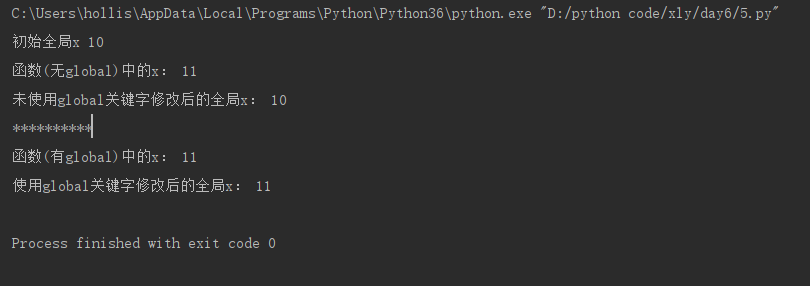
1. 把今天讲的列表，元组，字典，字符串的所有接口基本功能练习一遍，并用思维导出总结好（不清楚哪些是基础功能，就是照着上课代码练习一遍）
2. 编写代码理解 局部变量与全局变量

x=10  
  
def modify\_x():  
 x=10  
 x+=1  
 print("函数(无global)中的x：",x)  
  
def modify\_gx():  
 global x  
 x+=1  
 print("函数(有global)中的x：",x)  
  
print("初始全局x",x)  
modify\_x()  
print("未使用global关键字修改后的全局x：",x)  
print('\*'\*10)  
modify\_gx()  
print("使用global关键字修改后的全局x：",x)



1. 求两个有序数字列表的公共元素

import random as r  
s=[];s1=[]  
for i in range(10):  
 s.append(r.randint(0+10\*i,10+10\*i))  
 s1.append(r.randint(0+10\*i,10+10\*i))  
  
print("s",s,"\ns1:",s1)  
print("两者的公共元素为：",set(s).intersection(s1))

1. 给定一个n个整型元素的列表a，其中有一个元素出现次数超过n / 2，求这个元素

class Solution:

def majorityElement(self, nums)

votes = 0

for num in nums:

if votes == 0: x = num

votes += 1 if num == x else -1

return x

这里有力扣的详细解析

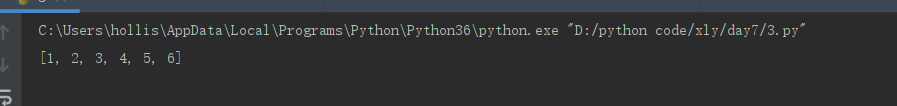
<https://leetcode-cn.com/problems/shu-zu-zhong-chu-xian-ci-shu-chao-guo-yi-ban-de-shu-zi-lcof/solution/mian-shi-ti-39-shu-zu-zhong-chu-xian-ci-shu-chao-3/>

1. 列表、元组，字典的相同点，不同点有哪些，请罗列

均为容器，列表、元组均可容纳不同的数据类型，相比于列表，元组不可原地更改（可以以切片的方式实现间接更改），可以看做一个被冻结的列表。字典以键值对作为元素，存储映射关系。

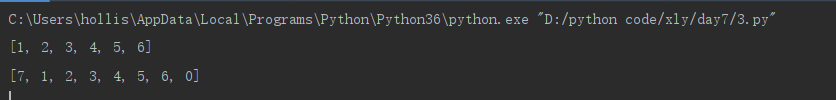
1. 将元组 (1,2,3) 和集合 {4,5,6} 合并成一个列表。

print(list((1,2,3))+list({4,5,6}))



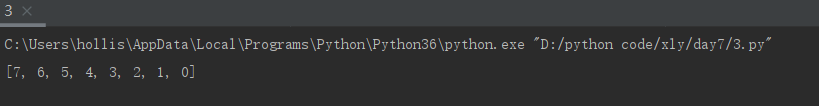
1. 在列表 [1,2,3,4,5,6] 首尾分别添加整型元素 7 和 0。

a=[1,2,3,4,5,6]  
print(a)  
a.insert(0,7)  
a.append(0)  
print(a)



