# Python复习题

## 第一章基础知识

**知识点：**

1. 变量的创建、变量的命名规则、变量的类型、python采用的是基于值得内存管理方式
2. 运算符与表达式
3. 常见的内置函数
4. 模块的导入与使用
5. Python代码编写规范

**复习题习题：**

1. 为什么说python采用的是基于值得内存管理模式？

答：Python采用的是基于值的内存管理方式，如果为不同变量赋值相同值，则在内存中只有一份该值，多个变量指向同一块内存地址，例如下面的代码。

1. Python中导入模块有哪几种方式？

答：常用的有三种方式，分别为

* import 模块名 [as 别名]
* from 模块名 import 对象名[ as 别名]
* from math import \*

1. 在python3.x中input（）函数接收到的用户输入数据一律为什么类型？str
2. 表达式isinstance(“hello word”,str)的值为（）？True
3. Python的设计具有很强的可读性，相比其他语言具有的特色语法有以下选项，正确的是()：ABC

A.交互式 B.解释型 C.面向对象 D.服务端语言

1. Python崇尚优美、清晰、是一个优秀并广泛使用的语言，得到行内众多领域的认可，下列属于Python主要应用领域的是:()ABCD

A．系统运维 B.科学计算、人工智能 C.云计算 D.金融量化

1. 关于Python变量，下列说法错误的是（)B
2. 变量不必事先声明但区分大小写
3. 变量无须先创建和赋值而直接使用
4. 变量无须指定类型
5. 可以使用del关键字释放变量
6. 有如下类定义，下列描述错误的是？ C  
   class A(object):  
   pass  
   class B(A):  
   pass  
   b = B()
7. isinstance(b,A)==True
8. isinstance(b,object)==True
9. isinstance(B,A)==True
10. isinstance(b,B)==True

## 第二章 python序列

**知识点：**

1. 有序序列：支持双向索引
2. 可变序列：
3. 列表：列表中元素的类型可以各不相同，并且支持复杂数据类型对象的嵌套
4. 创建列表、删除列表对象
5. 列表中的方法：添加元素（‘+’、append()、extend()、insert()）、删除元素（del、pop()、remove()、）、修改元素（指定元素重新赋值）、查找元素（下标访问元素、index()）、计数（count()）、成员资格判断(in、not in)、切片操作（添加、修改、删除、获取元素）、列表的排序、常用的内置函数（max,min,sum,zip,enmerate）列表推导式
6. 元组：元组中的元素类型可以各不相同，并且支持复杂数据类型对象的嵌套
7. 创建元组（元组中只有一个元素的情况）、删除元组对象
8. 元组中的方法：查找元素（下标访问元素、index()）
9. 支持切片操作，通过切片操作只能获取元素。
10. 序列解包
11. 生成器推导式
12. 字典的定义，字典的键是不可变数据类型，键不嫩重复，值是可以重复的。
13. 字典的创建，删除
14. 字典元素的读取（get(),字典的键访问值）、字典元素的添加（updata()、指定键为下标为字典元素赋值，若键存在表示修改，不存在则表示添加）
15. 字典中的方法：items(),keys(),values()
16. 集合：无序可变的序列，集合中的元素不允许重复，每个元素是唯一的。
17. 集合的创建与删除
18. 集合的方法：添加元素（add()、update()）、删除元素（remove()、pop()、clear()、popitems()、discard()）
19. 集合的操作：交集（&、intersection()）、并集（|、union()）、差集（-、difference()）、对称差集（^、symmertric\_difference()）
20. 集合的包含关系（子集、真子集）
21. 深入理解排序、lambda表达式

**复习题：**

1. 使用列表推导式生成包含10个数字5的列表，语句可以写为（）. [5 for i in range(10)]
2. 假设有列表a=[‘name’,”age”,sex’],和b=[“Dong”,38,’Male’]，请使用一个语句将这两个列表的内容转换为字典，并且以列表a中的元素为字典的键，以列表b里面的元素为字典的值，这个语句可以写为（）。dict(zip(a,b))
3. 假设列表对象aList的值为[3,4,5,6,7,9,11,13,15,17],那么切片aList[3:7]得到的结果为（）。[6, 7, 9, 11]
4. 执行以下代码的结果为（）。(1, 2, 3, [1, 2, 4])

