《Python 编程精通与实战》习题及其答案

为了便于读者更加深刻地理解 Python，针对教材内容，特提供习题及其答案，习题类型包括：选择题、填空题、简答题、判断题、应用题。

1　单项选择题

1.1 习题

1. Python 中，可以表示真或假的两个特殊值是（　　）。
   A. true 和 false　　B. yes 和 no　　C. on 和 off　　D. 1 和 0
2. Python 中用来读取用户输入的函数是（　　）。
   A. input()　　B. print()　　C. read()　　D. scan()
3. Python 中用来分割字符串的方法是（　　）。
   A. split()　　B. strip()　　C. slice()　　D. divide()
4. Python 中用来连接字符串的符号是（　　）。
   A. +　　B. -　　C. *　　D. /
5. Python 中的 if 语句后面要加上（　　）。
   A. 逗号（,）　　B. 冒号（:）　　C. 分号（;）　　D. 点号（.）
6. 在 Python 中，以下（　　）选项表示将一个变量赋值为列表类型。
   A. my_list=[1,2,3]　　B. my_list={“apple”,“banana”,“cherry”}
   C. my_list=(1,2,3)　　D. my_list=<1,2,3>
7. 下列（　　）Python 内置函数可用于去除字符串首位空格。
   A. trim()　　B. strip()　　C. delete()　　D. erase()
8. 在 Python 中，（　　）关键字用于定义函数。
   A. function　　B. def　　C. define　　D. procedure
9. 以下（　　）运算符可以用于比较两个字符串是否相等。
   A. ==　　B. !=　　C. <　　D. >
10. 在 Python 中，以下（　　）选项表示将两个整数相加。
    A. 2+3　　B. "2" + "3"　　C. 2+3.0　　D. "2" + "3” +"4”
11. 在 Python 中，下列（　　）定义一个空的列表。
    A. list()　　B. []　　C. empty_list()　　D. {}
12. 在 Python 中，以下（　　）是正确的字符串拼接方法。
    A. str1.join(str2)　　B. str1 + str2　　C. str1.concat(str2)　　D. str1.append(str2)

13. 下列（　　）选项，打印字符串 "Hello, World!"。

A. print("Hello, World!")　　B. echo("Hello, World!")

C. display("Hello, World!")　　D. show("Hello, World!")

14. 下面选项中，（　　）是 Python 的注释符号。

A. //　　B. #　　C. --　　D. /* */

15. 在下面选项中，（　　）获取一个字符串的长度。

A. str.size()　　B. str.len()　　C. len(str)　　D. str.length()

16. 下列选项中，（　　）不是合法的变量名？

A. my_variable　　B. 2variable　　C. _variable　　D. variable_2

17. 下列选项中，（　　）将字符串转换为小写。

A. str.lower()　　B. str.to_lower()　　C. str.casefold()　　D. str.convert_lower()

18. 下面（　　）选项，可以创建一个包含连续整数的列表，从 1 到 10（包括 1 和 10）。

A. list(range(1, 11))　　B. list(range(1, 10))

C. list(range(11))　　D. list(range(0, 10))

19. 以下（　　）选项，用于从控制台获取用户输入。

A. get_input()　　B. input()　　C. user_input()　　D. read_input()

20. 下面（　　）选项，用于删除字典中指定键的条目。

A. delete()　　B. remove()　　C. del()　　D. discard()

21. pandas 中用于读取 Excel 文件的函数是：

A. read_csv()　　B. read_excel()　　C. read_table()　　D. read_sql()

22. 下列关于 Python 的描述错误的是（　　）。

A. Python 语言采用严格的“缩进” 来表明程序的格式框架。

B. Python 语言中， 字符串是用一对双引号""或者一对单引号'' 括起来的零个或者多个字符。

C. 在 Python 语言中， 赋值与二元操作符不可以组合。

D. Python 语言的多行注释以''' （三个单引号） 开头和结尾。

23. 以下（　　）为合法的 Python 变量名。

A. true　　B. False　　C. import　　D. if

24. 执行表达式 3+'3' 后， 结果为（　　）。

A. 6　　B. ' 33'　　C. 33　　D. 报错

25. 以下表示式（　　）不能计算 a 的 b 次方。

A. a**b　　B. math.pow(a, b)　　C. a^b　　D. pow(a, b)

26. 在炒菜机器人程序中，“第一步，加油少许”违背了算法的（　　）特征。

A. 确定性　　B. 可行性　　C. 有穷性　　D. 有输入和输出的

27. 以下选项中，（　　）不符合 Python 语言变量命名规则。

A. am　　B. in　　C. _AI　　D. str1

28. 下列关于 Python 语言描述正确的是（　　）。

A. Python 语言是一种面向机器的程序设计语言。

B. Python 语言比 Java、C/C++等程序设计语言好。

C. Python 编写的语言可读性强，因此它是一种自然语言。

D. Python 语法简洁，类库丰富。

29. 下列选项中， 能作为 Python 变量名的是（ ）。

A. Ab B. 4Ab C. if D. s+1

30. 下列各语句中，（ ）输出结果为 False。

A. print(7>2) B. print(5>0)

C. print((1>2) or(2>1)) D. print(2==0)

31. 以下选项中，（ ）不是 Python 逻辑运算符。

A. not B. and C. break D. or

32. 在 Python 中， 变量命名错误的是（ ）。

A. for B. a2 C. _a2 D. a_2

33. 在 Python 中，具有输出功能的函数是（ ）。

A. input() B. print() C. float() D. int()

34. 在 Python 中，能将字符串型数据转换为不含小数点的数字型数据的函数是（ ）。

A. str() B. print() C. float() D. int()

35. 在 Python 中，下列程序段的运行结果是（ ）。

```
a=5
b=2
print(a%b)
```

A. 2. 5 B. 2 C. 1 D. 3

36. 在 Python 语言中，表示取余运算的运算符是（ ）。

A. * B. / C. // D. %

37. Python 语言中，实现代码快速缩进的方法是（ ）。

A. 连续空格键 B. Tab C. Shift+Ctrl D. Alt+Tab

38. 在 Python 中，语句 x=input("请输入你的身高：") 的数据类型是（ ）。

A. 整数型 B. 浮点数型 C. 字符型 D. 引用型

39. 下列可以用作表示多行注释的是（ ）。

A. 前后加# B. 前后加''' C. 前后加/// D. 以上都不是

40. 以下选项中，（ ）不是 Python 语言的保留字。

A. for B. do C. in D. while

41. 在算法的流程图描述方法中， 表示输入输出的通常是（ ）。

A. 矩形框 B. 菱形框 C. 平行四边形框 D. 圆角矩形框

42. 下列（ ）不是 Python 的应用。

A. 移动终端开发 B. 图形图像处理

C. 游戏开发 D. 图形程序的开发

43. 下列关于变量的说法错误的是（ ）。

A. 变量用来暂时表示一个数据。

B. 变量名可以是字母、数字、下划线。

C. Python 的变量名不区分大小写。

D. 数字不能作为变量名的开头。

44. 拟在屏幕上打印输出“Hello World”， 以下选项中正确的是（ ）

A. print("Hello World") B. print(Hello World)

C. printf(“Hello World”) D. printf(‘Hello World’)

45. 在 Python 中，if 结构被用在（ ）

A. 语句相继被执行时 B. 执行一些语句之前必须先做出判断时

C. A 和 B 都是 D. A 和 B 都不是

46. 下面哪一行代码的输出结果不是“Python3.7”（ ）。

A. print("Python3.7") B. print("Python+3.7")

C. print("Python"+str(3.7)) D. print("Python"+"3.7")

47. 在 Python 中，表达式 b*b-4*a*c 用数学方法表示为（ ）。

A. 2b-4a-4c B. b×b-4ac C. 2b-4ac D. b2-ac4

48. 在 Python 中，下列哪个值是整数（ ）

A. 5. 0 B. “5. 0” C. -5 D. 以上都不是

49. 在 Python 程序中，已知列表 m=[2, 4, 0, 23, 1, 20]，那么 m[1]表示的元素是（ ）

A. 2 B. 4 C. 20 D. 1

50. 定义了列表 m=[1, 2, 3, 7, 5]，则当前列表中元素 m[2]的值是（ ）

A. 2 B. 3 C. 7 D. 1

51. 字符串 s=' hello'，则 len(s)的值是（ ）。

A. 2 B. 6 C. 5 D. 4

52. 下面代码“print(35-10)”，输出的结果是（ ）。

A. 35-10 B. ' 35-10' C. "35-10" D. 25

53. print 的作用是（ ）

A. 在屏幕上打印出来相应的文本或者数字等

B. 在打印机里打印相关文本或者数字等

C. 可以用来画图

D. 输出一个命令行

54. 在 Python 中，x=2.6，表达式 int(x)的结果是（ ）

A. 3 B. 2.6 C. 2.0 D. 2

55. 关于 Python 语句中，比较是否相等的运算符是（ ）

A. = B. == C. != D. <>

56. 在 Python 语言中，以下（ ）是乘方运算的运算符

A. * B. / C. // D. **

57. 在 Python 中，要使用（ ）转换为浮点数。

A. int B. float C. print D. for

58. 下列表达式中，值不是 1 的是（ ）

A. 6//5 B. 17%8 C. 7//3 D. 1**0

59. 关于变量的说法，错误的是（ ）。

A. 变量必须要命名

B. 变量第二次赋值后，第一次赋的值将被删除
C. 变量只能用来存储数字，不能表示存储文字
D. 在同一个程序里，变量名不能重复

60. 关于 Python 语句“A=-A”，这条语句描述的是（　　）。
A. A 和 A 的负数相等　　B. A 和 A 的绝对值相等
C. 给 A 赋值为它的负数　　D. A 的值为 0

61. 下面代码的输出结果是（　　）。
```
x=10
y=3
print(x%y, x**y)
```
A. 3 1000　　B. 1 30　　C. 3 30　　D. 1 1000

62. 下列（　　）符号可以用来修改变量的值。
A. >=　　B. ==　　C. =　　D. !=

63. Python 运算符中，整除的符号是（　　）
A. /　　B. %　　C. //　　D. **

64. Python 中的==代表的是（　　）
A. 把左边的值赋值给右边　　B. 把右边的值赋值给左边；
C. 比较左右两边是否相等　　D. 左右两边值进行交换；

65. 下列有关算法的叙述中，错误的是（　　）
A. 算法是解决问题的方法和步骤　　B. 解决某个具体问题的算法只有一个
C. 算法都必须在有限步骤内结束　　D. 算法都至少包含一个输出

66. 在 Python 中，表达式 a<=b 的含义是（　　）
A. a≮b　　B. a<b　　C. a=b　　D. a≤b

67. 在 Python 中，下列程序段的运行结果是（　　）
```
a=10
b=5
print(a+b)
```
A. 105　　B. 15　　C. ' 105'　　D. "15"

68. 在 Python 中，下列字符串常量引用错误的是（　　）
A. 256　　B. "OK"　　C. "256"　　D. ' OK'

69. 在 Python 中，可处理的数据类型有（　　）
①字符串型　②数值型　③列表
A. ①②　　B. ②③　　C. ①　　D. ①②③

70. 使用计算机解决问题，需要经历四个主要阶段，正确的是（　　）
A. 分析问题→设计算法→编写代码→运行程序
B. 设计算法→分析问题→编写代码→运行程序
C. 分析问题→编写代码→设计算法→运行程序
D. 设计算法→编写代码→分析问题→运行程序

71. 下列有 while 循环结构的说法，错误的是（　　）

A. while 循环格式一般：while(表达式)：语句或者语句组

B. 执行过程中，表达式一般是一个关系表达式或逻辑表达式

C. 表达式为真，执行循环体；为假，退出循环

D. 表达式为假，执行循环体；为真，退出循环

72. 在 Python 中，表达式 75>70 and 75<85 的值为（　　）

A. 0　　B. 1　　C. True　　D. False

73. 在 Python 中，与 1≤x≤10 相对应的表达式为（　　）

A. 1<x<=10　　B. x>=1 and x<=10

C. x<=1 and x>=10　　D. x>=1 or x<=10

74. 下列 Python 表达式中，值为字符串类型的是（　　）

①"abc"*2 ②"123+456" ③123+456 ④"123"+"456"

A. ①②③　　B. ②③④　　C. ①②④　　D. ②①④

75. 在 Python 中，字符串运算符“+”的作用是把字符串进行连接，则表达式"20"+"21"+"20+21"的运算结果是（　　）

A. 202120+21　　B. 4141　　C. 20212021　　D. 202141

76. Python 中，打开文件的常用方式有（　　）种。

A. 一种　　B. 两种　　C. 三种　　D. 四种

77. 以下（　　）选项是正确的文件打开模式

A. "r"　　B. "w"　　C. "a"　　D. "x"

78. 在 Python 中， 如何读取一个文本文件的内容？

A. 使用 open()函数打开文件，然后使用 read()方法读取文件内容。

B. 使用 os. read()函数打开文件，然后使用 read()方法读取文件内容。

C. 使用 open()函数打开文件，然后使用 readlines()方法读取文件内容。

D. 使用 os. readlines()函数打开文件，然后使用 readlines()方法读取文件内容。

79. 下面关于组合数据类型的叙述，错误的是（　　）

A. 列表能处理任意长度，混合类型数据的能力；因此当需要使用序列类型时，应首先考虑列表

B. 字典是一种“键-值”数据项的元素组合，其中的键允许有重复值

C. 当所需数据样本中不能有重复数据时，应选择使用集合类型

D. 当需要对元素数据进行保护或者用于固定搭配的场景时，尽量使用元组类型

80. 在列表类型中，不能添加元素的方法是（　　）

A. list.insert()

B. list.extend()

C. list.append()

D. list.add()

81. 下列是 Python 合法标识符的是（　　）。

A. 2variable　　B. variable2　　C. $anothervar　　D. if

82. random 库中，用于从序列类型中随机返回一个元素的函数是：（　　）

A. random()　　B. randint()　　C. choice()　　D. sample()

83. 以下（　　）不是 Python 的数值运算操作符。

A. $　　B. %　　C. **　　D. &

84. 设有字符串 a='Nanjing'，b='jing'，则执行下面（　　）语句会得到 True。

A. b in a　　B. b is a　　C. a in b　　D. a is b

85. 以下关于 Python 整数的定义，错误的是（　　）。

A. 0o1019　　B. 0b1010　　C. 0x1af　　D. 3670

86. 在 Python 中，下列（　　）操作可以打开文件并用写入模式写入文件内容。

A. open("file.txt", "r")　　B. open("file.txt", "w")

C. open("file.txt", "a")　　D. open("file.txt", "rb")

87. 下列选项中属于元组的是（　　）。

A. (21,32,43,45)　　B. 'Hello'　　C.[21,32,43,45]　　D.21

88. 在一个应用程序中，定义 a=[1,2,3,4,5,6,7,8,9,10]，为了打印输出列表后 a 的最后一个元素，下面正确的代码是（　　）。

A. print(a[10])　　B.print(a[9])　　C.print(a[len(a)])　　D.print(a(9))

89. 下列（　　）方法可以用于读取文件的全部内容。

A. read()　　B. readlines()　　C. readline()　　D. readall()

90. 若要在写入文件时，保证写入的内容不会被覆盖，而是追加在文件原有内容的末尾，应该使用（　　）模式打开文件。

A. 读取模式　　B. 写入模式　　C. 追加模式　　D. 二进制模式

91. 下面代码的输出结果是（　　）。

```
a = [5,1,3,4]
print(sorted(a,reverse = True))
```

A. [5, 1, 3, 4]　　B. [5, 4, 3, 1]　　C. [4, 3, 1, 5]　　D. [1, 3, 4, 5]

92. 同时去掉字符串左边和右边空格的函数，是（　　）。

A. center()　　B. count()　　C. fomat()　　D. strip()

93. 当需要在处理文件操作完成后自动关闭文件，应该使用下列（　　）语句。

A. for 循环　　B. try-except 语句　　C. with 语句　　D. while 循环

94. 下面（　　）语句可以打出“Hello,World!”。

A. Print(“Hello,World!”)　　B. echo(“Hello,World!”)

C. printf(“Hello,World!”)　　D. System.out(“Hello,World!”)

95. 在 Python 中，以下（　　）语句可以将两个列表 list1 和 list2 中的元素合并到一个新的列表 newList 中。

A. newList = list1 + list2　　B. newList = list1.append(list2)

C. newList = list1.extend(list2)　　D. newList = list1.join(list2)

96. 在 Python 中，（　　）语句可以删除 myList 中的第一个元素。

A. Del myList[0]　　B. myList.pop(0)

C. myList.remove(0)　　D. myList.delete(0)

97. 在 Python 中，以下（　　）操作符用于计算两个数的余数。

A. %　　B. **　　C. /　　D. //

98. 在 Python 中，以下（　　）语句将字符串 str1 转换为一个整数类型的变量。

A. int(str1)　　B. float(str1)　　C. str1.int()　　D. str1.to_int()

99. 在 Python 中，以下（　　）函数可以返回一个列表中元素的个数。

A. list.count()　　B. list.size()　　C. list.items()　　D. List.len()

100. 下列 Python 表达式中，能正确表示不等式方程|x|>1 的解是（　　）。

A. x>1 or x<-1　　B. x>-1 or x<1　　C. x>1 and x<-1　　D. x>-1 and x<1

101. 在 Python 中，运行下列程序，正确的结果是（　　）。

```
s=0
for i in range(1,5):
    s=s+i
print('i=',i,'s=',s)
```

A. i=4, s=10　　B. i=5, s=10　　C. i=5, s=15　　D. i=6, s=15

102. 在 Python 中，已知 a=3，b=5，运行下列程序段后，a 和 b 的值为（　　）。

```
a= a*b
b=a//b
a=a//b
```

A. a=3 b=5　　B. a=15 b=3　　C. a=5 b=5　　D. a=5 b=3

103. Python 中，print(66!=66)结果是（　　）。

A. 1　　B. 0　　C. True　　D. False

104. 下列（　　）可以用于判断文件是否存在？

A. file_exists()　　B. exists()　　C. os.path.exists()　　D. os.file.exists()

105. 使用 Python 进行文件处理时，下列（　　）是最好的方式来关闭文件。

A. 使用 close()方法　　B. 使用 os.close()方法

C. 使用 with 语句　　D. 不关闭文件

106. 将文件 file.txt 重命名为 newfile.txt，应该使用（　　）方法。

A. rename()　　B. renameto()　　C. renamefile()　　D. mv()

107. range()函数的返回值是（　　）类型。

A. 列表　　B. 元组　　C. 字典　　D. 迭代器

108. range(1, 10, 2)的长度是（　　）。

A. 5　　B. 4　　C. 6　　D. 9

109. range(1, 10, 2)中的步长，是（　　）。

A. 1　　B. 2　　C. 3　　D. 4

110. range(1, 10, 3)中的最后一个元素，是（　　）。

A. 7　　B. 8　　C. 9　　D. 10

111. 以下关于模块说法，错误的是（　　）

A. 一个 xx.py 就是一个模块

B. 任何一个普通的 xx.py 文件可以作为模块导入

C. 模块文件的扩展名不一定是 .py

D. 运行时会从指定的目录搜索导入的模块，如果没有，会报错异常

112. 下列（　　）语句在 Python 中是非法的。

A. x = y = z = 1　　B. x = (y = z + 1)　　C. x, y = y, x　　D. x += y

113. 关于 Python 内存管理，下列（　　）错误。

A. 变量不必事先声明　　B. 变量无须先创建和赋值而直接使用

C. 变量无须指定类型　　D. 可以使用 del 释放资源

114. 下面（　　）不是 Python 合法的标识符。

A. int32　　B. 40XL　　C. self　　D. name

115. 下列（　　）是错误的。

A. 除字典类型外，所有标准对象均可以用于布尔测试

B. 空字符串的布尔值是 False

C. 空列表对象的布尔值是 False

D. 值为 0 的任何数字对象的布尔值是 False

116. Python 不支持的数据类型有（　　）。

A. char　　B. int　　C. float　　D. list

117. 关于 Python 中的复数，下列（　　）说法错误。

A. 表示复数的语法是 real + image j　　B. 实部和虚部都是浮点数

C. 虚部必须后缀 j，且必须是小写　　D. 方法 conjugate 返回复数的共轭复数

118. 关于字符串下列（　　）说法错误。

A. 字符应该视为长度为 1 的字符串

B. 字符串以\0 标志字符串的结束

C. 既可以用单引号，也可以用双引号创建字符串

D. 在三引号字符串中可以包含换行回车等特殊字符

119. 以下（　　）语句不能创建一个字典。

A. dict1 = {}　　B. dict2 = { 3 : 5 }

C. dict3 = {[1,2,3]: "uestc"}　　D. dict4 = {(1,2,3): "uestc"}

120. 下列 Python 语句中，正确的是（　　）。

A. min = x if x < y else y　　B. max = x > y ? x : y

C. if (x > y) print x　　D. while True : pass

121. 计算机中信息处理和信息储存用（　　）。

A. 二进制代码　　B. 十进制代码　　C. 十六进制代码　　D. ASCII 代码

122. Python 源程序执行的方式是：（　　）。

A. 编译执行　　B. 解析执行　　C. 直接执行　　D. 边编译边执行

123. Python 语言中，语句块的标记是（　　）。

A. 分号　　B. 逗号　　C. 缩进　　D. /

124. 以下选项中，（　　）是字符转换成字节的方法。

A. decode()　　B. encode()　　C. upper()　　D. rstrip()

125. 以下（　　）是正确的字符串。

A. 'abc"ab"　　B. 'ab\c"ab'　　C. "abc"ab"　　D. "abc\"ab"

126. "ab" + "c" *2 结果是：（　　）。

A. abc2　　B. abcabc　　C. abcc　　D. ababcc

127. 以下会出现错误的是（　　）。

A. '北京'.encode()　　B. '北京'.decode()

C. '北京'.encode().decode()　　D. 以上都正确

128. 如下程序：

```
str1 = "Runoob example…wow!!!"
str2 = "exam"
print(str1.find(str2, 5))
```

打印的结果是（　　）。

A. 6　　B. 7　　C. 8　　D. -1

129. 下面对 count(). index(). find()方法的描述，正确的是（　　）。

A. count()方法，用于统计字符串里某个字符出现的次数。

B. find()方法，检测字符串中是否包含子字符串 str，如果包含子字符串 str，则返回开始的索引值，否则会报一个异常。

C. index()方法，检测字符串中是否包含子字符串 str，如果 str 不在，返回-1。

D. 以上都错误。

130. 以下（　　）不是 Python 中的关键字。

A. raise　　B. with　　C. import　　D. final

131. 调用以下函数，返回的值是（　　）

```
def myfun():
pass
```

A. 0　　B. 出错不能运行　　C. 空字符串　　D. None

132. 函数如下：

```
def showNnumber(numbers):
for n in numbers:
print(n)
```

下面（　　）选项在调用这个函数时会报错。

A. showNumer([2,4,5])　　B. showNnumber('abcesf ')

C. showNnumber(3.4)　　D. showNumber((12,4,5))

133. 函数如下

```
def chanageInt(number2):
    number2 = number2+1
    print("changeInt: number2= ",number2)
#调用
number1 = 2
chanageInt(number1)
print("number:",number1)
```