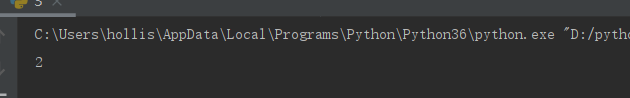
1. 反转列表 [0,1,2,3,4,5,6,7] 。

#5  
a=[0,1,2,3,4,5,6,7]  
a.reverse()  
print(a)



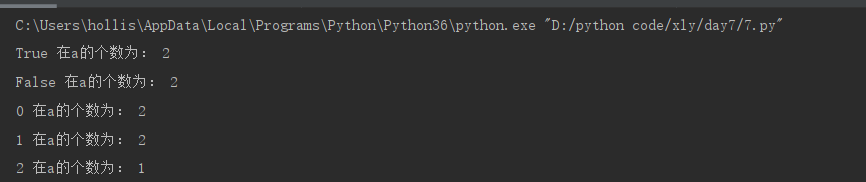
1. 反转列表 [0,1,2,3,4,5,6,7] 后给出中元素 5 的索引号。

#6  
a=[0,1,2,3,4,5,6,7]  
print(list(reversed(a)).index(5))



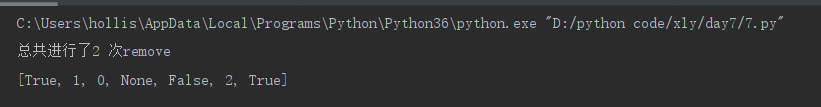
1. 分别统计列表 [True,False,0,1,2] 中 True,False,0,1,2的元素个数，发现了什么？

#python中True即1,0即False，用相应的关键字来替换  
a=[True,False,0,1,2]  
for i in a:  
 print(i,"在a的个数为：",a.count(i))



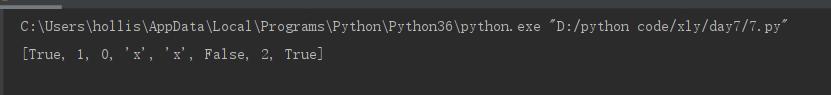
1. 从列表 [True,1,0,‘x’,None,‘x’,False,2,True] 中删除元素‘x’。

a=[True,1,0,'x',None,'x',False,2,True]  
ctr=0  
while 'x' in a:  
 ctr+=1  
 a.remove('x')  
print("总共进行了%d 次remove" % ctr)  
print(a)



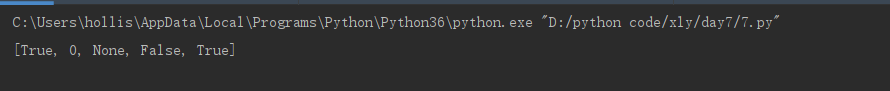
1. 从列表 [True,1,0,‘x’,None,‘x’,False,2,True] 中删除索引号为4的元素。

a=[True,1,0,'x',None,'x',False,2,True]  
a.pop(4)  
print(a)



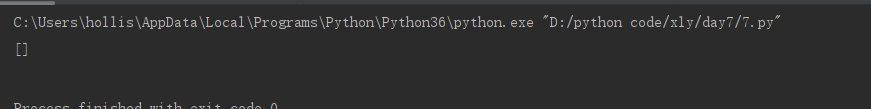
1. 删除列表中索引号为奇数（或偶数）的元素。

a=[True,1,0,'x',None,'x',False,2,True]  
b=[]  
for i in range(0,len(a),2): #步长为2，从0开始  
 b.append(a[i])  
print(b)



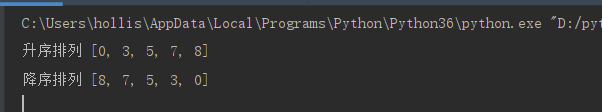
1. 清空列表中的所有元素。

a=[True,1,0,'x',None,'x',False,2,True]  
del a[:]  
print(a)



