目 录

基础知识点

Python 变量类型

变量赋值

1. Python 中的变量赋值不需要类型声明。
2. 每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。
3. 每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。
4. 等号（=）用来给变量赋值。
5. 等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值。例如：

```
counter = 100   # 赋值整型变量
miles = 1000.0   # 浮点型
name = "John"   # 字符串
print (counter)
print (miles)
print (name)
```

多个变量赋值

1. Python 允许你同时为多个变量赋值。例如：

```
a = b = c = 1
```

说明：以上实例，创建一个整型对象，值为 1，三个变量被分配到相同的内存空间上。

2. 您也可以为多个对象指定多个变量。例如：

```
a, b, c = 1, 2, "john"
```

说明：以上实例，两个整型对象 1 和 2 的分配给变量 a 和 b，字符串对象 "john" 分配给变量 c。

标准数据类型

1. 在内存中存储的数据可以有多种类型。例如，一个人的年龄可以用数字来存储，他的名字可以用字符来存储。
2. Python 定义了一些标准类型，用于存储各种类型的数据。
3. Python 有五个标准的数据类型：
   a) Numbers（数字）
   b) String（字符串）
   c) List（列表）
   d) Tuple（元组）
   e) Dictionary（字典）

Python 数字

1. 数字数据类型用于存储数值。
   他们是不可改变的数据类型，这意味着改变数字数据类型会分配一个新的对象。

当你指定一个值时，Number 对象就会被创建：

```
var1 = 1
var2 = 10
```

2. 可以使用 del 语句删除一些对象的引用。

del 语句的语法是：

del var1[,var2[,var3[....,varN]]]

可以通过使用 del 语句删除单个或多个对象的引用。例如：

del var

del var_a, var_b

3. Python 支持四种不同的数字类型：

a) int（有符号整型）

b) long（长整型[也可以代表八进制和十六进制]）

c) float（浮点型）

d) complex（复数）

Python 字符串

```
str = 'Hello World!'
print(str) # 输出完整字符串
print(str[0] )# 输出字符串中的第一个字符
print(str[2:5])# 输出字符串中第三个至第五个之间的字符串
print(str[2:])  # 输出从第三个字符开始的字符串
print(str * 2)  # 输出字符串两次
print(str + "TEST")# 输出连接的字符串
```

Python 列表

List（列表） 是 Python 中使用最频繁的数据类型。

列表的功能

可以完成大多数集合类的数据结构实现。它支持字符，数字，字符串甚至可以包含列表（即嵌套）。

列表的表示

用 **[ ]** 标识，是 python 最通用的复合数据类型。

列表的切片与组合

✧ 可以用到变量 **[头下标:尾下标]**，就可以截取相应的列表，从左到右索引默认 0 开始，从右到左索引默认 -1 开始，下标可以为空表示取到头或尾。

✧ 加号 + 是列表连接运算符，星号 * 是重复操作。

如下实例：

```
list = ['runoob', 786, 2.23, 'john', 70.2]
tinylist = [123, 'john']
print(list)# 输出完整列表
print(list[0])# 输出列表的第一个元素
print(list[1:3])# 输出第二个至第三个元素
print(list[2:])# 输出从第三个开始至列表末尾的所有元素
print(tinylist * 2 )# 输出列表两次
print(list + tinylist)# 打印组合的列表
```

更新列表

你可以对列表的数据项进行修改或更新，你也可以使用 append()方法来添加列表项，如下所示：

```
list = [] ##  空列表
list.append('Google') ##  使用  append()  添加元素
list.append('Runoob')
print(list)
```

删除列表元素

可以使用 del 语句来删除列表的元素，如下实例：

```
list1 = ['physics', 'chemistry', 1997, 2000]
print(list1)
del list1[2]
print("After deleting value at index 2 : ")
print(list1)
```

Python 列表脚本操作符

列表对 + 和 * 的操作符与字符串相似。+ 号用于组合列表，* 号用于重复列表。如下所示：

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print(x)	1 2 3	迭代

Python 列表函数&方法

Python 包含以下函数:

a) cmp(list1, list2)比较两个列表的元素
b) len(list)列表元素个数
c) max(list)返回列表元素最大值
d) min(list)返回列表元素最小值
e) list(seq)将元组转换为列表

Python 包含以下方法:

a) list.append(obj)在列表末尾添加新的对象
b) list.count(obj)统计某个元素在列表中出现的次数
c) list.extend(seq)在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
d) list.index(obj)从列表中找出某个值第一个匹配项的索引位置
e) list.insert(index, obj)将对象插入列表
f) list.pop(obj=list[-1])移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
g) list.remove(obj)移除列表中某个值的第一个匹配项
h) list.reverse()反向列表中元素

i) list.sort([func])对原列表进行排序

Python 元组

元组是另一个数据类型，类似于 List（列表）。

元组用"()"标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表

修改元组

元组中的元素值是不允许修改的，但我们可以对元组进行连接组合，如下实例:

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# 以下修改元组元素操作是非法的。# tup1[0] = 100

# 创建一个新的元组
tup3 = tup1 + tup2
print(tup3)
```

删除元组

元组中的元素值是不允许删除的，但我们可以使用 del 语句来删除整个元组，如下实例:

```
tup = ('physics', 'chemistry', 1997, 2000)
print(tup)
del tup
print("After deleting tup : ")
print(tup)
```

元组运算符

与字符串一样，元组之间可以使用 + 号和 * 号进行运算。这就意味着他们可以组合和复制，运算后会生成一个新的元组。

Python 表达式	**结果**	**描述**
len((1, 2, 3))	3	计算元素个数
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	连接
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	复制
3 in (1, 2, 3)	True	元素是否存在
for x in (1, 2, 3): print x	1 2 3	迭代

元组索引，截取

因为元组也是一个序列，所以我们可以访问元组中的指定位置的元素，也可以截取索引中的一段元素，如下所示：

元组：

```
L = ('spam', 'Spam', 'SPAM!')
```

Python 表达式	结果	描述
L[2]	'SPAM!'	读取第三个元素
L[-2]	'Spam'	反向读取；读取倒数第二个元素
L[1:]	('Spam', 'SPAM!')	截取元素

无关闭分隔符

任意无符号的对象，以逗号隔开，默认为元组，如下实例：

```
print('abc', -4.24e93, 18+6.6j, 'xyz')
x, y = 1, 2
print("Value of x , y : ", x,y)
```

元组内置函数

Python 元组包含了以下内置函数

序号	函数
1	cmp(tuple1, tuple2)　比较两个元组元素。
2	len(tuple)　计算元组元素个数。
3	max(tuple)　返回元组中元素最大值。
4	min(tuple)　返回元组中元素最小值。
5	tuple(seq)　将列表转换为元组。

Python 字典

字典的定义

字典(dictionary)是除列表以外 python 之中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。

两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。字典用"{ }"标识。字典由索引(key)和它对应的值 value 组成。

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}
print(dict['one']) # 输出键为'one' 的值
print(dict[2] )# 输出键为 2 的值
print(tinydict)# 输出完整的字典
print(tinydict.keys()) # 输出所有键
print(tinydict.values())  # 输出所有值
```

修改字典

向字典添加新内容的方法是增加新的键/值对，修改或删除已有键/值对如下实例:

实例

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8
dict['School'] = "DPS School"  #增加新键值对
print(dict['Age'] )
print(dict['School'])
```

删除字典元素

能删单一的元素也能清空字典，清空只需一项操作。显示删除一个字典用 del 命令，如下实例：

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name'] # 删除键是'Name'的条目
dict.clear() # 清空词典所有条目
del dict # 删除词典
print("dict['Age']: ", dict['Age'])
print("dict['School']: ", dict['School'])
```

字典键的特性

字典值可以没有限制地取任何 python 对象，既可以是标准的对象，也可以是用户定义的，但键不行。

注意：不允许同一个键出现两次。创建时如果同一个键被赋值两次，后一个值会被记住，如下实例：

```
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print("dict['Name']: ", dict['Name'])
```

字典内置函数&方法

Python 字典包含了以下内置函数：

序号	函数及描述
1	cmp(dict1, dict2)　比较两个字典元素。
2	len(dict)　计算字典元素个数，即键的总数。
3	str(dict)　输出字典可打印的字符串表示。
4	type(variable)　返回输入的变量类型，如果变量是字典就返回字典类型。

Python 字典包含了以下内置方法：

序号	函数及描述
1	dict.clear()　删除字典内所有元素
2	dict.copy()　返回一个字典的浅复制

3	dict.fromkeys(seq[, val]) 创建一个新字典，以序列 seq 中元素做字典的键，val 为字典所有键对应的初始值
4	dict.get(key, default=None) 返回指定键的值，如果值不在字典中返回 default 值
5	dict.has_key(key) 如果键在字典 dict 里返回 true，否则返回 false
6	dict.items() 以列表返回可遍历的(键, 值) 元组数组
7	dict.keys() 以列表返回一个字典所有的键
8	dict.setdefault(key, default=None) 和 get()类似, 但如果键不存在于字典中，将会添加键并将值设为 default
9	dict.update(dict2) 把字典 dict2 的键/值对更新到 dict 里
10	dict.values() 以列表返回字典中的所有值
11	pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key 值必须给出。 否则，返回 default 值。
12	popitem() 随机返回并删除字典中的一对键和值。

Python 数据类型转换

有时候，我们需要对数据内置的类型进行转换，数据类型的转换，你只需要将数据类型作为函数名即可。

以下几个内置的函数可以执行数据类型之间的转换。这些函数返回一个新的对象，表示转换的值。

函数	描述
int(x [,base])	将 x 转换为一个整数
long(x [,base])	将 x 转换为一个长整数
float(x)	将 x 转换到一个浮点数
complex(real [,imag])	创建一个复数
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效 Python 表达式,并返回一个对象

tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表
set(s)	转换为可变集合
dict(d)	创建一个字典。d 必须是一个序列 (key,value)元组。
frozenset(s)	转换为不可变集合
chr(x)	将一个整数转换为一个字符
unichr(x)	将一个整数转换为 Unicode 字符
ord(x)	将一个字符转换为它的整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串

Python 运算

Python 算术运算符

以下假设变量： **a=10，b=20：**

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x 除以 y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回 x 的 y 次幂	a**b 为 10 的 20 次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

Python 比较运算符

以下假设变量 a 为 10，变量 b 为 20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
>	大于 - 返回 x 是否大于 y	(a > b) 返回 False。
<	小于 - 返回 x 是否小于 y。所有比较运算符返回 1 表示真，返回 0 表示假。这分别与特殊的变量 True 和 False 等价。	(a < b) 返回 true。
>=	大于等于 - 返回 x 是否大于等于 y。	(a >= b) 返回 False。
<=	小于等于 - 返回 x 是否小于等于 y。	(a <= b) 返回 true。

Python 运算符优先级

以下表格列出了从最高到最低优先级的所有运算符：

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not or and	逻辑运算符

Python 条件语句

```
if 判断条件：
    执行语句……
else：
    执行语句……
if 判断条件 1:
    执行语句 1……
elif 判断条件 2:
    执行语句 2……
elif 判断条件 3:
    执行语句 3……
else:
    执行语句 4……
```

Python 循环语句

Python 提供了 for 循环和 while 循环（在 Python 中没有 do..while 循环）：

循环类型	描述
while 循环	在给定的判断条件为 true 时执行循环体，否则退出循环体。
for 循环	重复执行语句
嵌套循环	你可以在 while 循环体中嵌套 for 循环

1）while 循环

```
while 判断条件：
    执行语句……
```

while 语句时还有另外两个重要的命令 continue，break 来跳过循环，continue 用于跳过该次循环，break 则是用于退出循环，此外"判断条件"还可以是个常值，表示循环必定成立，具体用法如下：

```
# continue 和 break 用法
i = 1
while i < 10:
  i += 1
  if i%2 > 0: # 非双数时跳过输出
    continue
  print(i) # 输出双数 2、4、6、8、10

i = 1
while 1: # 循环条件为 1 必定成立
    print(i) # 输出 1~10
    i += 1
    if i > 10: # 当 i 大于 10 时跳出循环
```

```
        break
```

循环使用 else 语句

在 python 中，while … else 在循环条件为 false 时执行 else 语句块：

2）For 循环

循环使用 else 语句

在 python 中，for … else 表示这样的意思，for 中的语句和普通的没有区别，else 中的语句会在循环正常执行完（即 for 不是通过 break 跳出而中断的）的情况下执行，while … else 也是一样，如：

```
for i in sequence:
    statements
else:
    statements
```

Python for 循环嵌套语法：

```
for i in sequence:
    for i in sequence:
        statements
```

Python while 循环嵌套语法：

```
while expression:
    while expression:
        statement
```

你可以在循环体内嵌入其他的循环体，如在 while 循环中可以嵌入 for 循环， 反之，你可以在 for 循环中嵌入 while 循环。

循环控制语句

循环控制语句可以更改语句执行的顺序。Python 支持以下循环控制语句：

控制语句	描述
break 语句	在语句块执行过程中终止循环，并且跳出整个循环
continue 语句	在语句块执行过程中终止当前循环，跳出该次循环，执行下一次循环。
pass 语句	pass 是空语句，是为了保持程序结构的完整性。

Python break 语句

Python break 语句，就像在 C 语言中，打破了最小封闭 for 或 while 循环。

break 语句用来终止循环语句，即循环条件没有 False 条件或者序列还没被完全递归完，也会停止执行循环语句。

break 语句用在 while 和 for 循环中。

如果您使用嵌套循环，break 语句将停止执行最深层的循环，并开始执行下一行代码。

Python continue 语句

Python continue 语句跳出本次循环，而 break 跳出整个循环。

continue 语句用来告诉 Python 跳过当前循环的剩余语句，然后继续进行下一轮循环。

continue 语句用在 while 和 for 循环中。

Python pass 语句

Python pass 是空语句，是为了保持程序结构的完整性。

pass 不做任何事情，一般用做占位语句。

Python 日期和时间

Python 程序能用很多方式处理日期和时间，转换日期格式是一个常见的功能。

Python 提供了一个 time 和 calendar 模块可以用于格式化日期和时间。

时间间隔是以秒为单位的浮点小数。

每个时间戳都以自从 1970 年 1 月 1 日午夜（历元）经过了多长时间来表示。

Python 的 time 模块下有很多函数可以转换常见日期格式。如函数 time.time()用于获取当前时间戳, 如下实例:

```
import time;  # 引入 time 模块
ticks = time.time()
print ("当前时间戳为:", ticks)
```

上述也就是 struct_time 元组。这种结构具有如下属性：

序号	属性	值
0	tm_year	2008
1	tm_mon	1 到 12
2	tm_mday	1 到 31
3	tm_hour	0 到 23
4	tm_min	0 到 59
5	tm_sec	0 到 61 (60 或 61 是闰秒)
6	tm_wday	0 到 6 (0 是周一)
7	tm_yday	1 到 366(儒略历)
8	tm_isdst	-1, 0, 1, -1 是决定是否为夏令时的旗帜

获取当前时间

从返回浮点数的时间戳方式向时间元组转换，只要将浮点数传递给如 localtime 之类的函数。

```
import time
localtime = time.localtime(time.time())
print ("本地时间为 :", localtime)
```

格式化日期

我们可以使用 time 模块的 strftime 方法来格式化日期：

```
time.strftime(format[, t])
```

```
import time
# 格式化成 2016-03-20 11:45:39 形式
print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
# 格式化成 Sat Mar 28 22:24:24 2016 形式
print(time.strftime("%a %b %d %H:%M:%S %Y", time.localtime()))
# 将格式字符串转换为时间戳
a = "Sat Mar 28 22:24:24 2016"
print(time.mktime(time.strptime(a,"%a %b %d %H:%M:%S %Y")))
```

python 中时间日期格式化符号：

a) %y 两位数的年份表示（00-99）
b) %Y 四位数的年份表示（000-9999）
c) %m 月份（01-12）
d) %d 月内中的一天（0-31）
e) %H 24 小时制小时数（0-23）
f) %I 12 小时制小时数（01-12）
g) %M 分钟数（00=59）
h) %S 秒（00-59）
i) %a 本地简化星期名称
j) %A 本地完整星期名称
k) %b 本地简化的月份名称
l) %B 本地完整的月份名称
m) %c 本地相应的日期表示和时间表示
n) %j 年内的一天（001-366）
o) %p 本地 A.M.或 P.M.的等价符
p) %U 一年中的星期数（00-53）星期天为星期的开始
q) %w 星期（0-6），星期天为星期的开始
r) %W 一年中的星期数（00-53）星期一为星期的开始
s) %x 本地相应的日期表示
t) %X 本地相应的时间表示
u) %Z 当前时区的名称
v) %% %号本身

获取某月日历

Calendar 模块有很广泛的方法用来处理年历和月历，例如打印某月的月历：

```
import calendar
cal = calendar.month(2016, 1)
print("以下输出 2016 年 1 月份的日历:"+cal)
```

Time 模块

Time 模块包含了以下内置函数，既有时间处理的，也有转换时间格式的：

序号	函数及描述
1	time.altzone 返回格林威治西部的夏令时地区的偏移秒数。如果该地区在格林威治东部会返回负值（如西欧，包括英国）。对夏令时启用地区才能使用。
2	time.asctime([tupletime]) 接受时间元组并返回一个可读的形式为"Tue Dec 11 18:07:14 2008"（2008 年 12 月 11 日 周二 18 时 07 分 14 秒）的 24 个字符的字符串。
3	time.clock( ) 用以浮点数计算的秒数返回当前的 CPU 时间。用来衡量不同程序的耗时，比 time.time()更有用。
4	time.ctime([secs]) 作用相当于 asctime(localtime(secs))，未给参数相当于 asctime()
5	time.gmtime([secs]) 接收时间戳（1970 纪元后经过的浮点秒数）并返回格林威治天文时间下的时间元组 t。注：t.tm_isdst 始终为 0
6	time.localtime([secs]) 接收时间戳（1970 纪元后经过的浮点秒数）并返回当地时间下的时间元组 t（t.tm_isdst 可取 0 或 1，取决于当地当时是不是夏令时）。
7	time.mktime(tupletime) 接受时间元组并返回时间戳（1970 纪元后经过的浮点秒数）。
8	time.sleep(secs) 推迟调用线程的运行，secs 指秒数。
9	time.strftime(fmt[,tupletime]) 接收以时间元组，并返回以可读字符串表示的当地时间，格式由 fmt 决定。
10	time.strptime(str,fmt='%a %b %d %H:%M:%S %Y') 根据 fmt 的格式把一个时间字符串解析为时间元组。
11	time.time( ) 返回当前时间的时间戳（1970 纪元后经过的浮点秒数）。
12	time.tzset() 根据环境变量 TZ 重新初始化时间相关设置。

Time 模块包含了以下 2 个非常重要的属性：

序号	属性及描述
1	**time.timezone** 属性 time.timezone 是当地时区（未启动夏令时）距离格林威治的偏移秒数（>0，美洲;<=0 大部分欧洲，亚洲，非洲）。
2	**time.tzname** 属性 time.tzname 包含一对根据情况的不同而不同的字符串，分别是带夏令时的本地时区名称，和不带的。

日历（Calendar）模块

此模块的函数都是日历相关的，例如打印某月的字符月历。

星期一是默认的每周第一天，星期天是默认的最后一天。更改设置需调用 calendar.setfirstweekday()函数。模块包含了以下内置函数：

序号	函数及描述
1	calendar.calendar(year,w=2,l=1,c=6) 返回一个多行字符串格式的 year 年年历，3 个月一行，间隔距离为 c。 每日宽度间隔为 w 字符。每行长度为 21* W+18+2* C。l 是每星期行数。
2	calendar.firstweekday( ) 返回当前每周起始日期的设置。默认情况下，首次载入 caendar 模块时返回 0，即星期一。
3	calendar.isleap(year)　是闰年返回 True，否则为 false。
4	calendar.leapdays(y1,y2)　返回在 Y1，Y2 两年之间的闰年总数。
5	calendar.month(year,month,w=2,l=1) 返回一个多行字符串格式的 year 年 month 月日历，两行标题，一周一行。每日宽度间隔为 w 字符。每行的长度为 7* w+6。l 是每星期的行数。
6	calendar.monthcalendar(year,month) 返回一个整数的单层嵌套列表。每个子列表装载代表一个星期的整数。Year 年 month 月外的日期都设为 0;范围内的日子都由该月第几日表示，从 1 开始。
7	calendar.monthrange(year,month) 返回两个整数。第一个是该月的星期几的日期码，第二个是该月的日期码。日从 0（星期一）到 6（星期日）;月从 1 到 12。
8	calendar.prcal(year,w=2,l=1,c=6)　相当于 print calendar.calendar(year,w,l,c).