打印结果（　　）是正确的。

A. changeInt: number2= 3 number: 3　　B. changeInt: number2= 3 number: 2

C. number: 2 changeInt: number2= 2　　D. number: 2 changeInt: number2= 3

134. 定义类如下：

```
class Hello():
pass
```

下面说明正确的是（　　）。

A. 该类实例中不包含__dir__()方法

B. 该类实例中不包含__hash__()方法

C. 该类实例中包含__dir__()，包含__hash__()

D. 该类没有定义任何方法，所以该实例中没有包含任何方法

135. 关于 Python 类，说法错误的是（　　）。

A. 类的实例方法必须创建对象后才可以调用

B. 类的实例方法必须创建对象前才可以调用

C. 类的类方法可以用对象和类名来调用

D. 类的静态属性可以用类名和对象来调用

136. 有关异常，下列说法中（　　）正确。

A. 程序中抛出异常终止程序　　B. 程序中抛出异常不一定终止程序

C. 拼写错误会导致程序终止　　D. 缩进错误会导致程序终止

137. 导入模块的方式，错误的是（　　）。

A. import mo　　B. from mo import *

C. import mo as m　　D. import m from mo

138. 在 Python 中，下列（　　）选项可以用于将一个字典对象写入文件。

A. write_dict()　　B. save_dict()　　C. dump()　　D. write()

139. 下列关于 Python 面向对象的说法中，错误的是：（　　）。

A. Python 是一种纯粹的面向对象编程语言。

B. Python 中一切皆为对象。

C. Python 中的类和对象可以继承其他类的属性和方法。

D. Python 中的类是一种数据类型，可以作为参数传递给函数。

140. 在 Python 中，可以使用（　　）关键字来定义一个类：

A. class

B. def

C. object

D. instance

141. 下列哪个选项中，（　　）的说法是正确的。

A. 类是对象的实例化

B. 对象是类的实例化

C. 和对象是完全不同的概念

D. 类和对象是一样的东西

142. 在 Python 中，可以使用（　　）方法来初始化一个类的实例。

A. __init__()　　B. __new__()　　C. __create__()　　D. __start__()

143. 下列选项中，（　　）的说法是正确的。

A. Python 中的继承是单继承的。　　B. Python 中的继承是多继承的。

C. Python 中没有继承的概念。　　D. Python 中的继承是选择性的。

144. 下列（　　）选项是正确的 numpy 导入方式。

A. import np　　B. import numpy as np

C. from numpy import np　　D. import numpy

145. 下列（　　）选项，可以通过 numpy，创建一个包含 10 个零的一维数组。

A. numpy.zeros(10)　　B. numpy.zeros((10,))

C. numpy.zeros((10))　　D. numpy.array([0] * 10)

146. 在 numpy 中，下列（　　）可以获取数组的形状. 大小和维度。

A. arr.shape, arr.size, arr.dim　　B. arr.shape, arr.size, arr.ndim

C. arr.shape, arr.length, arr.ndim　　D. arr.size, arr.shape, arr.dim

147. 下列（　　）选项，可以在 numpy 数组中进行元素访问。

A. 通过索引　　B. 通过切片　　C. 通过循环　　D. 以上都可以

148. 在 numpy 中，（　　）函数可以创建具有指定形状的数组，并用随机数填充。

A. numpy.random.rand　　B. numpy.random.random

C. numpy.random.randn　　D. numpy.random.randint

149. 在 numpy 中，（　　）可以进行数组的水平叠加。

A. numpy.hstack　　B. numpy.concat

C. numpy.append　　D. numpy.concatenate

150. 在 numpy 中，（　　）计算数组的平均值、最大值和最小值。

A. arr.mean(),arr.max(),arr.min()

B. arr.average(),arr.maximum(),arr.minimum()

C. numpy.mean(arr), numpy.max(arr), numpy.min(arr)

D. numpy.average(arr), numpy.maximum(arr), numpy.minimum(arr)

151. 在 numpy 中，（　　）进行数组的转置操作。

A. arr.transpose()　　B. numpy.transpose(arr)　　C. arr.T　　D. numpy.T(arr)

152. 在 numpy 中，（　　）使用 numpy 进行数组的逐元素乘法操作。

A. arr1 * arr2　　B. numpy.multiply(arr1, arr2)

C. numpy.dot(arr1, arr2)　　D. numpy.cross(arr1, arr2)

153. 在 numpy 中，（　　）对数组进行排序操作。

A. arr. sort()　　B. numpy.sort(arr)

C. numpy.sort(arr, axis=0)　　D. arr. sorted()

154. 以下关于 Python 语言中“缩进”说法正确的是（　　）。

A. 缩进是程序中长度统一且强制使用。

B. 缩进是非强制的，仅为了提高代码可读性。

C. 缩进可以用在任何语句之后，表示语句间的包含关系。

D. 缩进统一为四个空格。

155. 给定字符串 S，以下（　　）表示 S 从右侧向左第三个字符。

A. S[3]　　B. S[-3]　　C. S[:-3]　　D. S[0:-3]

156. 以下（　　）不是 Python 合法命名。

A. _MyGod_　　B. 5MyGod　　C. MyGod5　　D. MyGod

157. 在 Python 中，（　　）是用于获取用户输入的函数。

A. eval()　　B. input()　　C. print()　　D. get()

158. 下列（　　）不属于 Python 保留字。

A. elif　　B. type　　C. import　　D. def

159. 以下（　　）不是 Python 数据类型。

A. 整数　　B. 字符串　　C. 列表　　D. 实数

160. 下列（　　）选项给出的保留字不直接用于表示分支结构。

A. elif　　B. if　　C. else　　D. in

161. 利用 print()格式化输出，（　　）选项用于控制浮点数的小数点后两位输出。

A. {.2}　　B. {:.2}　　C. {.2f}　　D. {:.2f}

162. Python 是（　　）类语言。

A. 编译型语言　　B. 解释型语言　　C. 汇编语言　　D. 机器语言

163. 下列（　　）关键字用于定义函数？

A. def　　B. function　　C. define　　D. func

164. 在 Python 中，（　　）获取列表的长度。

A. length(list)　　B. list.size()　　C. len(list)　　D. sizeOf(list)

165. 下列（　　）选项用于向列表末尾添加元素。

A. list.add()　　B. list.append()　　C. list.insert()　　D. list.extend()

166. Python 中用于条件判断的关键字是（　　）。

A. check　　B. switch　　C. if　　D. select

167. 下列（　　）运算符用于整数除法。

A. /　　B. //　　C. %　　D. *

168. 通过（　　）语句，在 Python 中打开一个文件。

A. open(file, 'r')　　B. read(file)　　C. file.open('r')　　D. open('r', file)

169. 在 Python 中，通过（　　）语句创建一个空集合。

A. new set()　　B. {}　　C. set()　　D. emptySet()

170. 下列（　　）函数用于从键盘读取用户输入？

A. read_input()　　B. input()　　C. get_input()　　D. user_input()

171. 在 Python 中，（　　）语句引用第二个元素。

A. list[1]　　B. list[2]　　C. list(2)　　D. list.second()

172. 下列（　　）模块用于绘制图表？

A. drawlib　　B. plot　　C. graph　　D. matplotlib

173. 下列（　　）语句在 Python 中处理异常。

A. try/except　　B. if/else　　C. handle/exception　　D. catch/throw

174. 以下关于函数参数的描述，正确的是：（　　）。

A. 调用函数时，按参数名称传递的参数，要按照定义顺序进行传递。

B. 定义函数可选参数的时候，不限制可选参数在参数列表中的位置。

C. 函数在定义时可以不指定可选参数的默认值，调用函数的时候再传入参数。

D. 在一个函数内部定义的变量，到另一个函数中不能引用。

175. 以下程序的输出结果是：（　　）。

```
def add_Run(L=None):
    if L is None:
        L = []
    L.append('Run')
    return L
add_Run()
add_Run()
print(add_Run(['Lying']))
```

A. ['Lying', 'Run', 'Run']　　B. ['Run']['Run']['Lying', 'Run']

C. ['Lying']　　D. ['Lying', 'Run']

176. 以下程序的输出结果是：（　　）。

```
L = []
x = 3
def pri_val(x):
L.append(x)
x = 5
pri_val(x)
print('L = {}, x = {}'.format(L, x))
```

A. L = [5], x = 5　　B. L = [3], x = 5

C. L = 3, x = 5　　D. L = 3, x = 3

177. 关于组合数据类型的描述，正确的选项是（　　）。

A. Python 中最常用的映射类型的典型代表是字典类型。

B. 列表类型里的元素要求是同一种数据类型。

C. 元组采用大括号方式表示。

D. 序列类型的元素可以用 reverse()方法交换相邻元素的位置。

178. 以下程序的输出结果是（　　）。

```
l = [1,2,3,4,5,6,7]
print(l[3:2])
print(l[-5:-3])
```

A. [] [3, 4]　　B. [3, 4] []

C. [3, 4] [3, 4]　　D. [] []

179. 以下关于程序设计语言的描述，正确的选项是（　　）。

A. Python 语言是一种面向过程，也是面向对象的语言。

B. Python 语言的生态库都是官方开发的。

C. Python 语言是网络通用语言。

D. Python 语言与平台相关。

180. 以下关于 Python 程序的基本语法元素，错误的描述是（　　）。

A. Python 语言只能用 4 个空格的缩进来实现程序的强制可读性。

B. 变量是由用户定义的用来保存和表示数据的一种语法元素。

C. 变量的命名规则之一是名字的首位不能是数字。

D. 变量标识符是一个字符串，长度是没有限制的。

181. 关于 Python 语言的注释语句的描述，正确的是（　　）。

A. #注释符可以注释多行。

B. 以#开头的语句是注释。

C. #之后的语句被解释器解释，但不执行。

D. '''开头的语句也表示注释，用法跟#一样。

182. 关于以下程序输出结果的描述，正确的选项是（　　）。

```
l = [1,2,3,4,5,6,7]
print(l.pop(0), len(l))
```

A. 1 6　　B. 1 7　　C. 0 7　　D. 0 6

183. 以下程序的输出结果，可能的选项是（　　）。

```
ds = {'av':2,'vr':4,'ls':9,'path':6}
print(ds.popitem(), len(ds))
```

A. ('path', 6) 3　　B. ('av', 2) 4

C. ('path', 6) 4　　D. ('vr', 2) 3

184. 执行下述程序的输出结果是（　　）。

```
ds = {'eng':2,'math':6,'comp':9,'PE':4}
print(ds.pop(max(ds.keys()), 0))
```

A. 6　　B. math　　C. PE　　D. 4

185. 关于打开文件函数 open(<文件路径名>,<打开模式>)的打开模式，正确的是（　　）。

A. 'r'表示只读模式打开文件，如果文件不存在，就会返回异常。

B. 'w'表示写模式打开文件，如果文件存在，就会在文件尾继续写删除文件已有内容。

C. 'a'表示追加模式打开文件，如果文件不存在，就返回异常。如果文件不存在，就创建一个文件，如果存在就在文件尾追加内容。

D. 'b'表示二进制文件模式打开文件，可以单独作为 open 函数的参数。

186. 设 a.txt 的内容是：a,b,c,d。以下程序执行结果是（　　）。

```
with open('a.txt','r') as f:
print(f.read().split(','))
```

A. ['a', 'b', 'c', 'd']　　B. [a, b, c, d]

C. 'a', 'b', 'c', 'd'　　D. a, b, c, d

187. 在程序之间交换数据，常使用的第三方库是（　　）。

A. PyGame　　B. Pandas　　C. json　　D. Flask

188. 文件 data.csv 里的内容如下：

1,三轴机,17,5
2,壳体热套,10,2
3,泵体安装,19,3
关于以下程序在屏幕上输出结果的描述，正确的选择是（　　）。
```
with open('data.csv','r') as f:
print(f.readlines())
```
A. 输出一行列表，里面包括三个字符串元素
B. 输出三行列表，每行列表里面有一个字符串元素
C. 输出一行列表，里面包括一个字符串元素
D. 输出三行字符串

189. 执行以下代码，output.txt 文件中的内容是（　　）。
```
aaa =[8, 5, 2, 2]
with open('output.txt', 'w') as f:
    for aa in aaa:
        f.write(';'.join(str(aa)))
```
A. 8,5,2,2　　B. 8522　　C. 8;5;2;2　　D. 8 5 2 2

190. 可用来获取网页内容的 Python 第三方库是（　　）。
A. OpenCV　　B. Matplotlib　　C. requests　　D. SciPy

191. 以下属于 Python 标准时间库的选项是（　　）。
A. calender　　B. time　　C. datetime　　D. logging

192. 下述（　　）不属于数据库设计的内容。
A. 数据库物理结构　　B. 数据库管理系统
C. 数据库概念结构　　D. 数据库逻辑结构

193. 以下关于 turtle 库的描述，错误的是（　　）。
A. 用 import turtle 语句之后，用 turtle.circle()函数画图
B. turtle.sethead()函数的别名是 turtle.seth()
C. 用 circle()函数只能画圆，不能画一个圆弧
D. 用 import turtle as t 语句之后，用 t.circle()函数画图

194. 将 E-R 图转换成关系模式时，实体与联系都可以表示成（　　）。
A. 属性　　B. 域　　C. 关系　　D. 键

195. 关于基本输入输出函数的描述，错误的选项是（　　）。
A. print()函数的参数可以是一个函数，执行结果是显示函数返回的值。
B. eval()函数的参数是“3*4”的时候，返回的值是整数“12”。
C. 当用户输入一个整数“6”的时候，input()函数返回的也是整数“6”。
D. 当 print()函数输出多个变量的时候，可以用逗号分隔多个变量名。

196. 关于 Python 数据类型的描述，正确的选项是（　　）。
A. 内置函数 divmod(x,y)的运算结果是两个整数：x 除 y 的整数商以及余数。
B. 函数 ord(x)是返回字符串 x 对应的 Unicode 编码。
C. 运算符+、 -、 *、 / 等跟赋值符号 = 相连，形成增强赋值操作符。

D. 函数 lower(x)是将字符串 x 的首字母小写。

197. 关于<模板字符串>.format(<逗号分隔的参数>)中的内容描述，正确的是（　　）。

A. 格式控制信息{2:0>7f}里面的‘f’表示这个位置是一个整数。

B. 模板字符串里的格式控制信息语法格式是：{<参数序号>:<格式控制标记>}，参数序号是从 1 开始。

C. 格式控制信息{:*^10}表示这个位置是最大长度为 10 的整数。

D. 逗号分隔的参数可以是变量，也可以是函数。

198. 以下关于控制结构的描述，错误的是（　　）。

A. if 条件满足情况下要执行的语句块，要放在 if 语句后面，并缩进。

B. if 条件不满足情况下要执行的语句块，放在 else 语句后面。

C. 分支结构中的判断条件，只能是产生 True 或 False 的表达式或函数。

D. 语句 if 1，这种表达式是可以执行的。

199. 假设：x = 'ab'；xy ='ab93kdfd'；k = 0。下列代码选项中，使得 k 的值是 0 的选项是：

A. if xy.count(x) >=1 : k = 1

B. if xy > x : k = 1

C. if x in xy : k = 1

D. if xy in x : k = 1

200. 关于 Python 语言的特点， 以下选项中描述错误的是（　　）。

A. Python 语言是脚本语言　　B. Python 语言是多模型语言

C. Python 语言是跨平台语言　　D. Python 语言是非开源语言

201. 以下选项中， 不是 Python IDE 的是（　　）。

A. PyCharm　　B. Jupyter Notebook　　C. R studio　　D. Spyder

202. 下列选项中可以获取 Python 整数类型帮助的是（　　）。

A. >>> help(int)　　B. >>> dir(str)

C. >>> help(float)　　D. >>> dir(int)

203. 关于 Python 语言的注释， 以下选项中描述错误的是（　　）。

A. Python 语言有两种注释方式： 单行注释和多行注释

B. Python 语言的多行注释以''' （三个单引号） 开头和结尾

C. Python 语言的单行注释以单引号 ' 开头

D. Python 语言的单行注释以#开头

204. 以下选项中， 不是 Python 语言合法命名的是（　　）。

A. MyGod5　　B. 5MyGod　　C. MyGod　　D. _MyGod_

205. 在一行上写多条 Python 语句使用的符号是（　　）。

A. 分号　　B. 点号　　C. 逗号　　D. 冒号

206. 关于赋值语句， 以下选项中描述错误的是（　　）。

A. 在 Python 语言中，"="表示赋值，即将"="右侧的计算结果赋值给左侧变量，包含"="的语句称为赋值语句。

B. 设 a = 10; b = 20，执行"a, b = a, a + b; print(a, b) " 和 "a = b;b = a + b; print(a, b) "之后，得到同样的输出结果：10 30

C. 设 x = "alice"; y = "kate" ，执行"x, y = y, x"可以实现变量 x 和 y 值的互换。

D. 在 Python 语言中，有一种赋值语句，可以同时给多个变量赋值。

207. 关于 Python 赋值语句，以下选项中不合法的是（ ）。

A. x, y=y, x　　B. x=1; y=1　　C. x=(y=1)　　D. x=y=1

208. 下面代码的输出结果是（ ）。

```
>>> a = b = c =123
>>> print(a, b, c)
```

A. 0 0 123　　B. 出错　　C. 123 123 123　　D. 1 1 123

209. 语句 x, y=eval(input()) 执行时， 输入数据格式错误的是（ ）。

A. 3 4　　B. （3, 4）　　C. 3, 4　　D. [3, 4]

210. 利用 print()格式化输出，能够控制浮点数的小数点后两位输出的是（ ）。

A. {. 2}　　B. {: . 2f}　　C. {. 2f}　　D. {: . 2}

211. type(1+0xf*3. 14)的结果是（ ）。

A. <class 'int' >　　B. <class 'long' >

C. <class 'str' >　　D. <class 'float' >

212. 在一个同时包含整数和浮点数的表达式中，Python 要进行的转换是（ ）。

A. 浮点数转换为整数　　B. 整数转换为浮点数

C. 浮点数和整数转换为字符串　　D. 整数转换为字符串

213. 整形变量 x 中存放了一个两位数， 要将这个两位数的个位数的个位数字和十位数字交换位置， 例如， 13 变成 31，正确的 Python 表达式是（ ）。

A. (x%10) *10+x//10　　B. (x%10) //10+x//10

C. (x/10) %10+x//10　　D. (x%10) *10+x%10

214. 给出如下代码：

```
>>> x = 3. 14
>>> eval('x + 10')
```

上述代码的输出结果是（ ）。

A. 系统报错　　B. TypeError: must be str, not int

C. 3.1410　　D. 13.14

215. 以下选项中， 不是 Python 数据类型的是（ ）。

A. 实数　　B. 列表　　C. 字符串　　D. 整数

216. 下面代码的输出结果是（ ）。

```
a = 5/3+5//3
print(a)
```

A. 2.666666666666667　　B. 14　　C. 5.4　　D. 3.333333

217. 下面代码的输出结果是（ ）。

```
>>> True / False
```

A. True　　B. 系统报错　　C. 0　　D. -1

218. 下面代码的输出结果是（　　）。

```
x = 1
x *= 3+5**2
print(x)
```

A. 28　　B. 14　　C. 13　　D. 29

219. 下面代码的输出结果是（　　）。

```
x=10
y=3
print(divmod(x, y) )
```

A. (3, 1)　　B. 1, 3　　C. 3, 1　　D. (1, 3)

220. 下面代码的输出结果是（　　）。

```
x=3.1415926
print(round(x,2), round(x) )
```

A. 3.14 3　　B. 6.28 3　　C. 2 2　　D. 3 3.14

221. 下面代码的输出结果是（　　）。

```
x = 12.34; print(type(x) )
```

A. <class 'complex'>　　B. <class 'bool'>

C. <class 'float'>　　D. <class 'int'>

222. 下面代码的输出结果是（　　）。

```
x=10
y=-1+2j
print(x+y)
```

A. (9+2j)　　B. 11　　C. 2j　　D. 9

223. 下面代码的执行结果是（　　）。

```
a = 10.99
print(complex(a) )
```

A. (10.99+0j)　　B. 0.99　　C. 10.99　　D. 10.99+0j

224. 下面代码的输出结果是（　　）。

```
z = 12.12 + 34j
print(z.real)
```

A. 12.12　　B. 34.0　　C. 12　　D. 34

225. 下列选项中输出结果是 True 的是（　　）。

A. >>> isinstance(255,int)　　B. >>> chr(10).isnumeric()

C. >>> "Python".islower()　　D. >>> chr(13).isprintable()

226. 下列属于 math 库中的数学函数的是（　　）。

A. time()　　B. round()　　C. sqrt()　　D. random()

227. 下列函数中， 不属于基本的 Python 内置函数是（　　）。

A. hex()　　B. close()　　C. sum()　　D. exec()

228. 关于 eval 函数， 以下选项中描述错误的是（　　）。

A. eval 函数的定义为： eval(source, globals=None, locals=None, /)。

B. 执行">>> eval("Hello") "和执行">>> eval("'Hello' ") "得到相同的结果。

C. 如果用户希望输入一个数字， 并用程序对这个数字进行计算，可以采用 eval(input(<输入提示字符串>)) 组合。

D. eval 函数的作用是将输入的字符串转为 Python 语句， 并执行该语句。

229. 给出如下代码：

```
TempStr ="Hello World"
```

可以输出"World"子串的是（ ）。

A. print(TempStr[-5:])　　B. print(TempStr[-4: -1])

C. print(TempStr[-5: 0])　　D. print(TempStr[-5: -1])

230. 给出如下代码

```
s = 'Python is beautiful!'
```

可以输出"python"的是（ ）。

A. print(s[0: 7])　　B. print(s[: -14])

C. print(s[-21: -14].lower)　　D. print(s[0: 6].lower())

231. 给出如下代码

```
s = 'Python is Open Source!'
print(s[0: ].upper() )
```

上述代码的输出结果是（ ）。

A. PYTHON IS OPEN SOURCE!　　B. PYTHON IS OPEN SOURCE

C. Python is Open Source!　　D. PYTHON

232. Python 语句 print(r"\nGood") 的运行结果是（ ）。

A. 新行和字符串 Good　　B. r"\nGood"

C. \nGood　　D. 字符 r、新行和字符串 Good

233. 字符串 s='a\nb\tc'， 则 len(s) 的值是（ ）。

A. 7　　B. 6　　C. 5　　D. 4

234. 以下选项中可访问字符串 s 从右侧向左第三个字符的是（ ）。

A. s[3]　　B. s[: -3]　　C. s[0: -3]　　D. s[-3]

235. 下面代码的输出结果是（ ）。

```
a = "alex"
b = a.capitalize()
print(a, end=", ")
print(b)
```

A. alex, Alex　　B. ALEX, alex　　C. alex, ALEX　　D. Alex, Alex

236. 下面代码的输出结果是（ ）。

```
a = "ac"
b = "bd"
c = a + b
print(c)
```

A. acbd　　B. dbac　　C. bdac　　D. abcd

237. 以下选项中， 输出结果为 False 的是（　　）。

A. >>> 'python123' > 'python'

B. >>> 'ABCD' == 'abcd'.upper()

C. >>> ' ' < 'a'

D. >>> 'python' < 'pypi'

238. 下面代码的输出结果是（　　）。

```
str1 = "mysqlsqlserverPostgresQL"
str2 = "sql"
ncount = str1.count(str2, 10)
print(ncount)
```

A. 3　　B. 0　　C. 4　　D. 2

239. 给出如下代码

```
s= "abcdefghijklmn"
print(s[1: 10: 3])
```

上述代码的输出结果是（　　）。

A. beh　　B. behk　　C. adg　　D. adgj

240. 下面代码的输出结果是（　　）。

```
s1 = "The python language is a scripting language. "
s2 = s1.replace('scripting' , 'general' )
print(s2)
```

A. The python language is a scripting language.

B. 系统报错

C. ['The', 'python', 'language', 'is', 'a', 'scripting', 'language.']

D. The python language is a general language.

241. 下列表达式中， 有 3 个表达式的值相同， 另一个不相同， 与其他 3 个表达式不同的是（　　）。

A. "ABC"+"DEF"　　B. "".join(("ABC", "DEF"))

C. "ABC"-"DEF"　　D. 'ABCDEF' *1

242. 给出如下代码

```
s = "Alice"
print(s[:: -1])
```

上述代码的输出结果是（　　）。

A. ALICE　　B. ecilA　　C. Alic　　D. Alice

243. 下面代码的输出结果是（　　）。

```
s = "The python language is a cross platform language. "
print(s.find('language' , 30) )
```

A. 11　　B. 系统报错　　C. 10　　D. 40

244. 下面代码的输出结果是（　　）。

```
>>> a, b, c, d, e, f = 'Python'
>>> b
```

A. 'y'　　B. 出错　　C. 1　　D. 0

245. 设 s="Happy New Year"， 则 s[3: 8]的值为（　　）。

A. 'ppy Ne'　　B. 'py Ne'　　C. 'ppy N'　　D. 'py New'

246. 设 s="Python Programming"， 那么 print(s[-5:]) 的结果是（　　）。

A. mming　　B. Python　　C. mmin　　D. Pytho

247. 执行下列语句后的显示结果是（　　）。

```
world="word"
print("hello"+world)
```

A. helloword　　B. "hello"world　　C. hello world　　D. "hello"+world

248. 语句 print('x=${: 7.2f}'.format(123.5678))，执行后的输出结果（选项的□代表空格）是（　　）。

A. x=123.56　　B. $123.57　　C. x=$□123.57　　D. x=$□123.56

249. 下面代码的输出结果是（　　）。

```
a ="Python"
b = "A Superlanguage"
print("{:->10} : {:-<19}".format(a,b) )
```

A. ----Python: A Superlanguage----

B. ----Python: ----A Superlanguage

C. The python language is a multimodel language.

D. Python----: ----A Superlanguage

250. print('{:7.2f} {:2d}'.format(101/7,101%8))的运行结果是（　　）。

A. {: 7.2f} {: 2d}　　B. □□14.43□5(□代表空格)

C. □14.43□□5（□代表空格）　　D. □□101/7□101%8（□代表空格）

251. 下列 Python 保留字中，不用于表示分支结构的是（　　）。

A. if　　B. in　　C. else　　D. elif

252. 关于 Python 程序中与“缩进”有关的说法中， 以下选项中正确的是（　　）。

A. 缩进统一为 4 个空格。

B. 缩进在程序中长度统一且强制使用。

C. 缩进可以用在任何语句之后，表示语句间的包含关系。

D. 缩进是非强制性的，仅为了提高代码可读性。

253. 下面代码的输出结果是（　　）。

```
a = 2
b = 2
c = 2.0
print(a == b, a is b, a is c)
```

A. True False True　　B. True False False

C. True True False　　D. False False True

254. 以下选项中描述正确的是（　　）。

A. 条件 35<=45<75 是合法的，且输出为 False。

B. 条件 24<=28<25 是合法的，且输出为 True。

C. 条件 24<=28<25 是不合法的。

D. 条件 24<=28<25 是合法的，且输出为 False。

255. 下面代码的输出结果是（　　）。

```
a = 1.0
if isinstance(a,int) :
    print("{} is int".format(a))
else:
    print("{} is not int".format(a))
```