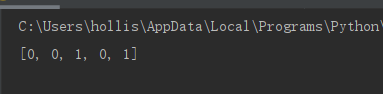
1. 对列表 [3,0,8,5,7] 分别做升序和降序排列。

a=[3,0,8,5,7]  
print("升序排列",sorted(a))  
print("降序排列",sorted(a,reverse=True))



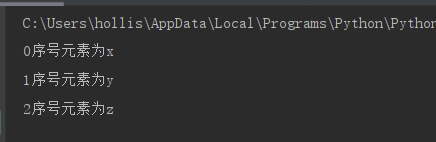
1. 将列表 [3,0,8,5,7] 中大于 5 元素置为1，其余元素置为0。

a=[3,0,8,5,7]  
a=[1 if x>5 else 0 for x in a]  
print(a)



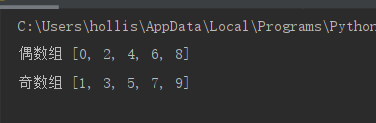
1. 遍历列表 [‘x’,‘y’,‘z’]，打印每一个元素及其对应的索引号。

a=['x','y','z']  
for i in range(len(a)):  
 print("{}序号元素为{}" .format(i,a[i]))



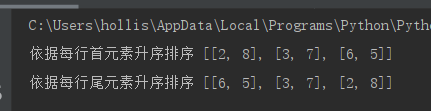
1. 将列表 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] 拆分为奇数组和偶数组两个列表。

a=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
l1=a[::2]  
l2=a[1::2]  
print("偶数组",l1)  
print("奇数组",l2)



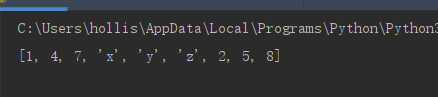
1. 分别根据每一行的首元素和尾元素大小对二维列表 [[6, 5], [3, 7], [2, 8]] 排序。

a=[[6, 5], [3, 7], [2, 8]]  
print("依据每行首元素升序排序",sorted(a,key=lambda x:x[0]))  
print("依据每行尾元素升序排序",sorted(a,key=lambda x:x[1]))



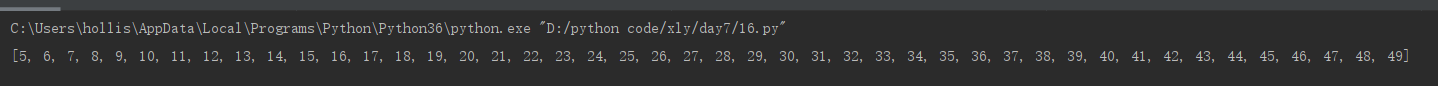
1. 从列表 [1,4,7,2,5,8] 索引为3的位置开始，依次插入列表 [‘x’,‘y’,‘z’] 的所有元素。

a=[1,4,7,2,5,8]  
b=['x','y','z']  
a=a[:3]+b+a[3:]  
print(a)



1. 快速生成由 [5,50) 区间内的整数组成的列表。

a=list(range(5,50))  
print(a)



1. 若 a = [1,2,3]，令 b = a，执行 b[0] = 9， a[0]亦被改变。为何？如何避免？----讲了深COPY和浅COPY再做

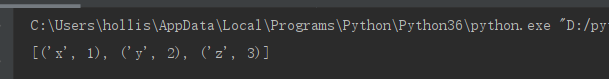
#对于复杂类型，如多层嵌套，import copy

#b=copy.deepcopy(a)

a=[1,2,3]  
b=a[:]  
b[0]=0  
print("a",a)  
print("b",b)

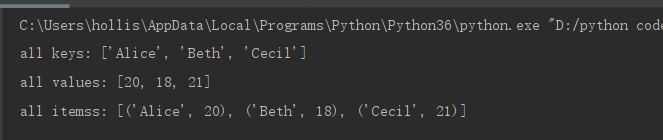
1. 将列表 [‘x’,‘y’,‘z’] 和 [1,2,3] 转成 [(‘x’,1),(‘y’,2),(‘z’,3)] 的形式。

a=[1,2,3]  
b=['x','y','z']  
print(list(zip(b,a)))