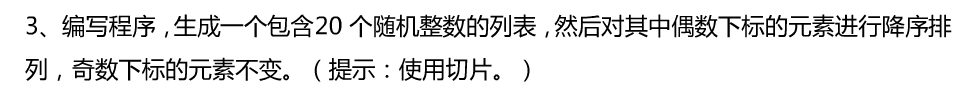
t=(1,2,3,[1,2,3])

t[-1][-1]=4

print(t)

1. 以下声明错误的是（）
2. dic={}
3. dic={100:200}
4. dic={(1,2,3):’test’}
5. dict={[1,2,3]:’test’}
6. 若a=(1,2,3),下列哪些操作是合法的？
7. a[1:-1]
8. a\*3
9. a[2]=4 元组不可变
10. list(a)
11. 下列代码运行结果是？ (c)  
    a = map(lambda x: x\*\*3, [1, 2, 3]) ，print(list(a)) 立方
12. [1,6,9]
13. [1,12,27]
14. [1,8,27]
15. (1,6,9)
16. 下列程序打印结果为（    A  ）  
    nl = [1,2,5,3,5]  
    nl.append(4) 最后增加  
    nl.insert(0,7) 指定位置增加  
    nl.sort() 排序  
    print nl
17. [1,2,3,4,5,5,7]
18. [0,1,2,3,4,5,5]
19. [1,2,3,4,5,7]
20. [7,5,4,3,2,1]
21. 下面哪个是Python中的不变的数据结构？(D)
22. set
23. list
24. dict
25. tuple元组
26. 对于以下代码，描述正确的是：  
    list = ['1', '2', '3', '4', '5']  
    print list[10:]
27. 导致编译错误
28. 输出['1', '2', '3', '4', '5']
29. []
30. 导致IndexError
31. 表达式{1,2,}\*2的结果为{1,2,1,2}吗？**×集合中元素不能重复**
32. 集合可以作为字典的值吗？能
33. 使用del命令或者列表对象的remove（)方法删除列表中元素时会影响列表中部分元素的索引。√
34. list(map(str,[1,2,3]))的执行结果为（）。['1', '2', '3'] map函数就是映射一对多，让[1，2，3]都变成字符串
35. 元组是不可变的，不支持列表对象的inset()、remove()等方法，也不支持del命令删除其中的元素，但可以使用del命令删除整个元组对象。√
36. 表达式 sorted([111, 2, 33], key=lambda x: len(str(x))) 的值为\_\_\_\_\_\_\_ [2, 33, 111]\_\_。根据字符长度排序返回一个排好的新列表 sort原地排序 sorted返回新列表
37. 字典中多个元素之间使用\_\_\_\_\_逗号\_\_\_\_\_\_\_分隔开，每个元素的“键”与“值”之间使用\_\_\_\_冒号\_\_\_\_分隔开。
38. 表达式 [index for index, value in enumerate([3,5,7,3,7]) if value == max([3,5,7,3,7])] 的值为\_\_\_\_\_\_[2, 4]\_\_\_\_\_\_\_\_\_\_\_\_。 enumerate函数可以返回下标
39. 已知 x = {'a':'b', 'c':'d'}，那么表达式 'b' in x.values() 的值为\_\_\_\_True\_\_\_\_\_\_\_\_\_\_。判断b是不是字典x中的values值。keys是键。
40. 已知 x = [1, 11, 111]，那么执行语句 x.sort(key=lambda x: len(str(x)), reverse=True) 之后，x的值为\_\_\_\_\_\_\_\_\_ [111, 11, 1]\_\_\_\_\_\_\_\_\_。根据字符长度排序 reverse=Ture是逆序
41. 表达式 [index for index, value in enumerate([3,5,7,3,7]) if value == max([3,5,7,3,7])] 的值为\_\_\_\_\_\_\_\_[2, 4]\_\_\_\_\_\_\_\_\_\_。
42. 表达式 {1, 2, 3, 4} - {3, 4, 5, 6}的值为\_\_\_{1, 2}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
43. 表达式set([1, 1, 2, 3])的值为\_\_\_\_\_\_ {1, 2, 3}\_\_\_\_\_\_\_\_\_\_\_\_\_\_。Set（）函数转换为集合，集合元素不重复。
44. 