9	calendar.prmonth(year,month,w=2,l=1) 相当于 print calendar.calendar（year，w，l，c）。
10	calendar.setfirstweekday(weekday) 设置每周的起始日期码。0（星期一）到 6（星期日）。
11	calendar.timegm(tupletime) 和 time.gmtime 相反：接受一个时间元组形式，返回该时刻的时间戳（1970 纪元后经过的浮点秒数）。
12	calendar.weekday(year,month,day) 返回给定日期的日期码。0（星期一）到 6（星期日）。月份为 1（一月） 到 12（12 月）。

Python 函数

函数的定义

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。

函数能提高应用的模块性，和代码的重复利用率。你已经知道 Python 提供了许多内建函数，比如 print()。但你也可以自己创建函数，这被叫做用户自定义函数。

定义一个函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

1) 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号()。
2) 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
3) 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
4) 函数内容以冒号起始，并且缩进。
5) return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的 return 相当于返回 None。

语法如下：

```
def functionname(parameters):
    "函数_文档字符串"
    function_suite
    return [expression]
```

参数传递

在 python 中，类型属于对象，变量是没有类型的：Python 参数分为可更改(mutable)与不可更改(immutable)对象。

在 python 中，整数、字符串、元组是不可更改的对象，而列表，字典等则是可以修改的对象。

1) **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。
2) **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 模块

什么是 python 模块

Python 模块(Module)，是一个 Python 文件，以 .py 结尾，包含了 Python 对象定义和 Python 语句。

模块让你能够有逻辑地组织你的 Python 代码段。

把相关的代码分配到一个模块里能让你的代码更好用，更易懂。

模块能定义函数，类和变量，模块里也能包含可执行的代码。

模块的定义

例子：

下例是个简单的模块 support.py：

```
def print_func(par):
    print("Hello : "+par)
    return
```

模块的引入

模块定义好后，我们可以使用 import 语句来引入模块，语法如下：

我们有三种方法来引入模块：

1.import module1[, module2[,... moduleN]

比如要引用模块 math，就可以在文件最开始的地方用 **import math** 来引入。在调用 math 模块中的函数时，必须这样引用：

模块名.函数名

2.from…import 语句

Python 的 from 语句让你从模块中导入一个指定的部分到当前命名空间中。语法如下：

例如，要导入模块 fib 的 fibonacci 函数，使用如下语句：

from fib import fibonacci

3.from…import* 语句

from…import* 语句把一个模块中所有函数都导入进来;注:相当于导入的是一个文件夹中所有文件,所有函数都是绝对路径

dir()函数

dir() 函数一个排好序的字符串列表，内容是一个模块里定义过的名字。

返回的列表容纳了在一个模块里定义的所有模块，变量和函数。

globals() 和 locals() 函数

根据调用地方的不同，globals() 和 locals() 函数可被用来返回全局和局部命名空间里的名字。

如果在函数内部调用 locals()，返回的是所有能在该函数里访问的命名。

如果在函数内部调用 globals()，返回的是所有在该函数里能访问的全局名字。

两个函数的返回类型都是字典。所以名字们能用 keys() 函数摘取。

reload() 函数

当一个模块被导入到一个脚本，模块顶层部分的代码只会被执行一次。

因此，如果你想重新执行模块里顶层部分的代码，可以用 reload() 函数。该函数会重新导入之前导入过的模块。

在这里，module_name 要直接放模块的名字，而不是一个字符串形式。比如想重载 hello 模块，如下：

reload(hello)

Python 中的包

包是一个分层次的文件目录结构，它定义了一个由模块及子包，和子包下的子包等组成的 Python 的应用环境。

简单来说，包就是文件夹，但该文件夹下必须存在 __init__.py 文件，该文件的内容可以为空。__init__.py 用于标识当前文件夹是一个包。

Python 文件 I/O

本章只讲述所有基本的的 I/O 函数，更多函数请参考 Python 标准文档。

打印到屏幕

最简单的输出方法是用 print 语句，你可以给它传递零个或多个用逗号隔开的表达式。此函数把你传递的表达式转换成一个字符串表达式

读取键盘输入

Python 提供了一个内置函数从标准输入读入一行文本，默认的标准输入是键盘。如下：

input 函数

input([prompt]) 函数从标准输入读取一个行，并返回一个字符串（去掉结尾的换行符），可以接收一个 Python 表达式作为输入，并将运算结果返回。

打开和关闭文件

现在，您已经可以向标准输入和输出进行读写。现在，来看看怎么读写实际的数据文件。

Python 提供了必要的函数和方法进行默认情况下的文件基本操作。你可以用 **file** 对象做大部分的文件操作。

open 函数

你必须先用 Python 内置的 open()函数打开一个文件，创建一个 file 对象，相关的方法才可以调用它进行读写。

语法：

file object = open(file_name [, access_mode][, buffering])

File 对象的属性

一个文件被打开后，你有一个 file 对象，你可以得到有关该文件的各种信息。

以下是和 file 对象相关的所有属性的列表：

属性	描述
file.closed	返回 true 如果文件已被关闭，否则返回 false。
file.mode	返回被打开文件的访问模式。
file.name	返回文件的名称。
file.softspace	如果用 print 输出后，必须跟一个空格符，则返回 false。否则返回 true。

close()方法

File 对象的 close（）方法刷新缓冲区里任何还没写入的信息，并关闭该文件，这之后便不能再进行写入。

当一个文件对象的引用被重新指定给另一个文件时，Python 会关闭之前的文件。用 close（）方法关闭文件是一个很好的习惯。

语法：

fileObject.close()

write()方法

write()方法可将任何字符串写入一个打开的文件。需要重点注意的是，Python 字符串可以是二进制数据，而不是仅仅是文字。

write()方法不会在字符串的结尾添加换行符('\n')：

fileObject.write(string)

read()方法

read（）方法从一个打开的文件中读取一个字符串。需要重点注意的是，Python 字符串可以是二进制数据，而不是仅仅是文字。

语法：fileObject.read([count])

Python 的 os 模块提供了帮你执行文件处理操作的方法，比如重命名和删除文件。

要使用这个模块，你必须先导入它，然后才可以调用相关的各种功能。

rename()方法：

rename()方法需要两个参数，当前的文件名和新文件名。

os.rename(current_file_name, new_file_name)

你可以用 remove()方法删除文件，需要提供要删除的文件名作为参数。

os.remove(file_name)

Python 里的目录：

所有文件都包含在各个不同的目录下，不过 Python 也能轻松处理。os 模块有许多方法能帮你创建，删除和更改目录。

mkdir()方法

可以使用 os 模块的 mkdir()方法在当前目录下创建新的目录们。你需要提供一个包含了要创建的目录名称的参数。

chdir()方法

可以用 chdir()方法来改变当前的目录。chdir()方法需要的一个参数是你想设成当前目录的目录名称。

rmdir()方法

rmdir()方法删除目录，目录名称以参数传递。

在删除这个目录之前，它的所有内容应该先被清除。

文件、目录相关的方法

File 对象和 OS 对象提供了很多文件与目录的操作方法。

1. File 对象方法: file 对象提供了操作文件的一系列方法。
2. OS 对象方法: 提供了处理文件及目录的一系列方法

Python 经典题库

一、简答题

1、 写出python导入模块的关键字。

2、 写出 Python 运算符&的两种功能？

3、 简单解释 Python 基于值的自动内存管理方式？

4、 在 Python 中导入模块中的对象有哪几种方式？

5、 解释 Python 脚本程序的“__ name__ ”变量及其作用？

6、 为什么应尽量从列表的尾部进行元素的增加与删除操作？

7、 分析逻辑运算符“or”的短路求值特性？

8、 简单解释 Python 中的字符串驻留机制？

9、 异常和错误有什么区别？

10、使用 pdb 模块进行 Python 程序调试主要有哪几种用法？

11、阅读下面的代码，并分析假设文件“D:\test.txt”不存在的情况下两段代码可能发生的问题。

代码 1：

```
 try:
    fp = open(r'd:\test.txt')
    print('Hello world!', file=fp)
finally:
    fp.close()
```

代码2：

```
try:
    fp = open(r'd:\test.txt', 'a+')
    print('Hello world!', file=fp)
finally:
    fp.close()
```

12、下面的代码本意是把当前文件夹中所有 html 文件都改为 htm 文件，仔细阅读代码，简要说明可能存在的问题。

```
import os
file_list=os.listdir(".")
for filename in file_list:
    pos = filename.rindex(".")
    if filename[pos+1:] == "html":
        newname = filename[:pos+1]+"htm"
        os.rename(filename,newname)
        print(filename+"更名为： "+newname)
```

【解析】

一、简答题

1、（1）import　（2）from * import *

2、（1）数字位运算；2）集合交集运算

3、Python 采用的是基于值的内存管理方式，在 Python 中可以为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一个内存地址；Python 具有自动内存管理功能，会自动跟踪内存中所有的值，对于没有任何变量指向的值，Python 自动将其删除。

4、（1）import 模块名 [as 别名]；（2）from 模块名 import 对象名[as 别名]；（3）from 模块名 import *

5、每个 Python 脚本在运行时都有一个“__name__”属性。如果脚本作为模块被导入，则其“__name__”属性的值被自动设置为模块名；如果脚本独立运行，则其“__name__”属性值被自动设置为“__main__”。利用“__name__”属性即可控制 Python 程序的运行方式。

6、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。

7、假设有表达式“表达式 1 or 表达式 2”，如果表达式 1 的值等价于 True，那么无论表达式 2 的值是什么，整个表达式的值总是等价于 True。因此，不需要再计算表达式 2 的值。

8、对于短字符串，将其赋值给多个不同的对象时，内存中只有一个副本，多个对象共享改副本。

9、python 的错误包括语法代码逻辑错误，或不合法的输入错误，或者试图访问不存在的文件等，导致程序无法正常运行；

python 异常分为异常的产生和异常的处理，异常产生，检查到错误且解释器认为是异常，抛出异常；异常处理，截获异常，忽略或者终止程序处理异常。

10、

1) 在交互模式下使用 pdb 模块提供的功能可以直接调试语句块、表达式、函数等多种脚本。
2) 在程序中嵌入断点来实现调试功能。在程序中首先导入 pdb 模块，然后使用 pdb.set_trace()。在需要的位置设置断点。如果程序中存在通过该方法调用显式插入的断点，那么在命令提示符环境下执行该程序或双击执行程序时将自动打开 pdb 调试环境，即使该程序当前不处于调试状态。
3) 使用命令行调试程序。在命令行提示符下执行“python –m pdb 脚本文件名”，则直接进入调试环境；当调试结束或程序正常结束以后，pdb 将重启该程序。）

11、假设文件“D:\test.txt”不存在，那么第一段代码会抛出异常，提示 fp 没有定义；第二段代码执行正常。原因是第二段代码使用内置函数 open()打开指定文件时如果不存在则会创建该文件，从而不会抛出异常。

12、答：对于字符串对象，如果要查找的子字符串不存在，则 rindex()方法会抛出异常。所以，如果当前文件夹中有不包含圆点的文件名或者子文件夹名，上面的代码会抛出异常而崩溃。

二、填空题

1、 Python 安装扩展库常用的是_____工具。

2、 Python 标准库 math 中用来计算平方根的函数是_____。

3、 Python 程序文件扩展名主要有_____和_____两种，其中后者常用于 GUI 程序。

4、 Python 源代码程序编译后的文件扩展名为_____。

5、 使用 pip 工具升级科学计算扩展库 numpy 的完整命令是_____。

6、 使用 pip 工具查看当前已安装的 Python 扩展库的完整命令是_____。

7、 在 IDLE 交互模式中浏览上一条语句的快捷键是_____。

8、 在 Python 中_____表示空类型。

9、 列表、元组、字符串是 Python 的_____(有序？无序)序列。

10、查看变量类型的 Python 内置函数是_____。

11、查看变量内存地址的 Python 内置函数是_____。

12、以 3 为实部 4 为虚部，Python 复数的表达形式为_____或_____。

13、Python 运算符中用来计算整商的是 _____。

14、Python 运算符中用来计算集合并集的是_____。

15、使用运算符测试集合包含集合 A 是否为集合 B 的真子集的表达式可以写作_____。

16、_____命令既可以删除列表中的一个元素，也可以删除整个列表。

17、表达式 int('123', 16) 的值为_____。

18、表达式 int('123', 8) 的值为_____。

19、表达式 int('123') 的值为_____。

20、表达式 int('101',2) 的值为_____。

21、表达式 abs(-3) 的值为_____。

22、Python 3.x 语句 print(1, 2, 3, sep=':') 的输出结果为_____。

23、表达式 int(4**0.5) 的值为_____。

24、Python 内置函数_____可以返回列表、元组、字典、集合、字符串以及 range对象中元素个数。

25、Python 内置函数_____用来返回序列中的最大元素。

26、Python 内置函数_____用来返回序列中的最小元素。

27、Python 内置函数_____用来返回数值型序列中所有元素之和。

28、为了提高 Python 代码运行速度和进行适当的保密，可以将 Python 程序文件编译为扩展名___的文件。

29、已知 x = 3，那么执行语句x += 6 之后，x 的值为_____。

30、表达式 3 | 5 的值为_____。

31、表达式 3 & 6 的值为_____。

32、表达式 3 ** 2 的值为_____。

33、表达式 3 * 2 的值为_____。

34、表达式 3<<2 的值为_____。

35、表达式 65 >> 1 的值为_____。

36、表达式 chr(ord('a')^32) 的值为_____。

37、表达式 chr(ord('a')-32) 的值为_____。

38、表达式 abs(3+4j) 的值为_____。

39、表达式 callable(int) 的值为_____。

40、假设有 Python 程序文件 abc.py，其中只有一条语句 print(__name__)，那么直接运行该程序时得到的结果为_____。

41、表达式 16**0.5 的值为_____。

42、表达式 type({3})的值为_____。

43、表达式 isinstance('Hello world', str)的值为_____。

44、表达式[1, 2, 3]*3 的执行结果为_____。

45、list(map(str, [1, 2, 3]))的执行结果为_____。

46、语句 x = 3==3, 5 执行结束后，变量 x 的值为_____。

47、已知 x = 3，并且 id(x)的返回值为 496103280，那么执行语句 x += 6 之后，表达式 id(x) == 496103280 的值为_____。

48、已知 x = 3，那么执行语句 x *= 6 之后，x 的值为_____。

49、表达式[3] in [1, 2, 3, 4]的值为_____。

50、列表对象的 sort()方法用来对列表元素进行原地排序，该函数返回值为_____。

【解析】

1. pip
2. sqrt
3. py、pyw
4. pyc
5. pip install--upgrade numpy
6. pip list
7. Alt+P
8. None
9. 有序
10. type( )
11. id( )
12. 3+4j、3+4J
13. //
14. |
15. A<B
16. del
17. 291
18. 83
19. 123
20. 5
21. 3
22. 1:2:3
23. 2
24. len( )
25. max( )
26. min( )
27. sum( )
28. pyc
29. 9
30. 7
31. 2
32. 9
33. 6
34. 12
35. 32
36. ‘A’
37. ‘A’
38. 5.0
39. True
40. __main__
41. 4.0
42. Set
43. True
44. [1,2,3,1,2,3,1,2,3]
45. [‘1’, ‘2’, ‘3’]
46. (True, 5）
47. False
48. 18
49. False
50. None

51、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]得到的值是____。

52、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_____。

53、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“键”，以列表 b 中的元素为“值”，这个语句可以写为_____。