A. 1.0 is not int　　B. 1.0 is int　　C. 无输出　　D. 出错

256. 执行下列 Python 语句后的显示结果是（　　）。

```
x=2
y=2.0
if(x==y) : print("Equal")
else: print("Not Equal")
```

A. Equal　　B. Not Equal　　C. 编译错误　　D. 运行时错误

257. 下列表式的值为 True 的是（　　）。

A. 2!=5 or 0　　B. 3>2>2　　C. 5+4j>2-3j　　D.'abc'>'xyz'

258. 关于 Python 的分支结构， 以下选项中描述错误的是（　　）。

A. 分支结构可以向已经执行过的语句部分跳转。

B. Python 中 if-elif-else 语句描述多分支结构。

C. Python 中 if-else 语句用来形成二分支结构。

D. 分支结构使用 if 保留字。

259. 执行下列 Python 语句后的显示结果是（　　）。

```
i=1
if(i): print(True)
else: print(False)
```

A. 输出 1　　B. 输出 True　　C. 输出 False　　D. 输出 0

260. 以下不合法的布尔表达式是（　　）。

A. x in[1, 2, 3, 4, 5]　　B. 3=a

C. e>5 and 4==f　　D. (x-6) >5

261. 给出下面代码：

```
a = input("").split(", ")
if isinstance(a, list) :
    print("{} is list".format(a))
else:
    print("{} is not list". format(a) )
```

代码执行时， 从键盘获得 1, 2, 3， 则代码的输出结果是：（　　）。

A. 1, 2, 3 is list　　B. ['1','2','3'] is list

C. 执行代码出错　　D. 1, 2, 3 is not list

262. 下面 if 语句统计"成绩（mark）优秀的男生以及不及格的男生"的人数， 正确的语句为（　　）。

A. if(gender=="男" and mark<60 or mark>=90) : n+=1

B. if(gender=="男" and mark<60 and mark>=90) : n+=1

C. if(gender=="男" and (mark<60 or mark>=90)) : n+=1

D. if(gender=="男" or (mark<60 or mark>=90)) : n+=1

263. 在 Python 中， 逻辑量有（　　）。

A. Yes, No　　B. True, false　　C. T, F　　D. 1, 0

264. 给出如下代码：

```
sum = 0
for i in range(1, 11) :
    sum += i
print(sum)
```

以下选项中描述正确的是：（　　）。

A. 循环内语句块执行了 11 次。

B. sum += i 可以写为 sum =+ i。

C. 输出的最后一个数字是 55。

D. 如果 print(sum)语句完全右对齐，输出结果不变。

265. 下面代码的输出结果是（　　）。

```
x2 = 1
for i in range(4, 0, -1) :
    x1 = (x2 + 1) * 2
    x2 = x1
print(x1)
```

A. 46　　B. 23　　C. 190　　D. 94

266. 给出下面代码：

```
for i in range(1, 10) :
    for j in range(1, i+1) :
        print("{} *{} ={} \t". format(j, i, i*j) , end = ' ' )
    print("")
```

以下选项中描述错误的是：（　　）。

A. 执行代码，输出九九乘法表。

B. 内层循环 i 用于控制一共打印 9 行。

C. 执行代码出错。

D. 也可使用 while 嵌套循环实现打印九九乘法表。

267. 关于 break 语句与 continue 语句的说法中， 以下选项中不正确的是（　　）。

A. 当多个循环语句嵌套时，break 语句只适用于最里层的语句。

B. break 语句结束循环，继续执行循环语句的后续语句。

C. continue 语句结束循环，继续执行循环语句的后续语句。

D. continue 语句类似于 break 语句，也必须在 for、while 循环中使用。

268. 关于 Python 遍历循环，以下选项中描述错误的是（　　）。

A. 遍历循环通过 for 实现。

B. 无限循环无法实现遍历循环的功能。

C. 遍历循环可以理解为从遍历结构中逐一提取元素，放在循环变量中，对于所提取的每个元素只执行一次语句块。

D. 遍历循环中的遍历结构可以是字符串、文件、组合数据类型和 range()函数等。

269. 以下 for 语句中， 不能完成 1～10 的累加功能的是（　　）。

A. for i in range(10, 0) : sum+=i。

B. for i in range(1, 11) : sum+=i。

C. for i in range(10, 0, -1) : sum+=i。

D. for i in (10, 9, 8, 7, 6, 5, 4, 3, 2, 1) : sum+=i。

270. 下列循环语句中有语法错误的是（　　）。

A. while(x==y) : 5　　B. while(0) : pass

C. for i in [1, 2, 3]: print(i)　　D. for True: x=30

271. 下面代码的输出结果是（　　）。

```
for a in 'mirror' :
    print(a, end="")
    if a == 'r' :
        break
```

A. mir　　B. mirror　　C. mi　　D. mirro

272. 下面代码的输出结果是（　　）。

```
for s in "HelloWorld":
    if s=="W":
        continue
    print(s, end="")
```

A. Helloorld　　B. HelloWorld　　C. World　　D. Hello

273. 下面代码的输出结果是（　　）。

```
for i in range(1, 10, 2) :
    print(i, end=", ")
```

A. 1, 3, 5, 7, 9,　　B. 1, 4, 7,　　C. 1, 4,　　D. 1, 3,

274. 下面代码的输出结果是（　　）。

```
for i in range(1, 6) :
    if i%3 == 0:
        break
    else:
```

```
        print(i, end =", ")
```

A. 1, 2,　　B. 1, 2, 3, 4, 5, 6

C. 1, 2, 3, 4, 5,　　D. 1, 2, 3,

275. 下面代码的输出结果是（　　）。

```
a = 2.0
b = 1.0
s = 0
for n in range(1,4):
    s += a / b
    t = a
    a = a + b
    b = t
print(round(s, 2) )
```

A. 5.17　　B. 8.39　　C. 6.77　　D. 3.5

276. 以下选项中能够实现 Python 循环结构的是（　　）。

A. loop　　B. do…for　　C. if　　D. while

277. 给出下面代码：

```
age=23
start=2
if age%2!=0:
    start=1
for x in range(start, age+2, 2) :
    print(x)
```

上述程序输出值的个数是：（　　）。

A. 10　　B. 16　　C. 14　　D. 12

278. 下面代码的输出结果是（　　）。

```
for num in range(2, 10) :
    if num > 1:
        for i in range(2, num) :
            if (num % i) == 0:
                break
            else:
                print(num,end=',')
```

A. 2, 4, 6, 8, 10,　　B. 4, 6, 8, 9,

C. 3,5,5,5,7,7,7,7,7,9,　　D. 2, 4, 6, 8,

279. 下面代码的输出结果是（　　）。

```
s = 0
while(s<=1) :
    print(' 计数： ' , s)
```

```
    s = s + 1
```
A. 计数： 0　　B. 出错

C. 计数： 1　　D. 计数： 0　计数： 1

280. 下面代码的输出结果是（　　）。

```
sum = 0
for i in range(2, 101) :
    if i % 2 == 0:
        sum += i
    else:
        sum -= i
print(sum)
```

A. 51　　B. -50　　C. 49　　D. 50

281. 下面代码的输出结果是（　　）。

```
for i in "Python":print(i, end=" ")
```

A. P y t h o n　　B. P, y, t, h, o, n,

C. P y thon　　D. Python

282. 给出如下代码：

```
a=3
while a > 0:
    a -= 1
    print(a, end=" ")
```

以下选项中描述错误的是：（　　）。

A. a -= 1 可由 a = a - 1 实现。

B. 使用 while 保留字可创建无限循环。

C. 条件 a > 0 如果修改为 a < 0 程序执行会进入死循环。

D. 这段代码的输出内容为 2 1 0。

283. 设有程序段

```
k=10
while(k) :
    k=k-1
```

则下面描述中正确的是（　　）。

A. while 循环执行 10 次　　B. 循环是无限循环

C. 循环体语句一次也不执行　　D. 循环体语句执行一次

284. 基本的 Python 内置函数 range(a, b, s) 的作用是（　　）。

A. 返回组合类型的逆序迭代形式。

B. 返回 a 的四舍五入值，b 表示保留小数的位数。

C. 返回 a 的 b 次幂。

D. 产生一个整数序列，从 a 到 b（不含） 以 s 为步长。

285. 生成一个[0.0, 1.0) 之间的随机小数的函数是（　　）。

A. random.randint(0.0,1.0)　　B. random.random()
C. random.randrange(0.0,1.0)　　D. random.uniform(0.0, 1.0)

286. random.uniform(a,b) 的作用是（　　）。
A. 生成一个[a, b]之间的随机小数。
B. 生成一个[a, b) 之间以 1 为步数的随机整数。
C. 生成一个[a, b]之间的随机整数。
D. 生成一个[0. 0, 1. 0) 之间的随机小数。

287. 生成一个[10, 99]之间的随机整数的函数是（　　）。
A. random.randint(10,99)　　B. random.random()
C. random.randrange(10,99,2)　　D. random.uniform(10,99)

288. random 库的 seed(a) 函数的作用是（　　）。
A. 生成一个[0.0, 1.0)之间的随机小数 B. 生成一个 k 比特长度的随机整数
C. 生成一个随机整数　　D. 设置初始化随机数种子 a

289. 生成一个 k 比特长度的随机整数的函数是（　　）。
A. random.choice(k)　　B. random.shuffle(k)
C. random.getrandbits(k)　　D. random.sample(pop,k)

290. random 库的 random.randrange(start,stop[,step]) 函数的作用是（　　）。
A. 生成一个[start, stop)之间的随机小数。
B. 将序列类型中元素随机排列，返回打乱后的序列。
C. 生成一个[start, stop)之间以 step 为步数的随机整数。
D. 从序列类型（例如列表）中随机返回一个元素。

291. random 库的 random.sample(pop,k)函数的作用是（　　）。
A. 从 pop 类型中随机选取 k-1 个元素，以列表类型返回。
B. 从 pop 类型中随机选取 k 个元素，以列表类型返回。
C. 生成一个随机整数。
D. 随机返回一个元素。

292. 给出如下代码：

```
import random
num = random. randint(1, 10)
while True:
    guess = input()
    i = int(guess)
    if i == num:
        print("你猜对了")
        break
    elif i < num:
        print("小了")
    elif i > num:
        print("大了")
```

以下选项中描述错误的是（　　）。

A. random.randint(1,10)生成[1,10]之间的整数。

B. "while True: "创建了一个永远执行的 While 循环。

C. "import random"这行代码是可以省略的。

D. 这段代码实现了简单的猜数字游戏。

293. random.uniform(a,b) 的作用是（　　）。

A. 生成一个[a,b]之间的随机整数。

B. 生成一个(a,b)之间的随机数。

C. 生成一个均值为 a，方差为 b 的正态分布。

D. 生成一个[a,b]之间的随机小数。

294. 下面代码的输出结果是（　　）。

```
for a in ["torch", "soap", "bath"]:
    print(a,end=" ")
```

A. torch soap bath

B. torch, soap, bath

C. torch, soap, bath,

D. torch

295. 下面代码的输出结果是（　　）。

```
a = []
for i in range(2, 10) :
    count = 0
    for x in range(2, i-1) :
        if i % x == 0:
            count += 1
    if count != 0:
        a. append(i)
print(a)
```

A. [3 , 5 , 7 , 9]　　B. [4, 6, 8, 9]

C. [2 , 3 , 5 , 7]　　D. [4 , 6 , 8 , 9 , 10]

296. 对下列语句不符合语法要求的表达式是（　　）。

```
for var in ___ :
    print(var)
```

A. range(0, 10)　　B. 1 2 3　　C. (1, 2, 3)　　D. {1, 2, 3, 4, 5}

297. 下面代码的输出结果是（　　）。

```
a = [1, 2, 3]
if isinstance(a, float) :
    print("{} is float".format(a))
else:
    print("{} is not float".format(a))
```

A. 出错　　B. a is float
C. [1, 2, 3] is not float　　D. a is <class 'float t' >

298. 给出下面代码：

```
a = input("") . split(", ")
x = 0
while x < len(a) :
    print(a[x], end="")
    x += 1
```

代码执行时， 从键盘获得 a, b, c, d， 则代码的输出结果是：（　　）。

A. a, b, c, d　　B. 无输出　　C. 执行代码出错　　D. abcd

299. 下面代码的输出结果是（　　）。

```
for i in ["pop star"]:
    pass
print(i, end = "")
```

A. pop star　　B. popstar　　C. 无输出　　D. 出错

300. 下面代码的输出结果是（　　）。

```
list1 = [m+n for m in 'AB' for n in 'CD' ]
print(list1)
```

A. ['AC', 'AD', 'BC', 'BD']　　B. ABCD
C. 错误　　D. AABBCCDD

301. 下面代码的输出结果是（　　）。

```
vlist = list(range(5) )
print(vlist)
```

A. [0, 1, 2, 3, 4]　　B. 0; 1; 2; 3; 4;
C. 0, 1, 2, 3, 4,　　D. 0 1 2 3 4

302. 下面代码的输出结果是（　　）。

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

A. [6, 9, 5, 4]　　B. [9, 6, 5, 4]
C. [5, 4, 6, 9]　　D. [4, 5, 9, 6]

303. 下面代码的输出结果是（　　）。

```
s =["seashell", "gold", "pink", "brown", "purple", "tomato"]
print(len(s) , min(s) , max(s) )
```

A. 6 seashell gold　　B. 5 purple tomato
C. 5 pink brown　　D. 6 brown tomato

304. 下面代码的输出结果是（　　）。

```
list1 = []
for i in range(1, 11) :
    list1.append(i**2)
print(list1)
```

A. [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

B. ----Python: ----A Superlanguage

C. 错误

D. [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

305. 下面代码的输出结果是（　　）。

```
L = [1, 2, 3, 4, 5]
s1 = ',' . join(str(n) for n in L)
print(s1)
```

A. [1, 2, 3, 4, 5]　　B. 1, , 2, , 3, , 4, , 5

C. [1, , 2, , 3, , 4, , 5]　　D. 1,2,3,4,5

306. 关于 Python 的列表， 以下选项中描述错误的是（　　）。

A. Python 列表是一个可以修改数据项的序列类型。

B. Python 列表用中括号[]表示。

C. Python 列表的长度不可变。

D. Python 列表是包含 0 个或者多个对象引用的有序序列。

307. 关于 Python 序列类型的通用操作符和函数，以下选项中描述错误的是（　　）。

A. 如果 s 是一个序列，x 是 s 的元素，x in s 返回 True。

B. 如果 s 是一个序列，s =[1, "kate", True]，s[-1] 返回 True。

C. 如果 s 是一个序列，s =[1, "kate", True]，s[3] 返回 True。

D. 如果 s 是一个序列，x 不是 s 的元素，x not in s 返回 True。

308. 序列 s，返回 s 中从 i 到 j 步长为 k 的元素子序列，表达式是（　　）。

A. s[i,j,k]　　B. s(i,j,k)　　C. s[i:j:k]　　D. s[i;j;k]

309. 下面代码的输出结果是（　　）。

```
li = ["hello", 'se', [["m","n"],["h","kelly"],'all'], 123, 446]
print(li[2][1][1])
```

A. h　　B. m　　C. n　　D. kelly

310. 关于 Python 的元组类型，以下选项中描述错误的是（　　）。

A. 元组一旦创建就不能被修改。

B. 一个元组可以作为另一个元组的元素，可以采用多级索引获取信息。

C. 元组中元素不可以是不同类型。

D. Python 中元组采用逗号和圆括号（可选）来表示。

311. 下面代码的输出结果是（　　）。

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
print(list1+list2)
```

A. [5, 7, 9]　　B. [4, 5, 6]　　C. [1, 2, 3, 4, 5, 6]　　D. [1, 2, 3]

312. 下面代码的输出结果是（　　）。

```
list1 = [(m,n) for m in 'ABC' for n in 'ABC' if m!=n]
print(list1)
```

A. ['AC','AD','BC','BD']

B. [('A', 'B'), ('A', 'C'), ('B', 'A'), ('B', 'C'), ('C', 'A'), ('C', 'B')]

C. 错误

D. [('A','C'),('A','D'),('B','C'),('B','D')]

313. 下面代码的输出结果是（　　）。

```
s =["seashell", "gold", "pink", "brown", "purple", "tomato"]
print(s[4: ])
```

A. ['purple', 'tomato']

B. ['gold', 'pink','brown','purple','tomato']

C. ['seashell','gold','pink','brown']

D. ['purple']

314. 下面代码的输出结果是（　　）。

```
a = []
for i in range(2, 10) :
    count = 0
    for x in range(2, i-1) :
        if i % x == 0:
            count += 1
    if count == 0:
        a.append(i)
print(a)
```

A. [3 , 5 , 7 , 9]　　B. [4, 6 , 8 , 9 , 10]

C. [2, 3, 5, 7]　　D. [2 , 4 , 6 , 8]

315. 下面代码的输出结果是（　　）。

```
a = [1, 3]
b = [2, 4]
a.extend(b)
print(a)
```

A. [1, 3, 2, 4]　　B. [4 , 3 , 2 , 1]

C. [4 , 2 , 3 , 1]　　D. [1 , 2 , 3 , 4]

316. 设序列 s，以下选项中对 max(s)描述正确的是（　　）。

A. 一定能够返回序列 s 的最大元素。

B. 返回序列 s 的最大元素，如果有多个相同，则返回一个列表类型。

C. 返回序列 s 的最大元素，如果有多个相同，则返回一个元组类型。

D. 返回序列 s 的最大元素，但要求 s 中元素之间可比较。

317. 元组变量 t=("cat","dog","tiger","human")，t[::-1]的结果是（　　）。

A. ('human', 'tiger', 'dog', 'cat')　　B. 运行出错

C. {'human', 'tiger', 'dog', 'cat'}　　D. ['human', 'tiger', 'dog', 'cat']

318. 下面代码的输出结果是（　　）。

```
vlist = list(range(5))
for e in vlist:
    print(e, end=", ")
```

A. [0, 1, 2, 3, 4]　　B. 0; 1; 2; 3; 4;

C. 0, 1, 2, 3, 4,　　D. 0 1 2 3 4

319. 以下选项中，（　　）不是具体的 Python 序列类型。

A. 字符串类型　　B. 列表类型　　C. 数组类型　　D. 元组类型

320. 下面代码的输出结果是（　　）。

```
list1 = [i*2 for i in 'Python']
print(list1)
```

A. ['PP', 'yy', 'tt', 'hh', 'oo', 'nn']　　B. Python Python

C. 错误　　D. [2, 4, 6, 8, 10, 12]

321. 下面代码的输出结果是（　　）。

```
s =["seashell", "gold", "pink", "brown", "purple", "tomato"]
print(s[1: 4: 2])
```

A. ['gold', 'brown']　　B. ['gold', 'pink', 'brown','purple', 'tomato']

C. ['gold', 'pink']　　D. ['gold', 'pink','brown']

322. 下面代码的输出结果是（　　）。

```
a = [1, 2, 3]
for i in a[: : -1]:
    print(i, end=", ")
```

A. 3, 2, 1,　　B. 3, 1, 2　　C. 2, 1, 3　　D. 1, 2, 3

323. 将以下代码保存成 Python 文件， 运行后输出的是（　　）。

```
li = ['alex', 'eric', 'rain']
s = "_". join(li)
print(s)
```

A. _alex_eric_rain　　B. _alex_eric_rain_

C. alex_eric_rain　　D. alex_eric_rain_

324. 给出如下代码：

```
import random as ran
listV = []
ran.seed(100)
for i in range(10) :
    i = ran.randint(100, 999)
    listV.append(i)
```

以下选项中能输出随机列表元素最大值的是（　　）。

A. print(listV.reverse(i))　　B. print(max(listV))

C. print(listV.pop(i))　　D. print(listV.max())

325. 下列选项中，与 s[0:-1]表示的含义相同的是（　　）。

A. s[-1]　　B. s[:]　　C. s[:len(s)-1]　　D. s[0:len(s)]

326. 下列 Python 语句的输出结果是（　　）。

```
a=[1,2,3,None,(),[]]
print(len(a))
```

A. 4　　B. 5　　C. 6　　D. 7

327. 下列 Python 语句的输出结果是（　　）。

```
s1=[4, 5, 6]
s2=s1
s1[1]=0
print(s2)
```

A. [4, 5, 6]　　B. [4, 0, 6]　　C. [0, 5, 6]　　D. [4, 5, 0]

328. 二维列表 ls=[[1,2,3],[4,5,6],[7,8,9]]，能获取其中元素 5 选项是（　　）。

A. ls[1][1]　　B. ls[-2][-1]　　C. ls[-1][-1]　　D. ls[4]

329. 下列程序执行后，p 的值是（　　）。

A. 45　　B. 15　　C. 6　　D. 28

330. 二维列表 ls=[[1,2,3],[4,5,6],[7,8,9]]，以下（　　）能获取其中元素 9。

A. ls[0][-1]　　B. ls[-2][-1]　　C. ls[-1][-1]　　D. ls[-1]

331. Python 语句 print(type((1,2,3,4)))的输出结果是（　　）。

A. <class 'tuple'>　　B. <class 'dict'>

C. <class 'set'>　　D. <class 'list'>

332. Python 语句 print(type([1,2,3,4]))的输出结果是（　　）。

A. <class 'tuple'>　　B. <class 'dict'>

C. <class 'set'>　　D. <class 'list'>

333. tuple(range(2,10,2))的返回结果是（　　）。

A. [2, 4, 6, 8]　　B. [2, 4, 6, 8, 10]

C. (2, 4, 6, 8)　　D. (2, 4, 6, 8, 10)

334. 对于列表 L=[1,2,'Python',[1,2,3,4,5]]，L[-3]是（　　）。

A. 1　　B. 2　　C. 'Python'　　D. [1,2,3,4,5]

335. 表达式",".join(ls)中，ls 是列表类型，（　　）选项对其功能描述正确。

A. 在列表 ls 每个元素后增加一个逗号。

B. 将逗号字符串增加到列表 ls 中。

C. 将列表所有元素连接成一个字符串，元素之间增加一个逗号。

D. 将列表所有元素连接成一个字符串，每个元素后增加一个逗号。

336. 给出如下代码：

```
DictColor={"seashell":"海贝色","gold":"金色","pink":"粉红色","brown":"棕色
```

","purple":"紫色","tomato":"西红柿色"}
以下选项中，能输出“海贝色”的是（　　）。
A. print(DictColor["seashell"])　　B. print(DictColor.values())
C. print(DictColor["海贝色"])　　D. print(DictColor.keys())
337. 下面代码的输出结果是（　　）。
```
dict={'a':1,'b':2,'b':'3'}
temp=dict['b']
print(temp)
```
A. 1　　B. {'b':2}　　C. 3　　D. 2
338. 下面代码的输出结果是（　　）。
```
a = {}
if isinstance(a, list) :
    print("{} is list".format(a))
else:
    print("{} is {} ".format("a",type(a)))
```
A. 出错　　B. a is list　　C. 无输出　　D. a is <class 'dict'>
339. 给定字典 d，以下选项中对 d.items()的描述正确的是（　　）。
A. 返回一种 dict_items 类型，包括字典 d 中所有键值对。
B. 返回一个集合类型，每个元素是一个二元元组，包括字典 d 中所有键值对。
C. 返回一个元组类型，每个元素是一个二元元组，包括字典 d 中所有键值对。
D. 返回一个列表类型，每个元素是一个二元元组，包括字典 d 中所有键值对。
340. S 和 T 是两个集合，对 S|T 的描述正确的是（　　）。
A. S 和 T 的并运算，包括在集合 S 和 T 中的所有元素。
B. S 和 T 的补运算，包括集合 S 和 T 中的非相同元素。
C. S 和 T 的交运算，包括同时在集合 S 和 T 中的元素。
D. S 和 T 的差运算，包括在集合 S 但不在 T 中的元素。
341. 关于 Python 字典，以下选项中描述错误的是（　　）。
A. Python 字典是包含 0 个或多个键值对集合，没有长度限制，可根据"键"索引"值"的内容。
B. 如果想保持一个集合中元素的顺序，可以使用字典类型。
C. 字典中对某个键值的修改，可以通过中括号[]的访问和赋值实现。
D. Python 通过字典实现映射。
342. 下面代码的输出结果是（　　）。
```
str1="k:1|k1:2|k2:3|k3:4"
str_list=str1.split('|')
d={}
for l in str_list:
    key, value=l.split(':')
    d[key] =value
```

print(d)

A. [k: 1, k1: 2, k2: 3, k3: 4]

B. {k: 1, k1: 2, k2: 3, k3: 4}

C. ['k': '1', 'k1': '2', 'k2': '3', 'k3': '4']

D. {'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}

343. S 和 T 是两个集合，对 S&T 的描述正确的是（　　）。

A. S 和 T 的并运算，包括在集合 S 和 T 中的所有元素。

B. S 和 T 的补运算，包括集合 S 和 T 中的非相同元素。

C. S 和 T 的交运算，包括同时在集合 S 和 T 中的元素。

D. S 和 T 的差运算，包括在集合 S 但不在 T 中的元素。

344. 给定字典 d，以下选项中对 x in d 的描述正确的是（　　）。

A. x 是一个二元元组，判断 x 是否是字典 d 中的键值对。

B. 判断 x 是否是在字典 d 中以键或值方式存在。

C. 判断 x 是否是字典 d 中的值。

D. 判断 x 是不是字典 d 中的键。

345. 下面代码的输出结果是（　　）。

