110 道 Python 面试笔试题超强汇总！

大数据分析和人工智能 • 6 小时前 • 15 次点击

> 编译：嘉美伯爵

这几天好多学员留言问有没有Python面试题，今天统一给大家分享一遍，希望能帮助此时仍在找工作的同学，尽快找到工作，希望对基本知识不熟悉的同学，能认真做一遍，肯定会有不少收获。

1、一行代码实现1--100之和？

利用sum()函数求和

```
In [1]: sum(range(0,101))
Out[1]: 5050
```

2、如何在一个函数内部修改全局变量？

利用global 修改全局变量

```
In [2]: a = 5

In [3]: def fn():
   ...:     global a
   ...:     a = 4
```

```
    ...:

In [4]: fn()

In [5]: print(a)
4
```

3、列出5个Python标准库？

os：提供了不少与操作系统相关联的函数

sys：通常用于命令行参数

re：正则匹配

math：数学运算

datetime：处理日期时间

4、字典如何删除键和合并两个字典？

del和update方法

```
In [17]: dic
Out[17]: {'age': 18, 'name': '1s'}
```

5、谈下Python的GIL？

GIL是Python的全局解释器锁，同一进程中假如有多个线程运行，一个线程在运行Python程序的时候会霸占Python解释器（加了一把锁即GIL），使该进程内的其他线程无法运行，等该线程运行完后其他线程才能运行。如果线程运行过程中遇到耗时操作，则解释器锁解开，使其他线程运行。所以在多线程中，线程的运行仍是有先后顺序的，并不是同时进行。

多进程中因为每个进程都能被系统分配资源，相当于每个进程有了一个Python解释器，所以多进程可以实现多个进程的同时运行，缺点是进程系统资源开销大

6、Python实现列表去重的方法？

先通过集合去重，在转列表

```
In [7]: list = [11,12,13,12,15,16,13]

In [8]: a = set(list)

In [9]: a
Out[9]: {11, 12, 13, 15, 16}

In [10]: [x for x in a]
Out[10]: [16, 11, 12, 13, 15]
```

集合去重

7、fun(*args,**kwargs)中的*args,**kwargs什么意思？

*args 和 **kwargs主要用于函数定义。你可以将不定数量的参数传递给一个函数。这里的不

定的意思是:预先并不知道,函数使用者会传递多少个参数给你,所以在这个场景下使用这两个关键字。*args是用来发送一个非键值对的可变数量的参数列表给一个函数.这里有个例子帮你理解这个概念:

```
In [6]: def demo(args_f,*args_v):
        print args_f
        for x in args_v:
                print x
  ...:

In [7]: demo('a','b','c','d')
a
b
c
d
```

kwargs允许你将不定长度的键值对,作为参数传递给一个函数。如果你想要在一个函数里处理带名字的参数, 你应该使用kwargs。这里有个例子帮你理解这个概念:

kwargs允许你将不定长度的键值对,作为参数传递给一个函数。如果你想要在一个函数里处理带名字的参数, 你应该使用kwargs。这里有个例子帮你理解这个概念:

```
In [1]: def demo(**args_v):
  ...:         for k,v in args_v.items():
  ...:                 print k,v
  ...:

In [2]: demo(name='njcx')
name njcx
```

8、Python2和Python3的range（100）的区别？

Python2返回列表，Python3返回迭代器，节约内存。

9、一句话解释什么样的语言能够用装饰器?