54、任意长度的 Python 列表、元组和字符串中最后一个元素的下标为_____。

55、Python 语句 list(range(1,10,3))执行结果为_____。

56、表达式 list(range(5)) 的值为_____。

57、已知 a = [1, 2, 3]和 b = [1, 2, 4]，那么 id(a[1])==id(b[1])的执行结果为_____。

58、切片操作 list(range(6))[::2]执行结果为_____。

59、使用切片操作在列表对象 x 的开始处增加一个元素 3 的代码为_____。

60、语句 sorted([1, 2, 3], reverse=True) == reversed([1, 2, 3]) 执行结果为_____。

61、表达式 sorted([111, 2, 33], key=lambda x: len(str(x))) 的值为_____。

62、表达式 sorted([111, 2, 33], key=lambda x: -len(str(x))) 的值为_____。

63、语句 x = (3,) 执行后 x 的值为_____。

64、语句 x = (3) 执行后 x 的值为_____。

65、已知 x=3 和 y=5，执行语句 x, y = y, x 后 x 的值是_____。

66、可以使用内置函数_____查看包含当前作用域内所有全局变量和值的字典。

67、可以使用内置函数_____查看包含当前作用域内所有局部变量和值的字典。

68、字典中多个元素之间使用_____分隔开，每个元素的“键”与“值”之间使用_____分隔开。

69、字典对象的_____方法可以获取指定“键”对应的“值”，并且可以在指定“键”不存在的时候返回指定值，如果不指定则返回 None。

70、字典对象的_____方法返回字典中的“键-值对”列表。

71、字典对象的_____方法返回字典的“键”列表。

72、字典对象的_____方法返回字典的“值”列表。

73、已知 x = {1:2}，那么执行语句 x[2] = 3 之后，x 的值为_____。

74、表达式{1, 2, 3, 4} - {3, 4, 5, 6}的值为_____。

75、表达式 set([1, 1, 2, 3])的值为_____。

76、使用列表推导式得到 100 以内所有能被 13 整除的数的代码可以写作_____。

77、已知 x = [3, 5, 7]，那么表达式 x[10:]的值为__。

78、已知 x = [3, 5, 7]，那么执行语句 x[len(x):] = [1, 2]之后，x的值为_____。

79、已知 x = [3, 7, 5]，那么执行语句 x.sort(reverse=True)之后，x的值为_____。

80、已知 x = [3, 7, 5]，那么执行语句 x = x.sort(reverse=True)之后，x 的值为_____。

81、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_____。

82、表达式 list(zip([1,2], [3,4])) 的值为_____。

83、已知 x = [1, 2, 3, 2, 3]，执行语句 x.pop() 之后，x 的值为_____。

84、表达式 list(map(list,zip(*[[1, 2, 3], [4, 5, 6]]))) 的值为_____。

85、表达式 [x for x in [1,2,3,4,5] if x<3] 的值为_____。

86、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__。

87、已知 x = [3,5,3,7] ，那么表达式[x.index(i) for i in x if i==3]的值为___。

88、已知列表 x = [1, 2]，那么表达式 list(enumerate(x)) 的值为_____。

89、已知 vec = [[1,2], [3,4]] ，则表达式 [col for row in vec for col in row] 的值为_____。

90、已知 vec = [[1,2], [3,4]]，则表达式 [[row[i] for row in vec] for i in range(len(vec[0]))]的值为_____。

91、已知 x = list(range(10))，则表达式 x[-4:] 的值为_____。

92、已知 x = [3, 5, 7]，那么执行语句 x[1:] = [2]之后，x 的值为_____。

93、已知 x = [3, 5, 7]，那么执行语句 x[:3] = [2]之后，x 的值为_____。

94、已知 x 为非空列表， 那么执行语句 y = x[:] 之后， id(x[0]) == id(y[0]) 的值为_____。

95、已知 x = [1, 2, 3, 2, 3]，执行语句 x.remove(2) 之后，x 的值为_____。

96、表达式 len([i for i in range(10)]) 的值为_____。

97、表达式 len(range(1,10)) 的值为_____。

98、表达式 range(10)[-1] 的值为_____。

99、表达式 range(10,20)[4] 的值为_____。

100、表达式 round(3.4) 的值为_____。

【解析】

51. [6,7,9,11]
52. [5 for i in range(10)]
53. c = dict(zip(a, b))
54. -1
55. [1,4,7]
56. [0,1,2,3,4]
57. True
58. [0,2,4]
59. x[0:0]= [3]
60. False
61. [2, 33, 111]
62. [111, 33, 2]
63. (3,)
64. 3
65. 5
66. globals()
67. locals()
68. 逗号、冒号
69. get( )
70. items( )
71. keys( )
72. values( )
73. {1:2,2:3}
74. {1,2}
75. {1,2,3}
76. [i for i in range(100) if i%13==0]
77. []
78. [3,5,7,1,2]
79. [7,5,3]
80. None
81. [111, 11, 1]
82. [(1,3),(2,4)]
83. [1,2,3,2]
84. [[1,4],[2,5],[3,6]]
85. [1,2]
86. [2,4]
87. [0,0]
88. [(0,1),(1,2)]
89. [1,2,3,4]
90. [[1, 3], [2, 4]]
91. [6, 7, 8, 9]
92. [3,2]
93. [2]
94. True
95. [1,3, 2, 3]
96. 10
97. 9
98. 9
99. 14
100. 3

101、表达式 round(3.7) 的值为_____。

102、已知 x = (3)， 那么表达式 x * 3 的值为_____。

103、已知 x = (3,)，那么表达式 x * 3 的值为_。

104、假设列表对象 x = [1, 1, 1]，那么表达式 id(x[0]) == id(x[2])的值为_____。

105、已知列表 x = list(range(10))，那么执行语句 del x[::2]之后，x 的值为_____。

106、已知列表 x = [1, 2, 3, 4]，那么执行语句 del x[1] 之后 x 的值为_____。

107、表达式 [1] * 2 的值为_____。

108、表达式 [1, 2] * 2 的值为_____。

109、已知列表 x = [1, 2, 3]，那么执行语句 x.insert(1, 4) 只有，x 的值为_____。

110、已知列表 x = [1, 2, 3]，那么执行语句 x.insert(0, 4) 只有，x 的值为_____。

111、已知列表 x = [1, 2, 3]，那么执行语句 x.pop(0) 之后，x 的值为_____。

112、已 知 x =[[1]]*3，那么执 行 语 句 x[0][0]=5之后，变量x的值为_____。

113、表达式 list(map(lambda x: x+5, [1, 2, 3, 4, 5])) 的值为_____。

114、表达式 {1, 2, 3, 4, 5} ^ {4, 5, 6, 7} 的值为_____。

115、已知 x = [1, 2, 3]， 那么执行语句 x[len(x)-1:] = [4, 5, 6] 之后， 变量 x 的值为_____。

116、表达式 len(range(1, 10)) 的值为_____。

117、已知 x 是一个列表对象，那么执行语句 y = x[:]之后表达式 id(x) == id(y) 的值为_____。

118、表 达 式sorted([13, 1, 237, 89, 100], key=lambda x: len(str(x)))的值为_____。

119、已知 x = {1:2, 2:3}，那么表达式 x.get(3, 4) 的值为_____。

120、已知 x = {1:2, 2:3}，那么表达式 x.get(2, 4) 的值为_____。

121、表达式 {1, 2, 3} | {3, 4, 5} 的值为_____。

122、表达式 {1, 2, 3} | {2, 3, 4} 的值为_____。

123、表达式 {1, 2, 3} & {3, 4, 5} 的值为_____。

124、表达式 {1, 2, 3} & {2, 3, 4} 的值为_____。

125、表达式 {1, 2, 3} - {3, 4, 5} 的值为_____。

126、表达式 {1, 2, 3} < {3, 4, 5} 的值为_____。

127、表达式 {1, 2, 3} < {1, 2, 4} 的值为_____。

128、表达式 [1,2,3].count(4) 的值为_____。

129、Python 标准库 random 中的_____方法作用是从序列中随机选择 1 个元素。

130、Python标准库random中的sample(seq, k)方法作用是从序列中选择_____(重复？不重复？)的 k 个元素。

131、random 模块中_____方法的作用是将列表中的元素随机乱序。

132、执行代码 x, y, z = sorted([1, 3, 2]) 之后，变量 y 的值为_____。

133、表达式 (1, 2, 3)+(4, 5) 的值为_____。

134、表达式 dict(zip([1, 2], [3, 4])) 的值为_____。

135、语句 x, y, z = [1, 2, 3] 执行后，变量 y 的值为_____。

136、已知 x = [[1,3,3], [2,3,1]]，那么表达式 sorted(x, key=lambda item:item[0]+item[2])的值为_____。

137、已知x = [[1,3,3], [2,3,1]]，那么表达式 sorted(x, key=lambda item:(item[1],item[2]))的值为_____。

138、已知 x = [[1,3,3], [2,3,1]]，那么表达式sorted(x, key=lambda item:(item[1], -item[2]))的值为_____。

139、已知 x = {1, 2, 3}，那么执行语句 x.add(3) 之后，x 的值为____。

140、已知 x = {1:1}，那么执行语句 x[2] = 2 之后，len(x)的值为__。

141、已知 x = {1:1, 2:2}，那么执行语句 x[2] = 4 之后，len(x)的值为__。

142、假设已从标准库functools 导入reduce()函数，那么表达式 reduce(lambda x, y: x-y, [1, 2, 3])的值为____。

143、假设已从标准库functools 导入reduce()函数，那么表达式 reduce(lambda x, y: x+y, [1, 2, 3]) 的值为____。

144、假设已从标准库functools 导入reduce()函数，那么表达式 reduce(lambdax,y:max(x,y),[1,2,3,4,4,5])的值为__。

145、 已知有函数定义 def demo(*p):return sum(p)，那么表达式 demo(1, 2, 3) 的值为____、表达式 demo(1, 2, 3, 4) 的值为_____。

146、已知列表 x = [1, 2]，那么连续执行命令 y = x 和 y.append(3) 之后，x 的值为____。

147、已知列表 x = [1, 2]，那么连续执行命令 y = x[:] 和 y.append(3) 之后，x 的值为__。

148、已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 id(x) == id(y) 的值为__。

149、已知列表 x = [1, 2]，执行语句 y = x 后，表达式 id(x) == id(y) 的值为____。

150、已知列表 x = [1, 2]，执行语句 y = x 后，表达式 x is y 的值为_。

【解析】

101. 4
102. 9
103. (3,3,3)
104. True
105. [1,3,5,7,9]
106. [1,3, 4]
107. [1, 1]
108. [1, 2, 1, 2]
109. [1, 4, 2, 3]
110. [4, 1, 2, 3]
111. [2, 3]
112. [[5], [5], [5]]
113. [6, 7, 8, 9, 10]
114. {1, 2, 3, 6, 7}
115. [1,2,4,5,6]
116. 9
117. False
118. [1,13,89,237,100]
119. 4
120. 3
121. {1,2,3,4,5}
122. {1,2,3,4}
123. {3}
124. {2,3}
125. {1,2}
126. False
127. False
128. 0
129. choice()
130. 不重复
131. shuffle( )
132. 2
133. (1,2,3,4,5)
134. {1:3,2:4}
135. 2
136.[[2, 3, 1], [1, 3, 3]]
137.[[2, 3, 1], [1, 3, 3]]
138.[[1, 3, 3], [2, 3, 1]]
139. {1, 2, 3}
140. 2
141. 2
142. -4
143. 6
144. 5
145. 6、10
146. [1, 2, 3]
147. [1,2]
148. False
149. True
150. True

151、已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 x is not y 的值为_____。

152、表达式 sorted(random.sample(range(5), 5)) 的值为___。

153、表达式 [i for i in range(10) if i>8] 的值为_____。

154、已知有列表x=[[1,2,3],[4,5,6]]，那么表达式[[row[i] for row in x] for i in range(len(x[0]))]的值为____。

155、执行语句 x,y,z = map(str, range(3)) 之后，变量 y 的值为___。

156、已知列表 x = [1, 2]，那么执行语句 x.extend([3]) 之后， x 的值为_____。

157、已知列表 x = [1, 2]，那么执行语句 x.append([3]) 之后，x 的值为___。

158、表达式 print(0b10101) 的值为___。
159、已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[:3] 之后，x 的值为____。
160、已知 x = range(1,4) 和 y = range(4,7)，那么表达式sum([i*j for i,j in zip(x,y)])的值为____。
161、表达式 [5 for i in range(3)] 的值为_____。
162、表达式 {1, 2, 3} == {1, 3, 2} 的值为_____。
163、表达式 [1, 2, 3] == [1, 3, 2] 的值为_____。
164、已知 x = [1, 2, 1]，那么表达式 id(x[0]) == id(x[2])的值为_____。
165、表达式 3 not in [1, 2, 3]的值为_____。
166、已知 x = [1, 2]，那么执行语句 x[0:0] = [3, 3]之后，x 的值为____。
167、已知 x = [1, 2]，那么执行语句 x[0:1] = [3, 3]之后，x 的值为____。
168、已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[1:3] 之后，x 的值为_____。
169、已知 x = [[1, 2, 3,], [4, 5, 6]]，那么表达式 sum([i*j for i,j in zip(*x)]) 的值为__。
170、已知列表 x = [1, 2, 3]和y = [4, 5, 6]，那么表达式[(i,j) for i, j in zip(x,y) if i==3]的值为____。
171、已知列表 x = [1.0, 2.0, 3.0]，那么表达式 sum(x)/len(x)的值为_____。
172、已知 x = {1:2, 2:3, 3:4}，那么表达式 sum(x) 的值为_____。
173、已知 x = {1:2, 2:3, 3:4}，那么表达式 sum(x.values()) 的值为_____。
174、已知 x = [3, 2, 3, 3, 4] ，那么表达式[index for index, value in enumerate(x) if value==3]的值为_____。
175、表达式 1234%1000//100 的值为_____。
176、表达式 3 // 5 的值为_____。
177、表达式 [1, 2] + [3] 的值为_____。
178、表达式 (1,) + (2,) 的值为_____。
179、表达式 (1) + (2) 的值为_____。
180、已知 x, y = map(int, ['1', '2'])，那么表达式 x + y 的值为__。
181、已知列表 x = list(range(5))，那么执行语句 x.remove(3) 之后，表达式 x.index(4)的值为__
_____。
182、已知列表 x = [1, 3, 2]，那么执行语句 x.reverse() 之后，x 的值为___。
183、已知列表 x = [1, 3, 2]，那么执行语句 x = x.reverse() 之后，x 的值为___。
184、已知 x 为非空列表，那么表达式 x.reverse() == list(reversed(x)) 的值为__。
185、已知 x 为非空列表，那么表达式 x.sort() == sorted(x) 的值为____。
186、已知列表 x = [1, 3, 2] ， 那么执行语句 y = list(reversed(x)) 之后， x 的值为 ____。
187、已知列表 x = [1, 3, 2] ， 那么执行语句 y = list(reversed(x)) 之后， y 的值为 ____。
188、已知列表 x 中包含超过 5 个以上的元素，那么表达式 x == x[:5]+x[5:] 的值为_____。
189、已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 sum(x) 的值为____。
190、已知字典 x = {i:str(i+3) for i in range(3)} ， 那么表达式 sum(item[0] for item in x.items())的值为____。
191、已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 ''.join([item[1] for item in x.items()])的值为_____。
192、已知列表 x = [1, 3, 2] ， 那么表达式 [value for index, value in enumerate(x) if index==2]的值为____。
193、已知列表 x = [1, 3, 2]，那么执行语句 a, b, c = sorted(x) 之后，b 的值为_____。
194、已知列表 x = [1, 3, 2]，那么执行语句 a, b, c = map(str,sorted(x))之后，c的值为_____。
195、表达式set([1,2,3]) == {1, 2, 3} 的值为_____。

196、表达式 set([1,2, 2,3]) == {1, 2, 3} 的值为_____。
197、表达式 '%c'%65 == str(65) 的值为_____。
198、表达式 '%s'%65 == str(65) 的值为_____。
199、表达式 chr(ord('b')^32) 的值为_____。
200、表达式 'abc' in 'abdcefg' 的值为_____。

【解析】

151.	True	163.	False	176.	0	189.	3
152.	[0,1,2,3,4]	164.	True	177.	[1,2,3]	190.	3
153.	[9]	165.	False	178.	(1,2)	191.	'345'
154.	[[1, 4], [2, 5],	166.	[3,3,1,2]	179.	3	192.	[2]
[3, 6]]		167.	[3,3,2]	180.	3	193.	2
155.	‘1’	168.	[1,4,5]	181.	3	194.	'3'
156.	[1,2,3]	169.	32	182.	[2,3,1]	195.	True
157.	[1,2,[3]]	170.	[(3,6)]	183.	None	196.	True
158.	21	171.	2.0	184.	False	197.	False
159.	[4,5]	172.	6	185.	False	198.	True
160.	32	173.	9	186.	[1,3,2]	199.	‘B’
161.	[5,5,5]	174.	[0,2.3]	187.	[2,3,1]	200.	False
162.	True	175.	2	188.	True		

201、已知 x 为整数变量，那么表达式 int(hex(x), 16) == x 的值为_____。
202、已知 x, y = 3, 5，那么执行 x, y = y, x 之后，x 的值为_____。
203、已知 x = 'abcd'和y='abcde'，那么表达式[i==j for i,j in zip(x,y)]的值为_____。
204、已知 x = list(range(20))，那么表达式 x[-1]的值为_____。
205、已知 x = 3+4j 和 y = 5+6j，那么表达式 x+y 的值为_____。
206、已知 x = [3]，那么执行 x += [5]之后 x 的值为_____。
207、已知 x = [3, 3, 4]，那么表达式 id(x[0])==id(x[1])的值为_____。
208、表达式 int('11', 2)的值为_____。
209、表达式 int('11', 8)的值为_____。
210、表达式 int(bin(54321), 2)的值为_____。
211、表达式 chr(ord('A')+1)的值为_____。
212、表达式 int(str(34)) == 34 的值为_____。
213、表达式 list(str([3, 4])) == [3, 4]的值为_____。
214、表达式{1, 2, 3, 4, 5, 6} ^ {5, 6, 7, 8}的值为_____。
215、表达式 15 // 4 的值为_____。
216、表达式 sorted({'a':3, 'b':9, 'c':78})的值为_____。
217、表达式 sorted({'a':3, 'b':9, 'c':78}.values())的值为_____。
218、已知 x = [3, 2, 4, 1]，那么执行语句 x = x.sort()之后，x 的值为_____。
219、表达式 list(filter(lambda x: x>5, range(10)))的值为_____。
220、已知 x = list(range(20))，那么语句 print(x[100:200])的输出结果为_____。
221、已知 x = list(range(20))，那么执行语句 x[:18] = []后列表 x 的值为_____。
222、已知 x = [1, 2, 3]，那么连续执行 y = x[:]和 y.append(4)这两条语句之后，x 的值为_____。
223、已知 x = [1, 2, 3]，那么连续执行y = x和 y.append(4)这两条语句之后，x 的值为_____。