```
>>> s = set()
>>> type(s)
```

A. <class 'dict'>　　B. <class 'tuple'>

C. <class 'list'>　　D. <class 'set'>

346. 给定字典 d，以下选项中对 d.keys() 的描述正确的是（　　）。

A. 返回一种 dict_keys 类型，包括字典 d 中所有键。

B. 返回一个集合类型，包括字典 d 中所有键。

C. 返回一个元组类型，包括字典 d 中所有键。

D. 返回一个列表类型，包括字典 d 中所有键。

347. 下面代码的输出结果是_____。

```
l1=[1, 2, 3, 2]
l2=['aa','bb','cc','dd','ee']
d={}
for index in range(len(l1)):
    d[l1[index]]=l2[index]
print(d)
```

A. {1: 'aa', 2: 'dd', 3: 'cc'}。

B. {1: 'aa', 2: 'bb', 3: 'cc', 2: 'bb' }。

C. {1: 'aa', 2: 'bb', 3: 'cc', 2: 'dd' }。

D. {1: 'aa', 2: 'bb', 3: 'cc'}。

348. 下面代码的输出结果是（　　）。

```
i = ['a', 'b','c']
l = [1, 2, 3]
```

```
b = dict(zip(i, l) )
print(b)
```

A. {'a': 1, 'b': 2, 'c': 3}　　B. 不确定

C. 报出异常　　D. {1: 'a', 2: 'd', 3:'c' }

349. 以下选项中不能生成一个空字典的是（　　）。

A. {}　　B. {[]}　　C. dict([])　　D. dict()

350. 字典 d={'abc':123,'def':456,'ghi':789}，len(d)的结果是（　　）。

A. 3　　B. 12　　C. 9　　D. 6

351. 给定字典 d，以下选项中对 d.get(x,y)的描述正确的是（　　）。

A. 返回字典 d 中键值对为 x:y 的值。

B. 返回字典 d 中值为 y 的值，如果不存在，则返回 x。

C. 返回字典 d 中键为 y 的值，如果不存在，则返回 y。

D. 返回字典 d 中键为 x 的值，如果不存在，则返回 y。

352. 下面代码的输出结果是（　　）。

```
>>> s = {}
>>> type(s)
```

A. <class 'dict'>　　B. <class 'tuple'>

C. <class 'list'>　　D. <class 'set'>

353. 给定字典 d，以下选项中可以清空该字典并保留变量的是（　　）。

A. d.remove()　　B. del d　　C. d.clear()　　D. d.pop()

354. 下列类型的数据中其元素可以改变的是（　　）。

A. 列表　　B. 元组　　C. 字符串　　D. 单个字符

355. 下列 Python 语句的运行结果是（　　）。

```
nums=set([1,2,2,3,3,3,4])
print(len(nums))
```

A. 1　　B. 2　　C. 3　　D. 4

356. 字典 D={'A':10,'B':20,'C':30,'D':40}，对第 4 个字典元素访问形式是（　　）。

A. D[3]　　B. D[4]　　C. D[D]　　D. D['D']

357. 对于字典 D={'A':10,'B':20,'C':30,'D':40}，len(D)的值是（　　）。

A. 4　　B. 8　　C. 10　　D. 12

358. 语句 print(type({1:1,2:2,3:3,4:4}))的输出结果是（　　）。

A. <class 'tuple'>　　B. <class 'dict'>

C. <class 'set'>　　D. <class 'list'>

359. 设 a=set([1,2,2,3,3,3,4,4,4,4])，则 sum(a)的值是（　　）。

A. 10　　B. 20　　C. 30　　D. 40

360. 以下不能创建字典的语句是（　　）。

A. dict1={}　　B. dict2={3:5}

C. dict3=dict([2,5],[3,4])　　D. dict4=dict(([1,2],[3,4]))

361. 下列 Python 语句的运行结果是（　　）。

d={1:'a',2:'b',3:'c'}
print(len(d))
A. 0　　B. 1　　C. 3　　D. 6
362. 字典 D={'A':10,'B':20,'C':30,'D':40}，sum(list(D.values()))值是（　　）。
A. 10　　B. 100　　C. 40　　D. 200
363. 已知 s={'a',1,'b',2}，print(s['b'])运行结果是（　　）。
A. 语法错误　　B. 'b'　　C. 1　　D. 2
364. 以下不能创建集合的语句是（　　）。
A. s1=set()　　B. s2=set("abcd")
C. s3=(1,2,3,4)　　D. s4=frozenset((3,2,1))
365. 给出如下代码：
MonthandFlower={"1 月":"梅花","2 月":"杏花","3 月":"桃花","4 月": "牡丹花", "5 月":"石榴花","6 月":"莲花","7 月":"玉簪花","8 月":"桂花","9 月":"菊花","10 月":"芙蓉花","11 月":"山茶花","12 月":"水仙花"}
n=input("请输入 1-12 的月份:")
print(n+"月份代表花："+MonthandFlower.get(str(n)+"月"))
以下选项中描述正确的是（　　）。
A. MonthandFlower 是一个集合。
B. MonthandFlower 是一个元组。
C. MonthandFlower 是一个列表。
D. 代码实现了从键盘上获取一个整数（1-12）来表示月份， 输出该月份对应的代表花名。
366. 下面代码的输出结果是（　　）。
>>>f=lambda x, y: y+x
>>>f(10, 10)
A. 10　　B. 100　　C. 10,10　　D. 20
367. 执行下面的代码， 以下选项中正确的是_____。
def f(x, y = 0, z = 0) :
　　pass # 空语句， 定义空函数体
A. f(1,x=1,z=3)　　B. f(1,y=2,t=3)
C. f(x=1,y=2,z=3)　　D. f(x=1,2)
368. 下面代码的输出结果是（　　）。
def func(a, b) :
　　a *= b
　　return a
s = func(5, 2)
print(s)
A. 20　　B. 12　　C. 1　　D. 10
369. 给出如下代码：

```
ls = ["car", "truck"]
def funC(a) :
    ls =[]
    ls.append(a)
    return
funC("bus")
print(ls)
```

以下选项中描述错误的是（　　）。

A. 代码函数定义中，ls.append(a)中的 ls 是局部变量。

B. 执行代码输出结果为['car','truck','bus']。

C. ls.append(a) 代码中的 ls 是列表类型。

D. 执行代码输出结果为['car', 'truck']。

370. 以下选项中，对于递归程序的描述错误的是（　　）。

A. 书写简单　　B. 递归程序都可以有非递归编写方法

C. 一定要有基例　　D. 执行效率高

371. 给出如下代码：

```
ls = ["car", "truck"]
def funC(a):
    ls.append(a)
    return
funC("bus")
print(ls)
```

以下选项中描述错误的是（　　）。

A. ls.append(a)代码中的 ls 是全局变量。

B. funC(a)中的 a 为非可选参数。

C. ls.append(a)代码中的 ls 是列表类型。

D. 执行代码输出结果为['car', 'truck']。

372. 关于 lambda 函数，以下选项中描述错误的是（　　）。

A. lambda 函数也称为匿名函数。　　B. lambda 不是 Python 的保留字。

C. 定义了一种特殊的函数。　　D. lambda 函数将函数名作为函数结果返回。

373. 下面代码的输出结果是（　　）。

```
def fib(n):
    a, b = 1, 1
    for i in range(n-1) :
        a, b = b, a+b
    return a
print (fib(7))
```

A. 5　　B. 21　　C. 13　　D. 8

374. 关于函数参数传递（parameter passing），以下选项错误的是（　　）。

A. 形式参数是函数定义时提供的参数。

B. 函数调用时，需要将形式参数传递给实际参数。

C. Python 参数传递时不构造新数据对象，而是让形式参数和实际参数共享同一对象。

D. 实际参数是函数调用时提供的参数。

375. 下面代码实现的功能描述为（　　）。

```
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)
num =eval(input("请输入一个整数："))
print(fact(abs(int(num))))
```

A. 接受用户输入的整数 N，输出 N 的阶乘值。

B. 接受用户输入的整数 N，判断 N 是否为水仙花数。

C. 接受用户输入的整数 N，判断 N 是否为完数并输出结论。

D. 接受用户输入的整数 N，判断 N 是否为素数并输出结论。

376. 关于 Python 的 lambda 函数， 以下选项中描述错误的是（　　）。

A. lambda 用于定义简单的、能够在一行内表示的函数。

B. lambda 函数将函数名作为函数结果返回。

C. f = lambda x, y: x+y 执行后，f 的类型为数字类型。

D. 可以使用 lambda 函数定义列表的排序原则。

377. 假设函数中不包括 global 保留字，对于改变参数值的方法，错误的是（　　）。

A. 参数是列表类型时，改变原参数的值。

B. 参数的值是否改变与函数中对变量的操作有关，与参数类型无关。

C. 参数是组合类型（可变对象）时，改变原参数的值。

D. 参数是整数类型时，不改变原参数的值。

378. 下面代码的执行结果是（　　）。

```
def greeting(args1, *tupleArgs, **dictArgs) :
    print(args1)
    print(tupleArgs)
    print(dictArgs)
names = ['HTY', 'LFF','ZH' ]
info = {'schoolName':'NJRU','City':'Nanjing' }
greeting('Hello,', *names, **info)
```

A. 出错

B. Hello,
('HTY', 'LFF', 'ZH')
{'schoolName': 'NJRU', 'City': 'Nanjing'}

C. ['HTY', 'LFF', 'ZH']

D. 无输出

379. 下面代码的执行结果是（　　）。

```
def area(r,pi=3.14159):
    return pi*r*r
area(pi=3.14,r=4)
```

A. 出错　　B. 50.24　　C. 39.4384　　D. 无输出

380. 下面代码的执行结果是（　　）。

```
def area(r,pi = 3.14159) :
    return pi*r*r
area(3.14,4)
```

A. 出错　　B. 50.24　　C. 39.4384　　D. 无输出

381. 关于函数的参数，以下选项中描述错误的是（　　）。

A. 定义函数时，如果有些参数存在默认值，可以在定义函数时直接为这些参数指定默认值。

B. 一个元组可以传递给带有星号的可变参数。

C. 可选参数可以定义在非可选参数的前面。

D. 在定义函数时，可以设计可变数量参数，通过在参数前增加星号（*）实现。

382. 关于函数局部变量和全局变量的使用规则，以下选项中描述错误的是（　　）。

A. 对于基本数据类型的变量，无论是否重名，局部变量与全局变量不同。

B. return 不可以传递任意多个函数局部变量返回值。

C. 对于组合数据类型的变量，如果局部变量未真实创建，则是全局变量。

D. 可以通过 global 保留字在函数内部声明全局变量。

383. 关于函数的返回值，以下选项中描述错误的是（　　）。

A. 函数可以返回 0 个或多个结果。

B. return 可以传递 0 个返回值，也可以传递任意多个返回值。

C. 函数可以有 return，也可以没有。

D. 函数必须有返回值。

384. 下面代码的输出结果是（　　）。

```
MA = lambda x, y : (x > y) * x + (x < y) * y
MI = lambda x, y : (x > y) * y + (x < y) * x
a = 10
b = 20
print(MA(a, b) )
print(MI(a, b) )
```

A. 20 10　　B. 20 20　　C. 10 10　　D. 10 20

385. 下面代码的运行结果是（　　）。

A. 10　　B. int　　C. 出错　　D. 11

386. 关于递归函数基例的说明， 以下选项中错误的是（　　）。

A. 递归函数必须有基例　　B. 递归函数的基例决定递归的深度

C. 每个递归函数都只能有一个基例　　D. 递归函数的基例不再进行递归

387. 给出如下代码：

```
def func(a, b) :
    c=a**2+b
    b=a
    return c
a=10
b=100
c=func(a, b) +a
```

以下选项中描述错误的是（　　）。

A. 执行该函数后，变量 c 的值为 200　　B. 执行该函数后，变量 a 的值为 10

C. 执行该函数后，变量 b 的值为 100　　D. 该函数名称为 func

388. 下面代码的输出结果是（　　）。

```
def hello_world() :
    print('ST', end="*")
def three_hellos() :
    for i in range(3) :
        hello_world()
three_hellos()
```

A. ST*ST*ST*　　B. ***　　C. ST*　　D. ST*ST*

389. 关于递归函数的描述，以下选项中正确的是（　　）。

A. 包含一个循环结构　　B. 函数名称作为返回值

C. 函数内部包含对本函数的再次调用　　D. 函数比较复杂

390. 关于 return 语句，以下选项中描述正确的是（　　）。

A. 函数中最多只有一个 return 语句　　B. 函数可以没有 return 语句

C. return 只能返回一个值　　D. 函数必须有一个 return 语句

391. 关于形参和实参的描述，以下选项中正确的是（　　）。

A. 函数定义中参数列表里面的参数是实际参数，简称实参。

B. 程序在调用时，将形参复制给函数的实参。

C. 程序在调用时，将实参复制给函数的形参。

D. 参数列表中给出要传入函数内部的参数，这类参数称为形式参数，简称形参。

392. 关于函数的关键字参数使用限制，以下选项中描述错误的是（　　）。

A. 关键字参数必须位于位置参数之前　　B. 关键字参数顺序无限制

C. 不得重复提供实际参数　　D. 关键字参数必须位于位置参数之后

393. 下面代码的输出结果是（　　）。

```
def f2(a) :
    if a > 33:
        return True
li = [11, 22, 33, 44, 55]
```

```
res = filter(f2, li)
print(list(res))
```

A. [44, 55]　　B. [11, 33, 55]

C. [22, 33, 44]　　D. [33, 44, 55]

394. 在 Python 中，关于全局变量和局部变量，以下选项中描述错误的是（　　）。

A. 一个程序中的变量包含两类：全局变量和局部变量。

B. 全局变量不能和局部变量重名。

C. 全局变量在程序执行的全过程有效。

D. 全局变量一般没有缩进。

395. 关于函数，以下选项中描述错误的是（　　）。

A. 函数也是数据。

B. 一个函数定义一个用户自定义函数对象。

C. 函数名称不可赋给其他变量。

D. 函数定义语句可执行。

396. 下面代码的输出结果是（　　）。

```
def exchange(a, b) :
    a, b = b, a
    return (a, b)
x = 10
y = 20
x, y = exchange(x, y)
print(x, y)
```

A. 20 10　　B. 20 20　　C. 10 10　　D. 20, 10

397. 关于下面代码， 以下选项中描述正确的是（　　）。

```
def fact(n, m=1) :
    s = 1
    for i in range(1, n+1):
        s *= i
    return s//m
print(fact(m=5, n=10))
```

A. 参数按照名称传递　　B. 按可变参数调用

C. 执行结果为 10886400　　D. 按位置参数调用

398. 执行下面代码，错误的是（　　）。

```
def f(x, y = 0, z = 0) :
    pass # 空语句， 定义空函数体
```

A. f(1, 2, 3)　　B. f(1)　　C. f(1, , 3)　　D. f(1, 2)

399. 给出如下代码：

```
def fact(n) :
    s = 1
```

```
    for i in range(1, n+1) :
        s *= i
    return s
```

以下选项中描述错误的是（　　）。

A. 代码中 n 是可选参数　　B. range()函数是 Python 内置函数

C. s 是局部变量　　D. fact(n)函数功能为求 n 的阶乘

400. 以下选项中，对于函数的定义错误的是（　　）。

A. def vfunc(a, b=2):　　B. def vfunc(*a, b):

C. def vfunc(a, *b):　　D. def vfunc(a, b):

401. 下面代码的输出结果是（　　）。

```
def func(a, b) :
    return a>>b
s = func(5, 2)
print(s)
```

A. 20　　B. 12　　C. 1　　D. 6

402. 在 Python 中，关于函数的描述，以下选项中正确的是（　　）。

A. 一个函数中只允许有一条 return 语句。

B. 函数 eval()可以用于数值表达式求值，例如 eval("2*3+1")。

C. Python 函数定义中没有对参数指定类型，这说明，参数在函数中可以当作任意类型使用。

D. Python 中，def 和 return 是函数必须使用的保留字。

403. 下列语句的运行结果是（　　）。

```
f1=lambda x: x*2
f2=lambda x: x**2
print(f1(f2(2)))
```

A. 3　　B. 4　　C. 6　　D. 8

404. 下列程序执行后， y 的值是（　　）。

```
def f(x, y) :
    return x**2+y**2
y=f(f(1, 3) , 5)
```

A. 100　　B. 125　　C. 35　　D. 9

405. 关于函数的说法中正确的是（　　）。

A. 函数定义时必须有形参。

B. 函数中定义的变量只在该函数体中起作用。

C. 函数定义时必须带 return 语句。

D. 实参与形参的个数可以不相同，类型可以任意。

406. 下列选项中不属于函数优点的是（　　）。

A. 减少代码重复　　B. 使程序模块化

C. 使程序便于阅读　　D. 便于发挥程序员的创造力

407. 以下关于函数说法正确的是（　　）。

A. 函数的实际参数和形式参数必须同名。

B. 函数的形式参数既可以是变量也可以是常量。

C. 函数的实际参数不可以是表达式。

D. 函数的实际参数可以是其他函数的调用。

408. 已知 f=lambda x, y: x+y，则 f([4],[1,2,3])的值是（　　）。

A. [1, 2, 3, 4]　　B. 10

C. [4, 1, 2, 3]　　D. {1, 2, 3, 4}

409. 以下选项中， 不是 Python 对文件的读操作方法的是（　　）。

A. read　　B. readtext　　C. readlines　　D. readline

410. 以下选项对应的方法可以用于从 CSV 文件中解析一二维数据的是（　　）。

A. split()　　B. exists()　　C. format()　　D. join()

411. 以下选项中，不是 Python 中文件操作的相关函数是_____。

A. open()　　B. write()　　C. read()　　D. load()

412. 关于文件的打开方式，以下选项中描述正确的是（　　）。

A. 文件只能选择二进制或文本方式打开。

B. 所有文件都可能以二进制方式打开。

C. 所有文件都可能以文本方式打开。

D. 文本文件只能以文本方式打开。

413. 给出如下代码：

```
fname = input("请输入要打开的文件: ")
fi = open(fname, "r")
for line in fi.readlines():
    print(line)
fi.close()
```

以下选项中描述错误的是（　　）。

A. 用户输入文件路径，以文本文件方式读入文件内容并逐行打印。

B. 上述代码中 fi.readlines()可以优化为 fi。

C. 通过 fi.readlines()方法将文件的全部内容读入一个列表 fi。

D. 通过 fi.readlines()方法将文件的全部内容读入一个字典 fi。

414. 关于 Python 文件的"+"打开模式，以下选项中描述正确的是（　　）。

A. 只读模式。

B. 与 r/w/a/x 一同使用，在原功能基础上增加同时读写功能。

C. 追加写模式。

D. 覆盖写模式。

415. 在以下选项中，对 CSV 格式的描述正确的是（　　）。

A. CSV 文件以英文逗号分隔元素　　B. CSV 文件以英文特殊符号分隔元素

C. CSV 文件以英文分号分隔元素　　D. CSV 文件以英文空格分隔元素

416. 执行如下代码：

```
fname = input("请输入要写入的文件: ")
fo = open(fname, "w+")
ls=["清明时节雨纷纷,","路上行人欲断魂,","借问酒家何处有？","牧童遥指杏花村。"]
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

以下选项中描述错误的是（　　）。

A. 执行代码时，从键盘输入"清明.txt"，则清明.txt 被创建。

B. 代码主要功能为向文件写入一个列表类型，并打印输出结果。

C. fo.seek(0)这行代码可以省略，不影响输出效果。

D. fo.writelines(ls)将元素全为字符串的 ls 列表写入文件。

417. 以下选项中，不是 Python 文件打开的合法模式组合的是（　　）。

A. "r+"　　B. "a+"　　C. "t+"　　D. "w+"

418. 以下选项中，不是 Python 中文件操作的相关函数是（　　）。

A. write()　　B. writeline()

C. readlines()　　D. open()

419. 关于 Python 文件打开模式的描述，以下选项中错误的是（　　）。

A. 只读模式 r　　B. 创建写模式 n

C. 追加写模式 a　　D. 覆盖写模式 w

420. 关于 CSV 文件的描述，以下选项中错误的是（　　）。

A. CSV 文件格式是一种通用的、相对简单的文件格式，应用于在程序之间转移表格数据。

B. 整个 CSV 文件是一个二维数据。

C. CSV 文件通过多种编码表示字符。

D. CSV 文件的每一行是一维数据，可以使用 Python 中的列表类型表示。

421. 表达式 writelines(lines)，能够将一个元素是字符串的列表 lines 写入文件，以下选项中描述正确的是（　　）。

A. 在生成的文件中，列表 lines 中各元素之间默认采用空格分隔。

B. 在生成的文件中，列表 lines 中各元素之间无分隔符。

C. 在生成的文件中，列表 lines 中各元素之间默认采用换行分隔。

D. 在生成的文件中，列表 lines 中各元素之间默认采用逗号分隔。

422. 以下选项中，不是 Python 文件处理.seek()方法的参数是（　　）。

A. 0　　B. 2　　C. 1　　D. -1

423. 关于下面代码中的变量 x，以下选项中描述正确的是（　　）。

```
fo = open(fname, "r")
for x in fo:
    print(x)
```

fo. close()

A. 变量 x 表示文件中的一个字符。　　B. 变量 x 表示文件中的一组字符。

C. 变量 x 表示文件中的全体字符。　　D. 变量 x 表示文件中的一行字符。

424. 关于文件关闭的.close()方法，以下选项中描述正确的是（　　）。

A. 文件处理结束之后，一定要用.close() 方法关闭文件。

B. 文件处理遵循严格的“打开—操作—关闭”模式。

C. 文件处理后可以不用.close()方法关闭文件，程序退出时会默认关闭。

D. 如果文件是以只读方式打开，仅在这种情况下可以不用.close()方法关闭文件。

425. 以下文件操作方法中，不能向 CSV 格式文件写入数据的是（　　）。

A. write　　B. seek 和 write

C. writeline　　D. writelines

426. 当打开一个不存在的文件时， 以下选项中描述正确的是（　　）。

A. 一定会报错　　B. 文件不存在则创建文件

C. 不存在文件无法被打开　　D. 根据打开类型不同，可能不报错

427. 以下选项中， 不是 Python 对文件的打开模式的是（　　）。

A. 'r'　　B. 'c'　　C. '+'　　D. 'w'

428. 下列程序的输出结果是（　　）。

```
f=open('f.txt','w' )
f.writelines(['python programming.'])
f.close()
f=open('f.txt','rb')
f.seek(10,1)
print(f.tell())
```

A. 1　　B. 10　　C. gramming　　D. Python

429. 在读写文件之前， 用于创建文件对象的函数是（　　）。

A. open　　B. create　　C. file　　D. folder

430. 关于语句 f=open('demo.txt', 'r')，下列说法错误的是（　　）。

A. demo.txt 文件必须已经存在。

B. 只能从 demo.txt 文件读数据，而不能向该文件写数据。

C. 只能向 demo.txt 文件写数据，而不能从该文件读数据。

D. "r"方式是默认的文件打开方式。

431. Python 异常处理机制中没有（　　）语句。

A. try　　B. throw　　C. assert　　D. finally

432. 下列关于 Python 异常处理的描述中，错误的是（　　）。

A. 异常处理可以通过 try-except 语句实现。

B. 任何需要检测的语句必须在 try 语句块中执行，并由 except 语句处理异常。

C. raise 语句引发异常后，它后面的语句不再执行。

D. except 语句处理异常最多有两个分支。

433. 下列程序的输出结果是（　　）。

```
try:
    x=1/2
except ZeroDivisionError:
    print('AAA')
```