函数可以作为参数传递的语言，可以使用装饰器。

10、Python内建数据类型有哪些？

整型--int

布尔型--bool

字符串--str

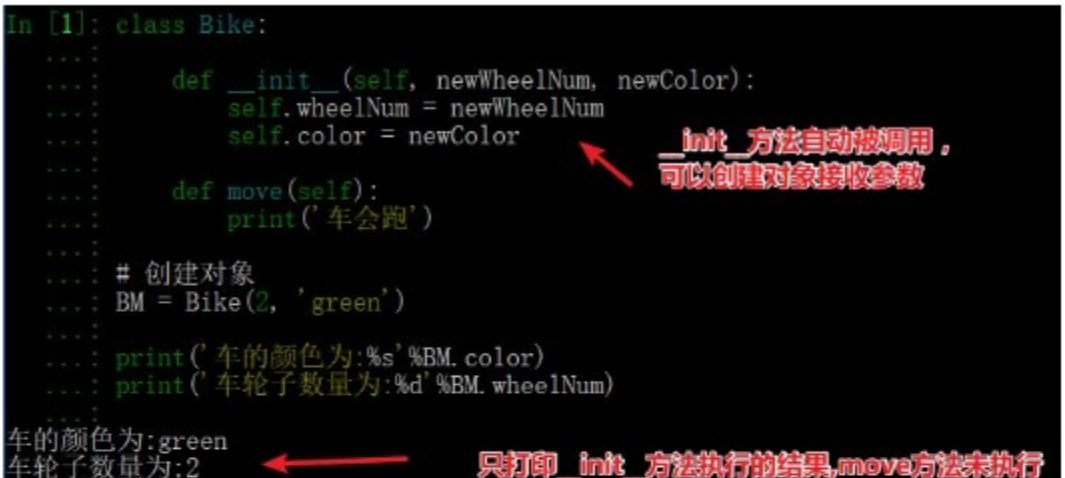
列表--list

元组--tuple

字典--dict

11、简述面向对象中__new__和__init__区别？

__init__是初始化方法，创建对象后，就立刻被默认调用了，可接收参数，如图

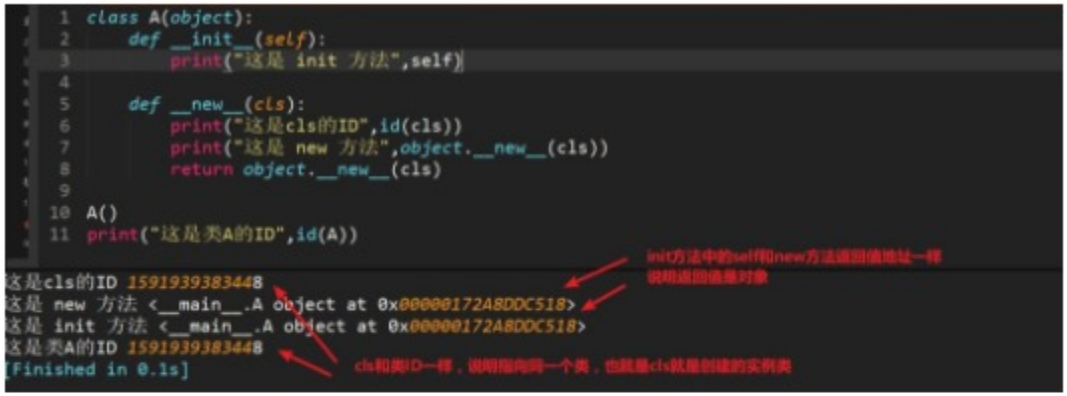


1）__new__至少要有一个参数cls，代表当前类，此参数在实例化时由Python解释器自动识别。

2）__new__必须要有返回值，返回实例化出来的实例，这点在自己实现__new__时要特别注意，可以return父类（通过super(当前类名, cls)）__new__出来的实例，或者直接是object的__new__出来的实例。

3）__init__有一个参数self，就是这个__new__返回的实例，__init__在__new__的基础上可以完成一些其它初始化的动作，__init__不需要返回值。

4）如果__new__创建的是当前类的实例，会自动调用__init__函数，通过return语句里面调用的__new__函数的第一个参数是cls来保证是当前类实例，如果是其他类的类名，；那么实际创建返回的就是其他类的实例，其实就不会调用当前类的__init__函数，也不会调用其他类的__init__函数。



12、简述with方法打开处理文件帮我我们做了什么？

```
f=open("./1.txt","wb")
try:
    f.write("hello world")
except:
    pass
finally:
    f.close()
```

打开文件在进行读写的时候可能会出现一些异常状况，如果按照常规的f.open写法，我们需要try,except,finally，做异常判断，并且文件最终不管遇到什么情况，都要执行finally f.close()关闭文件，with方法帮我们实现了finally中f.close（当然还有其他自定义功能，有兴趣可以研究with方法源码）。