224、已知 x = [1, 2, 3]，那么连续执行y = [1, 2, 3]和y.append(4)这两条语句之后，x的值为_____。
225、已知 x = [[]] * 3 ，那么执行语句x[0].append(1) 之后， x 的值为_____。
226、已知 x = [[] for i in range(3)]，那么执行语句 x[0].append(1) 之后， x 的值为_____。
227、已知 x = ([1], [2])，那么执行语句 x[0].append(3)后 x 的值为_____。
228、已知 x = {1:1, 2:2}，那么执行语句 x.update({2:3, 3:3})之后，表达式 sorted(x.items())的值为_____。
229、已知 x = {1:1, 2:2}，那么执行语句 x[3] = 3 之后，表达式 sorted(x.items())的值为_____。
230、已知 x = [1, 2, 3]，那么表达式 not (set(x*100)-set(x))的值为_____。
231、已知 x = [1, 2, 3]，那么表达式 not (set(x*100)&set(x))的值为_____。
232、表达式{'x': 1, **{'y': 2}}的值为_____。
233、表达式{*range(4), 4, *(5, 6, 7)}的值为_____。
234、已知x = [1,2,3,4,5] ，那么执行语句x[::2] =range(3)之后，x的值为_____。
235、已知x = [1,2,3,4,5]，那么执行语句 x[::2] = map(lambda y:y!=5,range(3))之后，x的值为___。
236、已知 x = [1,2,3,4,5]，那么执行语句 x[1::2] = sorted(x[1::2], reverse=True)之后，x的值为_。
237、表达式 True*3 的值为_____。
238、表达式 False+1 的值为_____。
239、表达式 'ab' in 'acbed' 的值为_____。
240、假设 n 为整数，那么表达式 n&1 == n%2 的值为_____。
241、关键字_____用于测试一个对象是否是一个可迭代对象的元素。
242、表达式 3<5>2 的值为_____。
243、已知 x = {'a':'b', 'c':'d'}，那么表达式 'a' in x 的值为_____。
244、已知 x = {'a':'b', 'c':'d'}，那么表达式 'b' in x 的值为_____。
245、已知 x = {'a':'b', 'c':'d'}，那么表达式 'b' in x.values() 的值为_____。
246、表达式 1<2<3 的值为_____。
247、表达式 3 or 5 的值为_____。
248、表达式 0 or 5 的值为_____。
249、表达式 3 and 5 的值为_____。
250、表达式 3 and not 5 的值为_____。

【解析】

201. True
202. 5
203. [True, True, True, True]
204. 19
205. 8+10j
206. [3,5]
207. True
208. 3
209. 9
210. 54321
211. 'B'
212. True
213. False
214. {1,2,3,4,7,8}
215. 3
216. ['a', 'b', 'c']
217. [3,9,78]
218. None
219. [6,7,8,9]
220. []
221. [18,19]
222. [1,2,3]
223. [1,2,3,4]
224. [1,2,3]
225. [[1], [1], [1]]
226. [[1], [], []]
227. ([1,3], [2])
228. [(1, 1), (2, 3), (3, 3)]
229. [(1, 1), (2, 2), (3, 3)]
230. True
231. False
232. {'x': 1, 'y': 2}
233. {0, 1, 2, 3, 4, 5, 6, 7}
234. [0, 2, 1, 4, 2]
235. [True, 2, True, 4, True]
236. [1, 4, 3, 2, 5]
237. 3
238. 1
239. False
240. True
241. in
242. True
243. True
244. False
245. True
246. True
247. 3
248. 5
249. 5
250. False

251、Python 中用于表示逻辑与、逻辑或、逻辑非运算的关键字分别是____、_____、_____。

252、Python 3.x 语句for i in range(3):print(i, end=',')的输出结果为_____。
253、Python 3.x 语句 print(1, 2, 3, sep=',')的输出结果为_____。
254、对于带有 else 子句的 for 循环和 while 循环，当循环因循环条件不成立而自然结束时___(会？不会？)执行 else 中的代码。
255、在循环语句中，_____语句的作用是提前结束本层循环。
256、在循环语句中，_____语句的作用是提前进入下一次循环。
257、表达式 5 if 5>6 else (6 if 3>2 else 5) 的值为_。
258、Python 关键字 elif 表示____和_两个单词的缩写。
259、表达式 3 in {1, 2, 3} 的值为_。
260、表达式 'ac' in 'abce' 的值为__。
261、表达式 not 3 的值为____。
262、表达式 'abc' in ('abcdefg') 的值为___。
263、表达式 'abc' in ['abcdefg'] 的值为___。
264、表达式 '\x41' == 'A' 的值为_____。
265、Python 语句''.join(list('hello world!'))执行的结果是____。
266、转义字符 r'\n'的含义是__。
267、已知列表对象 x = ['11', '2', '3']，则表达式 max(x) 的值为____。
268、表达式 min(['11', '2', '3']) 的值为_。
269、已知列表对象x = ['11', '2', '3']，则表达式max(x, key=len) 的值为_。
270、已知 path = r'c:\test.html' ， 那么表达式 path[:-4]+'htm' 的值为__。
271、表达式 list(str([1,2,3])) == [1,2,3] 的值为_____。
272、表达式 str([1, 2, 3]) 的值为__。
273、表达式 str((1, 2, 3)) 的值为__。
274、表达式 sum(range(1, 10, 2)) 的值为__。
275、表达式 sum(range(1, 10)) 的值为_____。
276、表达式 '%c'%65 的值为_。
277、表达式 '%s'%65 的值为_____。
278、表达式 '%d,%c' % (65, 65) 的值为___。
279、表达式 'Thefirst:{1}, the second is {0}'.format(65,97) 的值为_____。
280、表达式 '{0:#d},{0:#x},{0:#o}'.format(65) 的值为_____。
281、表达式 isinstance('abcdefg', str) 的值为___。
282、表达式 isinstance('abcdefg', object) 的值为___。
283、表达式 isinstance(3, object) 的值为__。
284、表达式 'abcabcabc'.rindex('abc') 的值为_。
285、表达式 ':'.join('abcdefg'.split('cd')) 的值为_____。
286、表达式 'Hello world. I like Python.'.rfind('python') 的值为__。
287、表达式 'abcabcabc'.count('abc') 的值为___。
288、表达式 'apple.peach,banana,pear'.find('p') 的值为__。
289、表达式 'apple.peach,banana,pear'.find('ppp') 的值为___。
290、表达式 'abcdefg'.split('d') 的值为___。
291、表达式 ':'.join('1,2,3,4,5'.split(',')) 的值为_____。
292、表达式 ','.join('a b ccc\n\n\nddd'.split()) 的值为_____。
293、表达式 'Hello world'.upper() 的值为__。

294、表达式 'Hello world'.lower() 的值为__。
295、表达式 'Hello world'.lower().upper() 的值为___。
296、表达式 'Hello world'.swapcase().swapcase() 的值为___。
297、表达式 r'c:\windows\notepad.exe'.endswith('.exe') 的值为__。
298、表达式 r'c:\windows\notepad.exe'.endswith(('.jpg', '.exe')) 的值为___。
299、表达式 'C:\\Windows\\notepad.exe'.startswith('C:') 的值为__。
300、表达式 len('Hello world!'.ljust(20)) 的值为____。

【解析】

251. and、or、not
252. 0,1,2,
253. 1,2,3
254. 会
255. break
256. continue
257. 6
258. else、if
259. True
260. False
261. False
262. True
263. False
264. True
265. 'hello world!'
266. 回车换行
267. '3'
268. '11'
269. '11'
270. 'c:\\test.htm'
271. False
272. '[1, 2, 3]'
273. '(1, 2, 3)'
274. 25
275. 45
276. 'A'
277. '65'
278. '65,A'
279. 'The first: 97, the second is 65'
280. '65,0x41,0o101'
281. True
282. True
283. True
284. 6
285. 'ab:efg'
286. -1
287. 3
288. 1
289. -1
290. ['abc', 'efg']
291. '1:2:3:4:5'
292. 'a,b,ccc,ddd'
293. 'HELLO WORLD'
294. 'hello world'
295. 'HELLO WORLD'
296. 'Hello world'
297. True
298. True
299. True
300. 20

301、表达式 len('abcdefg'.ljust(3))的值为_____。
302、表达式 'a' + 'b' 的值为_____。
303、已知 x = '123' 和 y = '456'，那么表达式 x + y 的值为_____。
304、表达式 'a'.join('abc'.partition('a')) 的值为_____。
305、表达式 re.split('\.+', 'alpha.beta...gamma..delta') 的值为_____。
306、已知 x = 'a234b123c'， 并且 re 模块已导入， 则表达式 re.split('\d+', x) 的值为_____。
307、表达式 ''.join('asdssfff'.split('sd')) 的值为_____。
308、表达式 ''.join(re.split('[sd]','asdssfff')) 的值为_____。
309、假设 re 模块已导入， 那么表达式 re.findall('(\d)\\1+', '33abcd112')的值为_____。
310、语句 print(re.match('abc', 'defg')) 输出结果为_____。
311、表达式 'Hello world!'[-4] 的值为_____。
312、表达式 'Hello world!'[-4:] 的值为_____。
313、表达式 'test.py'.endswith(('.py', '.pyw')) 的值为_____。
314、表达式 len('abc'.ljust(20)) 的值为_____。
315、代码 print(re.match('^[a-zA-Z]+$','abcDEFG000'))的输出结果为_____。
316、当在字符串前加上小写字母_____或大写字母表示原始字符串，不对其中的任何字符进行转义。
317、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。
318、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_____。

319、表达式 len('中国'.encode('utf-8'))的值为_____。
320、表达式 len('中国'.encode('gbk'))的值为_____。
321、表达式 chr(ord('A')+2) 的值为_____。
322、表达式 'abcab'.replace('a','yy') 的值为_____。
323、已知 table = ''.maketrans('abcw', 'xyzc')，那么表达式 'Hellow world'.translate(table)的值为_____。
324、表达式 'hello world, hellow every one'.replace('hello', 'hi') 的值为_____。
325、已知字符串 x = 'hello world'，那么执行语句 x.replace('hello', 'hi')之后，x的值为_____。
326、正则表达式元字符_____用来表示该符号前面的字符或子模式 1 次或多次出现。
327、已知 x = 'a b c d'，那么表达式 ','.join(x.split()) 的值为_____。
328、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。
329、表达式 'abcab'.strip('ab') 的值为_____。
330、表达式 [str(i) for i in range(3)] 的值为_____。
331、表达式 'abc.txt'.endswith(('.txt', '.doc', '.jpg')) 的值为_____。
332、代码 print(1,2,3,sep=':') 的执行结果为_____。
333、代码 for i in range(3):print(i, end=',') 的执行结果为_____。
334、表达式 eval('''import ('math').sqrt(9)''') 的值为_____。
335、表达式 eval('''import ('math').sqrt(3**2+4**2)''') 的值为_____。
336、表达式 eval('3+5') 的值为_____。
337、表达式 eval('[1, 2, 3]') 的值为_____。
338、假设 math 标准库已导入，那么表达式 eval('math.sqrt(4)') 的值为_____。
339、已知 x 为非空列表，那么表达式 random.choice(x) in x 的值为_____。
340、表达式 'abc10'.isalnum() 的值为_____。
341、表达式 'abc10'.isalpha() 的值为_____。
342、表达式 'abc10'.isdigit() 的值为_____。
343、表达式 'C:\\windows\\notepad.exe'.endswith('.exe') 的值为_____。
344、表达式 '%s'%[1,2,3] 的值为_____。
345、表达式 'aaasdf'.lstrip('as') 的值为_____。
346、表达式 'aaasdf'.lstrip('af') 的值为_____。
347、表达式 'aaasdf'.strip('af') 的值为_____。
348、表达式 'aaasdf'.rstrip('af') 的值为_____。
349、表达式 len('SDIBT') 的值为_____。
350、表达式 'Hello world!'.count('l') 的值为_____。

【解析】

301. 7
302. 'ab'
303. '123456
304. 'aaabc'
305. ['alpha', 'beta', 'gamma', 'delta']
306. ['a', 'b', 'c']
307. 'assfff'
308. 'afff'
309. ['3', '1']
310. None
311. 'r'
312. 'rld!'
313. True
314. 20
315. None
316. r、R
317. ?
318.'a1bbbb1c1d1e'
319. 6
320. 4
321. 'C'
322. 'yybcyyb'
323. 'Helloc corld'
324. 'hi world, hiw every one'
325. 'hello world'
326. +
327. 'a,b,c,d'
328. *
329. 'c'
330. ['0', '1', '2']
331. True
332. 1:2:3
333. 0,1,2
334. 3.0

335. 5.0
336. 8
337. [1, 2, 3]
338. 2.0
339. True
340. True
341. False
342. False
343. True
344. '[1, 2, 3]'
345. 'df'
346. 'sdf
347. 'sd'
348. 'aaasd'
349. 5
350. 3

351、已知x='abcdefg'，则表达式 x[3:]+x[:3]的值为_____。
352、字符串编码格式 UTF8 使用____个字节表示一个汉字。
353、字符串编码格式 GBK 使用_个字节表示一个汉字。
354、已知字符串编码格式 utf8使用3个字节表示一个汉字、1个字节表示英语字母，那么表达式 len('abc 你好')的值为__。
355、已知字符串编码格式 utf8 使用 3 个字节表示一个汉字、1 个字节表示英语字母，那么表达式 len('abc 你好'.encode())的值为_____。
356、已知字符串编码格式 gbk 使用 2 个字节表示一个汉字、1 个字节表示英语字母，那么表达式 len('abc 你好'.encode('gbk'))的值为_____。
357、已知 ord('A') 的值为 65 并且 hex(65) 的值为'0x41' ， 那么表达式'\x41b' 的值为____。
358、已知 formatter = 'good {0}'.format，那么表达式 list(map(formatter, ['morning']))的值为___。
359、已知 x = 'hello world.'，那么表达式 x.find('x')和 x.rfind('x')的值都为_____。
360、表达式':'.join('hello world.'.split())的值为_____。
361、表达式':'.join('a b c d'.split(maxsplit=2))的值为_____。
362、已知 x = 'hello world'，那么表达式 x.replace('l', 'g')的值为_____。
363、假设已成功导入 Python 标准库 string ， 那么表达式 len(string.digits) 的值为_____。
364、表达式'aaaassddf'.strip('af')的值为____。
365、表达式 len('aaaassddf'.strip('afds'))的值为_____。
366、表达式 len('hello world'[100:])的值为____。
367、表达式 chr(ord('a')^32^32)的值为_____。
368、表达式 chr(ord('a')^32)的值为_____。
369、已知 x = 'aa b ccc dddd'，那么表达式''.join([v for i,v in enumerate(x[:-1]) if v==x[i+1]])的值为_____。
370、假设正则表达式模块 re 已正确导入，那么表达式''.join(re.findall('\d+', 'abcd1234'))的值为_____。
371、假设正则表达式模块 re 已正确导入，那么表达式 re.findall('\d+?', 'abcd1234')的值为_____。
372、假设正则表达式模块 re 已正确导入，那么表达式 re.sub('(.\s)\\1+', '\\1','a a a a a bb')的值为_____。
373、表达式 eval('*'.join(map(str, range(1, 6)))) 的值为_____。
374、正则表达式模块 re 的____方法用来编译正则表达式对象。
375、正则表达式模块 re 的_____方法用来在字符串开始处进行指定模式的匹配。
376、正则表达式模块 re 的____方法用来在整个字符串中进行指定模式的匹配。
377、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_____。
378、表达式'Beautiful is better than ugly.'.startswith('Be', 5) 的值为___。
379、已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 ''.join(x.values())的值为_____。
380、Python 中定义函数的关键字是_____。
381、在函数内部可以通过关键字_____来定义全局变量。

382、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_。
383、表达式 sum(range(10)) 的值为_____。
384、表达式 sum(range(1, 10, 2)) 的值为_____。
385、表达式 list(filter(None, [0,1,2,3,0,0])) 的值为_。
386、表达式 list(filter(lambda x:x>2, [0,1,2,3,0,0])) 的值为__。
387、达式 list(range(50, 60, 3)) 的值为____。
388、表达式list(filter(lambda x: x%2==0, range(10))) 的值为_____。
389、表达式 list(filter(lambda x: len(x)>3, ['a', 'b', 'abcd'])) 的值为__。
390、已知 g = lambda x, y=3, z=5: x*y*z，则语句 print(g(1)) 的输出结果为___。
391、表达式 list(map(lambda x: len(x), ['a', 'bb', 'ccc'])) 的值为_____。
392、已知 f = lambda x: x+5，那么表达式 f(3) 的值为____。
393、表达式 sorted(['abc', 'acd', 'ade'], key=lambda x:(x[0],x[2])) 的值为____。
394、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_____。
395、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__。
396、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__。
397、已知f = lambdan: len(bin(n)[bin(n).rfind('1')+1:]) ，那么表达式f(6)的值为_____。
398、已知f =lambda n: len(bin(n)[bin(n).rfind('1')+1:]) ，那么表达式f(7)的值为_____。
399、已知 g = lambda x, y=3, z=5: x+y+z，那么表达式 g(2) 的值为_____。
400、已知函数定义 def func(*p):return sum(p)，那么表达式 func(1,2,3) 的值为____。

【解析】

351. 'defgabc'
352. 3
353. 2
354. 5
355. 9
356. 7
357. 'Ab'
358. ['good morning']
359. -1
360. 'hello:world.'
361. 'a:b:c d'
362. 'heggo worgd'
363. 10
364. 'ssdd'
365. 0
366. 0
367. 'a'
368. 'A'
369. 'accddd'
370. '1234'
371. ['1', '2', '3', '4']
372. 'a bb'
373. 120
374. compile()
375. match()
376. search()
377. 'is is'
378. False
379. '345'
380. def globa
381. None
382. 45
383. 25
384. [1, 2, 3]
385. [3]
386. [50, 53, 56, 59]
387. [0, 2, 4, 6, 8]
388. ['abcd']
389. 15
390. [1, 2, 3]
391. 8
392. ['abc', 'acd', 'ade']
393. 8
394. 15
395. -2
396. 1
397. 0
398. 10
399. 6
400.