A. 0　　B. 0.5　　C. AAA　　D. 无输出

434. 以下关于异常处理 try 语句块的说法，错误的是（　　）。

A. finally 语句中的代码段始终要被执行。

B. 一个 try 块后接一个或多个 except 块。

C. 一个 try 块后接一个或多个 finally 块。

D. try 必须与 except 或 finally 块一起使用。

435. 关于程序的异常处理，以下选项中描述错误的是（　　）。

A. Python 通过 try、except 等保留字提供异常处理功能。

B. 编程语言中的异常和错误是完全相同的概念。

C. 异常语句可以与 else 和 finally 保留字配合使用。

D. 程序异常发生后经过妥善处理可以继续执行。

436. Python 异常处理中不会用到的关键字是（　　）。

A. try　　B. finally　　C. if　　D. else

437. 下列 Python 保留字，用于异常处理结构中捕获特定类型异常的是（　　）。

A. def　　B. pass　　C. while　　D. except

438. 如果以负数作为平方根函数 math.sqrt()的参数，将产生（　　）。

A. 死循环　　B. 得数　　C. ValueError 异常　　D. finally

439. 关于 jieba 库的全模式分词，以下选项中描述正确的是（　　）。

A. 适合用于搜索引擎分词。

B. 在精确模式基础上，对长词再次切分，提高召回率。

C. 将句子最精确地切开，适合文本分析。

D. 把句子中所有可以成词的词语都扫描出来，速度非常快，但是不能解决歧义。

440. 关于 jieba 库的函数 jieba.lcut(x,cut_all=True)，正确选项是（　　）。

A. 精确模式，返回中文文本 x 分词后的列表变量。

B. 向分词词典中增加新词 w。

C. 搜索引擎模式，返回中文文本 x 分词后的列表变量。

D. 全模式，返回中文文本 x 分词后的列表变量。

441. 以下函数中，不是 jieba 库函数的是（　　）。

A. sorted(x)　　B. add_word()

C. lcut_for_search()　　D. lcut()

442. 关于 jieba 库的函数 jieba.lcut(x)，正确选项是（　　）。

A. 精确模式，返回中文文本 x 分词后的列表变量。

B. 向分词词典中增加新词 w。

C. 搜索引擎模式，返回中文文本 x 分词后的列表变量。

D. 全模式，返回中文文本 x 分词后的列表变量。

443. 关于 jieba 库的精确模式分词，以下选项中描述正确的是（　　）。

A. 把句子中所有可以成词的词语都扫描出来，速度非常快。

B. 在精确模式基础上，对长词再次切分，提高召回率。

C. 将句子最精确地切开，适合文本分析。

D. 适合用于搜索引擎分词。

444. 关于 wordcloud 库的描述，以下选项中正确的是（　　）。

A. wordcloud 库是专用于根据文本生成词云的 Python 第三方库。

B. wordcloud 库是网络爬虫方向的 Python 第三方库。

C. wordcloud 库是机器学习方向的 Python 第三方库。

D. wordcloud 库是中文分词方向的 Python 第三方库。

1.2 答案

1. A;	2. A;	3. A;	4. A;	5. B;
6. A;	7. B;	8. B;	9. A;	10. A;
11. B;	12. B;	13. A;	14. B;	15. C;
16. B;	17. A;	18. A;	19. B;	20. C;
21. B;	22. C;	23. A;	24. D;	25. C;
26. A;	27. B;	28. D;	29. A;	30. D;
31. C;	32. A;	33. B;	34. D;	35. C;
36. D;	37. B;	38. C;	39. B;	40. B;
41. C;	42. D;	43. C;	44. A;	45. B;
46. B;	47. B;	48. C;	49. B;	50. B;
51. C;	52. D;	53. A;	54. D;	55. B;
56. D;	57. B;	58. C;	59. C;	60. C;
61. D;	62. C;	63. C;	64. C;	65. B;
66. D;	67. B;	68. A;	69. D;	70. A;
71. D;	72. C;	73. B;	74. C;	75. A;
76. B;	77. A;	78. A;	79. B;	80. D;
81. B;	82. C;	83. A;	84. A;	85. A;
86. B;	87. A;	88. B;	89. A;	90. C;
91. B;	92. D;	93. C;	94. A;	95. A;
96. B;	97. A;	98. A;	99. D;	100. A;
101. A;	102. D;	103. D;	104. C;	105. C;
106. A;	107. D;	108. A;	109. B;	110. A;
111. C;	112. B;	113. B;	114. B;	115. A;
116. A;	117. C;	118. B;	119. C;	120. D;
121. A;	122. B;	123. C;	124. B;	125. D;
126. C;	127. B;	128. B;	129. A;	130. D;
131. D;	132. C;	133. B;	134. C;	135. B;
136. B;	137. D;	138. C;	139. D;	140. A;
141. B;	142. A;	143. B;	144. B;	145. B;
146. B;	147. D;	148. A;	149. D;	150. C;
151. C;	152. B;	153. B;	154. A;	155. B;
156. B;	157. B;	158. B;	159. D;	160. D;
161. D;	162. B;	163. A;	164. C;	165. B;
166. C;	167. B;	168. A;	169. C;	170. B;
171. A;	172. D;	173. A;	174. D;	175. D;
176. A;	177. A;	178. A;	179. A;	180. A;
181. B;	182. A;	183. A;	184. A;	185. A;

186. A; 187. C; 188. A; 189. B; 190. C;
191. B; 192. B; 193. C; 194. C; 195. C;
196. C; 197. D; 198. C; 199. D; 200. D;
201. C; 202. A; 203. C; 204. B; 205. A;
206. B; 207. C; 208. C; 209. A; 210. B;
211. D; 212. B; 213. A; 214. D; 215. A;
216. A; 217. B; 218. A; 219. A; 220. A;
221. C; 222. A; 223. A; 224. A; 225. A;
226. C; 227. B; 228. B; 229. A; 230. D;
231. A; 232. C; 233. C; 234. D; 235. A;
236. A; 237. D; 238. B; 239. A; 240. D;
241. C; 242. B; 243. D; 244. A; 245. B;
246. A; 247. A; 248. C; 249. A; 250. B;
251. B; 252. B; 253. C; 254. D; 255. A;
256. A; 257. A; 258. A; 259. B; 260. B;
261. B; 262. C; 263. D; 264. C; 265. A;
266. C; 267. C; 268. B; 269. A; 270. D;
271. A; 272. A; 273. A; 274. A; 275. A;
276. D; 277. D; 278. C; 279. D; 280. A;
281. A; 282. C; 283. A; 284. D; 285. B;
286. A; 287. A; 288. D; 289. C; 290. C;
291. B; 292. C; 293. D; 294. A; 295. B;
296. B; 297. C; 298. D; 299. A; 300. A;
301. A; 302. C; 303. D; 304. A; 305. D;
306. C; 307. C; 308. C; 309. D; 310. C;
311. C; 312. B; 313. A; 314. C; 315. A;
316. D; 317. A; 318. C; 319. C; 320. A;
321. A; 322. A; 323. C; 324. B; 325. C;
326. C; 327. B; 328. A; 329. A; 330. C;
331. A; 332. D; 333. C; 334. B; 335. C;
336. A; 337. C; 338. D; 339. A; 340. A;
341. B; 342. D; 343. C; 344. D; 345. D;
346. A; 347. A; 348. A; 349. B; 350. A;
351. D; 352. A; 353. C; 354. A; 355. D;
356. D; 357. A; 358. B; 359. A; 360. C;
361. C; 362. B; 363. A; 364. C; 365. D;
366. D; 367. C; 368. D; 369. B; 370. D;
371. D; 372. B; 373. C; 374. B; 375. A;
376. C; 377. B; 378. B; 379. B; 380. C;

381. C;　382. B;　383. D;　384. A;　385. A;
386. C;　387. A;　388. A;　389. C;　390. B;
391. C;　392. A;　393. A;　394. B;　395. C;
396. A;　397. A;　398. C;　399. A;　400. B;
401. C;　402. B;　403. D;　404. B;　405. B;
406. D;　407. D;　408. C;　409. B;　410. A;
411. D;　412. A;　413. D;　414. B;　415. A;
416. C;　417. C;　418. B;　419. B;　420. C;
421. B;　422. D;　423. D;　424. D;　425. C;
426. D;　427. B;　428. B;　429. A;　430. C;
431. B;　432. D;　433. D;　434. C;　435. B;
436. C;　437. D;　438. C;　439. D;　440. D;
441. A;　442. A;　443. C;　444. A。

2　填空题

2.1 习题

1. Python 的单行注释符号，是（　　）。
2. 下列 Python 代码的输出结果，是（　　）。

```
nums=[1,2,3,4,5]
squares=[x**2 for x in nums]
print(squares)
```

3. 运行代码“print(3+2*2)”，输出的结果是（　　）。
4. Python 标准库 math 中，用来计算平方根的函数是（　　）。
5. 在 Python 中，（　　none　　）表示空类型。
6. 列表、元组、字符串是 Python 的（　　）序列。
7. 查看变量类型的 Python 内置函数，是（　　）。
8. 查看变量内存地址的 Python 内置函数，是（　　）。
9. 表达式“print([1, 2, 3]*3)”的运行结果，是（　　）。
10. 表达式“print(list(map(str, [1, 2, 3])))”的执行结果，是（　　）。
11. 已知 x=3，并且“print(id(x))”的返回值为 496103280，那么执行语句“x+=6”之后，表达式“print(id(x) ==496103280)”，输出结果为（　　）。
12. 已知 x=3，那么执行语句 x*=6 之后，x 的值为（　　）。
13. 表达式“print([3] in [1, 2, 3, 4])”的执行结果，为（　　）。
14. 表达式“print(3 in [1, 2, 3, 4])”的执行结果，为（　　）。
15. 列表对象 aList 的值为[3,4,5,6,7,9,11,13,15,17]，切片 aList[3:7]的值是（　　）。
16. 表达式“print([5 for i in range(10)])”的运行结果是（　　）。
17. 任意长度的 Python 列表、元组和字符串，最后一个元素的下标为（　　）。
18. 转义字符“\n”的含义，是（　　）。
19. 表达式“print(int(4**0.5))”的执行结果，为（　　）。
20. 表达式“3 ** 2”的值，为（　　）。
21. 表达式 3 * 2 的值，为（　　）。
22. 列表 a=['name', 'age', 'sex']、b=['Dong',38,'Male']，以列表 a 中的元素为“键”，以列表 b 中的元素为“值”，这两个列表转换为字典 c，对应的语句是（　　）。
23. 语句 print(' '.join(list('hello world!'))），执行结果是（　　）。
24. 语句 print(list(range(1,10,3)))，执行结果是（　　）。
25. 切片操作 list(range(6)) [: : 2]，执行结果为（　　）。
26. 表达式 print(' ab' in ' acbed')，运行结果为（　　）。
27. 语句 print(1, 2, 3, sep=' : ')，输出结果为（　　）。
28. 表达式 print(sorted([111,2,33],key=lambda x: -len(str(x)))），运行结果是（　　）。

29. 列表对象 x = [' 11' , ' 2' , ' 3']，表达式 print(max(x))的值为（　　）。
30. 表达式 min([' 11' , ' 2' , ' 3'])的值为（　　）。
31. 列表对象 x=['11','2','3']，则表达式 max(x, key=len)的值为（　　）。
32. 执行语句 x=(3,)，x 的值为（　　）。
33. 执行语句 x=(3)后，x 的值为（　　）。
34. 已知 x = {1: 2}，执行语句 x[2] = 3 之后，x 的值为（　　）。
35. 字典对象的（　　）方法，返回字典中的“键-值对”列表。
36. 使用列表推导式代码（　　），得到 100 以内所有能被 13 整除的数。
37. 已知 x=[3,5,7]，执行语句 x[len(x) :]=[1,2]之后，x 的值为（　　）。
38. 表达式 list(zip([1, 2] , [3,4]))的值，为（　　）。
39. 已知 x=[1,2,3,2,3]，执行语句 x.pop()之后，x 的值为（　　）。
40. 表达式[x for x in [1,2,3,4,5] if x<3]的值，为（　　）。
41. 表达式[index for index, value in enumerate([3,5,7,3,7]) if value==max([3,5,7,3,7])]的值，为（　　）。
42. 已知 path=r'c:\test.html'，表达式 print(path[: -4]+'htm')，运行结果为（　　）。
43. 表达式 print(' %d, %c' % (65, 65))，运行结果是：（　　）。
44. 表达式'The first:{1},the second is {0}'.format(65,97)的值为（　　）。
45. 表达式',' .join(' a b ccc\n\n\nddd '.split())的值为（　　）。
46. 已知 x='123'、y='456'，那么表达式 x+y 的值为（　　）。
47. 表达式'abcab'.replace('a','yy')的值为（　　）。
48. 已知 table=' ' .maketrans('abcw','xyzc')，那么表达式'Hellow world'.translate(table)的值，为（　　）。
49. 表达式' : '.join('abcdefg'.split('cd'))的值，为（　　）。
50. 表达式 isinstance(' abcdefg' , str)的值，为_____ 。 （True）
51. 表达式'Hello world. I like Python.'.find('python')的值，为（　　）。
52. 已知 x={'b':1,'a':2}，执行语句 x.update({'a':3,'d':4})之后，表达式 sorted(x.items())的值，为（　　）。
53. 已知 x=list(range(20))，语句 print(x[100:200])的输出结果，为（　　）。
54. 表达式 sorted({'a':9,'b':3,'c':78}.values())的值，为（　　）。
55. type(1+2*3.14)的值，是：（　　）。
56. （　　）年，第一个 Python 编译器诞生。
57. 查看变量类型的 Python 内置函数是（　　）。
58. 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。
59. 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（　　）。
60. 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（　　）。
61. 字典对象的（　　）方法，可以获取指定“键”相应的“值”，并且可

以在指定“键”不存在的时候返回指定值，假如不指定则返回 None。

62. 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（　　）。

63. random 模块中，（　　）方法的作用，是将列表中的元素随机乱序。

64. 函数定义以（　　）开始。

65. 下列程序的运行结果是：（　　）。

```
list1 = [1, 3, 5, 7]
list2 = [2, 4, 6, 8, 1, 7]
different_elements = set(list1) ^ set(list2)
print(different_elements)
```

66. 在 Python 中，用来计算商的运算符是（　　）。

67. 表达式[1, 2, 3]*3 的执行结果，为（　　）。

68. 已知 x = 3，那么执行语句 x *= 6 之后，x 的值为（　　）。

69. 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（　　）。

70. 有列表 d=[1,3,5]，执行以下操作 d.append(7)后，d 的内容为（　　）。

71. 列表中多个元素之间，使用（　　）分隔开。

72. 已知 x = { 'a': 'b', 'c': 'd' }，那么表达式'a' in x 的值为_____。

73. 在循环语句中，（　　）语句的作用是提前结束本层循环。

74. 表达式'a' + 'b' 的值为（　　）。

75. 表达式'Hello world!' [-4]的值，为（　　）。

76. Python 中定义函数的关键字是（　　）。

77. 表达式 eval('3+5')的，值为（　　）。

78. 在 Python 中常用的输入输出语句，分别是（　　）。

79. 表达式 {1, 2, 3} | {2, 3, 4} 的值，为（　　）。

80. 如果函数中没有 return 语句，那么该函数的返回值为（　　）。

81. 语句 x, y, z = [1, 2, 3] 执行后，变量 y 的值为（　　）。

82. 表达式 3 not in [1, 2, 3]的值，为（　　）。

83. 创建一个空列表的语句是（　　）。

84. Python 中用于计算字符串长度的函数是（　　）。

85. 在 Python 中，整数除法结果的类型是（　　）。

86. Python 的逻辑与运算符是（　　）。

87. 使用（　　）关键字，可以声明一个空字典。

88. 在 Python 中，获取字符串 string 的第一个字符，可以使用（　　）。

89. Python 中的模块用于组织和重用代码，可以使用（　　）语句导入它。

90. （　　）语句用于捕获所有异常。

91. 在 Python 中，用于从文件读取内容的函数是（　　）。

92. （　　）方法用于将元素添加到集合中。

93. 字符串 s 中最后一个字符的位置是（　　）。

94. Python 表达式 12/4-2+5*8/4%5/2 的值为（　　）。

95. 使用 math 模块库中的函数时， 必须要使用（　　）语句导入该模块。
96. Python 提供了一个内置的数学函数库 math，math 库不支持（　　）（整数/浮点数/复数）类型。
97. 需要使用 math 库时，可以使用（　　）和（　　）这两种方式来实现 math 库的导入。
98. Python 数学函数中，（　　）函数可以用于求和。
99. Python 数学函数中，（　　）函数可以用于求两个数的最大公约数。
100. Python 数学函数中，（　　）函数可以返回一个数的整数部分。
101. Python 数学函数中，（　　）函数可以向上取整。
102. Python 数学函数中，（　　）函数可以向下取整。
103. Python 数学函数中，math 库提供了数学常数，（　　）表示圆周率。
104. Python 数学函数中，math 库提供了数学常数，（　　）表示自然对数。
105. 设 s='abcdefg'，则 s[3]的值是（　　）。
106. 设 s='abcdefg'，则 s[3:5]的值是（　　）。
107. 设 s='abcdefg'，则 s[: 5]的值是（　　）。
108. 设 s='abcdefg'，则 s[3:]的值是（　　）。
109. 设 s='abcdefg'，则 s[: : 2]的值是（　　）。
110. 设 s='abcdefg'，则 s[: : -1]的值是（　　）。
111. 在 Python 中，传统除法运算符是（　　）。
112. 在 Python 中，整除除法运算符是（　　）。
113. Python 表达式 4.5/2 的值为（　　）。
114. Python 表达式 4.5%2 的值为（　　）。
115. Python 表达式 4.5//2 的值为（　　）。
116. 下列 Python 语句的执行结果是（　　）。

```
a, b=3, 4
a, b=b, a
print(a, b)
```

117. Python 中，如果语句太长，可以使用（　　）作为续行符。
118. 已知 s1='red hat'，print(s1.upper())的结果是（　　）。
119. 已知 s1='red hat'，print(s1.swapcase())的结果是（　　）。
120. 已知 s1='red hat'，print(s1.title())的结果是（　　）。
121. 已知 s1='red hat'，print(s1.replace('hat' , 'cat'))的结果是（　　）。
122. 语句 print('AAA', "BBB",sep='-',end='!')执行的结果是（　　）。
123. 语句 a,a=10, 20 执行后，a 的值是（　　）。
124. 下面语句的执行结果是（　　）。

```
s='A'
print(3*s.split())
```

125. 设 Python 中有模块 m，如果希望同时导入 m 中的所有成员，则可以采用的导入形式是（　　）。

126. 设 m, n 为整型数据，则与 m%n 等价的表达式为（　　）。
127. Python 包含了数量众多的模块，通过（　　）语句，可以导入模块，并使用其定义的功能。
128. 语句 y=x 的含义是（　　）。
129. Python 通过（　　）来区分不同的语句块。
130. x/=x*y+z 等价的语句是（　　）。
131. 在 Python 语句行从解释器提示符后的第（　　）列开始。
132. 在 Python 中，可以在同一行中使用多条语句，语句之间使用（　　）分隔。
133. 在 Python 中，赋值的含义是使变量（　　）一个数据对象。该变量是该数据对象的（　　）。
134. Python 基本数据类型包括数值型、字符串型和（　　）。
135. Python 复合数据类型有列表、元组、（　　）和集合。
136. "4"+"5"的值是（　　）。
137. 'AsDf888'.isalpha() 的值是（　　）。
138. 当 x=0, y=50 时，语句 z=x if x==y else y 执行后，z 的值是（　　）。
139. 执行循环语句 for i in range(1, 5) : pass 后，变量 i 的值是（　　）。
140. 执行下列 Python 语句后输出结果是（　　），循环执行了（　　）次。

```
i=-1
while(i<0) : i*=i
print(i)
```

141. Python 无穷循环 while True: 的循环体中可用（　　）语句退出循环。
142. 下列 Python 语句的运行结果是（　　）。

```
x=True
y=False
z=False
print(x or y and z)
```

143. 执行循环语句 for i in range(1, 5, 2) : print(i) , 循环体执行的次数是（　　）。
144. Python 表达式 1/4+2.75 的值为（　　）。
145. Python 表达式[i for i in range(5) if i%2!=0]的值为（　　）。
146. Python 表达式[i**2 for i in range(3)]的值为（　　）。
147. 循环语句 for i in range(-3, 21, 4) 的循环次数为（　　）。
148. 以下 while 循环的循环次数是___【1】 ___。

```
i=0
while(i<10) :
    if(i<1) : continue
    if(i==5) : break
    i+=1
```

149. 已知 ans='n'，则表达式 ans=='y' or 'Y'的值为（　　）。
150. 判断整数 i 能否同时被 3 和 5 整除的 Python 表达式为（　　）。

151. 循环语句 for i in range(25, -4, -2)循环执行（　　）次。
152. 已知 a=3，b=5，c=6，d=True，表达式 not d or a>=0 and a+c>b+3 值为（　　）。
153. 执行下列 Python 语句，程序运行的结果是（　　）。

```
m=True
n=False
p=True
b1=m| n^p; b2=n| m^p
print(b1, b2)
```

154. Python 语句 16-2*5>7*8/2 or "XYZ"!="xyz" and not(10-6>18/2)值为（　　）。
155. 下列 Python 语句的运行结果是（　　）。

```
x=0
y=True
print(x>y and ' A' <' B' )
```

156. 下列 Python 语句的运行结果为（　　）。

```
for i in range(3) : print(i, end=' ' )
for i in range(2, 5) : print(i, end=' ' )
```

157. 执行下面程序段后，k 值是（　　）。

```
k=1
n=263
while(n) :
    k*=n%10
    n//=10
print(k)
```

158. 表达式 2<=1 and 0 or not 0 的值是（　　）。
159. 对于 if 语句的条件表达式后面或 else 后面的语句块，应将它们（　　）。
160. 执行语句 first, *middles, last=range(6)后，middles 值为（　　）。
161. 执行语句 first, *middles, last=range(6)后，sum(middles) /len(middles)值为（　　）。
162. 列表 L=[1, 2, 3, 4, 5, 6, 7, 8, 9]，则 L[2: 4]的值是（　　）。
163. 列表 L=[1, 2, 3, 4, 5, 6, 7, 8, 9]，则 L[: : 2]的值是（　　）。
164. 列表 L=[1, 2, 3, 4, 5, 6, 7, 8, 9]，则 L[-1]的值是（　　）。
165. 列表 L=[1, 2, 3, 4, 5, 6, 7, 8, 9]，则 L[-1: -1-len(L) : -1]的值是（　　）。
166. 表达式“2 in [1, 2, 3, 4]”的值是（　　）。
167. 表达式 max((1, 2, 3) *2)的值是（　　）。
168. 表达式'Python Program'.count('P')的值是（　　）。
169. 下列语句执行后，s 值为（　　）。

```
s=[1, 2, 3, 4, 5, 6]
s[: 1]=[]
s[: 2]=' a'
s[2: ]=' b'
```

s[2: 3]=[' x' , ' y']
del s[: 1]
170. 下列 Python 语句的输出结果是（　　）。
x=y=[1, 2]
x.append(3)
print(x is y, x==y, end=' ')
z=[1, 2, 3]
print(x is z, x==z, y==z)
171. Python 语句 print(tuple(range(2)),list(range(2)))运行结果是（　　）。
172. 下列语句的运行结果是（　　）。
s=[1, 2, 3, 4]
s.append([5, 6])
print(len(s))
173. 序列元素的编号，从（　　）开始。
174. 访问序列元素的编号，用（　　）括起来。
175. 对于列表 x，x.append(a)等价于（　　）。
176. 下列语句的运行结果是（　　）。
s1=[1, 2, 3, 4]
s2=[5, 6, 7]
print(len(s1+s2))
177. 下列 Python 语句的输出结果是（　　）。
s=['a', 'b']
s.append([1, 2])
s.extend([5, 6])
s.insert(10, 8)
s.pop()
s.remove('b')
s[3:]=[]
s.reverse()
178. 已知 fruits=['apple','banana','pear']，则 print(fruits[-1][-1])结果是（　　）。
179. 已知 fruits=['apple','banana','pear']，则 print(fruits.index('apple'))结果是（　　）。
180. 已知 fruits=['apple','banana','pear']，则 print('Apple' in fruits)结果是（　　）。
181. 下列语句执行后，di['fruit'][1]值为（　　）。
di={'fruit': ['apple', 'banana','orange']}
di['fruit'].append('watermelon')
182. 字典是（　　）的集合。
183. Python 中，s1={1,2,3,5},s2={2,3,5}，则 s1.update(s2)结果为（　　）。
184. Python 中，s1={1,2,3,5},s2={2,3,5}，则 s1.intersection(s2)值为（　　）。
185. Python 中，s1={1,2,3,5},s2={2,3,5}，则 s1.difference(s2)值为（　　）。

186. 下面语句的输出结果是（　　）。

```
list1={}
list1[1]=1
list1['1']=3
list1[1]+=2
sum=0
for k in list1:
    sum+=list1[k]
print(sum)
```

187. {1, 2, 3, 4} &{3, 4, 5} 的值是（　　）。

188. {1,2, 3,4} | {3, 4, 5} 的值是（　　）。

189. {1, 2, 3, 4} -{3, 4, 5}的值是（　　）。

190. 下面语句的输出结果是（　　）。

```
d={1: 'a' , 2: 'b' , 3: 'c' }
del d[1]
d[1]=' x'
del d[2]
print(d)
```