1. 以列表形式返回字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中所有的键。

a= {'Alice': 20, 'Beth': 18, 'Cecil': 21}  
print("all keys:",list(a.keys()))  
print("all values:",list(a.values()))  
print("all itemss:",list(a.items()))



1. 以列表形式返回字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中所有的值。

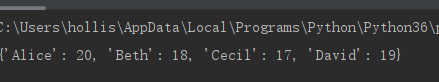
**见21题**

1. 以列表形式返回字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中所有键值对组成的元组。

**见21题**

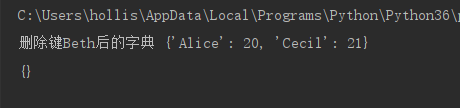
1. 向字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中追加 ‘David’:19 键值对，更新Cecil的值为17。

a= {'Alice': 20, 'Beth': 18, 'Cecil': 21}  
a['David']=19  
a['Cecil']=17  
print(a)



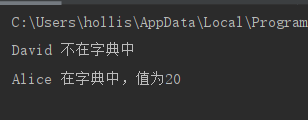
1. 删除字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中的Beth键后，清空该字典。

a= {'Alice': 20, 'Beth': 18, 'Cecil': 21}  
del a['Beth']  
print("删除键Beth后的字典",a)  
a.clear()  
print(a)



1. 判断 David 和 Alice 是否在字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21} 中。

a= {'Alice': 20, 'Beth': 18, 'Cecil': 21}  
keys=['David','Alice']  
for key in keys:  
 if a.get(key):  
 print("{} 在字典中，值为{}".format(key,a[key]))  
 else:  
 print(key,"不在字典中")

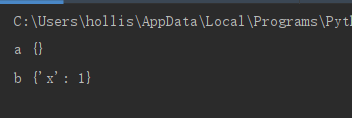


1. 遍历字典 {‘Alice’: 20, ‘Beth’: 18, ‘Cecil’: 21}，打印键值对。

a= {'Alice': 20, 'Beth': 18, 'Cecil': 21}  
for key,val in a.items():  
 print("key:{} value:{}".format(key,val))

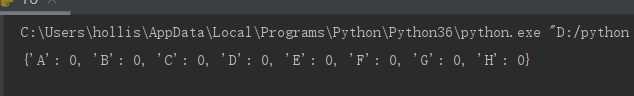
1. 若 a = dict()，令 b = a，执行 b.update({‘x’:1})， a亦被改变。为何？如何避免？----讲了深COPY和浅COPY再做

a={}  
b=a.copy()  
b.update({"x":1})  
print("a",a)  
print("b",b)



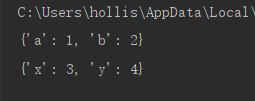
1. 以列表 [‘A’,‘B’,‘C’,‘D’,‘E’,‘F’,‘G’,‘H’] 中的每一个元素为键，默认值都是0，创建一个字典。

a={}.fromkeys('A B C D E F G H'.split(),0)  
print(a)



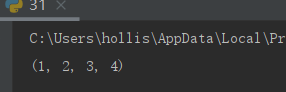
1. 将二维结构 [[‘a’,1],[‘b’,2]] 和 ((‘x’,3),(‘y’,4)) 转成字典。

a=[['a',1],['b',2]]  
b=(('x',3),('y',4))  
print(dict(a))  
print(dict(b))



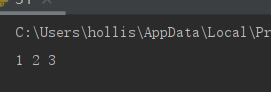
1. 将元组 (1,2) 和 (3,4) 合并成一个元组。

a=(1,2)  
b=(3,4)  
print(a+b)



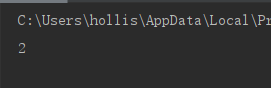
1. 将空间坐标元组 (1,2,3) 的三个元素解包对应到变量 x,y,z。

x,y,z=(1,2,3)  
print(x,y,z)