401、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_____。
402、已知函数定义 def func(**p):return sum(p.values())，那么表达式 func(x=1, y=2, z=3)的值为____。
403、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_____。
404、已知 f = lambda x: 5，那么表达式 f(3)的值为_____。

405、Python 使用_____关键字来定义类。
406、表达式 isinstance('abc', str) 的值为_____。
407、表达式 isinstance('abc', int) 的值为_____。
408、表达式 isinstance(4j, (int, float, complex)) 的值为_____。
409、表达式 isinstance('4', (int, float, complex)) 的值为_____。
410、表达式 type(3) in (int, float, complex) 的值为_____。
411、表达式 type(3.0) in (int, float, complex) 的值为___。
412、表达式 type(3+4j) in (int, float, complex) 的值为_____。
413、表达式 type('3') in (int, float, complex) 的值为_____。
414、表达式 type(3) == int 的值为____。
415、在 Python 定义类时， 与运算符“**” 对应的特殊方法名为_____。
416、在 Python 中定义类时，与运算符“//”对应的特殊方法名为_____。
417、表达式 type({}) == dict 的值为_____。
418、表达式 type({}) == set 的值为_____。
419、在Python 中，不论类的名字是什么，构造方法的名字都是_____。
420、如果在设计一个类时实现了 contains() 方法，那么该类的对象会自动支持_____运算符。
421、对文件进行写入操作之后，_____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。
422、Python 内置函数_____用来打开或创建文件并返回文件对象。
423、使用上下文管理关键字_____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。
424、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_____。
425、Python 标准库os.path中用来判断指定文件是否存在的方法是_____。
426、Python 标准库 os.path 中用来判断指定路径是否为文件的方法是____。
427、Python 标 准 库 os.path 中用来判断指定路径是否为文件夹的方法是_____。
428、Python 标准库 os.path 中用来分割指定路径中的文件扩展名的方法是_____。
429、Python 扩展库_____支持 Excel 2007 或更高版本文件的读写操作。
430、Python 标准库_____中提供了计算 MD5 摘要的方法 md5( )。
431、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'，_____) as dst:dst.write(src.read())。
432、Python 内建异常类的基类是_____。
433、Python 标准库_____对Socket 进行了二次封装，支持Socket 接口的访问，大幅度简化了网络程序的开发。
434、Python 扩展库_____中封装了Windows底层几乎所有 API 函数。
435、线程对象的_____方法用来阻塞当前线程，指定线程运行结束或超时后继续运行当前线程。
436、Python 用来访问和操作内置数据库 SQLite 的标准库是_____。
437、用于删除数据库表 test 中所有 name 字段值为’10001’ 的记录的 SQL 语句为_____。
438、Python 扩展库_____完美封装了图形库 OpenGL 的功能。
439、Python 扩展库_____和_____提供了图像处理功能。

【解析】

401. 10
402. 6
403. 'xyz'
404. 5
405. class
406. True
407. False
408. True
409. False
410. True
411. True
412. True
413. False
414. True
415. pow ()
416. floordiv ()
417. True
418. False
419. init
420. in
421. flush()
422. open()
423. with
424. listdir()
425. exists()
426. isfile()
427. isdir()
428. splitext()
429. openpyxl
430. hashlib
431. 'w'
432.BaseException
433. Socket
434. pywin32
435. join()
436. sqlite3
437. delete from test where name='10001'
438. pyopengl
439. PIL、pillow

三、判断题

1、Python 是一种跨平台、开源、免费的高级动态编程语言。（ ）

2、Python 3.x 完全兼容 Python 2.x。（ ）

3、Python 3.x 和 Python 2.x 唯一的区别就是：print 在 Python 2.x 中是输出语句，而在 Python 3.x中是输出函数。（ ）

4、在 Windows 平台上编写的 Python 程序无法在 Unix 平台运行。（ ）

5、不可以在同一台计算机上安装多个 Python 版本。（ ）

6、已知 x = 3，那么赋值语句 x = 'abcedfg' 是无法正常执行的。（ ）

7、Python 变量使用前必须先声明，并且一旦声明就不能再当前作用域内改变其类型。（ ）

8、Python 采用的是基于值得自动内存管理方式。（ ）

9、在任何时刻相同的值在内存中都只保留一份（ ）

10、Python不允许使用关键字作为变量名，允许使用内置函数名作为变量名，但这会改变函数名的含义。（ ）

11、在Python 中可以使用 if 作为变量名。（ ）

12、在Python 3.x 中可以使用中文作为变量名。（ ）

13、Python 变量名必须以字母或下划线开头，并且区分字母大小写。（ ）

14、加法运算符可以用来连接字符串并生成新字符串。（ ）

15、9999**9999 这样的命令在 Python 中无法运行。（ ）

16、3+4j 不是合法的 Python 表达式。（ ）

17、0o12f 是合法的八进制数字。（ ）

18、不管输入什么，Python 3.x 中 input()函数的返回值总是字符串。（ ）

19、pip 命令也支持扩展名为.whl 的文件直接安装 Python 扩展库。（ ）

20、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。（ ）

21、在Python 中 0xad 是合法的十六进制数字表示形式。（ ）

22、3+4j 是合法 Python 数字类型。（ ）

23、在Python 中 0oa1 是合法的八进制数字表示形式。（ ）

24、Python 使用缩进来体现代码之间的逻辑关系。（ ）

25、Python 代码的注释只有一种方式，那就是使用#符号。（ ）

26、放在一对三引号之间的任何内容将被认为是注释。（ ）

27、尽管可以使用 import 语句一次导入任意多个标准库或扩展库，但是仍建议每次只导入一个标准库或扩展库。（ ）

28、为了让代码更加紧凑，编写 Python 程序时应尽量避免加入空格和空行。（ ）

29、在 Python 3.5 中运算符+不仅可以实现数值的相加、字符串连接，还可以实现列表、元组的合并和集合的并集运算。（ ）

30、在 Python 中可以使用 for 作为变量名。（ ）

31、在 Python 中可以使用 id 作为变量名，尽管不建议这样做。（ ）

32、Python 关键字不可以作为变量名。（ ）

33、一个数字 5 也是合法的 Python 表达式。（对）

34、执行语句 from math import sin 之后，可以直接使用 sin() 函数，例如 sin(3)。（ ）

35、不同版本的 Python 不能安装到同一台计算机上。（ ）

36、一般来说，Python 扩展库没有通用于所有版本 Python 的，安装时应选择与已安装Python的版本对应的扩展库。（ ）

37、Python 变量名区分大小写，所以 student 和 Student 不是同一个变量。（ ）

38、在 Python 3.x 中 reduce()是内置函数。（ ）

39、如果只需要 math 模块中的 sin()函数，建议使用 from math import sin 来导入，而不要使用 import math 导入整个模块。（ ）

40、在 Python 3.x 中，使用内置函数 input()接收用户输入时，不论用户输入的什么格式，一律按字符串进行返回。（ ）

41、安装 Python 扩展库时只能使用 pip 工具在线安装，如果安装不成功就没有别的办法了。（ ）

42、ython 支持使用字典的“键”作为下标来访问字典中的值。（ ）

43、列表可以作为字典的“键”。（ ）

44、元组可以作为字典的“键”。（ ）

45、字典的“键”必须是不可变的。（ ）

46、已知 x 为非空列表，那么表达式 sorted(x, reverse=True) == list(reversed(x)) 的值一定是 True。（ ）

47、已知 x 为非空列表，那么 x.sort(reverse=True)和 x.reverse()的作用是等价的。（ ）

48、生成器推导式比列表推导式具有更高的效率，推荐使用。（ ）

49、Python 集合中的元素不允许重复。（ ）

50、Python 集合可以包含相同的元素。（ ）

【解析】

1、对	11、错	21、对	31、对	41、错
2、错	12、对	22、对	32、对	42、对
3、错	13、对	23、错	33、对	43、错
4、错	14、对	24、对	34、对	44、对
5、错	15、错	25、错	35、错	45、对
6、错	16、错	26、错	36、对	46、错
7、错	17、错	27、对	37、对	47、错
8、对	18、对	28、错	38、错	48、对
9、错	19、对	29、错	39、对	49、对
10、对	20、错	30、错	40、对	50、错

51、Python 字典中的“键”不允许重复。（ ）

52、Python 字典中的“值”不允许重复。（ ）

53、Python 集合中的元素可以是元组。（ ）

54、Python 集合中的元素可以是列表。（ ）

55、Python 字典中的“键”可以是列表。（ ）

56、Python 字典中的“键”可以是元组。（ ）

57、Python 列表中所有元素必须为相同类型的数据。（ ）

58、Python 列表、元组、字符串都属于有序序列。（ ）

59、已知A和B是两个集合，并且表达式 A<B 的值为 False，那么表达式 A>B的值一定为 True。（ ）

60、列表对象的 append()方法属于原地操作，用于在列表尾部追加一个元素。（ ）

61、对于列表而言，在尾部追加元素比在中间位置插入元素速度更快一些，尤其是对于包含大量元素的列表。（ ）

62、假设有非空列表x，那么x.append(3)、x = x+[3]与 x.insert(0,3)在执行时间上基本没有太大区别。（ ）

63、使用Python 列表的方法insert()为列表插入元素时会改变列表中插入位置之后元素的索引。（ ）

64、假设 x 为列表对象，那么 x.pop()和 x.pop(-1)的作用是一样的。（ ）

65、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。（ ）

66、已知列表 x = [1, 2, 3]，那么执行语句 x = 3 之后，变量 x 的地址不变。（ ）

67、使用列表对象的 remove()方法可以删除列表中首次出现的指定元素，如果列中不存在要删除的指定元素则抛出异常。（ ）

68、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。（ ）

69、Python 字典和集合属于无序序列。（ ）

70、无法删除集合中指定位置的元素，只能删除特定值的元素。（ ）

71、元组的访问速度比列表要快一些，如果定义了一系列常量值，并且主要用途仅仅是对其进行遍历二不需要进行任何修改，建议使用元组而不使用列表。（ ）

72、当以指定“键”为下标给字典对象赋值时，若该“键”存在则表示修改该“键”对应的“值”，若不存在则表示为字典对象添加一个新的“键-值对”。（ ）

73、假设 x 是含有 5 个元素的列表，那么切片操作 x[10:]是无法执行的，会抛出异常。（ ）

74、只能对列表进行切片操作，不能对元组和字符串进行切片操作。（ ）

75、只能通过切片访问列表中的元素，不能使用切片修改列表中的元素。（ ）

76、只能通过切片访问元组中的元素，不能使用切片修改元组中的元素。（ ）

77、字符串属于 Python 有序序列，和列表、元组一样都支持双向索引。（ ）

78、Python 字典和集合支持双向索引。（ ）

79、Python 集合不支持使用下标访问其中的元素。（ ）

80、相同内容的字符串使用不同的编码格式进行编码得到的结果并不完全相同。（ ）

81、删除列表中重复元素最简单的方法是将其转换为集合后再重新转换为列表。（ ）

82、已知列表 x 中包含超过 5 个以上的元素，那么语句 x = x[:5]+x[5:]的作用是将列表x中的元素循环左移5位。（ ）

83、对于生成器对象 x = (3 for i in range(5))，连续两次执行 list(x)的结果是一样的。（ ）

84、对于大量列表的连接，extend()方法比运算符+具有更高的效率。（ ）

85、表达式 {1, 3, 2} > {1, 2, 3} 的值为 True。（ ）

86、列表对象的 extend()方法属于原地操作，调用前后列表对象的地址不变。（ ）

87、对于数字 n，如果表达式 0 not in [n%d for d in range(2, n)] 的值为 True 则说明 n 是素数。（ ）

88、表达式 ‘a’+1 的值为’b’。（ ）

89、创建只包含一个元素的元组时，必须在元素后面加一个逗号，例如(3,)。（ ）

90、表达式 list('[1, 2, 3]') 的值是[1, 2, 3]。（ ）

91、同一个列表对象中的元素类型可以各不相同。（ ）

92、同一个列表对象中所有元素必须为相同类型。（ ）

93、已知 x 为非空列表，那么执行语句 x[0] = 3 之后，列表对象 x 的内存地址不变。（ ）

94、列表可以作为集合的元素。（ ）

95、集合可以作为列表的元素。（ ）

96、元组可以作为集合的元素。（ ）

97、集合可以作为元组的元素。（ ）

98、字典可以作为集合的元素。（ ）

99、集合可以作为字典的键。（ ）

100、集合可以作为字典的值。（ ）

【解析】

51、对	61、对	71、对	81、对	91、对
52、错	62、错	72、对	82、错	92、错
53、对	63、对	73、错	83、错	93、对
54、错	64、对	74、错	84、对	94、错
55、错	65、对	75、错	85、错	95、对
56、对	66、错	76、对	86、对	96、对
57、错	67、对	77、对	87、对	97、对
58、对	68、对	78、错	88、错	98、错
59、错	69、对	79、对	89、对	99、错
60、对	70、对	80、对	90、错	100、对

101、可以使用 del 删除集合中的部分元素。（ ）

102、列表对象的pop()方法默认删除并返回最后一个元素，如果列表已空则抛出异常。（ ）

103、表达式 {1, 2} * 2 的值为 {1, 2, 1, 2}。（ ）

104、Python 字典支持双向索引。（ ）

105、Python 集合支持双向索引。（ ）

106、Python 元组支持双向索引。（ ）

107、假设 re 模块已成功导入，并且有 pattern = re.compile('^'+'\.'.join([r'\d{1,3}' for i in range(4)])+'$')，那么表达式 pattern.match('192.168.1.103') 的值为 None。（ ）

108、假设 random 模块已导入，那么表达式 random.sample(range(10), 20)的作用是生成20个不重复的整数。（ ）

109、假设 random 模块已导入，那么表达式 random.sample(range(10), 7) 的作用是生成 7 个不重复的整数。（ ）

110、使用 random 模块的函数 randint(1, 100)获取随机数时，有可能会得到 100。（ ）

111、已知 x = (1, 2, 3, 4)，那么执行 x[0] = 5 之后，x 的值为(5, 2, 3, 4)。（ ）

112、已知 x = 3，那么执行 x += 6 语句前后 x 的内存地址是不变的。（ ）

113、成员测试运算符 in 作用于集合时比作用于列表快得多。（ ）

114、内置函数 len()返回指定序列的元素个数，适用于列表、元组、字符串、字典、集合以及 range、zip 等迭代对象。（ ）

115、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。（ ）

116、已知 x 和 y 是两个等长的整数列表，那么表达式[i+j for i,j in zip(x,y)]的作用时计算这两个列表所表示的向量的和。（ ）

117、表达式 int('1'*64, 2)与 sum(2**i for i in range(64))的计算结果是一样的，但是前者更快一些。（ ）

118、已知 x = list(range(20))，那么语句 del x[::2]可以正常执行。（ ）

119、已知 x = list(range(20))，那么语句 x[::2] = []可以正常执行。（ ）

120、已知 x = list(range(20))，那么语句 print(x[100:200])无法正常执行。（ ）

121、已知 x 是个列表对象，那么执行语句y = x之后，对 y 所做的任何操作都会同样作用到 x 上。（ ）

122、已知 x 是个列表对象，那么执行语句 y = x[:]之后，对y所做的任何操作都会同样作用到x上。（ ）

123、在Python中，变量不直接存储值，而是存储值的引用，也就是值在内存中的地址。（ ）

124、表达式(i**2 for i in range(100))的结果是个元组。（ ）

125、在 Python 中元组的值是不可变的，因此，已知 x = ([1], [2])，那么语句 x[0].append(3)是无法正常执行的。（ ）

126、Python 内置的字典 dict 中元素是按添加的顺序依次进行存储的。（ ）

127、Python 内置的集合 set 中元素顺序是按元素的哈希值进行存储的，并不是按先后顺序。（ ）

128、已知 x = {1:1, 2:2}，那么语句 x[3] =3 无法正常执行。（ ）

129、Python 内置字典是无序的，如果需要一个可以记住元素插入顺序的字典，可以使用 collections. OrderedDict。（ ）

130、已知列表 x = [1, 2, 3, 4]，那么表达式 x.find(5)的值应为-1。（ ）

131、列表对象的排序方法 sort()只能按元素从小到大排列，不支持别的排序方式。（ ）

132、已知 x 是一个列表，那么 x = x[3:] + x[:3]可以实现把列表 x 中的所有元素循环左移3位。（ ）

133、如果仅仅是用于控制循环次数，那么使用 for i in range(20)和 for i in range(20, 40)的作用是等价的。（ ）

134、在循环中 continue 语句的作用是跳出当前循环。（ ）

135、在编写多层循环时，为了提高运行效率，应尽量减少内循环中不必要的计算。（ ）

136、带有 else 子句的循环如果因为执行了 break 语句而退出的话，则会执行 else 子句中的代码。（ ）

137、对于带有 else 子句的循环语句，如果是因为循环条件表达式不成立而自然结束循环，则执行else子句中的代码。（ ）

138、在条件表达式中不允许使用赋值运算符“=”，会提示语法错误。（ ）

139、在 UTF-8 编码中一个汉字需要占用 3 个字节。（ ）

140、在 GBK 和 CP936 编码中一个汉字需要 2 个字节。（ ）

141、在 Python 中，任意长的字符串都遵守驻留机制。（ ）

142、Python 运算符%不仅可以用来求余数，还可以用来格式化字符串。（ ）

143、Python 字符串方法 replace()对字符串进行原地修改。（ ）

144、如果需要连接大量字符串成为一个字符串，那么使用字符串对象的 join()方法比运算符+具有更高的效率。（ ）

145、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。（ ）

146、已知 x 为非空字符串，那么表达式 ''.join(x.split()) == x 的值一定为 True。（ ）

147、已知 x 为非空字符串，那么表达式 ','.join(x.split(',')) == x 的值一定为 True。（ ）

148、当作为条件表达式时，[]与 None 等价。（ ）

149、表达式 [] == None 的值为 True。（ ）

150、当作为条件表达式时，{}与 None 等价。（ ）

【解析】

101、错	111、错	121、对	131、错	141、错
102、对	112、错	122、错	132、对	142、对
103、错	113、对	123、对	133、对	143、错
104、错	114、对	124、错	134、错	144、对
105、错	115、对	125、错	135、对	145、对
106、对	116、对	126、错	136、错	146、错
107、错	117、对	127、对	137、对	147、对
108、错	118、对	128、错	138、对	148、对
109、对	119、错	129、对	139、对	149、错
110、对	120、错	130、错	140、对	150、对

151、表达式 {}==None 的值为 True。（ ）

152、表达式 pow(3,2) == 3**2 的值为 True。（ ）

153、当作为条件表达式时，空值、空字符串、空列表、空元组、空字典、空集合、空迭代对象以及任意形式的数字 0 都等价于 False。（ ）

154、正则表达式对象的 match()方法可以在字符串的指定位置开始进行指定模式的匹配。（ ）

155、使用正则表达式对字符串进行分割时，可以指定多个分隔符，而字符串对象的split()方法无法做到这一点。（ ）

156、正则表达式元字符“^”一般用来表示从字符串开始处进行匹配，用在一对方括号中的时候则表示反向匹配，不匹配方括号中的字符。（ ）

157、正则表达式元字符“\s”用来匹配任意空白字符。（ ）

158、正则表达式 元字符“\d”用来匹配任意数字字符。（ ）

159、已知 x 和 y 是两个字符串，那么表达式 sum((1 for i,j in zip(x,y) if i==j))可以用来计算两个字符串中对应位置字符相等的个数。（ ）