191. 集合是一组无序排列的、（　　）的元素集。

192. 集合包含两种类型，即（　　）。

193. 下列语句执行后的结果是（　　）。

```
d1={1: 'food' }
d2={1: '食品' , 2: '饮料' }
d1.update(d2)
print(d1[1])
```

194. 下列语句执行后的结果是（　　）。

```
d={1: 'x' , 2: 'y' , 3: 'z' }
del d[1]
del d[2]
d[1]='A'
print(len(d))
```

195. 下列语句执行后的结果是（　　）。

```
fruits={'apple' : 3, 'banana' : 4, 'pear' : 5}
fruits['banana']=7
print(sum(fruits.values()))
```

196. Ptyton 语句 print(len({}))的结果，是（　　）。

197. Python 语句 print(set([1, 2, 1, 2, 3]))的结果，是（　　）。

198. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s.split(',')的值为（　　）。

199. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s.rsplit(',', 1)的值为（　　）。

200. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s.partition(',')的值是（　　）。
201. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s.rpartition(',')的值为（　　）。
202. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s3.join('abc')的值为（　　）。
203. 设 s='a,b,c'，s2=('x','y','z')，s3=':' ，则 s3.join(s2)的值为（　　）。
204. 设有 f=lambda x, y: {x: y}，则 f(5, 10)的值是（　　）。
205. 使用关键字（　　）可以在一个函数中设置一个全局变量。
206. 没有 return 语句的函数将返回（　　）。
207. 函数定义时确定的参数称为（　　）。
208. 函数调用时提供的参数称为（　　）。
209. 函数定义以关键字（　　）开始。
210. 函数定义最后以（　　）结束。
211. 在 Python 中，如果异常并未被处理或捕捉，程序就会用（　　）错误信息终止程序的执行。
212. Python 提供了一些异常类，所有异常都是（　　）类的成员。
213. Python 提供（　　)三种方法读取文本文件内容。
214. 下列程序的输出结果是（　　）。

```
counter=1
num=0
def Testvariable() :
    global counter
    for I in (1, 2, 3) : counter+=1
    num=10
Testvariable()
print(counter, num)
```

215. Python 提供了（　　）机制来专门处理程序运行时错误。
216. Python 处理程序运行时错误相应的语句是（　　）。
217. 二进制文件的读取，使用（　　）方法。
218. 二进制文件的写入，使用（　　）方法。
219. Python 的（　　）模块提供了许多文件处理方法。
220. 异常处理程序将可能发生异常的语句块放在（　　)语句中， 紧跟其后可放置若干个对应的 except 语句。如果引发异常，则系统依次检查各个 except 语句，试图找到与所发生异常相匹配的异常类型。
221. 根据文件数据组织形式，Python 文件可分为（　　)文件和二进制文件。
222. 根据文件数据组织形式，Python 程序文件是（　　）文件。
223. 根据文件数据组织形式，一幅 jpg 图像文件是一个（　　)文件。
224. seek(0) 将文件指针定位于（　　）。
225. seek(0,1)将文件指针定位于（　　）。
226. seek(0, 2)将文件指针定位于（　　）。
227. 下列程序的输出结果是（　　）。

```
try:
    print(2/' 0' )
except ZeroDivisionError:
    print('AAA' )
except Exception:
    print('BBB' )
```

228. 在 Python 中，字典和集合都是用一对（　　）作为定界符。

229. 字典的每个元素有两部分组成，即键和（　　），其中键不允许重复。

230. 列表 a = ['name', 'age','sex']和 b = ['Dong' , 38, 'Male']，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（　　）。

231. 假设有一个列表 a，现要求从列表 a 中每 3 个元素取 1 个，并且将取到的元素组成新的列表 b，可以使用语句（　　）。

232. 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（　　）。

233. 是否可以使用 del 命令来删除元组中的部分元素？答案是（　　）。

234. 下面语句的执行结果是（　　）。

```
s = '@ #$'
print(str.split(3*s) )
```

235. 执行 print(1.9-1==0.9)，结果是 False。原因是（　　）。

236. Python 提供了异常机制来专门处理程序运行时错误，相应的语句是（　　）。

237. 列表对象 sort()方法，用来对列表元素进行原地排序，该函数返回值为（　　）。

238. Python 语句 print(1,2,3,sep=','),输出结果为（　　）。

239. 已知“g=lambda x,y=3,z=5:x*y*z”，语句 print(g(1))输出结果为（　　）。

2. 2 答案

1. #； 2. [1, 4, 9, 16, 25]； 3. 7； 4. sqrt()； 5. none；
6. 有序； 7. type()； 8. id()； 9. [1, 2, 3, 1, 2, 3, 1, 2, 3]；
10. ['1', '2', '3']； 11. False； 12. 18； 13. False； 14. True；
15. [6, 7, 9, 11]； 16. [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]； 17. -1； 18. 回车换行；
19. 2； 20. 9； 21. 6； 22. c=dict(zip(a,b))；
23. h e l l o w o r l d !； 24. [1, 4, 7]； 25. [0, 2, 4]； 26. False；
27. 1 : 2 : 3； 28. [111, 33, 2]； 29. 3； 30. 11；
31. 11； 32. (3,)； 33. 3; 34. {1: 2, 2: 3}； 35. items()；
36. [i for i in range(100) if i%13==0]； 37. [3, 5, 7, 1, 2]；
38. [(1, 3), (2, 4)]； 39. [1, 2, 3, 2]； 40. [1, 2]； 41. [2, 4]；
42. c:\test.htm； 43. 65, A； 44. The first:97,the second is 65；
45. a,b,ccc,ddd； 46. 123456； 47. yybcyyb；
48. Helloc corld； 49. ab : efg； 50. True； 51. -1；
52. [('a', 3), ('b', 1), ('d', 4)]； 53. []； 54. [3, 9, 78]；
55. <class 'float'>； 56. 1991； 57. type()； 58. len()；
59. dict(zip(a,b))； 60. True； 61. get()； 62. [2, 4]；
63. shuffle()； 64. def； 65. {2, 3, 4, 5, 6, 8}；
66. //； 67. [1, 2, 3, 1, 2, 3, 1, 2, 3]； 68. 18；
69. -1； 70. [1, 3, 5, 7]； 71. 逗号（;）；
72. True； 73. break； 74. 'ab'； 75. 'r'；
76. def； 77. 8； 78. input()和 print()； 79. {1, 2, 3, 4}；
80. None； 81. 2； 82. False； 83. []； 84. len()；
85. int； 86. and； 87. dict； 88. string[0]； 89. import；
90. expect； 91. read()； 92. add； 93. -1； 94. 1.0；
95. from math import *； 96. 复数； 97. import math. from math import *；
98. fsum()； 99. gcd()； 100. trunc()； 101. ceil()；
102. floor()； 103. pi； 104. e； 105. d；
106. de； 107. abcde； 108. defg； 109. aceg；
110. gfedcba； 111. /； 112. //；
113. 2.25； 114. 0.5； 115. 2.0； 116. 4 3；
117. \； 118. RED HAT； 119. RED HAT； 120. Red Hat；
121. red cat； 122. AAA-BBB!； 123. 20； 124. ['A', 'A', 'A']；
125. from m import * 或 import m； 126. m-m//n*n 或 m-n*(m//n)；
127. import； 128. 将 x 赋值给 y； 129. 缩进对齐； 130. x=x/(x*y+z)；
131. 1 或 一； 132. ; 或 分号； 133. 指向. 别名；
134. 布尔型； 135. 字典； 136. 45； 137. False；
138. 50； 139. 4； 140. 1. 1； 141. break；

142. True；　143. 2；　144. 3.0；　145. [1, 3]；
146. [0, 1, 4]；　147. 6；　148. 无限；
149. 'Y'；　150. i%3==0 and i%5==0；　151. 15；
152. True；　153. True False；　154. True；　155. False；
156. 0 1 2 2 3 4；　157. 36；　158. True；　159. 缩进对齐；
160. [1, 2, 3, 4]；　161. 2.5；　162. [3, 4]；　163. [1, 3, 5, 7, 9]；
164. 9；　165. [9, 8, 7, 6, 5, 4, 3, 2, 1]；　166. True；
167. 3；　168. 2；　169. [4, 'x', 'y']；
170. True True False True True；　171. (0, 1) [0, 1]；　172. 5；
173. 0；　174. []；　175. x=x+[a]；
176. 7；　177. [5, [1, 2], 'a']；　178. r；　179. 0；　180. False；
181. banana；　182. 无序的“关键字: 值”对，或 key-value pair；
183. None；　184. {2, 3, 5}；　185. {1}；　186. 6；　187. {3, 4}；
188. {1, 2, 3, 4, 5}；　189. {1, 2}；　190. {3: 'c', 1: ' x'}；　191. 不重复；
192. 可变集合和不可变集合；　193. 食品；　194. 2；
195. 15；　196. 0；　197. {1, 2, 3}；　198. ['a', 'b', 'c']；
199. ['a,b', 'c']；　200. ('a', ',', 'b,c')；　201. ('a,b', ',', 'c')；
202. a:b:c；　203. x:y:z；　204. {5: 10}；　205. global；
206. None；　207. 形式参数，形参；　208. 实际参数，实参；
209. def；　210. 冒号；　211. SystemExit；　212. Exception；
213. read()、readline()、readlines()；　214. 4 0；　215. 异常；
216. try-except；　217. read()；　218. write()；　219. os；
220. try；　221. ASCII 文件，或文本文件；
222. ASCII 文件，或文本文件；　223. 二进制文件；
224. 起始位置；　225. 当前位置；　226. 文件末尾；　227. BBB；
228. 大括号；　229. 值；　230. c = dict(zip(a,b))；　231. b = a[: : 3]；
232. [5 for i in range(10)]；　233. 不可以；
234. ['@', '#$@', '#$@', '#$']；
235. 浮点数保存的是一个近似值，而非精确值；　236. try…except；
237. None；　238. 1,2,3；　239. 15。

3 简答题

3. 1 习题

1. 简述 Python 中列表与元组的区别。
2. 简述定义函数的规则。
3. 简述__new__和__init__的区别。
4. 简述 read、readline、readlines 之间的区别。
5. 去掉 old_list = [1,1,1,3,4] 中的重复元素。
6. 用两个元素之间有对应关系的 list，构造一个 dict。
7. 什么是封装？在 Python 中如何实现封装？
8. 什么是继承？在 Python 中如何实现继承？
9. 什么是多态？在 Python 中如何实现多态？
10. 简述生成器、迭代器、可迭代对象及其应用场景。
11. 列举 Python 相较于 Java、PHP、C、C#、C++等其他语言的特点。
12. 简述解释型和编译型编程语言。
13. 请列举至少 5 个 PEP8 规范。
14. Python 递归的最大深度。
15. 简述 ASCII、Unicode、UTF-8. GBK 的区别。
16. 简述字节码和机器码的区别。
17. 简述三元运算规则及其应用场景。
18. 文件操作时，xreadlines 和 readlines 的区别是什么？
19. 列举布尔值为 False 的常见值。
20. 列举字符串、列表、元组、字典每个常用的 5 个方法。
21. 简述 lambda 表达式格式及其应用场景。
22. pass 的作用是什么？
23. arg 和*kwarg 作用是什么？
24. 简述 is 和==的区别。
25. 简述 Python 的深浅拷贝及其应用场景。
26. Python 垃圾回收机制是什么？
27. 列举 Python 的可变类型和不可变类型。
28. 列举至少 8 个常用模块。
29. re 模块的 match 和 search 区别是什么？
30. 什么是正则的贪婪匹配？
31. 列举常见的内置函数。
32. 简述 filter、map、reduce 的作用。
33. 请使用一行代码实现九九乘法表。

34. 如何安装第三方模块？以及用过哪些第三方模块？
35. 如何实现"1,2,3" 变成 ['1','2','3']？
36. 如何实现['1', '2', '3']变成[1,2,3]？
37. 比较 a = [1,2,3] 和 b = [(1),(2),(3)] 以及 b = [(1,),(2,),(3,)] 的区别。
38. 如何用一行代码生成[1,4,9,16,25,36,49,64,81,100]？
39. 如何通过代码实现删除列表中重复的值?
40. 如何在函数中设置一个全局变量?
41. logging 模块的作用是什么？它的应用场景有哪些？
42. 求结果。
    a. [i % 2 for i in range(10)]
    b. (i % 2 for i in range(10))
43. 常用字符串的格式化方法有哪几种？
44. 下面程序运行的结果是什么？

```
def num():
    return [lambda x: i*x for i in range(4)]
print([m(2) for m in num()])
```

45. 以下程序的运行结果是什么？

```
v = dict.fromkeys(['k1','k2'],[])
v['k1'].append(666)
print(v)
v['k1'] = 777
print(v)
```

46. 请输出字符串变量 name 对应的值的前 3 个字符。
47. 请输出字符串变量 name 对应的值的后 2 个字符。
48. 移除字符串变量 name 对应的值两边的空格，并输出移除后的内容。
49. 判断字符串变量 name 对应的值 a 出现的次数，并输出结果。
50. 判断字符串变量 name 对应的值，以 a 进行分割，并输出结果。
51. 将字符串变量 name 对应的值 a 替换成 w，并输出结果。
52. 计算列表平均值。
53. 列表转字符串如何操作？
54. 查找最大值如何操作？
55. 检查是否全是数字如何操作？
56. 反转字符串如何操作？
57. 对一个列表中所有的元素求平方。
58. 判断是否为素数如何操作？
59. 字符串去重如何操作？
60. 计算字符串出现的次数。
61. 文件读取所有行如何操作？
62. 快速排序如何操作？

63. 生成斐波那契数列如何操作？
64. 字典键值对交换如何操作？
65. 求两个集合的交集。
66. 将字符串转换为整型列表如何操作？
67. 生成随机数如何操作？
68. 混淆字符串的字母顺序如何操作？
69. 将秒转换为时分秒如何操作？
70. 判断闰年如何操作？
71. 扁平化嵌套列表如何操作？
72. 并行处理列表如何操作？
73. 使用装饰器简化代码如何操作？
74. 利用生成器表达式节省内存如何操作？
75. 错误处理的简洁方式是什么？
76. 使用列表推导式和条件判断如何操作？
77. 简单介绍自定义迭代器及其操作。
78. 快速交换变量值如何操作？
79. 列表推导式简化循环如何操作？
80. 字典推导式构建字典如何操作？
81. 简单介绍三元条件运算符及其操作。
82. 使用 zip()合并列表。
83. 生成器表达式节省内存如何操作？
84.列表排序的方法有哪些？
85. 使用 enumerate()遍历带索引的列表如何操作？
86. 如何使用集合去除重复项？
87. 如何进行字符串分割和连接？
88. 使用 any()和 all()检查序列如何操作？
89. 如何使用一行代码反转列表？
90. 如何使用 map()函数将一个函数应用于序列？
91. 如何利用 filter()筛选序列？
92. 文本统计分析。假设有一个长文本文件，你想找出其中最常出现的单词，该如何操作？
93. 如何快速统计列表元素出现次数？
94. 如何实现列表一键去重？
95. 并行处理文件如何操作？
96. 如何实现简洁的日期时间格式化？
97. 如何实现优雅的列表拼接？
98. 如何实现一键字典排序？
99. 如何通过高级迭代方法，同时遍历两个列表？
100. 如何实现简易错误处理？
101. 如何通过列表推导式快速生成新列表？

102. 如何实现轻松读写 CSV 文件？
103. 简述核心概念 1：变量与数据类型。
104. 简述核心概念 2：运算符与表达式。
105. 简述核心概念 3：条件判断语句。
106. 简述核心概念 4：循环结构。
107. 简述核心概念 5：列表与元组。
108. 简述核心概念 6：字典与集合。
109. 简述核心概念 7：函数定义与调用。
110. 简述核心概念 8：模块与导入。
111. 简述核心概念 9：错误与异常处理。
112. 简述核心概念 10：面向对象编程简介。
113. 简述列表推导式。
114. 简述生成器表达式。
115.简述 lambda 函数。
116. 简述 map 函数。
117. 简述 filter 函数。
118. 简述 sorted 函数。
119. 简述 enumerate 函数。
120. 简述 zip 函数。
121. 简述 itertools 模块。
122. 简述 functools 模块。
123. 简述 operator 模块。
124. 简述内置函数 len()。
125. 简述内置函数 type()。
126. 简述内置函数 isinstance()。
127. 简述内置函数 dir()。
128. 简述内置函数 id()。
129. 简述内置函数 del。
130. 简述内置函数 globals()与 locals()。
131. 简述内置函数 all()与 any()。
132. 简述内置函数 enumerate()。
133. 简述内置函数 zip()。
134. 简述内置函数 format()。
135. 简述内置函数 join()。
136. 简述内置函数 split()。
137. 简述内置函数 strip()。
138. 简述内置函数 sorted()。
139. 简述内置函数 reversed()。
140. 简述内置函数 set()与 frozenset()。

141. 简述内置函数 assert。
142. 简述内置函数 traceback。
143. 简述内置函数 sys.exc_info()。
144. 简述内置函数 map()。
145. 简述内置函数 filter()。
146. 简述内置函数 reduce()。
147. 简述内置函数 lambda。
148. 简述内置函数__str__与__repr__。
149. 简述内置函数__getattr__。
150. 简述内置函数@property。
151. 简述内置函数 importlib。
152. 简述内置函数 pkgutil。
153. 简述内置函数 sys.path。

3. 2 答案

1. 简述 Python 中列表与元组的区别。

（1）列表使用方括号[ ]表示，元素之间使用逗号分隔。

（2）元组使用圆括号（）表示，元素之间也使用逗号分隔。

（3）列表是可变的，可以进行增删改操作。

（4）元组是不可变的，一旦创建就无法修改。

（5）列表的元素可以是不同类型的，元组的元素通常是同一类型的。

2. 简述定义函数的规则。

（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。

3. 简述__new__和__init__的区别。

（1）__new__是一个静态方法，而__init__是一个实例方法。

（2）__new__方法会返回一个创建的实例，而__init__什么都不返回。

（3）只有在__new__返回一个 cls 的实例时，后面的__init__才能被调用。

（4）当创建一个新实例时调用__new__，初始化一个实例时用__init__。

4. 简述 read、readline、readlines 之间的区别。

（1）read 读取整个文件。

（2）readline 读取下一行，使用生成器方法。

（3）readlines 读取整个文件到一个迭代器以供我们遍历。

5. 去掉 old_list = [1,1,1,3,4] 中的重复元素

```
new_list = list(set(old_list))
```

6. 用两个元素之间有对应关系的 list，构造一个 dict。

```
names = ['jianpx', 'yue']
ages = [23, 40]
m = dict(zip(names,ages))
```

7. 什么是封装？在 Python 中如何实现封装？

封装是面向对象编程中的一种概念， 它指的是将数据和方法封装到一个类中， 通过接口来控制对数据的访问。 在 Python 中，可以通过使用私有变量和私有方法来实现封装。

私有变量可以通过在变量名前面加上两个下划线来定义，私有方法也可以通过在方法名前面加上两个下划线来定义。

8. 什么是继承？在 Python 中如何实现继承？

继承是面向对象编程中的一种概念，它指的是一个类可以派生出子类，子类可以继承父类的属性和方法。

在 Python 中，可以通过在定义子类时将父类作为参数传递给子类来实现继承。子类可以直接访问父类的属性和方法，并可以重载父类的方法。

9. 什么是多态？在 Python 中如何实现多态？

多态是面向对象编程中的一种概念，它指的是同一个方法调用可以有不同的行为，这取决于调用这个方法的对象或者类。

在 Python 中，可以通过定义一个抽象基类来实现多态。 抽象基类定义了一个统一的接口，具体子类可以根据自己的需要来实现这个接口， 从而实现相同的方法调用有不同的行为。

10. 简述生成器、迭代器、可迭代对象及其应用场景。

（1）生成器：在 Python 中，一边循环一边计算的机制，称为生成器（generator），通过 next()取值，两种表现形式：将列表生成式的[]改为()；含有 yield 关键字的函数。应用场景：优化代码，节省内存。

（2）迭代器：是访问集合元素的一种方式。迭代器同时实现了__iter__和__next__方法。

（3）可迭代对象：只要实现了__iter__方法的对象就是可迭代对象。

11. 列举 Python 相较于 Java、PHP、C、C#、C++等其他语言的特点。

（1）Python 代码简介、明确、优雅、简单易懂。

（2）开发效率高。

（3）可扩展性强。

12. 简述解释型和编译型编程语言。

（1）解释型：在执行程序时，计算机才一条一条地将代码解释成机器语言给计算机来执行。

（2）编译型：是把源程序的每一条语句都编译成机器语言，并保存成二进制文件，这样计算机运行该程序时可以直接以机器语言来运行此程序，运行速度很快。

13. 请列举至少 5 个 PEP8 规范。

（1）缩进：每一级 4 个缩进。

连续跨行应该使用圆括号或大括号或者使用悬挂缩进。

（2）代码长度约束。

一行列数：PEP8 规定最大为 79 列，如果拼接 url 很容易超限

一个函数：不可以超过 30 行；直观来讲就是完整显示一个函数一个屏幕就够了，不需要上下拖动。

一个类：不要超过 200 行代码，不要超过 10 个方法

一个模块：不要超过 500 行

（3）import。

不要在一句 import 中引用多个库

（4）命名规范。

（5）注释。

总体原则，错误的注释不如没有注释。所以当一段代码发生变化时，第一件事就是要修改注释。

14. Python 递归的最大深度。

一般计算机默认的最大递归深度在 1000 左右，python 最大递归深度一般在 4000 左右，跟计算机的性能有关系，这个数不是一个定数，可通过以下方式测试。

```
import sys
print(sys.getrecursionlimit())
print(sys.setrecursionlimit(10000))
```

15. 简述 ASCII、Unicode、UTF-8、GBK 的区别。

（1）ASCII 码：使用一个字节编码，所以它的范围基本是只有英文字母、数字和一些特殊符号，只有 256 个字符。

（2）Unicode：能够表示全世界所有的字节

（3）GBK：只用来编码汉字，GBK 全称《汉字内码扩展规范》，使用双字节编码。

（4）UTF-8：一种针对 Unicode 的可变长度字符编码，又称万国码。

16. 简述字节码和机器码的区别。

（1）机器码：是电脑 CPU 直接读取运行的机器指令，运行速度最快，但是非常晦涩难懂。

（2）字节码：是一种中间状态（中间码）的二进制代码（文件）。需要直译器转译后才能成为机器码。

17. 简述三元运算规则及其应用场景。

（1）规则：为真时的结果 if 判定条件 else 为假时的结果。

（2）应用场景：在赋值变量的时候，可以直接加判断，然后赋值。

18. 文件操作时，xreadlines 和 readlines 的区别是什么？

（1）readlines 返回一个 list。

（2）xreadlines 方法返回一个生成器。

19. 列举布尔值为 False 的常见值。

0、[]、()、{}、''、False、None。

20. 列举字符串、列表、元组、字典每个常用的 5 个方法。

（1）字符串：repleace,strip,split,reverse,upper,lower,join。

（2）列表：append,pop,insert,remove,sort,count,index。

（3）元组：index,count,__len__(),__dir__()。

（4）字典：get,keys,values,pop,popitems,clear,update,items。

21. 简述 lambda 表达式格式及其应用场景。

（1）表达式格式：lambda 后面跟一个或多个参数，紧跟一个冒号，以后是一个表达式。冒号前是参数，冒号后是返回值。例如：lambda x : 2x。

（2）应用场景：经常与一些内置函数相结合使用，比如说 map(),filter(),sorted(),reduce()等。

22. pass 的作用是什么？

（1）空语句 do nothing。

（2）保证格式完整。

（3）保证语义完整。

23. arg 和*kwarg 作用是什么？

万能参数，解决了函数参数不固定的问题。

*arg：会把位置参数转化为 tuple。

**kwarg：会把关键字参数转化为 dict。

24. 简述 is 和==的区别。

（1）is：判断内存地址是否相等。

（2）==：判断数值是否相等。

25. 简述 Python 的深浅拷贝及其应用场景？

（1）copy()：浅 copy，浅拷贝指仅仅拷贝数据集合的第一层数据。

（2）deepcopy()：深 copy，深拷贝指拷贝数据集合的所有层。

26. Python 垃圾回收机制是什么？

Python 采用的是引用计数机制为主，标记-清除和分代收集（隔代回收、分代回收）两种机制为辅的策略：

（1）计数机制。

Python 的 GC 模块主要运用了引用计数来跟踪和回收垃圾。在引用计数的基础上，还可以通过“标记-清除”解决容器对象可能产生的循环引用的问题。通过分代回收以空间换取时间进一步提高垃圾回收的效率。

（2）标记-清除：标记-清除的出现打破了循环引用，也就是它只关注那些可能会产生循环引用的对象。缺点：该机制所带来的额外操作和需要回收的内存块成正比。

（3）隔代回收原理：将系统中的所有内存块根据其存活时间划分为不同的集合，每一个集合就成为一个“代”，垃圾收集的频率随着“代”的存活时间的增大而减小。也就是说，活得越长的对象，就越不可能是垃圾，就应该减少对它的垃圾收集频率。那么如何来衡量这个存活时间：通常是利用几次垃圾收集动作来衡量，如果一个对象经过的垃圾收集次数越多，可以得出：该对象存活时间就越长。

27. 列举 Python 的可变类型和不可变类型。

（1）不可变类型（数字、字符串、元组、不可变集合）。

（2）可变类型（列表、字典、可变集合）。

28.列举至少 8 个常用模块。

os,sys,time,random,re,hashlib,logging,json,pickle....