13、列表[1,2,3,4,5],请使用map()函数输出[1,4,9,16,25]，并使用列表推导式提取出大于10的数，最终输出[16,25]？

map（ ）函数第一个参数是fun，第二个参数是一般是list，第三个参数可以写list，也可以不写，根据需求。

```
list = [1,2,3,4,5]
def fn(x):
    return x**2

res = map(fn,list)
res = [ i for i in res if i > 10]
print(res)
```

```
[16, 25]
[Finished in 0.5s]
```

14、python中生成随机整数、随机小数、0--1之间小数方法？

随机整数：random.randint(a,b),生成区间内的整数

随机小数：习惯用numpy库，利用np.random.randn(5)生成5个随机小数

0-1随机小数：random.random(),括号中不传参

```
36 import random
37 import numpy as np
38 result = random.randint(10,20)
39 res = np.random.randn(5)
40 ret = random.random()
41 print("正整数",result)
42 print("5个随机小数",res)
43 print("0-1随机小数",ret)
```

区间

不能传数值

```
正整数 14
5个随机小数 [ 0.17721496 -0.76364375 -0.39721767 -0.23669313  0.21354755]
0-1随机小数 0.6659634699304517
[Finished in 1.9s]
```

15、避免转义给字符串加哪个字母表示原始字符串？

r，表示需要原始字符串，不转义特殊字符

16、

中国

，用正则匹配出标签里面的内容（“中国”），其中class的类名是不确定的。

```
45 import re
46 str = '<div class="nam">中国</div>'
```

```
47 res=re.findall(r'<div class=".*">(.*?)</div>',str)
48 print(res)
```

.代表可有可无
*代表任意字符
满足类名可以变化

(.*?)提取文本

```
['中国']
[Finished in 0.5s]
```

17、Python中断言方法举例？

assert（ ）方法，断言成功，则程序继续执行，断言失败，则程序报错。

```
50 a=3
51 assert(a>1)
52 print("断言成功，程序继续向下执行")
53
54 b=4
55 assert(b>7)
56 print("断言失败，程序报错")
57
```

```
['中国']
断言成功，程序继续向下执行
Traceback (most recent call last):
  File "C:\Users\ry-wu.junya\Desktop\test\gensimm.py", line 55, in <module>
    assert(b>7)
AssertionError
[Finished in 0.6s]
```

18、数据表student有id,name,score,city字段，其中name中的名字可有重复，需要消除重复行，请写sql语句？

select distinct name from student

19、10个Linux常用命令？

ls pwd cd touch rm mkdir tree cp mv cat more grep echo

20、Python2和Python3区别？列举5个

1、Python3 使用 print 必须要以小括号包裹打印内容，比如 print('hi')

Python2 既可以使用带小括号的方式，也可以使用一个空格来分隔打印内容，比如 print 'hi'

2、Python2 range(1,10)返回列表，python3中返回迭代器，节约内存

3、Python2中使用ascii编码，python中使用utf-8编码

4、Python2中unicode表示字符串序列，str表示字节序列

Python3中str表示字符串序列，byte表示字节序列

5、Python2中为正常显示中文，引入coding声明，python3中不需要

6、Python2中是raw_input()函数，python3中是input()函数

21、列出python中可变数据类型和不可变数据类型，并简述原理

不可变数据类型：数值型、字符串型string和元组tuple

不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象，而对于相同的值的对象，在内存中则只有一个对象（一个地址），如下图用id()方法可以打印对象的id。

```
In [1]: a = 3

In [2]: b = 3

In [3]: id(a)
```

```
Out[3]: 1365598496

In [4]: id(b)
Out[4]: 1365598496

In [5]:
```

可变数据类型：列表list和字典dict；

允许变量的值发生变化，即如果对变量进行append、+=等这种操作后，只是改变了变量的值，而不会新建一个对象，变量引用的对象的地址也不会变化，不过对于相同的值的不同对象，在内存中则会存在不同的对象，即每个对象都有自己的地址，相当于内存中对于同值的对象保存了多份，这里不存在引用计数，是实实在在的对象。

```
In [5]: a = [1,2]

In [6]: b = [1,2]

In [7]: id(a)
Out[7]: 2572957427336

In [8]: id(b)
Out[8]: 2572957321544

In [9]:
```