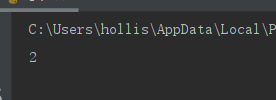
1. 返回元组 (‘Alice’,‘Beth’,‘Cecil’) 中 ‘Cecil’ 元素的索引号。

a=('Alice','Beth','Cecil')  
print(a.index('Cecil'))



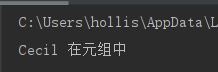
1. 返回元组 (2,5,3,2,4) 中元素 2 的个数。

a=(2,5,3,2,4)  
print(a.count(2))



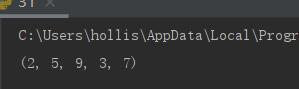
1. 判断 ‘Cecil’ 是否在元组 (‘Alice’,‘Beth’,‘Cecil’) 中。

a=('Alice','Beth','Cecil')  
if 'Cecil' in a:print("Cecil 在元组中")  
else:print("Cecil 不在元组中")



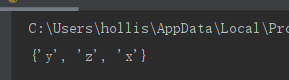
1. 返回在元组 (2,5,3,7) 索引号为2的位置插入元素 9 之后的新元组。

a=(2,5,3,7)  
a=a[:2]+(9,)+a[2:]  
print(a)



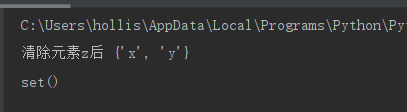
1. 创建一个空集合，增加 {‘x’,‘y’,‘z’} 三个元素。

s1=set().union(['x','y','z'])  
print(s1)



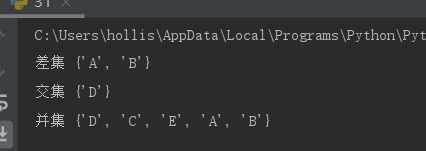
1. 删除集合 {‘x’,‘y’,‘z’} 中的 ‘z’ 元素，增加元素 ‘w’，然后清空整个集合。

s=set(['x','y','z'])  
s.remove('z')  
print("清除元素z后",s)  
s.clear()  
print(s)



1. 返回集合 {‘A’,‘D’,‘B’} 中未出现在集合 {‘D’,‘E’,‘C’} 中的元素（差集）。

a = set(['A', 'D', 'B'])  
b = set(['E', 'D', 'C'])  
print("差集",a.difference(b))  
print("交集",a.intersection(b))  
print("并集",a.union(b))



1. 返回两个集合 {‘A’,‘D’,‘B’} 和 {‘D’,‘E’,‘C’} 的并集。

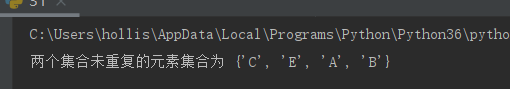
**见39题**

1. 返回两个集合 {‘A’,‘D’,‘B’} 和 {‘D’,‘E’,‘C’} 的交集。

**见39题**

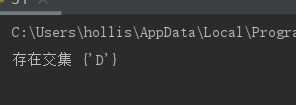
1. 返回两个集合 {‘A’,‘D’,‘B’} 和 {‘D’,‘E’,‘C’} 未重复的元素的集合。

a = set(['A', 'D', 'B'])  
b = set(['E', 'D', 'C'])  
pub=a.intersection(b)  
  
print("两个集合未重复的元素集合为",a.union(b).difference(pub))



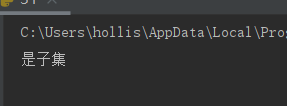
1. 判断两个集合 {‘A’,‘D’,‘B’} 和 {‘D’,‘E’,‘C’} 是否有重复元素。

a = set(['A', 'D', 'B'])  
b = set(['E', 'D', 'C'])  
pub=a.intersection(b)  
if a.intersection(b):  
 print("存在交集",pub)  
else:  
 print("不存在交集")