29. re 模块的 match 和 search 区别是什么？

（1）match：从字符串的开头位置匹配，必须以此为开头。

（2）search：从开头开始查，找到符合的就返回结果。

30. 什么是正则的贪婪匹配？

正则表达式一般趋向于最大长度匹配。

31. 列举常见的内置函数。

map,filter,zip,len,bin,oct,hex,int,float,bool,sum,min,max,str,list,tuple,dict,range,next,hash,help,id。

32. 简述 filter、map、reduce 的作用。

（1）filter(function，iterable)：过滤函数。

（2）map(function，iterable)：循环函数。

（3）reduce(function, iterable)：累积函数。

33. 请使用一行代码实现九九乘法表。

lis = ['%s%s=%s'%(i,j,ij) for i in range(1,10) for j in range(i,10)]

34. 如何安装第三方模块？以及用过哪些第三方模块？

pip3 imstall 模块名

django，Matplotlib，Tornado，PyGame

35. 如何实现"1,2,3"变成['1','2','3']？

a = "1,2,3"

li = a.split(',')

36. 如何实现['1', '2', '3']变成[1,2,3]？

li = ['1','2','3']

lis = list(map(lambda x:int(x),li))

37. 比较 a = [1,2,3] 和 b = [(1),(2),(3)] 以及 b = [(1,),(2,),(3,)] 的区别。

a = [1,2,3]正常的列表。

b = [(1),(2),(3)] 虽然列表的每个元素加上了括号，但是当括号内只有一个元素并且没有逗号时，其数据类型是元素本身的数据类型。

b = [(1,),(2,),(3,)]列表中的元素类型都是元组类型。

38. 如何用一行代码生成[1,4,9,16,25,36,49,64,81,100]？

li = [x*x for x in range(1,11)]

39. 如何通过代码实现删除列表中重复的值？

li = [1, 1, 1, 23, 3, 4, 4]

new_li = list(set(li))

new_li.sort(key=li.index)

40. 如何在函数中设置一个全局变量？

使用 Python 的内置语法 globals 全局变量。

41. logging 模块的作用？它的应用场景有哪些？

logging 模块的作用：

（1）程序调试。

（2）了解软件程序运行情况，是否正常。

（3）软件程序运行故障分析与问题定位。

应用场景：网站的运维工作，程序实时监控。

42. 求结果：

a. [i % 2 for i in range(10)] ===>[0,1,0,1,0,1,0,1,0,1]

b. (i % 2 for i in range(10))===>返回一个生成器的内存地址

43. 常用字符串的格式化方法有哪几种？

（1）%s %d

（2）format 格式化输出

（3）print(f'内容{变量名}')

44. 下面程序运行的结果是什么？

```
def num():
    return [lambda x: i*x for i in range(4)]
print([m(2) for m in num()])
```

程序运行结果是：[6, 6, 6, 6]

解析：问题的本质在与 python 中的属性查找规则，Python 在查找"名称"时，是按照 LEGB 规则查找的： Local-->Enclosed-->Global-->Built in。

（1）Local 指的就是函数或者类的方法内部。

（2）Enclosed 指的是嵌套函数（一个函数包裹另一个函数，闭包）。

（3）Global 指的是模块中的全局变量。

（4）Built in 指的是 Python 为自己保留的特殊名称。

在上面的例子中，i 就是在闭包作用域（enclousing），而 Python 的闭包是迟绑定，这意味着闭包中用到的变量的值，是在内部函数被调用时查询得到的所以：[lambda x: i*x for i in range(4)]打印出来是含有四个内存地址的列表，每个内存地址中的 i 在本内存中都没有被定义，而是通过闭包作用域中的 i 值，当 for 循环执行结束后，i 的值等于 3，所以在执行[m(2) for m in num()]时，每个内存地址中的 i 值等于 3，当 x 等于 2 时，打印出来的结果都是 6，从而得到结果[6, 6, 6, 6]。

45. 以下程序的运行结果是什么？

```
v = dict.fromkeys(['k1','k2'],[])
v['k1'].append(666)
print(v)
v['k1'] = 777
print(v)
```

程序运行结果：

```
{'k1': [666], 'k2': [666]}
{'k1': 777, 'k2': [666]}
```

46. 请输出字符串变量 name 对应的值的前 3 个字符。

```
print(name[: 3] )
```

47. 请输出字符串变量 name 对应的值的后 2 个字符。

```
print(name[-2: ] )
```

48. 移除字符串变量 name 对应的值两边的空格，并输出移除后的内容。

```
print(name.strip(‘’))
```

49. 判断字符串变量 name 对应的值 a 出现次数，并输出结果。

```
print(name.count(‘a’))
```

50. 判断字符串变量 name 对应的值以 a 进行分割，并输出结果。

```
print(name. split(‘a’ ) )
```

51. 将字符串变量 name 对应的值 a 替换成 w，并输出结果。

```
print(name.replace(‘a’,’w’))
```

52. 计算列表平均值。

```
numbers = [1, 2, 3, 4, 5]
```

```
average = sum(numbers) / len(numbers)
```
解释：使用 sum()计算列表总和，len()获取长度，然后相除得到平均值。Python 以其简洁优雅著称，能够用最少的代码行数实现强大的功能。

53. 列表转字符串如何操作？

```
my_list = ['Hello', 'world']
stringified = ' '.join(my_list)
```
解释：join()方法用于将列表中的元素连接成字符串，中间用指定字符（这里是空格）分隔。

54. 查找最大值如何操作？

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6]
max_value = max(numbers)
```
解释：直接使用 max()函数找到列表中的最大值。

55. 检查是否全是数字如何操作？

```
s = "12345"
is_all_digits = all(c.isdigit() for c in s)
```
解释：all()结合生成器表达式，检查序列中所有元素是否满足条件，这里检查每个字符是否为数字。

56. 反转字符串如何操作？

```
my_string = "hello"
reversed_string = my_string[::-1]
```
解释：切片操作[::-1]用于反转字符串。

57. 对一个列表中所有的元素求平方。

```
numbers = [1, 2, 3]
squared = [n**2 for n in numbers]
```
解释：列表推导式，对列表中的每个元素进行平方运算。

58. 判断是否为素数如何操作？

```
def is_prime(n):
    return all(n % i for i in range(2, int(n**0.5) + 1)) and n > 1
```
解释：使用 all()和生成器表达式判断 2 到根号 n 之间是否有因子。

59. 字符串去重如何操作？

```
my_string = "hello"
unique_chars = ''.join(sorted(set(my_string)))
```
解释：先用 set()去重，再排序，最后用 join()合并成字符串。

60. 计算字符串出现的次数。

```
text = "hello world"
count = text.count('o')
```
解释：count()方法统计子字符串在原字符串中出现的次数。

61. 文件读取所有行如何操作？

```
with open('example.txt', 'r') as file:
    lines = file.readlines()
```

解释：使用上下文管理器安全读取文件，readlines()读取所有行到列表中。
62. 快速排序如何操作？
```
def quick_sort(lst):
    return sorted(lst)
```
解释：虽然不是“一行内”完成，但使用内置的 sorted()函数快速排序，简洁有效。
63. 生成斐波那契数列如何操作？
```
fibonacci = lambda n: [0, 1] + [fibonacci(i - 1)[-1] + fibonacci(i - 2)[-1] for i in range(2, n)]
```
解释：递归定义斐波那契数列，注意效率较低，适用于教学目的。
64. 字典键值对交换如何操作？
```
my_dict = {'a': 1, 'b': 2}
swapped = {v: k for k, v in my_dict.items()}
```
解释：字典推导式，交换键值对。
65. 求两个集合的交集。
```
set1 = {1, 2, 3}
set2 = {2, 3, 4}
intersection = set1 & set2
```
解释：使用集合的交集运算符&。
66. 将字符串转换为整型列表如何操作？
```
s = "12345"
int_list = list(map(int, s))
```
解释：结合 map()和 list()，将字符串每个字符转换为整数并列表化。
67. 生成随机数如何操作？
```
import random
random_number = random.randint(1, 100)
```
解释：导入 random 模块，生成指定范围内的随机整数。
68. 混淆字符串的字母顺序如何操作？
```
from random import shuffle
s = "hello"
shuffled = ''.join(shuffle(list(s), random.random))
```
解释：将字符串转为列表，打乱顺序，再合并回字符串。
69. 将秒转换为时分秒如何操作？
```
seconds = 3661
hours, remainder = divmod(seconds, 3600)
minutes, seconds = divmod(remainder, 60)
time_format = f"{hours}:{minutes}:{seconds}"
```
解释：使用 divmod()函数进行多次除法和取余操作，格式化输出时间。
70. 判断闰年如何操作？
```
year = 2020
is_leap = year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
```

解释：根据闰年的规则，使用逻辑运算符组合判断条件。

71. 扁平化嵌套列表如何操作？

```
nested_list = [[1, 2], [3, 4, 5], [6]]
flattened = [item for sublist in nested_list for item in sublist]
```

解释：双层列表推导式，遍历每个子列表并展开其元素。

通过这些个实例，不仅学会了如何用 Python 的一行代码解决实际问题，还深入了解了 Python 的几个核心概念：列表、字符串操作、集合、字典、循环、条件语句、函数和模块的使用。这不仅提升了编码技巧，也更加欣赏 Python 的简洁之美。

72. 并行处理列表如何操作？

使用 concurrent.futures 模块可以并行执行函数，尽管严格来说不完全是一行代码，但可以简化并行计算的复杂性。

```
from concurrent.futures import ThreadPoolExecutor
def square(n):
    return n ** 2
numbers = [1, 2, 3, 4]
with ThreadPoolExecutor() as executor:
    results = list(executor.map(square, numbers))
```

解释：并行计算列表中每个数字的平方，适合大量数据处理。

73. 使用装饰器简化代码如何操作？

装饰器可以增强函数或类的功能，是 Python 的一大特色。

```
def my_decorator(func):
    def wrapper(*args, **kwargs):
        print("Something is happening before the function is called.")
        result = func(*args, **kwargs)
        print("Something is happening after the function is called.")
        return result
    return wrapper
@my_decorator
def say_hello(name):
    return f"Hello, {name}!"
print(say_hello("World"))
```

解释：定义一个装饰器并在函数上使用它，添加额外的行为而无需修改原始函数。

74. 利用生成器表达式节省内存如何操作？

当处理大数据流时，生成器比列表更高效。

```
data = (x for x in range(1000000) if x % 2 == 0)
for even_number in data:
    print(even_number)
```

解释：生成器表达式按需生成值，减少内存占用。

75. 错误处理的简洁方式是什么？

即使在一行内，也可以优雅地处理异常。

```
result = None
try:
    result = 10 / 0
except ZeroDivisionError:
    result = "Can't divide by zero."
print(result)
```

解释：基本的错误处理，确保程序不会因为未捕获的异常而崩溃。

76. 使用列表推导式和条件判断如何操作？

结合条件判断的列表推导式，可以简洁地筛选数据。

```
numbers = [1, 2, 3, 4, 5]
even_numbers = [x for x in numbers if x % 2 == 0]
```

解释：一行代码完成筛选偶数的任务，清晰易懂。

77. 简单介绍自定义迭代器及其操作。

Python 允许自定义迭代行为，尽管实现细节通常不止一行，但核心逻辑可以很简洁。

```
class CountDown:
    def __init__(self, start):
        self.start = start
    def __iter__(self):
        return self
    def __next__(self):
        if self.start <= 0:
            raise StopIteration
        self.start -= 1
        return self.start + 1
for number in CountDown(5):
    print(number)
```

解释：创建一个倒计时迭代器，展示了迭代器协议的基本实现。

总结

通过这些示例，我们不仅展示了 Python 如何用一行代码实现复杂的任务，还深入探讨了 Python 的高级特性，如装饰器、生成器、并行处理和错误处理等。这些技巧不仅能提升你的代码效率，还能增强代码的可读性和维护性。

78. 快速交换变量值如何操作？

在 Python 里，你可以用一行代码就完成两个变量值的交换。这招特别酷，省去了临时变量，简洁又高效。

```
a, b = 10, 20
a, b = b, a  # 交换 a 和 b 的值
print(a, b)  # 输出: 20 10
```

79. 列表推导式简化循环如何操作？

列表推导式是 Python 中的神器，它能让你用一行代码搞定原本需要多行循环才能完成的任务。比如，快速创建一个包含平方数的列表：

```
squares = [x**2 for x in range(10)]
print(squares)  # 输出: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

80. 字典推导式构建字典如何操作？

不仅列表，字典也有自己的推导式。想象一下，你需要构建一个字典，键是字母，值是字母的位置。这在一行代码里就能搞定：

```
char_positions = {char: idx for idx, char in enumerate('abcdefg')}
print(char_positions)  # 输出: {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6}
```

81. 简单介绍三元条件运算符及其操作。

在 Python 中，你可以用一行代码实现条件判断。这招叫做“三元条件运算符”，特别适合处理简单的 if-else 情况。

```
result = "True" if 5 > 3 else "False"
print(result)  # 输出: True
```

82. 使用 zip()合并列表

有时候我们需要将两个列表按位置组合成一个新的列表，这时 zip()函数就是你的救星。

```
list1 = ['apple', 'banana', 'cherry']
list2 = ['red', 'yellow', 'red']
combined_list = list(zip(list1, list2))
print(combined_list)  # 输出: [('apple', 'red'), ('banana', 'yellow'), ('cherry', 'red')]
```

83. 生成器表达式节省内存如何操作？

列表推导式很棒，但如果你处理的是大数据集，生成器表达式能帮你节省大量内存。它们在需要时才计算值，而不是一次性全部加载。

```
big_numbers = (x for x in range(1000000))
for number in big_numbers:
    print(number)  # 这里只打印了第一个数，因为生成器是懒惰计算的
```

84. 列表排序的方法有哪些？

列表排序可以变得非常灵活，只需一行代码，你就可以按照自定义规则排序。

```
names = ['Zoe', 'Adam', 'Charlie', 'Bella']
sorted_names = sorted(names, key=lambda name: name[-1])
print(sorted_names)  # 输出: ['Adam', 'Charlie', 'Bella', 'Zoe']
```

85. 使用 enumerate()遍历带索引的列表如何操作？

当你需要在循环中同时获取元素及其索引时，enumerate()函数是最佳选择。

```
fruits = ['apple', 'banana', 'cherry']
for index, fruit in enumerate(fruits):
    print(f"{index}: {fruit}")
# 输出:
# 0: apple
```

```
# 1: banana
# 2: cherry
```

86. 如何使用集合去除重复项？

集合是 Python 中的另一种数据类型，用于存储不重复的元素。一行代码可以去重。

```
numbers = [1, 2, 2, 3, 4, 4, 4, 5]
unique_numbers = list(set(numbers))
print(unique_numbers)  # 输出: [1, 2, 3, 4, 5]
```

87. 如何进行字符串分割和连接？

在处理文本时，字符串的分割和连接是家常便饭。Python 的 split()和 join()方法让这个过程变得异常简单。

```
sentence = "Hello, world! This is a test."
words = sentence.split()
joined_words = '-'.join(words)
print(joined_words)  # 输出: Hello,-world!-This-is-a-test.
```

88. 使用 any()和 all()检查序列如何操作？

any()和 all()函数可以帮助你快速检查序列中所有或任意元素是否满足条件。

```
bools = [True, False, True]
any_true = any(bools)  # 检查是否有 True
all_true = all(bools)  # 检查是否全为 True
print(any_true, all_true)  # 输出: True False
```

89. 如何使用一行代码反转列表？

反转列表是常见的操作，但在 Python 中，你完全可以用一行代码搞定。

```
numbers = [1, 2, 3, 4, 5]
reversed_numbers = numbers[::-1]
print(reversed_numbers)  # 输出: [5, 4, 3, 2, 1]
```

90. 如何使用 map()函数将一个函数应用于序列？

map()函数允许你将一个函数应用于序列中的每个元素，非常高效。

```
def square(x):
    return x ** 2
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(square, numbers))
print(squared_numbers)  # 输出: [1, 4, 9, 16, 25]
```

91. 如何利用 filter()筛选序列？

与 map()类似，filter()函数用于从序列中筛选出符合条件的元素。

```
def is_even(x):
    return x % 2 == 0
numbers = [1, 2, 3, 4, 5]
even_numbers = list(filter(is_even, numbers))
print(even_numbers)  # 输出: [2, 4]
```

92. 文本统计分析。假设有一个长文本文件，你想找出其中最常出现的单词，该如何操作？
利用上面学到的技巧，我们可以轻松实现：

```
with open('textfile.txt', 'r') as file:
    text = file.read().replace('\n', ' ').lower()  # 读取文件，转换为小写，替换换行符
    words = text.split()  # 分割单词
    word_counts = {word: words.count(word) for word in words}  # 计算每个单词的出现次数
    most_common_word = max(word_counts, key=word_counts.get)  # 找到出现次数最多的单词
    print(most_common_word, word_counts[most_common_word])  # 输出结果
```

这段代码展示了如何结合使用文件操作、字符串方法、字典推导式以及 max()函数来解决实际问题。

93. 如何快速统计列表元素出现次数？
你知道吗？不用循环，一行代码就能搞定元素计数。

```
numbers = [1, 2, 2, 3, 3, 3]
counts = {num: numbers.count(num) for num in set(numbers)}
print(counts)
```

这段代码首先用 set(numbers)去除重复元素，然后通过字典推导式快速统计每个元素出现的次数，超方便。

94. 如何实现列表一键去重？
遇到重复的列表元素，别急着一个一个删除，看这：

```
unique_list = list(set(my_list))
```

简单粗暴，利用 set 的特性直接去重，再转回列表，一气呵成。

95. 并行处理文件如何操作？
想要加速文件读取或处理？多线程来帮忙。

```
from concurrent.futures import ThreadPoolExecutor
def process_file(file):
    # 假设这是处理文件的函数
    pass
files = ['file1.txt', 'file2.txt', ...]
with ThreadPoolExecutor() as executor:
    executor.map(process_file, files)
```

这样，文件处理就并行起来了，大大提升了效率。

96. 如何实现简洁的日期时间格式化？
日期时间处理经常让人头大，但 Python 有妙招：

```
from datetime import datetime
now = datetime.now()
formatted = now.strftime("%Y-%m-%d %H:%M:%S")
print(formatted)
```

strftime 函数让你轻松定制日期时间的显示格式，是不是很贴心？
97. 如何实现优雅的列表拼接？
别再用+或 extend()了，试试这个集合操作：
```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
combined = [*list1, *list2]
print(combined)
```
星号操作符(*)可以展开列表，直接合并，简洁又高效。
98. 如何实现一键字典排序？
想要按字典的值排序？这招超实用。
```
my_dict = {'apple': 3, 'banana': 1, 'cherry': 2}
sorted_dict = dict(sorted(my_dict.items(), key=lambda x: x[1]))
print(sorted_dict)
```
这里用了 sorted()函数加上一个 lambda 表达式作为排序依据，轻松实现。
99. 如何通过高级迭代方法，同时遍历两个列表？
有时候我们需要对齐两个列表的数据，这样做：
```
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
for a, b in zip(list1, list2):
    print(f"{a} -> {b}")
```
zip 函数像魔术师一样把列表配对，一起遍历，省心又省力。
100. 如何实现简易错误处理？
写代码难免会出错，优雅地捕获异常是关键：
```
try:
    # 尝试执行的代码
    result = 10 / 0
except ZeroDivisionError:
    print("不能除以零哦！")
```
使用 try...except，错误不再令人头疼，而是成为你控制流程的好帮手。
101. 如何通过列表推导式快速生成新列表？
想要快速生成新列表？列表推导式是不二之选：
```
squares = [x**2 for x in range(1, 6)]
print(squares)
```
一行代码，将 1 到 5 的平方数尽收眼底，简洁高效。
102. 如何实现轻松读写 CSV 文件？
处理数据时，CSV 文件很常见，Python 内置模块来帮忙：
```
import csv
# 写入 CSV
with open('data.csv', 'w', newline='') as file:
```

```
    writer = csv.writer(file)
    writer.writerow(['Name', 'Age'])
    writer.writerow(['Alice', 24])
# 读取 CSV
with open('data.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

无需安装额外库，csv 模块轻松读写，数据处理就是这么简单。代码不在于长，而在于精。

103. 简述核心概念 1：变量与数据类型。

变量就像一个个小盒子，用来存放各种数据。给变量起个名字（如 age），再给它赋个值（如 age = 25），就宣告了一个变量的诞生。Python 支持多种数据类型：

整型（int），如 num = 42

浮点型（float），如 pi = 3.14

字符串（str），如 name = "Alice"

布尔型（bool），如 is_student = True

如果需要，还可以通过内置函数进行数据类型转换，如 int("123") 将字符串转为整数。

104. 简述核心概念 2：运算符与表达式。

运算符是数学运算的符号化表示，Python 中常见的有：

算术运算符：+（加）、-（减）、 *（乘）、 /（除）、 %（取模）、 **（幂运算）

比较运算符：==（等于）、!=（不等于）、>（大于）、<（小于）、>=（大于等于）、<=（小于等于）

赋值运算符：除了基本的=，还有+=（加后赋值）、-=（减后赋值）等复合赋值形式。

试试编写一个小程序，让用户输入两个数，然后计算它们的和、差、积、商、余数和幂次方吧。

105. 简述核心概念 3：条件判断语句。

面对复杂情况，我们需要做出决策。Python 提供 if...else 结构来实现条件判断：

```
temperature = 20
if temperature > 30:
    print("It's hot outside!")
elif temperature < 10:
    print("Brrr, it's cold!")
else:
    print("Ah, perfect weather!")
```

这个例子中，根据温度不同输出不同提示。学会用 if...elif...else，你就掌握了天气预报员的技能。

106. 简述核心概念 4：循环结构。

循环让我们能够重复执行某段代码。Python 提供了两种循环机制：

for 循环通常配合 range()函数使用，如：

```
for i in range(5):
    print(i)
```
这会输出从 0 到 4 的整数序列。
while 循环则根据某个条件反复执行，直到条件不再满足：
```
count = 0
while count < 5:
    print(count)
    count += 1
```
这个循环同样输出 0 到 4。别忘了 break 可以提前终止循环，continue 则跳过当前循环进入下一轮。
挑战一下，用循环绘制一个数字金字塔吧。
107. 简述核心概念 5：列表与元组。
列表是 Python 中最常用的序列类型，它允许存储一组有序、可变的数据：
```
grocery_list = ["apple", "banana", "orange"]
print(grocery_list[0])  # 输出 "apple"
grocery_list.append("pear")
grocery_list.sort()
```
列表可以索引访问、增删元素、排序等。而元组类似列表，但一旦创建便不可更改：
```
coordinates = (40.7128, -74.0060)  # 纽约市经纬度
print(coordinates[0])  # 输出 40.7128
```
练习一下，创建一个购物清单管理程序，实现添加商品、删除商品、按字母顺序排列等功能。
108. 简述核心概念 6：字典与集合。
字典是一种键值对（key-value）数据结构，非常适合存储对象的属性：
```
student = {"name": "Tom", "age": 18, "major": "Computer Science"}
print(student["name"])  # 输出 "Tom"
```
集合则用于存放无序且不重复的元素，支持交集、并集、差集等操作：
```
favorite_colors = {"red", "blue", "green"}
friend_colors = {"blue", "yellow", "green"}
shared_colors = favorite_colors.intersection(friend_colors)
```
动手设计一个学生信息数据库，用字典存储每个学生的数据，用集合记录所有学生的专业。
109. 简述核心概念 7：函数定义与调用。
函数是组织代码、实现特定功能的基本单元。定义一个函数如下：
```
def greet(name):
    """Greet a person by name."""
    print(f"Hello, {name}!")
greet("Alice")  # 调用函数，输出 "Hello, Alice!"
```
函数可以接受参数、返回值，还可以使用*args 和**kwargs 处理任意数量的位置参数和关键字参数。试着编写一个计算 BMI 指数的函数吧！

110. 简述核心概念 8：模块与导入。
为了提高代码复用性和可维护性，Python 支持将相关功能封装到模块中。导入模块的方式有：
```
import math
math.sqrt(16)  # 使用 math 模块的 sqrt 函数求平方根
from datetime import datetime
now = datetime.now()  # 直接使用 datetime 模块下的 now 函数
import my_module as mm  # 使用 as 为模块指定别名
mm.my_function()
```
尝试使用内置 math 模块解决一些实际问题，如计算圆面积、三角函数等。
111. 简述核心概念 9：错误与异常处理。
编程难免遇到错误，Python 通过异常机制优雅地处理这些问题。使用 try...except...finally 结构捕获并处理异常：
```
try:
    file = open("nonexistent_file.txt", "r")
except FileNotFoundError:
    print("The file you're looking for doesn't exist.")
finally:
    if 'file' in locals():
        file.close()
```
在上述代码中，当尝试打开不存在的文件时，FileNotFoundError 会被触发。except 子句捕获这个异常并打印友好提示，finally 子句确保无论是否发生异常，最后都会关闭文件。你还可以使用 raise 主动抛出异常，甚至自定义异常类以更精确地描述问题。
实践一下，编写一个文件读写程序，妥善处理可能出现的 IOError 和其他异常。
112. 简述核心概念 10：面向对象编程简介。
面向对象编程（OOP）是 Python 的重要特性之一，它将数据（属性）和操作数据的方法组织成类。创建一个简单的“动物”类体系：
```
class Animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species
    def speak(self):
        pass  # 子类应覆盖此方法
class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"
class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"
```