22、s = "ajldjlajfdljfddd"，去重并从小到大排序输出"adfjl"？

set去重，去重转成list，利用sort方法排序，reeverse=False是从小到大排。

list是不变数据类型，s.sort时候没有返回值，所以注释的代码写法不正确。

23、用lambda函数实现两个数相乘？

24、字典根据键从小到大排序？

dict={"name":"zs","age":18,"city":"深圳","tel":"1362626627"}

25、利用collections库的Counter方法统计字符串每个单词出现的次数"kjalfj;ldsjafl;hdsllfdhg;lahfbl;hl;ahlf;h"？

```
from collections import Counter
a = "kjalfj;ldsjafl;hdsllfdhg;lahfbl;hl;ahlf;h"
res=Counter(a)
print(res)

Counter({'l': 9, ';': 6, 'h': 6, 'f': 5, 'a': 4, 'j': 3, 'd': 3, 's': 2, 'k': 1, 'g': 1, 'b': 1})
[Finished in 0.1s]
```

26、字符串a = "not 404 found 张三 99 深圳"，每个词中间是空格，用正则过滤掉英文和数字，最终输出"张三 深圳"？

```
import re
a = "not 404 found 张三 99 深圳"
list = a.split(" ")
print(list)
res=re.findall('\d+|[a-zA-Z]+',a)
for i in res:
    if i in list:
        list.remove(i)
new_str=" ".join(list)
print(res)
print(new_str)



['not', '404', 'found', '张三', '99', '深圳']
['not', '404', 'found', '99']
张三 深圳
[Finished in 0.1s]
```


顺便贴上匹配小数的代码，虽然能匹配，但是健壮性有待进一步确认

```
#
import re
```

```
a = "not 404 50.56 found 张三 99 深圳"
list = a.split(" ")
print(list)
res=re.findall('\d+\.?\d*|[a-zA-Z]+',a)
for i in res:
    if i in list:
        list.remove(i)
new_str=" ".join(list)
print(res)
print(new_str)
```

匹配小数

```
['not', '404', '50.56', 'found', '张三', '99', '深圳']
['not', '404', '50.56', 'found', '99']
张三 深圳
[Finished in 0.1s]
```

27、filter方法求出列表所有奇数并构造新列表a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]？

filter() 函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。该接收两个参数，第一个为函数，第二个为序列，序列的每个元素作为参数传递给函数进行判，然后返回 True 或 False，最后将返回 True 的元素放到新列表。

```
a =  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
def fn(a):
        return a%2==1
newlist = filter(fn, a)
newlist=[i for i in newlist]
print(newlist)
```

```
[1, 3, 5, 7, 9]
[Finished in 0.1s]
```

28、列表推导式求列表所有奇数并构造新列表，a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]？

```
45  res = [i for i in a if i%2==1]
46  print(res)
[1, 3, 5, 7, 9]
[Finished in 0.1s]
```

29、正则re.complie作用？

re.compile是将正则表达式编译成一个对象，加快速度，并重复使用。

30、a=（1，）b=(1)，c=("1") 分别是什么类型的数据？

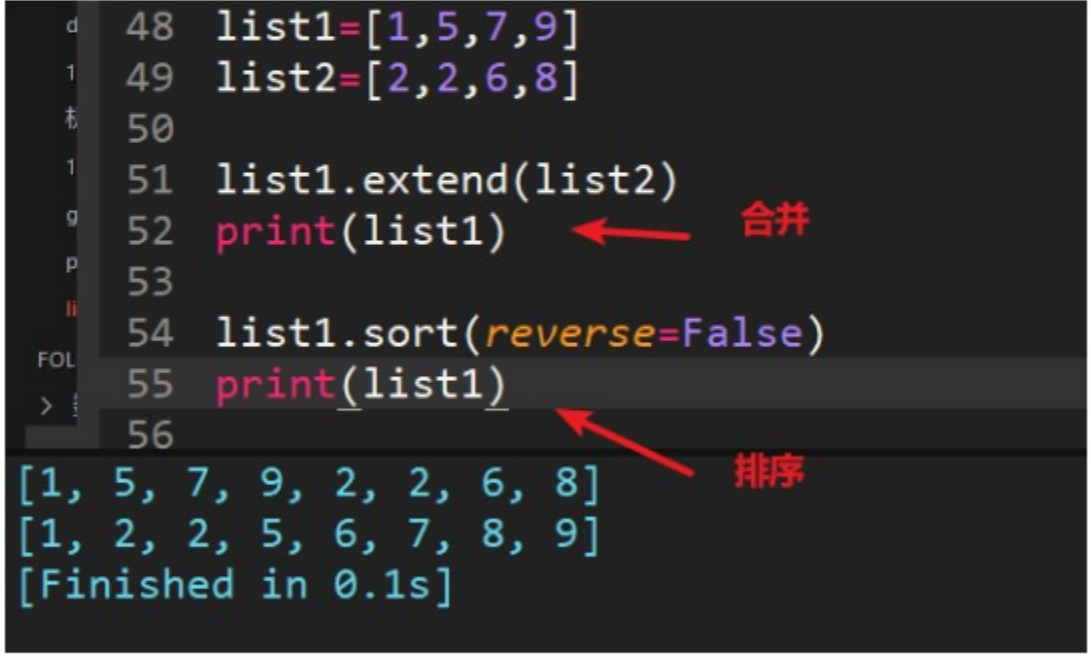
```
In [14]: type((1))
Out[14]: int

In [15]: type(("1"))
Out[15]: str

In [16]: type((1,))
Out[16]: tuple
```