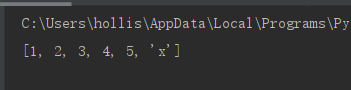
1. 判断集合 {‘A’,‘C’} 是否是集合 {‘D’,‘C’,‘E’,‘A’} 的子集。

a=set(['A','C'])  
b=set(['D','C','E','A'])  
  
if a.union(b)==b:  
 print("是子集")  
else:  
 print("不是子集")



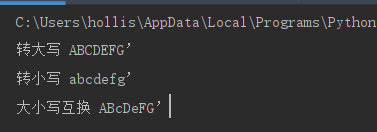
1. 去除数组 [1,2,5,2,3,4,5,‘x’,4,‘x’] 中的重复元素。

a=[1,2,5,2,3,4,5,'x',4,'x']  
a=list(set(a))  
print(a)



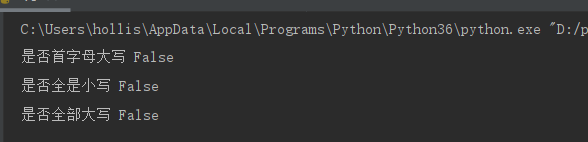
1. 返回字符串 ‘abCdEfg’ 的全部大写、全部小写和大下写互换形式。

s='abCdEfg’'  
print("转大写",s.upper())  
print("转小写",s.lower())  
print("大小写互换",s.swapcase())



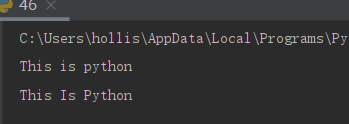
1. 判断字符串 ‘abCdEfg’ 是否首字母大写，字母是否全部小写，字母是否全部大写。

s='abCdEfg'  
print("是否首字母大写",s.capitalize()[0]==s[0])  
print("是否全是小写",s.islower())  
print("是否全部大写",s.isupper())



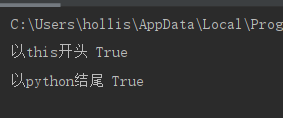
1. 返回字符串 ‘this is python’ 首字母大写以及字符串内每个单词首字母大写形式。

s='this is python'  
print(s.capitalize())  
print(s.title())



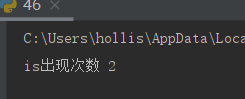
1. 判断字符串 ‘this is python’ 是否以 ‘this’ 开头，又是否以 ‘python’ 结尾。

s='this is python'  
print("以this开头",s.startswith("this"))  
print("以python结尾",s.endswith("python"))



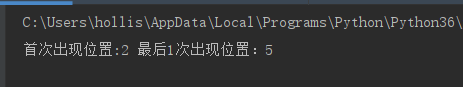
1. 返回字符串 ‘this is python’ 中 ‘is’ 的出现次数。

s='this is python'  
print("is出现次数",s.count('is'))



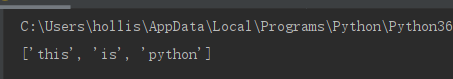
1. 返回字符串 ‘this is python’ 中 ‘is’ 首次出现和最后一次出现的位置。

s='this is python'  
print("首次出现位置:{} 最后1次出现位置：{}".format(s.index('is'),s.rindex('is')))



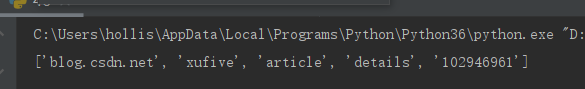
1. 将字符串 ‘this is python’ 切片成3个单词。

s='this is python'  
print(s.split())



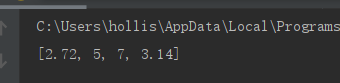
1. 返回字符串 ‘blog.csdn.net/xufive/article/details/102946961’ 按路径分隔符切片的结果。

s='blog.csdn.net/xufive/article/details/102946961'  
print(s.split('/'))