答：当列表增加或删除元素时，列表对象自动进行内存扩展或收缩，从而保证元素之间没有缝隙，但这涉及到列表元素的移动，效率较低，应尽量从列表尾部进行元素的增加与删除操作以提高处理速度。



## 第三章 选择与循环

**知识点：**

1. 选择结构（单分支、双分支、多分支、选择结构的嵌套）
2. 循环结构（for、while可跟else，也可不跟。如果循环是因为条件表达式不成立而自然结束，则执行else结构中的语句。 ）
3. break、continue

**复习题**：

1. 编写程序，要求用户输入一些数字，输出这些数字中唯一的数字。也就是说，如果某个数字出现多次，只保留一个。

from collections import OrderedDict

num = input('请输入数字：')

od = OrderedDict()

for k in num:

od[k] = od.get(k,0)+1

print(od)

1. 输入一行字符，统计字母、数字、空格和其他字符的个数。

s=input('输入一行字符:\n')

i=0

j=0

k=0

l=0

for c in s:

if c.isalpha():

i+=1

elif c.isspace():

j+=1

elif c.isdigit():

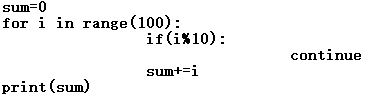
k+=1

else:

l+=1

print('英文=%d,空格=%d,数字=%d,其他字符=%d'%(i,j,k,l))

1. 在python中，下列流程控制语句没有的是（c）
2. If…..elif
3. while……else
4. do……while
5. if…..else
6. 下列程序运行的结果（）



1. 5050 B. 4950 C. 450 D. 45
2. 以下for语句中，（）能完成1—10的累加功能**。BC**
3. for i in range（10, 0）:

sum+=1

1. for i in range（1, 11）:

sum+=1

1. for i in range（10, 0,-1）:

sum+=1

1. for i in range（10, 9,8,7,6,5,4,3,2,1）:

sum+=1

1. 如下程序会打印多少个数:（D）

k = 1000

**while** k > 1:

    print k

    k = k//2

1. 1000
2. 10
3. 11
4. 9
5. Python语句x=0;y=True; print(x>y and “A”<”B”)的运行结果\_\_\_ False\_\_\_。

## 第四章 字符串与正则表达式

**知识点：**

1. 字符串是有序不可变类型，支持双向索引，切片操作获取元素
2. 字符串格式化
3. 对于短字符串，python支持驻留机制
4. 字符串的常见方法（查找、替换、统计、判断）
5. 正则表达式及常见方法

**复习题：**

1. 表达式 'Hello world'.lower().upper() 的值为\_\_\_\_\_\_'HELLO WORLD'\_\_\_\_\_。Lower函数转换为小写，upper函数转换为大写。
2. 表达式 r'c:\windows\notepad.exe'.endswith(('.jpg', '.exe')) 的值为\_\_ True\_\_\_\_\_。

Endswith函数判断字符串是否以指定字符或子字符串结尾

1. 表达式 'ab' in 'acbed' 的值为\_\_\_False\_\_\_\_\_。
2. 表达式 sorted([111, 2, 33], key=lambda x: len(str(x))) 的值为[2, 33, 111]\_。
3. 表达式 'abcabcabc'.rindex('abc') 的值为\_\_\_\_\_6\_\_\_\_\_\_\_。没找到会返回空值

Rindex函数返回最后出现的位置

1. 表达式 ':'.join('abcdefg'.split('cd')) 的值为\_\_\_\_\_'ab:efg'\_\_\_\_\_\_\_\_\_。

Join连接相邻两个字符串并插入指定字符串’:’