160、Python 3.x 中字符串对象的 encode()方法默认使用 utf8 作为编码方式。（ ）

161、已知 x = 'hellow world.'.encode()，那么表达式 x.decode('gbk')的值为'hellow world.'。（ ）

162、已知 x = 'Python 是一种非常好的编程语言'.encode()，那么表达式 x.decode('gbk')的值为

'Python 是一种非常好的编程语言'。（ ）

163、正则表达式'^http'只能匹配所有以'http'开头的字符串。（ ）

164、正则表达式'^\d{18}|\d{15}$'只能检查给定字符串是否为 18 位或 15 位数字字符，并不能保证一定是合法的身份证号。（ ）

165、正则表达式'[^abc]'可以一个匹配任意除'a'、'b'、'c'之外的字符。（ ）

166、正则表达式'python|perl'或'p(ython|erl)'都可以匹配'python'或'perl'。（ ）

167、字节串 b'hello world'和 b'hello world.'的 MD5 值相差很小。（ ）

168、调用函数时，在实参前面加一个型号*表示序列解包。（ ）

169、在 Python 3.x 中语句 print(*[1,2,3]) 不能正确执行。（ ）

170、函数是代码复用的一种方式。（ ）

171、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。（ ）

172、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。（ ）

173、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。（ ）

174、定义 Python 函数时必须指定函数返回值类型。（ ）

175、定义 Python 函数时，如果函数中没有 return 语句，则默认返回空值 None。（ ）

176、如果在函数中有语句 return 3，那么该函数一定会返回整数 3。（ ）

177、函数中必须包含 return 语句。（ ）

178、函数中的 return 语句一定能够得到执行。（ ）

179、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。（ ）

180、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。（ ）

181、函数内部定义的局部变量当函数调用结束后被自动删除。（ ）

182、在函数内部，既可以使用 global 来声明使用外部全局变量，也可以使用 global 直接定义全局变量。（ ）

183、在函数内部没有办法定义全局变量。（ ）

184、在函数内部直接修改形参的值并不影响外部实参的值。（ ）

185、在函数内部没有任何方法可以影响实参的值。（ ）

186、调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值。（ ）

187、在同一个作用域内，局部变量会隐藏同名的全局变量。（ ）

188、形参可以看做是函数内部的局部变量，函数运行结束之后形参就不可访问了。（ ）

189、假设已导入 random 标准库， 那么表达式 max([random.randint(1, 10) for i in range(10)])的值一定是10。（ ）

190、Python 标准库 random 的方法 randint(m,n)用来生成一个[m,n]区间上的随机整数。

（ ）

191、在函数内部没有任何声明的情况下直接为某个变量赋值，这个变量一定是函数内部的局部变量。（ ）

192、在 Python 中定义函数时不需要声明函数参数的类型。（ ）

193、在 Python 中定义函数时不需要声明函数的返回值类型。（ ）

194、在函数中没有任何办法可以通过形参来影响实参的值。（ ）

195、已知 x = 3，那么执行语句 x+=6 之后，x 的内存地址不变。（ ）

196、在定义函数时，某个参数名字前面带有一个*符号表示可变长度参数，可以接收任意多个普通实参并存放于一个元组之中。（ ）

197、在定义函数时，某个参数名字前面带有两个*符号表示可变长度参数，可以接收任意多个关键参数并将其存放于一个字典之中。（ ）

198、定义函数时，带有默认值的参数必须出现在参数列表的最右端，任何一个带有默认值的参数右边不允许出现没有默认值的参数。（ ）

199、在调用函数时，可以通过关键参数的形式进行传值，从而避免必须记住函数形参顺序的麻烦。（ ）

200、在调用函数时，必须牢记函数形参顺序才能正确传值。（ ）

【解析】

151、错	161、对	171、对	181、对	191、对
152、对	162、错	172、对	182、对	192、对
153、对	163、对	173、错	183、错	193、对
154、对	164、对	174、错	184、对	194、错
155、对	165、对	175、对	185、错	195、错
156、对	166、对	176、错	186、错	196、对
157、对	167、错	177、错	187、对	197、对
158、对	168、对	178、错	188、对	198、对
159、对	169、错	179、对	189、错	199、对
160、对	170、对	180、对	190、对	200、错

201、调用函数时传递的实参个数必须与函数形参个数相等才行。（ ）

202、在编写函数时，建议首先对形参进行类型检查和数值范围检查之后再编写功能代码，或者使用异常处理结构，尽量避免代码抛出异常而导致程序崩溃。（ ）

203、lambda 表达式中可以使用任意复杂的表达式，但是必须只编写一个表达式。（ ）

204、g = lambda x: 3 不是一个合法的赋值表达式。（ ）

205、包含 yield 语句的函数一般成为生成器函数，可以用来创建生成器对象。（ ）

206、在函数中 yield 语句的作用和 return 完全一样。（ ）

207、语句 pass 仅起到占位符的作用，并不会做任何操作。（ ）

208、无法使用 lambda 表达式定义有名字的函数。（ ）

209、在 Python 中定义类时，如果某个成员名称前有 2 个下划线则表示是私有成员。（ ）

210、在类定义的外部没有任何办法可以访问对象的私有成员。（ ）

211、Python 中一切内容都可以称为对象。（ ）
212、栈和队列的都具有先入后出的特点。（ ）
213、在一个软件的设计与开发中，所有类名、函数名、变量名都应该遵循统一的风格和规范。（ ）
214、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。（ ）
215、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。（ ）
216、Python 中没有严格意义上的私有成员。（ ）
217、在Python 中定义类时，运算符重载是通过重写特殊方法实现的。例如，在类中实现了 mul ()方法即可支持该类对象的**运算符。（ ）
218、在 IDLE 交互模式下，一个下划线“_”表示解释器中最后一次显示的内容或最后一次语句正确执行的输出结果。（ ）
219、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。（ ）
220、运算符“-”可以用于集合的差集运算。（ ）
221、如果定义类时没有编写析构函数，Python 将提供一个默认的析构函数进行必要的资源清理工作。（ ）
222、已知 seq 为长度大于 10 的列表，并且已导入 random 模块，那么[random.choice(seq) for i in range(10)]和 random.sample(seq,10)等价。（ ）
223、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。（ ）
224、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。（ ）
225、在 Python 中定义类时实例方法的第一个参数名称必须是 self。（ ）
226、在 Python 中定义类时实例方法的第一个参数名称不管是什么，都表示对象自身。（ ）
227、定义类时如果实现了 contains ()方法，该类对象即可支持成员测试运算 in。（ ）
228、定义类时如果实现了 len ()方法，该类对象即可支持内置函数 len()。（ ）
229、定义类时实现了 eq ( )方法，该类对象即可支持运算符==。（ ）
230、定义类时实现了 pow ( )方法，该类对象即可支持运算符**。（ ）
231、Python 类的构造函数是 init ( )。（ ）
232、定义类时，在一个方法前面使用@classmethod 进行修饰，则该方法属于类方法。（ ）
233、定义类时，在一个方法前面使用@staticmethod 进行休息，则该方法属于静态方法。（ ）
234、通过对象不能调用类方法和静态方法。（ ）
235、在 Python 中可以为自定义类的对象动态增加新成员。（ ）
236、Python 类不支持多继承。（ ）
237、属性可以像数据成员一样进行访问，但赋值时具有方法的优点，可以对新值进行检查。（ ）
238、只可以动态为对象增加数据成员，而不能为对象动态增加成员方法。（ ）
239、任何包含 call ()方法的类的对象都是可调用的。（ ）
240、在 Python 中函数和类都属于可调用对象。（ ）

241、函数和对象方法是一样的，内部实现和外部调用都没有任何区别。（ ）
242、在设计派生类时，基类的私有成员默认是不会继承的。（ ）
243、如果在设计一个类时实现类 len ()方法，那么该类的对象会自动支持 Python 内置函数 len()。（ ）
244、扩展库 os 中的方法 remove()可以删除带有只读属性的文件。（ ）
245、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。（ ）
246、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。（ ）
247、使用 print()函数无法将信息写入文件。（ ）
248、对文件进行读写操作之后必须显式关闭文件以确保所有内容都得到保存。（ ）
249、Python标准库os中的方法startfile()可以启动任何已关联应用程序的文件，并自动调用关联的程序。（ ）
250、Python 标准库 os 中的方法 startfile()可以用来打开外部程序或文件，系统会自动关联相应的程序来打开或执行指定的文件。（ ）

【解析】

201、错	211、对	221、对	231、对	241、错
202、对	212、错	222、错	232、对	242、对
203、对	213、对	223、对	233、对	243、对
204、错	214、对	224、对	234、错	244、错
205、对	215、错	225、错	235、对	245、错
206、错	216、对	226、对	236、错	246、错
207、对	217、错	227、对	237、对	247、错
208、错	218、对	228、对	238、错	248、对
209、对	219、对	229、对	239、对	249、对
210、错	220、对	230、对	240、对	250、对

251、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。（ ）
252、二进制文件不能使用记事本程序打开。（ ）
253、使用普通文本编辑器软件也可以正常查看二进制文件的内容。（ ）
254、二进制文件也可以使用记事本或其他文本编辑器打开，但是一般来说无法正常查看其中的内容。（ ）
255、Python 标准库 os 中的方法 isfile()可以用来测试给定的路径是否为文件。（ ）
256、Python 标准库 os 中的方法 exists()可以用来测试给定路径的文件是否存在。（ ）
257、Python 标准库 os 中的方法 isdir()可以用来测试给定的路径是否为文件夹。（ ）
258、Python 标准库os 中的方法listdir()返回包含指定路径中所有文件和文件夹名称的列表。（ ）
259、Python 扩展库 xlwt 支持对 Excel 2003 或更低版本的 Excel 文件进行写操作。（ ）
260、Python 扩展库 xlrd 支持对 Excel 2003 或更低版本的 Excel 文件进行读操作。（ ）
261、标准库 os 的 rename()方法可以实现文件移动操作。（ ）
262、标准库 os 的 listdir()方法默认只能列出指定文件夹中当前层级的文件和文件夹列表，而不能列出其子文件夹中的文件。（ ）

263、文件对象的 tell()方法用来返回文件指针的当前位置。（ ）
264、以写模式打开的文件无法进读操作。（ ）
265、假设已成功导入 os 和 sys 标准库，那么表达式 os.path.dirname(sys.executable) 的值为 Python 安装目录。（ ）
266、以读模式打开文件时，文件指针指向文件开始处。（ ）
267、以追加模式打开文件时，文件指针指向文件尾。（ ）
268、二进制文件也可以使用记事本程序打开，只是无法正确阅读和理解其中的内容。（ ）
269、文本文件是可以迭代的，可以使用 for line in fp 类似的语句遍历文件对象 fp 中的每一行。（ ）
270、Python 的主程序文件 python.exe 属于二进制文件。（ ）
271、对字符串信息进行编码以后，必须使用同样的或者兼容的编码格式进行解码才能还原本来的信息。（ ）
272、使用 pickle 进行序列化得到的二进制文件使用 struct 也可以正确地进行反序列化。（ ）
273、已知当前文件夹中有一个文件 readme.txt 具有只读属性，假设标准库 os 已正确导入，那么可以通过语句 os.chmod('readme.txt', 0o777)来删除该文件的只读属性。（ ）
274、Python 标准库 os 的函数 remove()不能删除具有只读属性的文件。（ ）
275、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。（ ）
276、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。（ ）
277、程序中异常处理结构在大多数情况下是没必要的。（ ）
278、带有 else 子句的异常处理结构，如果不发生异常则执行 else 子句中的代码。（ ）
279、异常处理结构也不是万能的，处理异常的代码也有引发异常的可能。（ ）
280、在异常处理结构中，不论是否发生异常，finally 子句中的代码总是会执行的。（ ）
281、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。（ ）
282、在 GUI 设计中，复选框往往用来实现非互斥多选的功能，多个复选框之间的选择互不影响。（ ）
283、 在 GUI 设计中，单选按钮用来实现用户在多个选项中的互斥选择，在同一组内多个选项中只能选择一个，当选择发生变化之后，之前选中的选项自动失效。（ ）
284、Python 代码可以内嵌在 asp 文件中。（ ）
285、无法配置 IIS 来支持 Python 程序的运行。（ ）
286、使用 TCP 协议进行通信时，必须首先建立连接，然后进行数据传输，最后再关闭连接。（ ）
287、TCP是可以提供良好服务质量的传输层协议，所以在任何场合都应该优先考虑使用。（ ）
288、可以使用 py2exe 或 pyinstaller 等扩展库把 Python 源程序打包成为 exe 文件，从而脱离 Python 环境在Windows 平台上运行。（ ）
289、Python 程序只能在安装了 Python 环境的计算机上以源代码形式运行。（ ）
290、继承自 threading.Thread 类的派生类中不能有普通的成员方法。（ ）
291、Python 标准库 threading 中的 Lock、RLock、Condition、Event、Semaphore 对象都可以用来实现线程同步。（ ）

292、在编写应用程序时，应合理控制线程数量，线程并不是越多越好。（ ）

293、在多线程编程时，当某子线程的 daemon 属性为 False 时，主线程结束时会检测该子线程是否结束，如果该子线程尚未运行结束，则主线程会等待它完成后再退出。（ ）

294、在4核CPU平台上使用多线程编程技术可以很轻易地获得 400%的处理速度提升。（ ）

295、多线程编程技术主要目的是为了提高计算机硬件的利用率，没有别的作用了。（ ）

296、Python只能使用内置数据库 SQLite，无法访问 MS SQLServer、ACCESS或Oracle、MySQL等数据库。（ ）

297、使用 OpenGL 画图时，画点是最基本的操作，具体生成的图形由 glBegin()函数指定的 mode 来决定。例如，mode 值为 GL_TRIANGLES 时表示将要绘制三角形。（ ）

298、OpenGL 采用的“状态机”工作方式，一旦设置了某种状态以后，除非显式修改该状态，否则该状态将一直保持。（ ）

【解析】

251、对	261、对	271、对	281、错	291、对
252、错	262、对	272、错	282、对	292、对
253、错	263、对	273、对	283、对	293、对
254、对	264、对	274、对	284、对	294、错
255、对	265、对	275、错	285、错	295、错
256、对	266、对	276、对	286、对	296、错
257、对	267、对	277、错	287、错	297、对
258、对	268、对	278、对	288、对	298、对
259、对	269、对	279、对	289、错	
260、对	270、对	280、对	290、错	

四、编程题

1、编写程序，在 D 盘根目录下创建一个文本文件 test.txt，并向其中写入字符串 hello world。

【解析】

```
fp = open(r"D:\test.txt", "a+")
print("hello world", file=fp)
fp.close()
```

2、写出下面代码的优化版本，提高运行效率。

```
x = list(range(500))
for item in x:
    t = 5**5
    print(item+t)
```

【解析】

```
x =list(range(500))
t =5**5
```

```
for item in x:
    print(item+t)
```

3、编写程序，生成一个包含 20 个随机整数的列表，然后对其中偶数下标的元素进行降序排列，奇数下标的元素不变。（提示：使用切片。）

【解析】

```
import random
x = [random.randint(0,100) for i in range(20)]
print(x)
y = x[::2]
y.sort(reverse=True)
x[::2] = y
print(x)
```

4、写出下面代码的执行结果。

```
def Join(List, sep=None):
    return (sep or ',').join(List)
print(Join(['a', 'b', 'c']))
print(Join(['a', 'b', 'c'],':'))
```

【解析】

a,b,c

a:b:c

5、写出下面代码的运行结果。

```
def Sum(a, b=3, c=5):
    return sum([a, b, c])
print(Sum(a=8, c=2))
print(Sum(8))
print(Sum(8,2))
```

【解析】

13

16

15

6、写出下面代码的运行结果。

```
def Sum(*p):
    return sum(p)
print(Sum(3, 5, 8))
print(Sum(8))
print(Sum(8, 2, 10))
```

【解析】

16

8

20

7、编写函数，判断一个数字是否为素数，是则返回字符串 YES，否则返回字符串 NO。

【解析】

```
import math
def IsPrime(v):
    n = int(math.sqrt(v)+1)
    for i in range(2,n):
        if v%i==0:
            return 'No'
        else:
            return 'Yes'
```

8、编写函数，模拟 Python 内置函数 sorted()。

【解析】

```
def Sorted(v):
    t = v[::]
    r = []
    while t:
        tt = min(t)
        r.append(tt)
        t.remove(tt)
    return r
```

9、编写程序，生成包含 20 个随机数的列表，然后将前 10 个元素升序排列，后 10 个元素降序排列，并输出结果。

【解析】

```
import random
x = [random.randint(0,100) for i in range(20)] print(x)
y = x[0:10]
y.sort()
x[0:10] = y
y = x[10:20]
y.sort(reverse=True)
x[10:20] = y
print(x)
```

10、编写程序，运行后用户输入 4 位整数作为年份，判断其是否为闰年。如果年份能被 400 整除，则为闰年；如果年份能被 4 整除但不能被 100 整除也为闰年。

【解析】

```
x = input('Please input an integer of 4 digits meaning the year:')
x = eval(x)
if x%400==0 or (x%4==0 and not x%100==0):
    print('Yes')
else:
```

```
    print('No')
```

11、编写程序，实现分段函数计算，如下表所示。

x	y
x<0	0
0<=x<5	x
5<=x<10	3x-5
10<=x<20	0.5x-2
20<=x	0

【解析】

```
x = input('Please input x:')
x = eval(x)
if x<0 or x>=20:
    print(0)
elif 0<=x<5:
    print(x)
elif 5<=x<10:
    print(3*x-5)
elif 10<=x<20:
    print(0.5*x-2)
```

12、阅读下面的程序，判断其是否可以正常运行，如果可以运行则写出执行结果，如果不能运行则写出理由。

```
class Test:
  def __init__(self, value):
      self. __value = value
  def value(self):
      return self. __value
t = Test(3)
t.value = 5
print(t.value)
```

【解析】不能运行。程序中定义的是只读属性，不能修改属性的值。

13、下面代码的功能是，随机生成 50 个介于[1,20]之间的整数，然后统计每个整数出现频率。请把缺少的代码补全。

```
import random
x = [random. __(1,20) for i in range(_)]
```

```
r = dict()
for i in x:
    r[i] = r.get(i,____)+1
for k, v in r.items():
    print(k, v)
```

【解析】分别填写 randint、50、0

14、假设有 Python 程序文件 demo.py，代码如下：

```
def main():
  if __name__== '__main__':
      print(1)
  else:
      print(2)
main()
```

将该程序文件直接运行时输出结果为__，作为模块导入时得到结果_____。

【解析】1、2

15、下面程序的执行结果是_____。

```
s = 0
for i in range(1,101):
    s += i
else:
    print(1)
```

【解析】1

16、下面程序的执行结果是　　。

```
s = 0
for i in range(1,101):
    s += i
  if i == 50:
      print(s)
      break
  else:
      print(1)
```

【解析】1275

17、下面的程序是否能够正常执行，若不能，请解释原因；若能，请分析其执行结果。

```
from random import randint
result = set()
while True:
    result.add(randint(1,10))
    if len(result)==20:
        break
print(result)
```