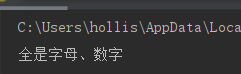
1. 将字符串 ‘2.72, 5, 7, 3.14’ 以半角逗号切片后，再将各个元素转成浮点型或整形。

a='2.72, 5, 7, 3.14'.split(',')  
conv=lambda x:float(x) if '.' in x else int(x)  
print(list(map(conv,a)))



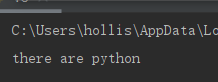
1. 判断字符串 ‘adS12K56’ 是否完全为字母数字，是否全为数字，是否全为字母？

a='adS12K56'  
if a.isnumeric():  
 print('全是数字')  
elif a.isalpha():  
 print('全是字母')  
elif a.isalnum():  
 print('全是字母、数字')



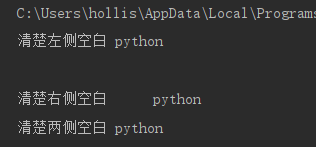
1. 将字符串 ‘there is python’ 中的 ‘is’ 替换为 ‘are’。

s='there is python'  
print(s.replace('is','are'))



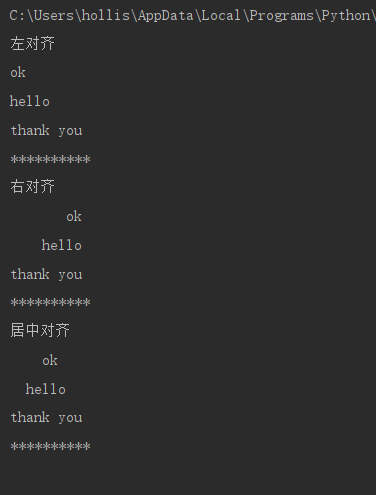
1. 清除字符串 ‘\t python \n’ 左侧、右侧，以及左右两侧的空白字符。

s='\t python \n'  
print("清楚左侧空白",s.lstrip())  
print("清楚右侧空白",s.rstrip())  
print("清楚两侧空白",s.strip())



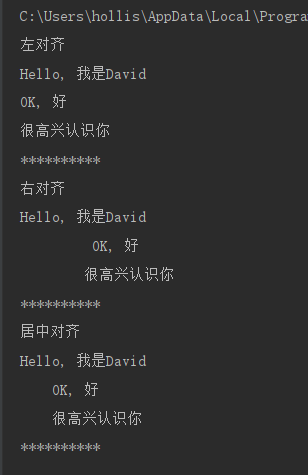
1. 将三个全英文字符串（比如，‘ok’, ‘hello’, ‘thank you’）分行打印，实现左对齐、右对齐和居中对齐效果。

s=['ok', 'hello', 'thank you']  
l=len(s[2])  
print("左对齐")  
for i in s:print(i.ljust(l))  
print('\*'\*10)  
  
print("右对齐")  
for i in s:print(i.rjust(l))  
print('\*'\*10)  
  
print("居中对齐")  
for i in s:print(i.center(l))  
print('\*'\*10)



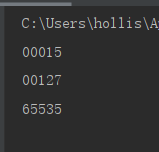
1. 将三个字符串（比如，‘Hello, 我是David’, ‘OK, 好’, ‘很高兴认识你’）分行打印，实现左对齐、右对齐和居中效果。

s=['Hello, 我是David', 'OK, 好', '很高兴认识你']  
l=max([len(x) for x in s])  
print("左对齐")  
for i in s:print(i.ljust(l))  
print('\*'\*10)  
  
print("右对齐")  
for i in s:print(i.rjust(l))  
print('\*'\*10)  
  
print("居中对齐")  
for i in s:print(i.center(l))  
print('\*'\*10)



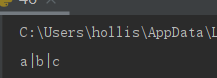
1. 将三个字符串 ‘15’, ‘127’, ‘65535’ 左侧补0成同样长度。

s=['15', '127', '65535']  
l=max([len(x) for x in s])  
for x in s:  
 print(x.rjust(l,'0'))