split函数拆分字符串，以’cd’为分隔符对字符串进行切片

1. 表达式 'Hello world. I like Python.'.rfind('python') 的值为\_\_\_-1\_\_\_\_\_。

find函数和rfind函数没找到返回-1

1. 表达式 'abcabcabc'.count('abc') 的值为\_\_\_\_\_\_3\_\_\_\_\_\_\_。

Count函数返回字符串出现次数

1. 代码 print(re.match('^[a-zA-Z]+$','abcDEFG000')) 的输出结果为\_ None\_。、

找到以字母开头字母结尾的字符串 并返回位置和该字符串，没有就返回None

1. 当在字符串前加上小写字母\_\_\_r\_\_或大写字母\_R\_\_\_\_表示原始字符串，不对其中的任何字符进行转义。
2. 在设计正则表达式时，字符\_\_?\_\_\_\_\_紧随任何其他限定符(\*、+、?、{n}、{n,}、{n,m})之后时，匹配模式是“非贪心的”，匹配搜索到的、尽可能短的字符串。
3. 假设正则表达式模块re已导入，那么表达式 re.sub('\d+', '1', 'a12345bbbb67c890d0e') 的值为\_\_\_\_’ a1bbbb1c1d1e’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

将数字串转换为1

1. 已知 table = ''.maketrans('abcw', 'xyzc')，那么表达式 'Hellow world'.translate(table) 的值为\_\_\_\_’ Helloc corld’\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。

maketrans()方法用于给translate()方法创建字符映射转换表1对1

1. 正则表达式元字符\_\_\_\_\*\_\_\_\_用来表示该符号前面的字符或子模式0次或多次出现。
2. 表达式 list(filter(lambda x:x>2, [0,1,2,3,4,0])) 的值为\_\_ [3，4]\_\_\_\_\_\_\_。

filter()函数过滤器，条件是大于2

表达式 re.search(r'\w\*?(?P<f>\b\w+\b)\s+(?P=f)\w\*?', 'Beautiful is is better than ugly.').group(0) 的值为\_\_\_\_ ‘is is’\_\_\_\_\_\_\_。

## 第五章 函数

**知识点**：

1. 函数的定义
2. 函数的参数类型
3. 函数的返回值
4. 变量的作用域

**练习题：**

1. 定义函数时，在形参前面加两个星号表示可以接收多个关键参数并存放于（C ）中。

A. 列表 B. 元组 C. 字典 D. 集合

1. 以下选项中，对于递归程序的描述错误的是（ C）。
2. 书写简单
3. 递归程序都可以有非递归编写方法
4. 执行效率高
5. 一定要有基例
6. 关于return说法正确的是(BD)
7. python函数中必须有return
8. return可以返回多个值
9. return没有返回值时，函数自动返回Null
10. 执行到return时，程序将停止函数内return后面的语句
11. 下面程序的功能是什么?（  A  ）

|  |  |
| --- | --- |
|  | def f(a, b):  **if** b == 0:  print a  **else**:  f(b, a%b)  a, b = input(“Enter two natural numbers: ”)  print f(a, b)   1. 求AB的最大公约数 2. 求AB的最小公倍数 3. 求A%B 4. 求A/B |

1. 在函数内部可以通过关键字\_\_\_\_\_global \_\_\_\_\_\_\_\_\_来定义全局变量。
2. 如果函数中没有return语句或者return语句不带任何返回值，那么该函数的返回值为\_\_\_\_\_\_\_\_None\_\_\_\_\_\_\_\_\_。
3. 已知函数定义 def func(\*p):return sum(p)，那么表达式 func(1,2,3, 4) 的值为\_\_10\_\_\_\_。
4. 已知函数定义 def func(\*\*p):return ''.join(sorted(p))，那么表达式 func(x=1, y=2, z=3)的值为\_\_\_\_\_ 'xyz'\_\_\_\_\_。
5. 表达式list(filter(lambda x: x>5, range(10)))的值为\_\_\_\_\_\_\_ [6, 7, 8, 9]\_\_\_\_\_\_\_\_\_\_。
6. 已知函数定义def demo(x, y, op):return eval(str(x)+op+str(y))，那么表达式demo(3, 5, '-')的值为\_\_\_\_\_\_\_\_\_ -2\_\_\_\_\_\_\_\_\_\_\_\_\_。
7. 定义函数时，即使该函数不需要接收任何参数，也必须保留一对空的圆括号来表示这是一个函数。√
8. 编写函数时，一般建议先对参数进行合法性检查，然后再编写正常的功能代码。√
9. 一个函数如果带有默认值参数，那么必须所有参数都设置默认值。×
10. 定义Python函数时必须指定函数返回值类型。×
11. 不同作用域中的同名变量之间互相不影响，也就是说，在不同的作用域内可以定义同名的变量。√
12. 全局变量会增加不同函数之间的隐式耦合度，从而降低代码可读性，因此应尽量避免过多使用全局变量。√
13. 下面的python3函数，如果输入的参数n非常大，函数的返回值会趋近于以下哪一个值（选项中的值用Python表达式来表示）（B）