31、两个列表[1,5,7,9]和[2,2,6,8]合并为[1,2,2,3,6,7,8,9]？

extend可以将另一个集合中的元素逐一添加到列表中，区别于append整体添加。



32、用Python删除文件和用linux命令删除文件方法？

python：os.remove(文件名)

linux:　　rm　文件名

33、log日志中，我们需要用时间戳记录error,warning等的发生时间，请用datetime模块打印当前时间戳“2018-04-01 11:38:54”？

顺便把星期的代码也贴上了

34、数据库优化查询方法？

外键、索引、联合查询、选择特定字段等等。

35、请列出你会的任意一种统计图（条形图、折线图等）绘制的开源库，第三方也行？

pychart、matplotlib

36、写一段自定义异常代码？

自定义异常用raise抛出异常

```
# raise自定义异常
def fn():
    try:
        for i in range(5):
            if i>2:
                raise Exception("数字大于2了")
    except Exception as ret:
        print(ret)
fn()
```

```
数字大于2了
[Finished in 0.1s]
```

37、正则表达式匹配中，（.*）和（.*?）匹配区别？

（.*）是贪婪匹配，会把满足正则的尽可能多的往后匹配。

（.*?）是非贪婪匹配，会把满足正则的尽可能少匹配。

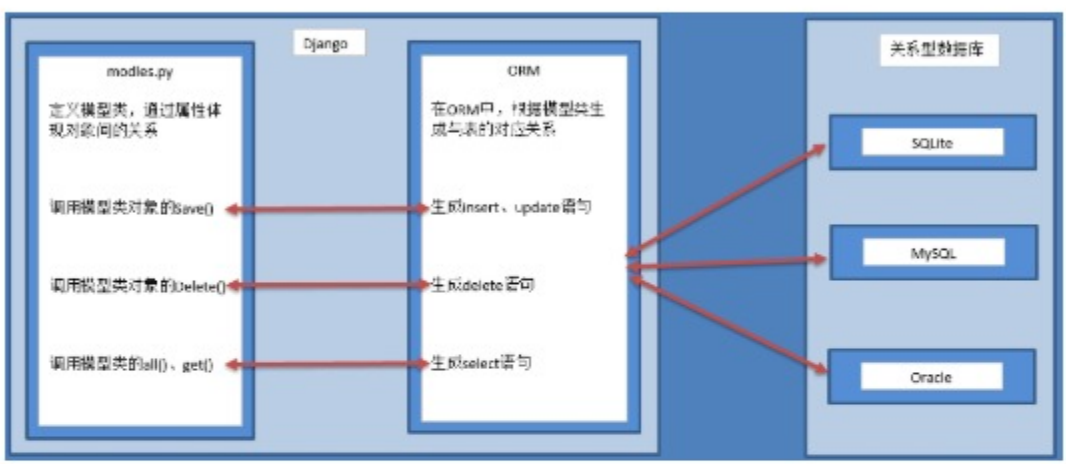
```
 97 s="<a>哈哈</a><a>呵呵</a>"
 98 import re
 99 res1=re.findall("<a>(.*)</a>",s)
100 print("贪婪匹配",res1)
101 res2=re.findall("<a>(.*?)</a>",s)
102 print("非贪婪匹配",res2)

贪婪匹配 ['哈哈</a><a>呵呵']
非贪婪匹配 ['哈哈', '呵呵']
[Finished in 0.1s]
```