【解析】无法正确执行，因为该程序的功能是从[1,10]区间中选择 20 个不同的随机整数，而该区间并没有这么多整数，所以程序死循环。

18、下面的代码是否能够正确运行，若不能请解释原因；若能，请分析其执行结果。

```
x = list(range(20))
for i in range(len(x)):
    del x[i]
```

【解析】无法正确执行，因为删除列表元素时会影响其他元素在列表中的索引，上面的代码会抛出下标越界的异常。

19、阅读下面的代码，解释其功能。

```
x = list(range(20))
for index, value in enumerate(x):
    if value == 3:
        x[index] = 5
```

【解析】将列表x中值为3的元素修改为5。

20、阅读下面的代码，解释其功能。

```
x = [range(3*i, 3*i+5) for i in range(2)]
x = list(map(list, x))
x = list(map(list, zip(*x)))
```

【解析】首先生成一个包含列表的列表，然后模拟矩阵转置。

21、阅读下面的代码，解释其功能。

```
import string
x = string.ascii_letters + string.digits
import random
print(''.join(random.sample(x, 10)))
```

【解析】输出由英文字母大小写或数字组成的长度为 10 且不重复的随机字符串。

22、阅读下面的代码，分析其执行结果。

```
def demo(*p):
    return sum(p)
print(demo(1,2,3,4,5))
print(demo(1,2,3))
```

【解析】输出结果为

15

6

23、阅读下面的代码，分析其执行结果。

```
def demo(a, b, c=3, d=100):
    return sum((a,b,c,d))
print(demo(1, 2, 3, 4))
print(demo(1, 2, d=3))
```

【解析】输出结果为

10
9

24、下面的代码输出结果为_____。

```
def demo():
    x = 5
x = 3
demo()
print(x)
```

【解析】3

25、下面函数的功能为_____。

```
def demo(lst, k):
    if k<len(lst):
        return lst[k:]+lst[:k]
```

【解析】将序列循环左移 k 位，得到新序列并返回。

26、编写函数，求任意整数的二进制形式中最后连续 0 的个数。

【解析】

```
def demo(n):
    b_n = bin(n)
    index = b_n.rfind('1') + 1
    return len(b_n[index:])
```

27、有n个乒乓球运动员打淘汰赛，编写函数计算至少需要多少场比赛才能决出冠军，不允许直接使用n-1。

【解析】

```
def demo(n):
    if n == 1:
        return 0
    if n == 2:
        return 1
    m, c = divmod(n, 2)
    return m + demo(c+m)
```

28、使用循环和列表推导式两种方法求解百钱买百鸡问题。假设大鸡 5 元一只，中鸡3元一只，小鸡1元三只，现有 100 元钱想买 100 只鸡，有多少种买法？

【解析】

（1）循环

```
for x in range(21):
    for y in range(34):
        z = 100-x-y
        if z%3==0 and 5*x + 3*y + z//3 == 100:
            print(x,y,z)
```

0 25 75
4 18 78
8 11 81
12 4 84

（2）列表推导式

[(x, y, 100-x-y) for x in range(21) for y in range(34) if (100-x-y)%3==0 and 5*x+3*y+(100-x-y)//3==100]

[(0, 25, 75), (4, 18, 78), (8, 11, 81), (12, 4, 84)]

29、编写函数，给定任意字符串，找出其中只出现一次的字符，如果有多个这样的字符，就全部找出。

【解析】

```
def searchOne(s):
# 创建空字典
    d = dict()
    # 遍历字符串，并分别记录每个字符的出现次数
    for ch in s:
    #  这里重点演示字典的 get()方法
    #  如果这个字符出现过，加 1
    # 如果这个字符第一次出现，0+1
        d[ch] = d.get(ch, 0) + 1
    # 列表推导式，查找所有只出现一次的字符
    chs = [ch for ch, n in d.items() if n==1]
    # 返回最终结果，所有只出现一次的字符
    return chs
print(searchOne('abcdddca'))
```

30、阅读以下冒泡法排序代码，尝试写出优化代码，提高代码运行效率。

```
from random import randint
def bubbleSort(lst):
    length = len(lst)
    for i in range(0, length):
        for j in range(0, length-i-1):
            #比较相邻两个元素大小，并根据需要进行交换
            if lst[j] > lst[j+1]:
                lst[j], lst[j+1] = lst[j+1], lst[j]
lst = [randint(1, 100) for i in range(20)]
print('Before sort:\n', lst)
bubbleSort(lst)
print('After sort:\n', lst)
```

【解析】

```
from random import randint
def bubbleSort(lst):
    length = len(lst)
    for i in range(0, length):
flag = True
        for j in range(0, length-i-1):
        #比较相邻两个元素大小，并根据需要进行交换
       if lst[j] > lst[j+1]:
            lst[j], lst[j+1] = lst[j+1], lst[j]
            flag = False
        if flag:
            break
lst = [randint(1, 100) for i in range(20)]
print('Before sort:\n', lst)
bubbleSort(lst)
print('After sort:\n', lst)
```

31、编写程序，用户输入带有千分位逗号的数字字符串，然后输出不带千分位逗号的数字字符串。如果输入字符串'0'则退出程序。

【解析】

```
def convert(strNumber):
    return ''.join(strNumber.split(','))
while True:
    x = input('输入带有千分位逗号的数字：')
    if x == '0':
        print('bye')
        break
     print(convert(x))
```

32、编写程序，用户输入不带千分位逗号的数字字符串，然后输出带千分位逗号的数字字符串。

【解析】

```
 def convert(strNumber):
    # 考虑小数的情况
    temp = strNumber.split('.', 1)
    # 整数部分
    first = temp[0]
    if not first.isdigit():
        return '不是有效数字'
    # 小数部分
```

```
    try:
        second = temp[1]
        if not second.isdigit():
            return '不是有效数字'
    except:
        second = ''
    # 增加千分位逗号
    def nested(s):
        result = []
        length = len(s)
        index = length % 3
        if index != 0:
            result.append(s[:index])
        for i in range(index, length, 3):
            result.append(s[i:i+3])
        return ','.join(result)
    first = nested(first)
    # 小数部分和整数部分的千分位不一样
    if second:
        second = ''.join(reversed(second))
        second = nested(second)
        second = ''.join(reversed(second))
        # 删除两侧可能的 0 和千分位逗号
        return '.'.join([first, second]).strip(',0')
    # 删除整数左侧可能的 0 和逗号
    return first.lstrip('0,')
# 测试
while True:
    x = input('输入不带千分位逗号的数字：')
    if x == '0':
        print('bye')
        break
    print(convert(x))
```

五、进阶编程题

编写程序，解决如下问题：

1、题目：有数字1、2、3、4，能组成多少个互不相同且无重复数字的三位数？都是多少？

【解析】

1.程序分析：可填在百位、十位、个位的数字都是 1、2、3、4。组成所有的排列后再去掉不满足条件的排列。

2.程序源代码：

```
for i in range(1,5):
    for j in range(1,5):
        for k in range(1,5):
            if( i != k ) and (i != j) and (j != k):
                print(i,j,k)
```

2、题目：一个整数，它加上100后是一个完全平方数，再加上268又是一个完全平方数，请问该数是多少？

【解析】

1.程序分析：在 10 万以内判断，先将该数加上 100 后再开方，再将该数加上 268 后再开方，如果开方后的结果满足如下条件，即是结果。请看具体分析：

2.程序源代码：

```
import math
for i in range(100000):
    #转化为整型值
    x = int(math.sqrt(i + 100))
    y = int(math.sqrt(i + 268))
    if(x * x == i + 100) and (y * y == i + 268):
     print(i)
```

3、题目：输入某年某月某日，判断这一天是这一年的第几天？

【解析】

1.程序分析：以 3 月 5 日为例，应该先把前两个月的加起来，然后再加上 5 天即本年的第几天，特殊情况，闰年且输入月份大于 3 时需考虑多加一天。

2.程序源代码：

```
year = int(input('year:\n'))
month = int(input ('month:\n'))
day = int(input ('day:\n'))

months = (0,31,59,90,120,151,181,212,243,273,304,334)
if 0 <= month <= 12:
        sum = months[month -1]
else:
```

```
        print ('data error')
sum += day
        leap = 0
if (year % 400 == 0) or ((year %4==0) and (year % 100 != 0)):
        leap = 1
if (leap == 1) and (month > 2):
        sum += 1
print ('it is the %dth day. ' % sum)
```

4、题目：输入三个整数x,y,z,请把这三个数由小到大输出。

【解析】

1.程序分析：我们想办法把最小的数放到 x 上，先将 x 与 y 进行比较，如果 x>y 则将 x 与 y 的值进行交换，然后再用 x 与 z 进行比较，如果 x>z 则将 x 与 z 的值进行交换，这样能使 x 最小。

2.程序源代码：

```
l =[]
for i in range(3):
        x = int(input ('integer:\n'))
        l.append(x)
l.sort()
print(l)
```

5、题目：用*号输出字母C的图案。

【解析】

1.程序分析：可先用'*'号在纸上写出字母 C,再分行输出。

2.程序源代码：

```
print( 'Hello Python world!\n')
print( '*'* 10)
for i in range(5):
      print ('*')
print ('*' *10)
```

6、题目：输出9*9口诀。

【解析】

1.程序分析：分行与列考虑，共 9 行 9 列，i 控制行，j 控制列。

2.程序源代码：

```
for i in range(1,10):
        for j in range(1,10):
            result = i * j
            print( '%d * %d = % -3d' % (i,j,result))
        print('')
```

7、题目：要求输出国际象棋棋盘。

【解析】

1.程序分析：用 i 控制行，j 来控制列，根据 i+j 的和的变化来控制输出黑方格还是白方格。

2.程序源代码：

```
import sys
for i in range(8):
  for j in range(8):
      if(i+j)%2== 0:
          sys.stdout.write(chr(219))
          sys.stdout.write(chr(219))
      else:
          sys.stdout.write(' ')
  print('')
```

8、题目：古典问题：有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？

【解析】

1.程序分析：兔子的规律为数列 1,1,2,3,5,8,13,21…

2.程序源代码：

```
f1 = 1
f2 = 1
for i in range(1,21):
      print( '%12d %12d' % (f1,f2))
      if (i%2) ==0:
          print('')
      f1 = f1 + f2
      f2 = f1 + f2
```

9、题目：判断101-200之间有多少个素数，并输所有素数。

【解析】

1.程序分析：判断素数的方法：用一个数分别去除 2 到 sqrt(这个数)，如果能被整除，则表明此数不是素数，反之是素数。

2.程序源代码：

```
h=0
leap = 1
from math import sqrt
from sys import stdout
for m in range(101,201):
      k = int(sqrt(m + 1))
      for i in range(2,k + 1):
          if m % i == 0:
```

```
                leap=0
                break
        if leap == 1:
            print( '%-4d'% m)
            h += 1
            if h % 10 == 0:
                print('')
        leap = 1
print( 'The total is %d' % h)
```

10、题目：打印出所有的“水仙花数”，所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。例如：153是一个“水仙花数”，因为153=1的三次方+5的三次方+3的三次方。

【解析】

1.程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。

2.程序源代码：

```
for n in range(100,1000):
    i = n // 100
    j = n // 10 % 10
    k = n % 10

    if i**3 + j**3 + k**3 == n:
        print (n)
```

11、题目：将一个正整数分解质因数。例如：输入90，打印出90=2*3*3*5。

【解析】

1.程序分析：对 n 进行分解质因数，应先找到一个最小的质数 k，然后按下述步骤完成：

(1) 如果这个质数恰等于 n，则说明分解质因数的过程已经结束，打印出即可。

(2) 如果 n>k，但 n 能被 k 整除，则应打印出 k 的值，并用 n 除以 k 的商，作为新的正整数 n，重复执行第一步。

(3) 如果 n 不能被 k 整除，则用 k+1 作为 k 的值，重复执行第一步。

2.程序源代码：

```
from sys import stdout
n = int(input ("input number:\n"))
print ("n = %d" % n)
for i in range(2,n + 1):
  while n != i:
      if n%i==0:
          stdout.write(str(i))
          stdout.write("*")
          n=n/i
```

```
        else:
            break
print( "%d" % n)
```

12、题目：利用条件运算符的嵌套来完成此题：学习成绩>=90分的同学用A表示，60-89分之间的用B表示，60分以下的用C表示。

【解析】

1.程序分析：(a>b)?a:b 这是条件运算符的基本例子。

2.程序源代码：

```
score = int(input ('input score:\n'))
if score >= 90:
   grade = 'A'
elif score >= 60:
   grade = 'B'
else:
   grade = 'C'
print( '%d belongs to %s' % (score,grade))
```

13、题目：输入一行字符，分别统计出其中英文字母、空格、数字和其它字符的个数。

【解析】

1.程序分析：利用 while 语句，条件为输入的字符不为'\n'

2.程序源代码：

```
import string
s = input('input a string:\n')
letters = 0
space = 0
digit = 0
others = 0
for c in s:
   if c.isalpha():
        letters += 1
   elif c.isspace():
        space += 1
   elif c.isdigit():
        digit += 1
   else:
        others += 1
print( 'char = %d,space = %d,digit = %d,others = %d' % (letters,space,digit,others))
```

14、题目：一球从100米高度自由落下，每次落地后反跳回原高度的一半；再落下，求它在第10次落地时，共经过多少米？第10次反弹多高？

【解析】

1.程序分析：使用循环，每次高度减少一半，并将得到的新的落地高度与之前的求和

2.程序源代码：

```
Sn = 100.0
Hn = Sn / 2

for n in range(2,11):
   Sn += 2 * Hn
   Hn /= 2

print( 'Total of road is %f '% Sn)
print ('The tenth is %f meter'% Hn)
```

15、题目：利用递归方法求5!。

【解析】

1.程序分析：递归公式：5！=5*4*3*2*1=5*4！

2.程序源代码：

```
def fact(j):
   sum = 0
   if j== 0:
        sum = 1
   else:
        sum = j * fact(j -1)
   return sum

for i in range(5):
   print '%d! = %d' % (i,fact(i))

print ('%d! = %d' % (5,fact(5)))
```

16、题目：利用递归函数调用方式，将所输入的5个字符，以相反顺序打印出来。

【解析】

1.程序分析：由递归的定义和使用方法解出

2.程序源代码：

```
def palin(n):
   next = 0
   if n <= 1:
        next = input()
        print(next)
   else:
        next = input()
        palin(n -1)
```

```
        print(next)
i = 5
palin(i)
```

17、题目：给一个不多于5位的正整数，要求：一、求它是几位数，二、逆序打印出各位数字。

【解析】

1.程序分析：分解出每一位数

2.程序源代码：

```
x = int(input ("input a number:\n"))
a = x//10000
b = x% 10000//1000
c = x% 1000 // 100
d = x % 100 //10
e = x % 10

if a !=0:
        print( "there are 5 ",e,d,c,b,a)
elif b !=0:
        print( "there are 4 ",e,d,c,b)
elif c !=0:
        print( "there are 3 ",e,d,c)
elif d !=0:
        print( "there are 2",e,d)
else:
        print ("there are 1",e)
```

18、题目：一个5位数，判断它是不是回文数。即12321是回文数，个位与万位相同，十位与千位相同。

【解析】

1.程序分析：同上例

2.程序源代码：

```
x = int(input ("input a number:\n"))
x = str(x)
for i in range(int(len(x)/2)):
   if x[i] != x[-i -1]:
        print ('this number is not a huiwen')
        break
print( 'this number is a huiwen')
```

19、题目：请输入星期几的第一个字母来判断一下是星期几，如果第一个字母一样，则继续判断第二个字母。

【解析】

1.程序分析：用情况语句比较好，如果第一个字母一样，则判断用情况语句或 if 语句判断第二个字母。

2.程序源代码：

```
from sys import stdin
letter = stdin.read(1)
stdin.flush()
while letter != 'Y':
  if letter == 'S':
      print('please input second letter')
      letter = stdin.read(1)
      stdin.flush()
      if letter == 'a':
          print( 'Saturday')
      elif letter == 'u':
          print('Sunday')
      else:
          print('data error')
          break
  elif letter == 'F':
      print('Friday')
      break
  elif letter == 'M':
      print('Monday')
      break
  elif letter == 'T':
      print('please input second letter')
      letter = stdin.read(1)
      stdin.flush()
      if letter == 'u':
          print('Tuesday')
      elif letter == 'h':
          print('Thursday')
      else:
          print('data error')
          break
  elif letter == 'W':
      print 'Wednesday'
  else:
      print('data error')
  letter = stdin.read(1)
  stdin.flush()
```

20、题目：求100之内的素数。

【解析】

1.程序分析：先将 1~100 的数量进行遍历，获得这个数是不是素数

除以这个小的整数（除了 1 和本身），获得余数，根据余数判断是否是素数

2.程序源代码：

```
list=[]
i=2
for i in range (2,100):
    j=2
    for j in range(2,i):
        if(i%j==0):
            break
    else:
        list.append(i)
print(list)
```

21、题目：对10个数进行排序。

【解析】

1.程序分析：可以利用选择法，即从后 9 个比较过程中，选择一个最小的与第一个元素交换,以此类推，即用第二个元素与后 8 个进行比较，并进行交换。

2.程序源代码：

```
if__name__== "__main__":
  N = 10
  # input data
  print('please input ten num:\n')
  l =[]
  for i in range(N):
      l.append(int(input ('input a number:\n')))
  for i in range(N):
      print(l[i])
  # sort ten num
  for i in range(N -1):
      min = i
      for j in range(i + 1,N):
          if l[min] > l[j]:min = j
      l[i],l[min] = l[min],l[i]
  print('after sorted')
  print(l)
```

22、题目：求一个3*3矩阵对角线元素之和。

【解析】

1.程序分析：利用双重 for 循环控制输入二维数组，再将 a[i][i]累加后输出。

2.程序源代码：

```
if__name__== '__main__':
   a = []
   sum = 0.0
   for i in range(3):
        a.append([])
        for j in range(3):
             a[i].append(float(input ("input num:\n")))
   for i in range(3):
        sum += a[i][i]
print(sum)
```

23、题目：将一个数组逆序输出。

【解析】

1.程序分析：用第一个与最后一个交换。

2.程序源代码：

```
if__name__== '__main__':
   a = [9,6,5,4,1]
   N = len(a)
   print(a)
   for i in range(int(len(a) / 2)):
        a[i],a[N - i -1] = a[N - i - 1],a[i]
   print(a)
```

24、题目：取一个整数a从右端开始的4～7位。

【解析】

1.程序分析：可以这样考虑：

⑴先使 a 右移 4 位,

⑵设置一个低 4 位全为 1,其余全为 0 的数。可用~(~0<<4),

(3)将上面二者进行&运算。

2.程序源代码：

```
if__name__== '__main__':
   a = int(input ('input a number:\n'))
   b = a ＞＞4
   c = ~(~0＜＜4)
   d = b & c
   print( '%o\t%o' %(a,d))
```