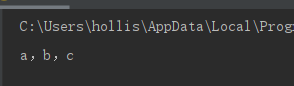
1. 将列表 [‘a’,‘b’,‘c’] 中各个元素用’|'连接成一个字符串。

a=['a','b','c']  
print('|'.join(a))



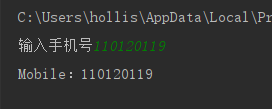
1. 将字符串 ‘abc’ 相邻的两个字母之间加上半角逗号，生成新的字符串。

a='abc'  
j='，'  
print(j.join(a))



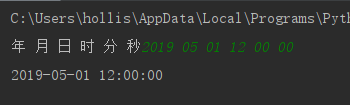
1. 从键盘输入手机号码，输出形如 ‘Mobile: 186 6677 7788’ 的字符串。

phone=input("输入手机号")  
print('Mobile：'+phone)



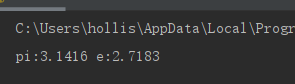
1. 从键盘输入年月日时分秒，输出形如 ‘2019-05-01 12:00:00’ 的字符串。

# 2019-05-01 12:00:00  
dt=input("年 月 日 时 分 秒").split()  
print('-'.join(dt[:3])+' '+':'.join(dt[3:]))



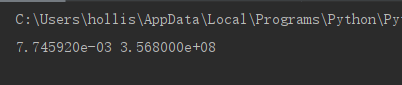
1. 给定两个浮点数 3.1415926 和 2.7182818，格式化输出字符串 ‘pi = 3.1416, e = 2.7183’。

a=3.1415926  
b=2.7182818  
print("pi:{:.4f} e:{:.4f}".format(a,b))



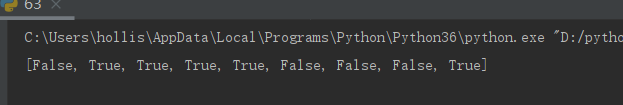
1. 将 0.00774592 和 356800000 格式化输出为科学计数法字符串。

a=0.00774592  
b=356800000  
print("{:e} {:e}".format(a,b))



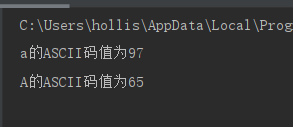
1. 将列表 [0,1,2,3.14,‘x’,None,’’,list(),{5}] 中各个元素转为布尔型。

a=[0,1,2,3.14,'x',None,'',list(),{5}]   
a=[bool(x) for x in a]  
print(a)



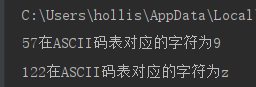
1. 返回字符 ‘a’ 和 ‘A’ 的ASCII编码值。

a=['a','A']  
for i in a:  
 print("{}的ASCII码值为{}".format(i,ord(i)))



1. 返回ASCII编码值为 57 和 122 的字符。

a=[57,122]  
for i in a:  
 print("{}在ASCII码表对应的字符为{}".format(i, chr(i)))

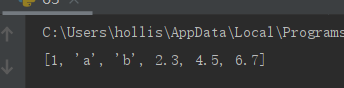


1. 将列表 [3,‘a’,5.2,4,{},9,[]] 中 大于3的整数或浮点数置为1，其余置为0。

a=[3,'a',5.2,4,{},9,[]]  
def convert\_num(x):  
 if isinstance(x,int):  
 return 1 if int(x)>3 else 0  
 elif isinstance(x,float):  
 return 1 if float(x) > 3 else 0  
 else:return 0  
a=list(map(convert\_num,a))  
print(a)

1. 将二维列表 [[1], [‘a’,‘b’], [2.3, 4.5, 6.7]] 转为 一维列表。

a=[[1], ['a','b'], [2.3, 4.5, 6.7]]  
a=[j for x in a for j in x]  
print(a)



1. 将等长的**键列表和值列表**转为字典。

a='A B C D E'.split()  
b=range(len(a))  
dic=dict(zip(a,b))  
print(dic)

1. 数字列表求和。

s=list(range(10))  
print(sum(s))