38、简述Django的orm？

ORM，全拼Object-Relation Mapping，意为对象-关系映射。

实现了数据模型与数据库的解耦，通过简单的配置就可以轻松更换数据库，而不需要修改代码只需要面向对象编程,orm操作本质上会根据对接的数据库引擎，翻译成对应的sql语句,所有使用Django开发的项目无需关心程序底层使用的是MySQL、Oracle、sqlite....，如果数据库迁移，只需要更换Django的数据库引擎即可。



39、[[1,2],[3,4],[5,6]]一行代码展开该列表，得出[1,2,3,4,5,6]？

列表推导式的骚操作😋
😋😋

运行过程：for i in a ,每个i是【1,2】，【3,4】，【5,6】，for j in i，每个j就是1,2,3,4,5,6,合并后就是结果。

```
121  # 展开列表
122  a=[[1,2],[3,4],[5,6]]
123  x=[j for i in a for j in i]
124  print(x)
```

```
[1, 2, 3, 4, 5, 6]
[Finished in 0.1s]
```

还有更骚的方法，将列表转成numpy矩阵，通过numpy的flatten（ ）方法，代码永远是只有更骚，没有最骚🧐🧐
🧐

```
121  # 展开列表
122  a=[[1,2],[3,4],[5,6]]
123  x=[j for i in a for j in i]
124  print(x)
125
126
127  import numpy as np
128  b=np.array(a).flatten().tolist()
129  print(b)
130
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
```

```
[Finished in 0.6s]
```

40、x="abc",y="def",z=["d","e","f"],分别求出x.join(y)和x.join(z)返回的结果？

join()括号里面的是可迭代对象，x插入可迭代对象中间，形成字符串，结果一致，有没有突然感觉字符串的常见操作都不会玩了🙂🙂🙂

顺便建议大家学下os.path.join()方法，拼接路径经常用到，也用到了join,和字符串操作中的join有什么区别，该问题大家可以查阅相关文档，后期会有答案。

```
86  x="abc"
87  y="def"
88  z=["d","e","f"]
89
90  m=x.join(y)
91  n=x.join(z)
92  print(m)
93  print(n)
```

```
dabceabcf
dabceabcf
[Finished in 0.1s]
```

41、举例说明异常模块中try except else finally的相关意义？

try..except..else没有捕获到异常，执行else语句。

try..except..finally不管是否捕获到异常，都执行finally语句。

```
132 try:
133     num = 100
134     print(num)
135 except NameError as errorMsg:
136     print('产生错误了:%s'%errorMsg)
137 else:
138     print('没有捕获到异常，则执行该语句')
139
140 try:
141     num = 100
142     print(num)
143 except NameError as errorMsg:
144     print('产生错误了:%s'%errorMsg)
145 finally:
146     print('不管是否捕获到异常，都执行该句')
147
```

```
100
没有捕获到异常，则执行该语句
100
不管是否捕获到异常，都执行该句
[Finished in 0.6s]
```