**import** random

def foo(n):

     random.seed()

     c1 = 0

     c2 = 0

**for** i in range(n):

        x = random.random()

        y = random.random()

        r1 = x \* x + y \* y

        r2 = (1 - x) \* (1 - x) + (1 - y) \* (1 - y)

**if** r1 <= 1 and r2 <= 1:

           c1 += 1

**else**:

           c2 += 1

**return**   c1 / c2

1. 4/3
2. (math.pi-2)/(4-math.pi)
3. Math.e\*\*(6/21)
4. Math.tan(53/180\*math.pi)

## 第六章 面向对象

**知识点：**

1. 类的定义
2. Self参数
3. 类成员和实例成员
4. 私有成员、公有成员
5. 实例化对象
6. 公有方法、私有方法、类方法、 静态方法、特殊方法
7. 继承的概念
8. 单继承
9. 子类重写父类中的同名方法和属性
10. 子类重写同名方法中调用父类同名方法和属性
11. Super关键字
12. 多继承
13. 多态

**练习题：**

1. Python使用\_\_\_\_\_\_\_\_def\_\_\_\_\_\_\_\_\_关键字来定义类。
2. 在Python中，不论类的名字是什么，构造方法的名字都是(\_\_init\_\_)
3. Python中一切内容都可以称为对象。√
4. 定义类时所有实例方法的第一个参数用来表示对象本身，在类的外部通过对象名来调用实例方法时不需要为该参数传值。√
5. 在面向对象程序设计中，函数和方法是完全一样的，都必须为所有参数进行传值。×
6. Python中没有严格意义上的私有成员。√
7. 对于Python类中的私有成员，可以通过“对象名.\_类名\_\_私有成员名”的方式来访问。√
8. 在派生类中可以通过“基类名.方法名()”的方式来调用基类中的方法。√
9. Python支持多继承，如果父类中有相同的方法名，而在子类中调用时没有指定父类名，则Python解释器将从左向右按顺序进行搜索。√
10. 有如下类定义，下列描述错误的是？ (D)  
    class A(object):  
    pass  
      
    class B(A):  
    pass  
      
    b = B()
11. isinstance(b,A)==True
12. issubclass(b,B)==True
13. isinstance(b,object)==True
14. issubclass(B,A)==True

10.谈谈你对面向对象的理解？

1. 定义一个僵尸Zombie类，该类可以实例化出多种僵尸对象，每个僵尸对象默认都有名字name、血量HP。其中僵尸对象有普通僵尸和铁桶僵尸两种，血量：默认是100滴，不需要外界提供。

定义一个角色User类，该类有名字name属性、以及打僵尸的beat方法。其中角色的名字可随意自定义，该beat方法需要传入一个僵尸对象。在方法内部可以实现：某某用户攻击了某某个僵尸，僵尸损失多少血，还剩多少血。每一次攻击，都固定扣除25滴血。

## 第七章 文件操作

**知识点：**

1. 文本文件和二进制文件的操作（文件操作三部曲）
2. With关键字
3. JSON格式的数据处理
4. Excel文件的读、写操作
5. Os模块中常用的文件操作函数
6. Os.path模块中常用的文件操作函数