25、题目：画图，学用circle画圆形。

【解析】

1.程序分析：使用 tkinter 库函数

2.程序源代码：

```
if__name__== '__main__':
    from Tkinter import *
    canvas = Canvas(width=800, height=600, bg='yellow')
    canvas.pack(expand=YES, fill=BOTH)
    k = 1
    j = 1
    for i in range(0,26):
        canvas.create_oval(310 - k,250 - k,310 + k,250 + k,width=l)
        k += j
        j += 0.3
    mainloop()
```

26、题目：画图，学用line画直线。

【解析】

1.程序分析：使用 tkinter 库函数

2.程序源代码：

```
if__name__== '__main__':
    from Tkinter import *
    canvas = Canvas(width=300, height=300, bg='green')
    canvas.pack(expand=YES, fill=BOTH)
    x0 = 263
    y0 = 263
    y1 = 275
    x1 = 275
    for i in range(19):
        canvas.create_line(x0,y0,x0,y1, width=l, fill='red')
        x0 = x0 - 5
        y0 = y0 - 5
        x1 = x1 + 5
        y1 = y1 + 5

    x0 = 263
    y1 = 275
    y0 = 263
    for i in range(21):
        canvas.create_line(x0,y0,x0,y1,fill = 'red')
        x0 += 5
        y1 += 5
        y1 +=5
```

```
mainloop()
```

27、题目：画图，学用rectangle画方形。

【解析】

1.程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。

2.程序源代码：

```
if__name__== '__main__':
    from Tkinter import *
    root = Tk()
    root.title('Canvas')
    canvas = Canvas(root,width = 400,height = 400, bg = 'yellow')
    x0 = 263
    y0 = 263
    y1 = 275
    x1 = 275
    for i in range(19):
        canvas.create_rectangle(x0,y0,x1,y1)
        x0 -=5
        y0 -=5
        xl +=5
        yl +=5

canvas.pack()
root.mainloop()
```

28、题目：输入3个数a,b,c,按大小顺序输出。

【解析】

1.程序分析：利用指针方法。

2.程序源代码：

```
if__name__== '__main__':
    n1 = int(input ('nl = :\n'))
    n2 = int(input ('n2 = :\n'))
    n3 = int(input ('n3 = :\n'))

    def swap(p1,p2):
        return p2,p1

    if   n1 > n2 : n1,n2 = swap(n1,n2)
    if   n1 > n3 : n1,n3 = swap(n1,n3)
    if   n2 > n3 : n2,n3 = swap(n2,n3)
```

```
    print( n1,n2,n3)
```

29、题目：输入数组，最大的与第一个元素交换，最小的与最后一个元素交换，输出数组。

【解析】

1.程序分析：遍历数组，找出最大值和最小值所对应的下标，分别与第一个和最后一个元素进行交换

2.程序源代码：

```
def inp(numbers):
    for i in range(9):
        numbers.append(int(input ('input a number:\n')))
    numbers.append(int(input ('input a numberin')))
p = 0
def max_min(array):
    max_index = min_index = 0
    for i in range(0,len(num_array)):
        p=i
        if num_array[p] > num_array[max_index]:
            max_index = p
        elif num_array[p] < num_array[min_index]:
            min_index = p
    last_index = len(num_array) -1
    min_value = num_array[min_index]
    max_value = num_array[max_index]

    new_array = num_array[:]
    new_array[min_index] = num_array[last_index]
    new_array[last_index] = min_value

    new_array[max_index] = num_array[0]
    new_array[0] = max_value
def outp(numbers):
    for i in range(len(numbers)):
        print(numbers[i])

if__name__== '__main__':
    array =[]
    inp(array)
    max__min(array)
    outp(array)
```

30、题目：有n个人围成一圈，顺序排号。从第一个人开始报数(从1到3报数)，凡报到3的人退出圈子，问最后留下的是原来第几号的那位。

【解析】

1.程序分析：生成一个有 n 个元素的列表，赋值为各自的序号（从 1 开始），遍历列表，每逢 3 且列表元素不为 0 时，将列表置零

2.程序源代码：

```
if__name__== '__main__':
   nmax = 50
   n = int(input ('please input the total of numbers: '))
   num =[]
   for i in range(n):
        num.append(i + 1)

   i = 0
   k = 0
   m = 0

 while m < n -1:
        if num[i] != 0 :
             k += 1
        if k==3:
             num[i] = 0
             k = 0
             m += 1
        i+=l
        if i == n :
             i = 0
   i = 0
   while num[i] == 0:
         i += 1
        print(num[i])
```

31、题目：写一个函数，求一个字符串的长度，在main函数中输入字符串，并输出其长度。

【解析】

1.程序分析：使用 len()函数可以直接得到字符串的长度

2.程序源代码：

```
if__name__== '__main__':
   s = input ('please input a string:\n')
   print ('the string has %d characters. ' % len(s))
```

32、题目：编写input()和output。函数输入，输出5个学生的数据记录。

【解析】

1.程序分析：使用列表来模拟结构，列表中的每个元素也是一个列表，代表一个学生的数据记录

2.程序源代码：

```
#使用 list 来模拟结构(不使用 class)
#stu = [string,string,list]

N = 5
#stu
   #num : string
   #name : string
   #score[4]: list
student =[]
for i in range(N):
   student.append(['', '',[]])

def input_stu(stu):
   for i in range(N):
         stu[i][0] = input('input student num:\n')
         stu[i][1] = input('input student name:\n')
         for j in range(3):
               stu[i][2].append(int(input('score:\n')))

def output_stu(stu):
   for i in range(N):
         print ('%-6s%-10s' % ( stu[i][0],stu[i][1]))
         for j in range(3):
               print ('%-8d'%stu[i][2][j])
if__name__== '__main__':
   input_stu(student)
   print(student )
   output_stu(student)
```

33、题目：编写一个函数，输入n为偶数时，调用函数求l/2+1/4+...+l/n,当输入n为奇数时，调用函数l/l+l/3+...+l/n(利用指针函数)。

【解析】

1.程序分析：判断输入的数的奇偶性，分别调用不同的函数(根据题目要求的公式编写函数)

2.程序源代码：

```
def peven(n):
   i = 0
   s = 0.0
   for i in range(2,n + 1,2):
```

```
        s += 1.0 / i
    return s
def podd(n):
    s = 0.0
    for i in range(1, n + 1,2):
        s += 1 / i
    return s

def dcall(fp,n):
    s = fp(n)
    return s

if__name__== '__main__':
    n = int(input('input a number:\n'))
    if n % 2 == 0:
        sum = dcall(peven,n)
    else:
        sum = dcall(podd,n)
    print(sum)
```

34、题目：某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的，加密规则如下：每位数字都加上5，然后用和除以10的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。

【解析】

1.程序分析：将输入的四位整数存在一个列表里面，对列表的元素进行上述操作，即可得到最终结果

2.程序源代码：

```
from sys import stdout
if__name__== '__main__':
    a = int(input('input a number:\n'))
    aa =[]
    aa.append(a % 10)
    aa.append(a % 100/10)
    aa.append(a % 1000/100)
    aa.append(a /1000)
    for i in range(4):
        aa[i] += 5
        aa[i] %= 10
    for i in range(2):
        aa[i],aa[3 - i] = aa[3 - i],aa[i]
```

```
    for i in range(3,-1,-1):
        print(int(aa[i]))
```

35、题目：计算字符串中子串出现的次数。

【解析】

1.程序分析：使用 count 函数可以直接得出次数

2.程序源代码：

```
if__name__== '__main__':
    str1 = input('input a string:\n')
    str2 = input('input a sub string:\n')
    ncount = str1.count(str2)
    print(ncount)
```

36、题目：从键盘输入一些字符，逐个把它们送到磁盘上去，直到输入一个#为止。

【解析】

1.程序分析：考察 python 的文件操作

2.程序源代码：

```
if__name__== '__main__':
    from sys import stdout
    filename = input('input a file name:\n')
    fp = open(filename, "w")
    ch = input ('input string:\n')
    while ch != '#':
        fp.write(ch)
        stdout.write(ch)
        ch = input ('')
    fp.close()
```

37、题目：从键盘输入一个字符串，将小写字母全部转换成大写字母，然后输出到一个磁盘文件“test”中保存。输入的字符串以！结束。

【解析】

1.程序分析：考察 python 的文件操作

2.程序源代码：

```
if__name__== '__main__':
    fp = open('test.txt', 'w')
    string = input ('please input a string:\n')
    string = string.upper()
    fp.write(string)
    fp = open('test.txt', 'r')
    print(fp.read())
```

```
    fp.close()
```

38、题目：有两个磁盘文件A和B，各存放一行字母，要求把这两个文件中的信息合并(按字母顺序排列)，输出到一个新文件C中。

【解析】

1.程序分析：考察 python 的文件操作

2.程序源代码：

```
if__name__== '__main__':
    import string
    fp = open('JCP099.py')
    a = fp.read()
    fp.close()

    fp = open('JCP098.py')
    b = fp.read()
    fp.close()

    fp = open('C.txt', 'W')
    l = list(a + b)
    l.sort()
    s =''
    s = s.join(l)
    fp.write(s)
    fp.close()
```

39、 100的阶乘是多少？

【解析】

```
def f(n):
    if(n==0):
        return 1
    else:
        return n*f(n-1)
import math
print(math.factorial(100))
```

40、 一只青蛙一次只能跳一级或者两级台阶，青蛙跳到100级台阶有多少中跳法？

【解析】

```
def frog(n):
    if(n==0): return 0
    if(n==1): return 1
```

```
    else:
        return frog(n-1)+frog(n-2)

if __name__ == '__main__':
    jump = [1, 2]
    while len(jump) < 100:
        jump.append(jump[-1]+jump[-2])
    print(jump[99])
```

41、. 100个人集中在一个房间，至少有两个人生日相同的概率有多大？

【解析】

```
def birthday(n):
    p=1
    if(n>365):
        return 1
    else:
        for i in range(n):
            p *= (365-i)/365
        return 1-p

print(birthday(100))
```

42、 有一个五位数abcde，乘以4以后变成edcba，abcde是多少？

【解析】

```
for i in range(10000, 100000):
    a_1 = i//10000
    b_1 = (i-10000*a_1)//1000
    c_1 = (i-10000*a_1-1000*b_1)//100
    d_1 = (i-10000*a_1-1000*b_1-100*c_1)//10
    e_1 = (i-10000*a_1-1000*b_1-100*c_1-10*d_1)
    i1 = i*4
    e_2 = i1//10000
    d_2 = (i1-10000*e_2)//1000
    c_2 = (i1-10000*e_1-1000*d_2)//100
    b_2 = (i1 - 10000 * e_2 - 1000 * d_2 - 100 * c_2)//10
    a_2 = (i1-10000*e_2-1000*d_2-100*c_2-10*b_2)
    if(a_1==a_2):
        if(b_1 == b_2):
            if(c_1 == c_2):
                if(d_1 == d_2):
                    if(e_1 == e_2):
```

```
                print(i)
    else:
        i = i+1
```

43、.运用Monte Carno 方法计算圆周率的近似值。

【解析】

```
import random
import math

def pi(n):
    i=0
    count =0
    while i <= n:
        x = random.random()
        y = random.random()
        if pow(x,2)+pow(y,2)<1:
            count += 1
        i += 1
    pi = 4*count/n
    return pi
if __name__ == '__main__':
    print("圆周率的值是：{}".format(pi(1000000)))
```

44、. 一普查员问一位女士,“你有多少个孩子,他们多少岁?”女士回答:“我有三个孩子,他们的岁数相乘是36,岁数相加就等于隔离间屋的门牌号码.”普查员立刻走到隔邻,看了一看,回来说:”我还需要多少资料.”女士回答:“我现在很忙,我最大的孩子正在楼上睡觉.”普查员说:”谢谢,我已知道了 问题:那三个孩子的岁数是多少?

【解析】

```
import numpy as np
from collections import Counter

age_1 = []  # age_1用于存储3人年龄乘积为36的情况
age_set = []  # age_set用于存储满足要求的3人年龄的集合（无序）。以防止重复存储年龄一
    样、顺序不同的情况
for age1 in range(1, 18):
    for age2 in range(1, 18):
        for age3 in range(1, 18):
            if age1 * age2 * age3 == 36:
                if {age1, age2, age3} not in age_set:  # 若满足条件的年龄未存储过，则存储
                    age_set.append({age1, age2, age3})
```

```
                age_l.append([age1, age2, age3])

age_array = np.array(age_l)
age_sum = np.sum(age_array, axis=1)  # 求所有情况的年龄和

m = Counter(age_sum)  # 数出每种和的个数。由题，知道年龄和还不足以判断年龄，所以一定存在多个和与正确情况相同
key_l = []  # 记录有多个结果相同的和
for key, value in m.items():
    if value > 1:
        key_l.append(key)

ans = []
for key in key_l:  # 对于这些可能的和，找出其对应的年龄存储到ans中
    for i in range(age_array.shape[0]):
        if age_sum[i] == key:
            ans.append(age_array[i])

for elem in ans:  # 由题，有一个年龄最大的孩子，找出最大值唯一的情况，打印。即得到结果
    maxi = np.max(elem)
    if np.sum(elem == maxi) == 1:
        print(elem)
```

45、 有两个序列a,b，大小都为n,序列元素的值任意整形数，无序；要求：通过交换a,b中的元素，使序列a元素的和与序列b元素的和之间的差最小。

【解析】

```
from itertools import combinations
import numpy as np
def f(a, b):
    """
通过交换a,b中的元素，使序列a元素的和与序列b元素的和之间的差最小
    :param a: 输入序列1
    :param b: 输入序列2
    :return: 交换后的序列1，交换后的序列2，两序列最终和的差
    """
    total = a + b  # total为两个序列的拼接结果

    count = 0
    # 用组合的方式找出a的所有可能结果，依次遍历
```

```
    for a_new in combinations(total, len(a)):
        a_new = list(a_new)
        b_new = total.copy()
        for e in a_new:
            b_new.remove(e)  # 计算此时与a_new对应的b_new

        d = abs(np.sum(a_new) - np.sum(b_new))  # 计算此时两个序列的和 的差 的绝对值
        if count == 0:  # 若是第一个循环，则直接保存结果。delta为两序列和的最小差值、
    a_ans和b_ans为最佳结果
            delta = d
            a_ans = a_new.copy()
            b_ans = b_new.copy()
        elif d < delta:  # 若当前差值小于最小差值，则更新结果
            delta = d
            a_ans = a_new.copy()
            b_ans = b_new.copy()

        count = count + 1

    return a_ans, b_ans, delta

# 测试序列
a_test = [100, 99, 98, 1, 2, 3]
b_test = [1, 2, 3, 4, 5, 40]

a_test_ans, b_test_ans, diff = f(a_test, b_test)
print('The result is: ')
print('a: {}'.format(a_test_ans))
print('b: {}'.format(b_test_ans))
print('Min difference of the two lists: {}'.format(diff))
```

46、 有三顶红帽子和两顶白帽子。将其中的三顶帽子分别戴在 A、B、C三人头上。这三人每人都只能看见其他两人头上的帽子，但看不见自己头上戴的帽子，并且也不知道剩余的两顶帽子的颜色。问A：“你戴的是什么颜色的帽子?” A回答说:“不知道。” 接着，又以同样的问题问B。B想了想之后，也回答说:“不知道。” 最后问C。C回答说:“我知道我戴的帽子是什么颜色了。” 当然，C是在听了A、B的回答之后而作出回答的。请尝试用编程方法解答此问题。

【解析】

```
from collections import Counter
```

```
conditions = []  # 用于存储所有可能的情况
for A_hat in ['r', 'w']:  # 遍历所有情况，'r'表示红帽子，'w'表示白帽子
    for B_hat in ['r', 'w']:
        for C_hat in ['r', 'w']:
            A_see = {'B': B_hat, 'C': C_hat}  # 记录A、B、C、看到的结果
            B_see = {'A': A_hat, 'C': C_hat}
            C_see = {'A': A_hat, 'B': B_hat}
            if (A_hat == 'w') + (B_hat == 'w') + (C_hat == 'w') == 3:  # 白帽子只有两顶，3人不可能都是白帽子
                continue
            elif Counter(A_see.values())['w'] == 2 or Counter(B_see.values())['w'] == 2 or \
                    Counter(C_see.values())['w'] == 2:  # 若看到另外两人都是白帽子，则能确定自己是红。故不可能。
                continue
            elif B_see['C'] == 'w':  # 由A所说知，B、C中至少一顶红帽子。若C是白帽子，则B能确定，故舍去
                continue
            elif C_see['B'] == 'w':  # 仅根据A所说的，C还不能确定自己的帽子，同B
                continue
            else:  # 剩余的情况均满足要求，以字典的形式保存到列表中
                conditions.append({'A': A_hat, 'B': B_hat, 'C': C_hat})

print('Possible conditions: {}'.format(conditions))  # 打印出所有可能的结果
C_set = []  # 总结出C可能戴的帽子
for e in conditions:
    C_set.append(e['C'])
C_set = set(C_set)
print('C: {}'.format(C_set))  # 打印C可能的帽子集合
```

47、 汉诺塔问题编程解答。

【解析】

```
def hanio(n, x, y, z):
    if n==1:
        print(x, '-->', z)
    else:
        hanio(n-1, x, z, y)
        print(x, '-->', z)
        hanio(n-1, y, x, z)
```

```
if __name__ == '__main__':
    n = int(input('请输入汉诺塔的层数：'))
    hanio(n, 'X', 'Y', 'Z')
```

48、 八皇后问题编程解答。

【解析】

```
def queen(A, cur=0):
    """
八皇后问题解答
    :param A: list. 存储每一行放置皇后的位置。A的长度表示棋盘的边长
    :param cur: 当前需要放置皇后的行号
    """
    if cur == len(A):  # 当前行号达到A的长度，说明皇后已经全部放置完成，打印结果
        print(A)
        return 0
    for col in range(len(A)):  # 遍历当前行的每一列col，看是否可以放置皇后
        A[cur], flag = col, True  # 在当前行把皇后暂且放在第col列。flag为True时表示暂未发生冲突
        for row in range(cur):  # 遍历前面已经放置过的所有行
            if A[row] == col or abs(col - A[row]) == cur - row:  # 若前面已经在第col列放置过，或当前位置的斜方向已放置过，则产生冲突，flag=False，跳出循环
                flag = False
                break
        if flag:  # 若falg为True，则当前位置不冲突，继续放置下一位置；否则，有冲突，当前位置无效，继续循环。
            queen(A, cur + 1)

queen([None] * 8)
```