# Python

## Tips

print(res['data']['base']['tag'])

if '4A景区' in res['data']['base']['tag']:

res = '4A景区'

if '5A景区' in res['data']['base']['tag']:

res = '5A景区' 错

用elis就是对的 连着两个if就不行

以value倒排的代码 lamba表达式 python3

day\_dict = city\_dict[city]

for day, tag\_dict in sorted(day\_dict.items(), key=lambda item: item[0]):

for tag, num in sorted(tag\_dict.items(), key=lambda item: item[1], reverse=True):

fw.write(city + "\t" + day + "\t" + tag + "\t" + str(num) + "\n")

pyhotn3 同一个文件 要么都用tab 要么都用空格 同一行也是要么都用tab 要么都用空格 不要混着用会报错

python 的json的\\替换和非json的\\替换是不一样的

如果只是普通的字符串\\n 替换成空 replace 需要\\\\n四个\

如果这个字符串被json反序列化后, 在json字段里面替换只需要\\n即可 是不需要再转义的

python map里的set list 是这个指针不会变, 扩容也不变 而是要先拿出来的操作 不用放回去

python主线程退出后 如果还有线程没执行完 进程是不会退出的 t1.join 就是 卡在这里 等t1执行完了再让主线程走后面的

# -\*- coding: utf8 -\*- 这个必须放第一行

Python2中默认的编码格式是 ASCII 格式，在没修改编码格式时无法正确打印汉字，所以在读取中文时会报错。

解决方法为只要在文件开头加入 # -\*- coding: UTF-8 -\*- 或者 # coding=utf-8 就行了

Python3.X 源码文件默认使用utf-8编码，所以可以正常解析中文，无需指定 UTF-8 编码

pip list 查看安装了多少包

pip3 show 包名 查看版本和位置

start

('lail,'('lail,', 3),

python的print不是行原子的 多个线程打印 一行之内也会被污染

with open(filepath + "/display\_father\_son\_ratio", 'r') as f:

for gline in f:

unicode\_str = gline.decode('gb18030')

line = unicode\_str.encode('utf-8')

意思都是先弄成unicode 然后在转存别的格式 字符串自带 decode和encode



def f(\*arg1, \*\*arg2):

for key in arg1:

print(key, "yyyyyyyy")

for key in arg2:

print(key, arg2[key], "wwwwwww")

def main():

f(1,2,3,4, a=1, b=2)

af.write("{}\_{}\t{}\n".format(info[0], info[1], info[2])) 这样的 format 字符串直接调用 {}

python 不能直接用key 得先判断下在不在 直接用不在的key会报错 无论get 还是del

a = {

"a": "b"

}

if ('a' in a) & len(a) > 0:

print("1111") 括号扩一下

列表[-1]是倒数第一个的意思

For x in x:

collection = {1, 2, 3, 4, 5} 集合 空集合定义要 s = set() 不能s = {} 这样会被当做dict

Python中pass的作用与使用教程

Python中pass的作用

空语句 do nothing

保证格式完整

保证语义完整

## 类型

在 python 中，类型属于对象，对象有不同类型的区分，变量是没有类型的：

a=[1,2,3]

a="Runoob"

以上代码中，[1,2,3] 是 List 类型，"Runoob" 是 String 类型，而变量 a 是没有类型，她仅仅是一个对象的引用（一个指针），可以是指向 List 类型对象，也可以是指向 String 类型对象

也就是py中没有基本数据类型 全是对象

可更改(mutable)与不可更改(immutable)对象

在 python 中，strings, tuples, 和 numbers 是不可更改的对象，而 list,dict 等则是可以修改的对象。

不可变类型：变量赋值 a=5 后再赋值 a=10，这里实际是新生成一个 int 值对象 10，再让 a 指向它，而 5 被丢弃，不是改变 a 的值，相当于新生成了 a。

可变类型：变量赋值 la=[1,2,3,4] 后再赋值 la[2]=5 则是将 list la 的第三个元素值更改，本身la没有动，只是其内部的一部分值被修改了。

python 函数的参数传递：

不可变类型：类似 C++ 的值传递，如整数、字符串、元组。如 fun(a)，传递的只是 a 的值，没有影响 a 对象本身。如果在 fun(a) 内部修改 a 的值，则是新生成一个 a 的对象。

可变类型：类似 C++ 的引用传递，如 列表，字典。如 fun(la)，则是将 la 真正的传过去，修改后 fun 外部的 la 也会受影响

python 中一切都是对象，严格意义我们不能说值传递还是引用传递，我们应该说传不可变对象和传可变对象。

## 面向对象

## 参数

Python中有三种类型的参数：必备参数、默认参数和关键字参数。它们之间的区别如下：

必备参数（positional arguments）是函数定义中要求传递的参数，调用时必须提供对应的参数值，并按照顺序传递。如果不提供或提供错误的参数，则会抛出 TypeError 异常。例如：

def my\_func(a, b):

print(a + b)

my\_func(1, 2) # 输出 3

默认参数（default arguments）是在函数定义时给定的参数值，当调用时没有提供对应参数的值时，将使用该默认值作为参数值。默认参数可以使函数调用更加灵活，但必须放置在所有必备参数后面。例如：

def greet(name, greeting='Hello'):

print(f'{greeting}, {name}!')

greet('Jack') # 输出 'Hello, Jack!'

greet('Lisa', 'Hi') # 输出 'Hi, Lisa!'

关键字参数（keyword arguments）是根据参数名来设置参数值的一种方法。它们与位置无关，只需指定参数名称和相应的值即可。这允许您跳过某些默认参数并直接为必备参数和其他关键字参数提供值。例如：

def my\_func(a, b, c):

print(f'a={a}, b={b}, c={c}')

my\_func(c=3, a=1, b=2) # 输出 'a=1, b=2, c=3'

总之，必备参数是函数所需的传入参数，不可省略；默认参数是在调用时可选的，如果没有提供，则将使用默认值；而关键字参数允许您按名称提供特定的参数值，并可以与必备参数和默认参数混合使用。

## 异常



还有上次说的KeyError都是异常

反正那些报错都是异常