**练习题：**

1. 使用上下文管理关键字\_\_\_\_\_\_\_with\_\_\_\_\_\_\_可以自动管理文件对象，不论何种原因结束该关键字中的语句块，都能保证文件被正确关闭。对文件进行写入操作之后，\_\_\_\_\_\_\_\_flush()\_\_\_\_\_\_\_方法用来在不关闭文件对象的情况下将缓冲区内容写入文件。
2. Python内置函数\_\_\_\_\_\_open()\_\_\_\_\_\_\_用来打开或创建文件并返回文件对象。
3. Python标准库os中用来列出指定文件夹中的文件和子文件夹列表的方式是\_\_ listdir()\_\_\_\_\_。
4. Python标准库os.path中用来判断指定路径是否为文件的方法是\_\_\_\_\_\_\_ isfile()\_\_\_\_\_\_\_\_。
5. 已知当前文件夹中有纯英文文本文件readme.txt，请填空完成功能把readme.txt文件中的所有内容复制到dst.txt中，with open('readme.txt') as src, open('dst.txt', \_\_\_’w’\_\_\_\_\_\_\_\_\_) as dst:dst.write(src.read())。
6. 使用内置函数open()且以”w”模式打开的文件，文件指针默认指向文件尾。×
7. 使用内置函数open()打开文件时，只要文件路径正确就总是可以正确打开的。×
8. Python调用(   )函数不可实现对文件内容的读取（D）
9. read()
10. readline（）
11. readlines()
12. readclose()
13. 读取文件的几种方法的区别?

## 第八章 异常处理结构

**知识点：**

**练习题：**

1. Python内建异常类的基类是\_\_\_\_\_\_ BaseException \_\_\_\_\_\_\_\_\_\_。
2. 程序中异常处理结构在大多数情况下是没必要的。×
3. 在try...except...else结构中，如果try块的语句引发了异常则会执行else块中的代码。×
4. 异常处理结构中的finally块中代码仍然有可能出错从而再次引发异常。√
5. 由于异常处理结构try...except...finally...中finally里的语句块总是被执行的，所以把关闭文件的代码放到finally块里肯定是万无一失，一定能保证文件被正确关闭并且不会引发任何异常。×

模拟题一

1. 单选题：
2. 关于 Python 的分支结构，以下选项中描述错误的是（B）。
3. Python 中 if-elif-else 语句描述多分支结构
4. 分支结构可以向已经执行过的语句部分跳转
5. Python 中 if-else 语句用来形成二分支结构
6. 分支结构使用 if 保留字
7. 关于 Python 对文件的处理，以下选项中描述错误的是（A）。
8. 当文件以文本(二进制)方式打开时，读写按照字节流方式。
9. Python 能够以文本和二进制两种方式处理文件。
10. Python 通过解释器内置的 open() 函数打开一个文件。
11. 文件使用结束后要用 close() 方法关闭，释放文件的使用授权。
12. 关于程序的异常处理，以下选项中描述错误的是（D）。
13. 异常语句可以与 else 和 finally 保留字配合使用。
14. Python 通过 try、except 等保留字提供异常处理功能。
15. 程序异常发生经过妥善处理可以继续执行。
16. 编程语言中的异常和错误是完全相同的概念。
17. 以下选项中不是 Python 对文件的写操作方法的是 （ D）。
18. Writelines C. write
19. write 和 seek D. writetext
20. 表达式list(zip([1,2], [3,4])) 的值为（C ）。Zip()函数打包为元组
21. ((1,2), (3,4)) C. [(1,3), (2,4)]
22. [[1, 3], [2,4]] D.[ 1,3,2,4]
23. os模块的（A）方法用来返回包含指定文件夹中所有的文件和子文件的列表。