42、Python中交换两个数值？

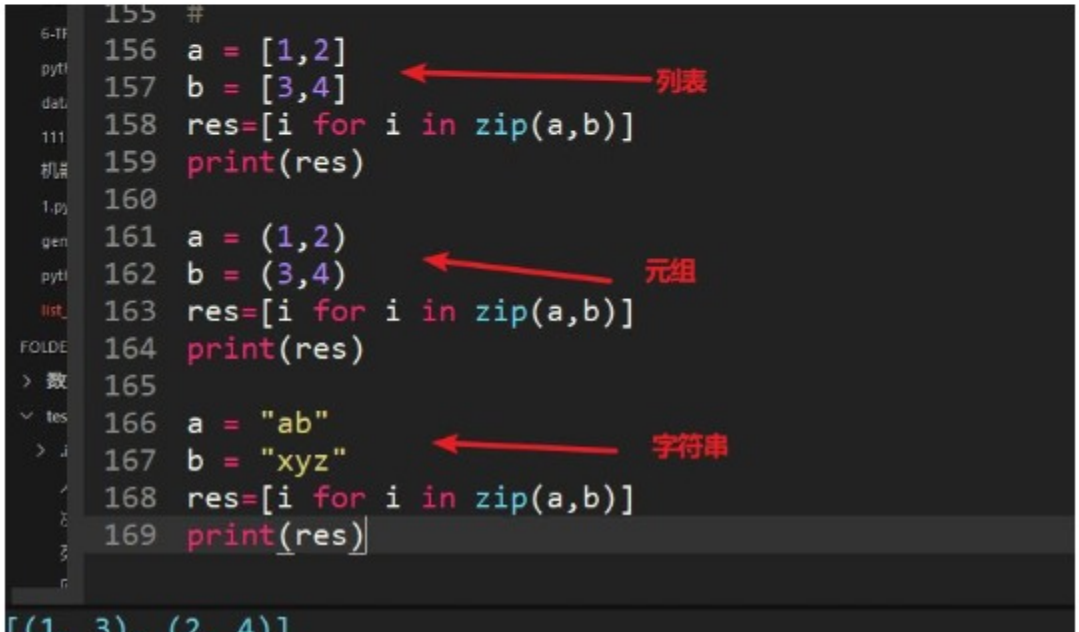
```
148 a,b=3,4
149 print(a,b)
150 a,b=b,a
151 print(a,b)
152
```

```
153
3 4
4 3
[Finished in 0.2s]
```

43、举例说明zip（）函数用法？

zip()函数在运算时，会以一个或多个序列（可迭代对象）做为参数，返回一个元组的列表。同时将这些序列中并排的元素配对。

zip()参数可以接受任何类型的序列，同时也可以有两个以上的参数;当传入参数的长度不同时，zip能自动以最短序列长度为准进行截取，获得元组。



```
[(1, 3), (2, 4)]
[('a', 'x'), ('b', 'y')]
[Finished in 0.2s]
```

44、a="张明 98分"，用re.sub，将98替换为100？

```
171 import re
172 a = "张明 98分"
173 ret = re.sub(r"\d+","100",a)
174 print(ret)
```

```
张明 100分
[Finished in 0.1s]
```

45、写5条常用sql语句？

show databases;

show tables;

desc 表名;

select * from 表名;

delete from 表名 where id=5;

update students set gender=0,hometown="北京" where id=5

46、a="hello"和b="你好"编码成bytes类型？

```
176 a=b"hello"
177 b="哈哈".encode()
178 print(a,b)
179 print(type(a),type(b))
```

```
张明 100分
b'hello' b'\xe5\x93\x88\xe5\x93\x88'
<class 'bytes'> <class 'bytes'>
[Finished in 0.1s]
```

47、[1,2,3]+[4,5,6]的结果是多少？

两个列表相加，等价于extend。

```
181 a = [1,2,3]
182 b = [4,5,6]
183 res=a+b
184 print(res)
```

```
[1, 2, 3, 4, 5, 6]
[Finished in 0.1s]
```

48、提高Python运行效率的方法？

1、使用生成器，因为可以节约大量内存；

2、循环代码优化，避免过多重复代码的执行；

3、核心模块用Cython PyPy等，提高效率；

4、多进程、多线程、协程；

5、多个if elif条件判断，可以把最有可能先发生的条件放到前面写，这样可以减少程序判断的次数，提高效率。

49、简述mysql和redis区别？

redis： 内存型非关系数据库，数据保存在内存中，速度快。

mysql：关系型数据库，数据保存在磁盘中，检索的话，会有一定的Io操作，访问速度相对慢。

50、遇到bug如何处理？

1、细节上的错误，通过print（ ）打印，能执行到print（ ）说明一般上面的代码没有问题，分段检测程序是否有问题，如果是js的话可以alert或console.log。

2、如果涉及一些第三方框架，会去查官方文档或者一些技术博客。

3、对于bug的管理与归类总结，一般测试将测试出的bug用teambin等bug管理工具进行记录，然后我们会一条一条进行修改，修改的过程也是理解业务逻辑和提高自己编程逻辑缜密性的方法，我也都会收藏做一些笔记记录。

4、导包问题、城市定位多音字造成的显示错误问题。