头：user-agent用户代理，手机、浏览器，cookie

http协议的底层协议TCP/ip，tcp是用来打包存储数据的，ip包接收的地址

Url地址域名，ip地址。域名地址好记，映射ip地址

Dns解析就是把域名地址转化成ip地址

端口号：服务太多了需要进行区分

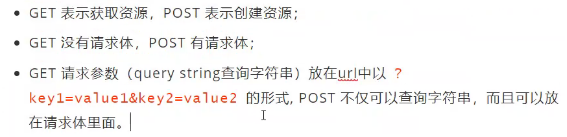
Https协议更安全，要安装证书

其余协议：Websocket、webservice

请求方法



GET和POST的区别



Get请求的参数直接放到URL中，？（问号）后面就是传的参数（query string查询字符串）

Body传参方式：form、json

Form表单就是键值对

Json标准化格式的字符串

响应的code（状态码）

2xx：成功。post返回201表示资源创建成功

3xx:重定向，跳转的其他的url上。301永久重定向，302临时重定向

4xx：客户端报错，请求报错可能传参问题。404找不到资源；401没有访问权限（如未登录情况）；403禁止访问（没有更高权限）；405方法不允许（如不允许post请求，但你发起了post请求）

5xx：服务端报错

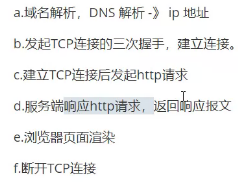
10月14日

Cookie 缓存，服务端生成返回的，类似会员卡，默认浏览器关掉就没有了；存储到浏览器客户端的

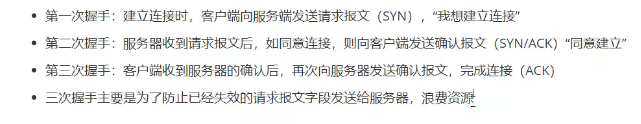
Session重要的信息记录到服务器（数据库），存储到服务端的，有效期服务端设置

Token跨平台，是会变的，保存在客户端本地，有效期服务端设置，一般保存在local storage

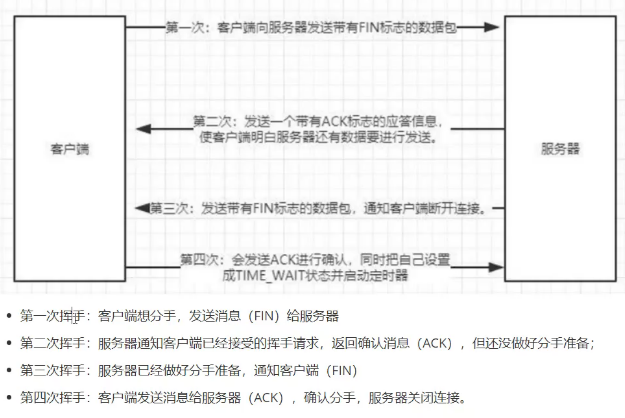
输入url后的过程



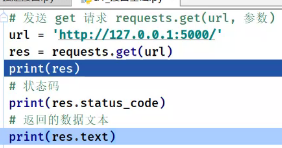
三次握手



四次挥手



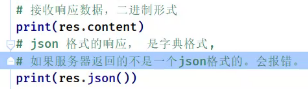
Requests第三方库，发送http请求的



加参数



Res.content.decode()讲二进制转换成字符串



带请求头



传递参数的三种方式，作为参数（写在=号前面）

Params

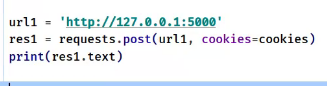
Data

Json

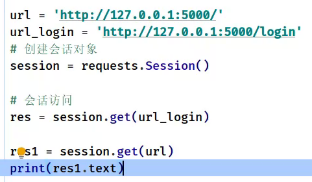
获取所有的cookie



传递cookie参数



Session会话管理cookie



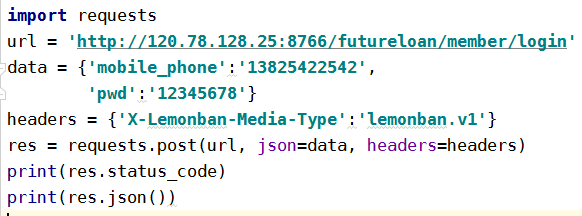
总结

Get只能通过params去传递参数，因为get没有请求体



Requests.get和s.get没有区别

请求，参数和头要分开接收



10月16日