A. listdir() B. fstat()

C. stat() D. remove()

1. 以下选项中不是 Python 数据分析的第三方库的是（A）。
2. Scipy B. requests

C. numpy D. pandas

1. Python语句f1 = lambda x:x\*3;f2=lambda x:x\*2; print(f1(f2(3)))的程序的运行结果是(D)。

A. 3       B. 6

C. 9          D. 18

1. 以下数据结构中，属于不可变类型的是（B ）。

A.字典 B.元组 C.列表 D.集合

1. 以下选项中，对于递归程序的描述错误的是（C ）。
2. 书写简单
3. 递归程序都可以有非递归编写方法
4. 执行效率高
5. 一定要有基例
6. 判断题
7. 假设os模块已导入，那么列表推导式 [filename for filename in os.listdir('C:\\Windows') if filename.endswith('.exe')] 的作用是列出C:\Windows文件夹中所有扩展名为.exe的文件。（V）
8. 只有Python扩展库才需要导入以后才能使用其中的对象，Python标准库不需要导入即可使用其中的所有对象和方法。（X）
9. 正则表达式模块re的match()方法是从字符串的开始匹配特定模式，而search()方法是在整个字符串中寻找模式，这两个方法如果匹配成功则返回match对象，匹配失败则返回空值None。（V）
10. 在函数内部没有办法定义全局变量。（X）
11. 在面向对象程序设计中，函数和方法是完全一样的，都必须为所有参数进行传值。（X）
12. 在函数调用时，在实参前面加上一个或两个星号(\*)可以进行序列解包，从而实现将序列中的元素值依次传递给相同数量的形参。（V）
13. 正则表达式'[^abc]'可以一个匹配任意除'a'、'b'、'c'之外的字符。（V）
14. 已知x和y是两个等长的整数列表，那么表达式sum((i\*j for i,j in zip(x, y)))的作用是计算这两个列表所表示的向量的内积。（V）
15. lambda表达式只能用来创建匿名函数，不能为这样的函数起名字。（X ）
16. Python字典中的“值”不允许重复。（X ）
17. 读程序写结果
18. 下面程序执行后的结果是( 11 )。

s="it's a beautiful day. TOM's mom is my ant"

flag=0

t=s.count(" ")

for ch in s:

if ch=="'":

flag=flag+1

counts=t+1+flag

print(counts) #输出结果

1. 以中国共产党第十九次全国代表大会报告中一句话作为字符串变量 s，完善 Python 程序，分别用 Python 内置函数及 jieba 库中已有函数计算字符串 s 的中文字符个数及中文词语个数。注意，中文字符包含中文标点符号。

import jieba

s = "中国特色社会主义进入新时代，我国社会主要矛盾已经转化为人民日益增长的美好生活需要和不平衡不充分的发展之间的矛盾。"

n = \_\_\_（1）\_\_\_ len(s)

m = \_\_\_\_（2）\_\_\_ len(jie.lcut(s))

print("中文字符数为{}，中文词语数为{}。".format(n,m))

1. 编程题
2. 小兰要为一条街的住户制作门牌号。这条街一共305位住户，门牌号是从1到305。小兰制作门牌号的方法是先制作0到9这几个数字字符，最后根据需要将字符粘贴到门牌上，例如门牌110需要依次粘贴字符1、1、0，即需要2个字符1，1个字符0。请问小兰制作完所有的1到305号门牌，总共需要多少个字符5？

**res = 0**

**for i in range(1, 305):**

**a = list(str(i))**

**res += a.count('5')**

**print(res)**

1. 小迪和小夏接到一个调研任务，需要按照省份统计班级同学的籍贯分布情况。他们决定分头统计男生和女生的籍贯分布，最后再汇总结果。已知小夏统计的女生籍贯分别是：江苏3人、北京2人、黑龙江1人、广东4人；小迪统计的男生籍贯分布是：江苏8人、浙江5人、山东5人、安徽4人、山西2人。请编写程序将两人的调研结果合并输出。

**from collections import Counter**

**xia\_dict={'江苏':3,'北京':2,'黑龙江':1,'广东':4}**

**di\_dict={'江苏':8,'浙江':5,'山东':5,'安徽':4,'山西':2}**

**xia\_dict,di\_dict=Counter(xia\_dict),Counter(di\_dict)**

**xia\_dict=dict(xia\_dict+di\_dict)**

**print(xia\_dict)**

1. 有一个列表students存放的是学生的成绩，要求将students里面的数据提取出来放到字典scores中，在屏幕上按照学号从小到大输出各位同学的成绩。

students=[{"id":"103","chinese":90,"math":85,"English":86}，

{"id":"102","chinese":95,"math":98,"English":96},

{"id":"101","chinese":80,"math":81,"English":76}]

输出结果：101：[80,81,76]

102：[95,98,96]

103：[90,85,86]

**students= [**

**{'id':'103','Chinese': 90,'Math':85,'English':86},**

**{'id':'102','Chinese': 95,'Math':98,'English':96},**

**{'id':'101','Chinese': 80,'Math':81,'English':76}**

**]**

**scores = {}**

**for student in students:**

**sv = student.items()**

**v = []**

**for it in sv:**

**if it[0] =='id':**

**k = it[1]**

**else:**

**v.append(it[1])**

**scores[k] = v**

**# print(scores)**

**so = list(scores.items())**

**so.sort(key = lambda x:x[0],reverse = False)**

**for l in so:**

**print('{}:{}'.format(l[0],l[1]))**