```
fido = Dog("Fido", "Canine")
felix = Cat("Felix", "Feline")
print(fido.speak())  # 输出 "Fido says Woof!"
print(felix.speak())  # 输出 "Felix says Meow!"
```

在这个例子中，Animal 是基类，Dog 和 Cat 继承自 Animal 并各自实现了 speak 方法。这就是 OOP 中的继承与多态。

113. 简述列表推导式。

列表推导式是一种快速创建新列表的方法，其基本语法是在一对方括号内，通过 for 循环对一个列表中的元素进行处理，并将处理结果加入新的列表中。

例如，我们要从一个包含整数的列表中找出所有的偶数，就可以这样做：

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = [num for num in numbers if num % 2 == 0]
print(even_numbers)  # 输出：[2, 4, 6]
```

这段代码中，[num for num in numbers if num % 2 == 0]就是一个列表推导式，它的工作原理就是遍历 numbers 列表中的每一个元素 num，然后检查 num 是否为偶数，如果是，则将其加入新的列表 even_numbers 中。

114. 简述生成器表达式。

生成器表达式与列表推导式类似，也是用来快速创建新列表的一种方法，但它不是一次性创建出整个列表，而是一个一个地生成结果。

例如，我们要计算从 1 到 10 的所有数字的平方，就可以这样做：

```
squares = (num ** 2 for num in range(1, 11))
print```python
for square in squares:
    print(square)
```

这段代码中，(num ** 2 for num in range(1, 11))就是一个生成器表达式，它的工作原理就是遍历 range(1, 11)中的每一个元素 num，然后计算 num 的平方，并将结果一次又一次地返回给调用者。

115. 简述 lambda 函数。

lambda 函数是一种没有名字的匿名函数，它的定义通常只有一行代码。

例如，我们要定义一个简单的函数，该函数接受两个参数 a 和 b，然后返回它们的和，我们可以这样做：

```
add = lambda a, b: a + b
print(add(1, 2))  # 输出：3
```

这段代码中，lambda a, b: a + b 就是一个 lambda 函数，它的工作原理就是接收 a 和 b 作为输入，然后返回 a+b 的结果。

116. 简述 map 函数。

map 函数是一个内置函数，它可以接受一个函数 f 和一个序列 s，然后对 s 中的每一个元素都执行 f，最后返回一个新的序列。

例如，我们要将一个列表中的所有元素都加上 10，就可以这样做：

```
numbers = [1, 2, 3, 4, 5]
new_numbers = map(lambda x: x + 10, numbers)
print(list(new_numbers))  # 输出：[11, 12, 13, 14, 15]
```
这段代码中，map(lambda x: x + 10, numbers)就是一个 map 函数，它的工作原理就是将 numbers 中的每一个元素 x 都传递给 lambda 函数，并将结果加入到一个新的列表中。

117. 简述 filter 函数。

filter 函数也是一个内置函数，它可以接受一个函数 f 和一个序列 s，然后对 s 中的每一个元素都执行 f，如果 f 返回 True，则将这个元素加入一个新的序列中。

例如，我们要找出一个列表中的所有偶数，就可以这样做：

```
numbers = [1, 2, 3, 4, 5]
even_numbers = filter(lambda x: x % 2 == 0, numbers)
print(list(even_numbers))  # 输出：[2, 4]
```
这段代码中，filter(lambda x: x % 2 == 0, numbers)就是一个 filter 函数，它的工作原理就是对 numbers 中的每一个元素 x 都执行 lambda 函数，如果 lambda 函数返回 True，则将这个元素加入一个新的列表中。

118. 简述 sorted 函数。

sorted 函数是一个内置函数，它可以接受一个可迭代对象，并返回一个新的排序后的列表。

例如，我们要对一个列表进行排序，就可以这样做：

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
sorted_numbers = sorted(numbers)
print(sorted_numbers)  # 输出：[1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```
这段代码中，sorted(numbers)就是一个 sorted 函数，它的工作原理就是接受 numbers 作为输入，然后返回一个排序后的列表。

119. 简述 enumerate 函数。

enumerate 函数是一个内置函数，它可以接受一个可迭代对象，并返回一个新的迭代器，该迭代器可以同时获取元素及其索引。

例如，我们要打印一个列表中每个元素的索引和值，就可以这样做：

```
fruits = ['apple', 'banana', 'cherry']
for index, fruit in enumerate(fruits):
    print(index, fruit)  # 输出：0 apple 1 banana 2 cherry
```
这段代码中，for index, fruit in enumerate(fruits):就是一个 enumerate 函数，它的工作原理就是接受 fruits 作为输入，然后返回一个新的迭代器，该迭代器可以同时获取 fruits 中每个元素的索引和值。

120. 简述 zip 函数。

zip 函数是一个内置函数，它可以接受任意多个可迭代对象，然后返回一个新的迭代器，该迭代器包含了各个输入对象中对应位置的元素。

例如，我们要将两个列表中的元素一一对应起来，就可以这样做：

```
fruits = ['apple', 'banana', 'cherry']
prices = [1, 2, 3]
```

```
zipped = zip(fruits, prices)
print(list(zipped))  # 输出：[(apple, 1), (banana, 2), (cherry, 3)]
```
这段代码中，zip(fruits, prices)就是一个 zip 函数，它的工作原理就是接受 fruits 和 prices 作为输入，然后返回一个新的迭代器，该迭代器包含了 fruits 和 prices 中对应位置的元素。

121. 简述 itertools 模块。

itertools 模块是 Python 的标准库之一，其中包含了许多用于生成迭代器的函数。

例如，我们想要将一个列表中的元素重复多次，就可以使用 itertools 模块的 repeat 函数：

```
import itertools
numbers = [1, 2, 3]
repeated_numbers = itertools.repeat(numbers, 3)
print(list(repeated_numbers))  # 输出：[[1, 2, 3], [1, 2, 3], [1, 2, 3]]
```
这段代码中，itertools.repeat(numbers, 3)就是一个 itertools 模块的函数，它的工作原理就是接收 numbers 作为输入，然后生成一个迭代器，该迭代器可以无限地重复 numbers 中的元素。

122. 简述 functools 模块。

functools 模块也包含了 Python 的标准库之一，其中包含了一些用于编写高阶函数的工具。

例如，我们想要实现一个函数，该函数接受一个函数和一个参数，然后返回一个新的函数，新函数的返回值是原始函数对参数的执行结果。

```
import functools
def double(x):
    return x * 2
tripled = functools.partial(double, 3)
print(tripled())  # 输出：6
```
这段代码中，functools.partial(double, 3)就是一个 functools 模块的函数，它的工作原理就是接收 double 和 3 作为输入，然后返回一个新的函数 tripped，新函数的返回值是原始函数 double 对参数 3 的执行结果。

123. 简述 operator 模块。

operator 模块包含了 Python 的一些标准运算符，如加法、减法、乘法等。

例如，我们想要实现一个函数，该函数接受一个字符串和一个字符，然后返回字符串中该字符出现的次数：

```
import operator
def count_char(s, char):
    return operator.count(s, char)
print(count_char('hello world', 'o'))  # 输出：2
```
这段代码中，operator.count(s, char)就是一个 operator 模块的函数，它的工作原理就是接收 s 和 char 作为输入，然后返回字符串中 char 出现的次数。

124. 简述内置函数 len()。

** len()**：想知道列表、字符串等容器有多长？只需一个 len()，它会告诉你元素个数。在 Python 世界里，内置函数就像一个个小巧玲珑的魔法盒，它们深藏不露，却又蕴含着强大

的能量。掌握并巧妙运用这些内置函数，不仅能简化代码，提升效率，更能展现优雅、地道的 Python 编程风格。

```
my_list = [1, 2, 3, 4, 5]
print(len(my_list))  # 输出：5
```

125. 简述内置函数 type()。

type()：想了解变量是什么类型？type()帮你快速识别。

```
x = "Hello, World!"
print(type(x))  # 输出：<class 'str'>
```

126. 简述内置函数 isinstance()。

isinstance()：判断对象是否属于指定类型（或其子类），确保类型安全。

```
def process_number(num):
    if isinstance(num, (int, float)):
        print(f"Processing number: {num}")
    else:
        print("Invalid input!")
process_number(42)  # 输出：Processing number: 42
process_number("42")  # 输出：Invalid input!
```

127. 简述内置函数 dir()。

dir()：想知道一个对象有哪些属性和方法？用 dir()列出所有成员。

```
import math
print(dir(math))  # 输出：['acos', 'acosh', 'asin', 'asinh', ...]
```

128. 简述内置函数 id()。

id()：获取对象独一无二的身份标识，理解 Python 中的“万物皆对象”。

```
a = [1, 2, 3]
b = a
print(id(a), id(b))  # 输出：两个相同的整数，表示 a 和 b 指向同一内存地址
a.append(4)
print(a, b)  # 输出：[1, 2, 3, 4], [1, 2, 3, 4]
c = [1, 2, 3]
print(id(c))  # 输出：不同于 a 和 b 的整数，c 是新的列表对象
```

hash()：计算对象的哈希值，用于字典、集合等数据结构的高效查找。

```
word = "python"
print(hash(word))  # 输出：-986773616
```

129. 简述内置函数 del。

del：删除对象引用，释放内存资源，或删除变量、列表元素等。

```
del my_list[0]  # 删除列表第一个元素
del my_variable  # 删除变量，使其不再存在于当前作用域
```

130. 简述内置函数 globals()与 locals()。

globals()与 locals()：查看全局/局部作用域内的变量名及其值。

```
x = "global"
def func():
    y = "local"
    print(globals())  # 输出：包含全局变量 x 的字典
    print(locals())  # 输出：包含局部变量 y 的字典
func()
```

131. 简述内置函数 all()与 any()。

all()与 any()：判断容器内所有/任意元素是否满足条件。

```
numbers = [1, 2, 0, 4]
print(all(number > 0 for number in numbers))  # 输出：False（存在非正数）
print(any(number > 0 for number in numbers))  # 输出：True（存在正数）
```

132. 简述内置函数 enumerate()。

enumerate()：同时获取容器内元素及其索引，便于循环处理。

```
fruits = ["apple", "banana", "cherry"]
for i, fruit in enumerate(fruits):
    print(f"Index {i}: {fruit}")
# 输出：
# Index 0: apple
# Index 1: banana
# Index 2: cherry
```

133. 简述内置函数 zip()。

zip()：将多个可迭代对象按元素打包成一个个元组，实现多数据源同步遍历。

```
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]
for name, age in zip(names, ages):
    print(f"{name} is {age} years old.")
# 输出：
# Alice is 25 years old.
# Bob is 30 years old.
# Charlie is 35 years old.
```

134. 简述内置函数 format()。

format()：灵活格式化字符串，插入变量、控制对齐、指定精度等。

```
name = "Alice"
age = 25
print("My name is {} and I am {} years old.".format(name, age))
# 输出：
# My name is Alice and I am 25 years old.
```

135. 简述内置函数 join()。

join()：将列表（或其他可迭代对象）中元素以指定字符连接成字符串。

```
words = ["Python", "is", "fun"]
sentence = " ".join(words)
print(sentence)
# 输出：
# Python is fun
```

136. 简述内置函数 split()。

split()：根据分隔符将字符串拆分为列表，常用于处理文本数据。

```
text = "A quick brown fox jumps over the lazy dog."
words = text.split(" ")
print(words)
# 输出：
# ['A', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog.']
```

137. 简述内置函数 strip()。

strip()：去除字符串两侧指定字符（默认空格），清理文本数据。

```
s = "      Hello, World!      "
clean_s = s.strip()
print(clean_s)
# 输出：
# Hello, World!
```

138. 简述内置函数 sorted()。

sorted()：对可迭代对象进行排序，返回一个新的排序后列表。

```
unsorted_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
sorted_list = sorted(unsorted_list)
print(sorted_list)
# 输出：
# [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```

139. 简述内置函数 reversed()。

reversed()：反转序列（如列表、元组、字符串）元素顺序。

```
numbers = [1, 2, 3, 4, 5]
reversed_numbers = list(reversed(numbers))
print(reversed_numbers)
# 输出：
# [5, 4, 3, 2, 1]
```

140. 简述内置函数 set()与 frozenset()。

set()与 frozenset()：创建无序、唯一元素集，后者不可变。

```
unique_elements = set([1, 2, 2, 3, 4, 4, 5])
print(unique_elements)  # 输出：{1, 2, 3, 4, 5}
immutable_set = frozenset(unique_elements)
```

141. 简述内置函数 assert。

assert：断言某个条件为真，否则触发 AssertionError，用于检查程序逻辑。

```
def divide(a, b):
    assert b != 0, "Cannot divide by zero!"
    return a / b
result = divide(10, 2)  #正常运行，结果为 5.0
result = divide(10, 0)  # 触发 AssertionError: Cannot divide by zero!
```

142. 简述内置函数 traceback。

traceback：捕获、打印及分析异常堆栈信息，辅助定位问题。

```
try:
    raise ValueError("This is an intentional error.")
except ValueError as e:
    import traceback
    traceback.print_exc()
# 输出类似如下：
# Traceback (most recent call last):
#   File "<stdin>", line 2, in <module>
# ValueError: This is an intentional error.
```

143. 简述内置函数 sys.exc_info()。

sys.exc_info()：获取当前正在处理的异常的详细信息（类型、值、堆栈跟踪）。

```
import sys
try:
    raise IndexError("Index out of range!")
except IndexError as e:
    exc_type, exc_value, exc_traceback = sys.exc_info()
    print(exc_type)  # 输出：<class 'IndexError'>
    print(exc_value)  # 输出：Index out of range!
    print(exc_traceback)  # 输出：详细的异常堆栈跟踪信息
```

144. 简述内置函数 map()。

map()：将函数应用到可迭代对象每个元素上，返回结果组成的迭代器。

```
numbers = [1, 2, 3, 4, 5]
squared = map(lambda x: x ** 2, numbers)
print(list(squared))  # 输出：[1, 4, 9, 16, 25]
```

145. 简述内置函数 filter()。

filter()：筛选出可迭代对象中满足条件的元素，返回过滤后的迭代器。

```
even_numbers = [1, 2, 3, 4, 5, 6]
filtered = filter(lambda x: x % 2 == 0, even_numbers)
print(list(filtered))  # 输出：[2, 4, 6]
```

146. 简述内置函数 reduce()。

reduce()（在 functools 模块中）：对可迭代对象元素应用二元函数累积结果。

```
from functools import reduce
product = reduce(lambda x, y: x * y, [1, 2, 3, 4, 5])
print(product)  # 输出：120
```

147. 简述内置函数 lambda。

lambda：定义小型匿名函数，简洁表达临时计算逻辑。

```
add_one = lambda x: x + 1
print(add_one(41))  # 输出：42
```

148. 简述内置函数__str__与__repr__。

__str__与__repr__：自定义对象的字符串表示形式，分别用于用户友好输出和调试。

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def __str__(self):
        return f"{self.name}, {self.age} years old"
    def __repr__(self):
        return f"Person(name={self.name!r}, age={self.age})"
p = Person("Alice", 25)
print(p)  # 输出：Alice, 25 years old
print(repr(p))  # 输出：Person(name='Alice', age=25)
```

149. 简述内置函数__getattr__。

__getattr__：当尝试访问不存在的属性时调用，提供自定义行为。

```
class MagicBox:
    def __getattr__(self, item):
        return f"Sorry, no such attribute '{item}'!"
box = MagicBox()
print(box.secret_key)  # 输出：Sorry, no such attribute 'secret_key'!
```

150. 简述内置函数@property。

@property：将方法包装成只读属性，实现属性访问控制与验证。

```
class Circle:
    def __init__(self, radius):
        self._radius = radius
    @property
    def radius(self):
        return self._radius
    @radius.setter
    def radius(self, value):
        if value < 0:
            raise ValueError("Radius must be non-negative.")
```

```
            self._radius = value
circle = Circle(5)
print(circle.radius)  # 输出：5
circle.radius = -1  # 会触发 ValueError
```

151. 简述内置函数 importlib。

importlib：动态导入、重载、查询模块信息，实现高级模块管理。

```
import importlib
module_name = "math"
module = importlib.import_module(module_name)
print(module.sqrt(16))  # 输出：4.0
```

152. 简述内置函数 pkgutil。

pkgutil：递归遍历包及其子包，查找模块、执行包级初始化等。

```
import pkgutil
package_name = "numpy"
package = pkgutil.get_loader(package_name)
print(package)  # 输出：numpy.__loader__
```

153. 简述内置函数 sys.path。

sys.path：查看 Python 解释器搜索模块的路径列表，调整路径以引入自定义模块。

```
import sys
print(sys.path)  # 输出：当前 Python 环境搜索模块的路径列表
sys.path.append("/path/to/custom/module")
```

挖掘 Python 内置函数，解锁编程新境界。Python 内置函数犹如一座宝藏库，等待你去发掘、利用。无论你是初学者还是资深开发者，熟练掌握并适时运用这些鲜为人知的内置函数，都能显著提升代码质量、开发效率，乃至编程思维。愿你在 Python 的世界里游刃有余，享受编程的乐趣与成就感。

4 判断题

4.1 习题

1. 在 Python 中，“//”运算符用于指数运算（　　）。
2. 列表和元组都是可变的数据类型（　　）。
3. 在 Python 中，None 表示布尔值（　　）。
4. 可以使用单引号和双引号来定义字符串，它们是等效的（　　）。
5. range(5)用于生成一个包含从 0 到 4 的整数的序列（　　）。
6. “//”是整数除法运算符，而不是指数运算。指数运算使用“**”（　　）。
7. 列表是可变的，但元组是不可变的（　　）。
8. 在 Python 中，None 表示空或缺失值，而不是布尔值 False（　　）。
9. 在 Python 中，可以使用单引号或双引号来定义字符串，它们是等效的（　　）。
10. range(5)生成一个包含 0 到 4 的整数序列，不包括 5（　　）。
11. 在 Python 中，+=运算符，可以用于字符串拼接（　　）。
12. 字典可以作为另一个字典的键（　　）。
13. break 语句用于结束整个程序的执行（　　）。
14. pop()方法可以用于从列表中删除指定位置的元素。
15. not in 运算符用于检查一个元素是否不在给定的序列中（　　）。
16. “+=”运算符不仅可以用于数字相加，还可以用于字符串拼接（　　）。
17. 字典的键必须是不可变的数据类型，而字典本身是可变的，因此不能作为另一个字典的键（　　）。
18. break 语句用于退出当前循环，而不是结束整个程序（　　）。
19. pop()方法用于从列表中删除指定位置的元素，如果不指定位置，默认删除最后一个元素（　　）。
20. Python 是一种跨平台、开源、免费的高级动态编程语言（　　）。
21. 在 Python 中，可以使用 if 作为变量名（　　）。
22. Python 使用缩进来体现代码之间的逻辑关系（　　）。
23. Python 代码的注释只有一种方式，那就是使用#符号（　　）。
24. 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（　　）。
25. 函数中的 return 语句一定能够得到执行（　　）。
26. 在函数内部没有办法定义全局变量（　　）。
27. 可以使用 insert()添加一个新的元素到列表的尾部（　　）。
28. 转义字符'\n'的含义是回车换行（　　）。
29. 顺序结构每条语句可以执行多次（　　）。
30. Python 中的类是一种封装了数据和方法的数据结构（　　）。

31. 在 Python 中，类可以继承其他类的属性和方法（　　）。
32. 在 Python 中，所有的东西都是对象（　　）。
33. 在 Python 中，类和对象是完全不同的概念（　　）。
34. 在 Python 中，可以使用 super()方法来调用父类的方法（　　）。
35. 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（　　）。
36. 形参可以被看作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（ ）。
37. 一个函数如果带有默认值参数，那么所有参数都必须设置默认值（　　）。
38. 在 Python 3.x 中，可以使用中文作为变量名（　　）。
39. Python 集合中的元素可以是元组（　　）。
40. 在函数内部直接修改形参的值并不影响外部实参的值（　　）。
41. 在函数内部直接修改形参的值并不影响外部实参的值（　　）。
42. 已知 x 为非空列表，执行语句 x[0] = 3 之后，列表对象 x 的内存地址不变（　　）。
43. 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（　　）。
44. 表达式 'a'+1 的值为'b'（　　）。

4. 2 答案

1. False；	2. False；	3. False；	4. True；	5. True；
6. False；	7. False；	8. True；	9. True；	10. True；
11. True；	12. False；	13. False；	14. False；	15. True；
16. True；	17. True；	18. True；	19. True；	20. True；
21. False；	22. True；	23. False；	24. True；	25. False；
26. False；	27. False；	28. True；	29. False；	30. True；
31. True；	32. True；	33. False；	34. True；	35. False；
36. True；	37. False；	38. True；	39. True；	40. True；
41. True；	42. True；	43. True；	44. False。	

5 应用题

5.1 习题

1.小蓝正在学习一门神奇的语言，这门语言中的单词都是由小写英文字母组成，有些单词很长，远远超过正常英文单词的长度。小蓝学了很长时间也记不住一些单词，他准备不再完全记忆这些单词，而是根据单词中哪个字母出现得最多来分辨单词。

现在，请你帮助小蓝，给了一个单词后，帮助他找到出现最多的字母和这个字母出现的次数。其实就是让你输入一段字符串后，得到当前字符串出现最多的字母和它的次数。

2.给定一个只包括(、)｛、｝[、]的字符串，判断字符串是否有效。

有效字符串需满足：

（1）左括号必须用相同类型的右括号闭合。

（2）左括号必须以正确的顺序闭合。

（3）空字符串可被认为是有效字符串。

示例 1：

输入：“()”

输出：True

示例 2：

输入：“()[]{}”

输出：True

示例 3：

输入：“(]”

输出：False

示例 4：

输入：“([)]”

输出：False

3.回文数是指正序（从左向右）和倒序（从右向左）都是一样的整数。例如，1221是回文数，而1222不是回文数。设计程序，判断输入的数是否为回文数。

4.输入边长n，打印对应边长的菱形。

5.split是Python字符串内置的一个非常有用的方法，它可以将一个字符串通过分隔符，切成我们想要的列表。比如现在我们有个字符串life-is-short-you-need-python，每一个单词之间，使用分隔符“-”进行分割。

我们去调用字符串split的方法之后，传入我们的分隔符“-”，那我们就会得到一个列表，里面每个元素都是通过分隔符切出来的子字符串。

6.美国数学家维纳（N.Wiener）11岁就上了大学。他曾在1935—1936年应邀来中国清华大学讲学。一次，他参加某个重要会议，年轻的脸孔引人注目。

有人询问他的年龄，他回答说：“我年龄的立方是个4位数。我年龄的4次方是个6位数。

这10个数字正好包含了从0到9这10个数字，每个都恰好出现1次。”

请你推算一下，他当时到底有多年轻？

7.给定一个正整数列表L，输出L内所有数字的乘积末尾0的个数（提示：不要直接相乘，数字很多，相乘得到的结果可能会很大）。

输入：L=[2,8,3,50]

输出：2

8.结尾非零数的奇偶性。给定一个正整数列表L，判断列表内所有数字乘积的最后一个非零数字的奇偶性。如果为奇数则输出1，如果为偶数则输出0。

输入：L=[2,8,3,50]

输出：0

9. 给定一个整数a，计算出a在二进制表示下1的个数，并将其输出。

输入：a=7

输出：3

10.大小写转换。给定一个字符串a，将a中的大写字母转换成小写字母，其他字符不变，并输出。

输入：a = ″KDJIskos234k,.;djfeiJ″

输出：kdjiskos234k,.;djfeij

11.判断三角形。给三个整数a、b、c，判断能否以它们为三个边长构成三角形。若能，则输出YES，否则输出NO。

输入：a=5，b=5，c=5

输出：YES

12.公约数的个数。给两个正整数a、b，输出它们公约数的个数。

输入：a=24，b=36

输出：6

13.回文子串。给一个字符串a和一个正整数n，判断a中是否存在长度为n的回文子串。如果存在，则输出YES，否则输出NO。 回文串的定义：记串str逆序之后的字符串是str1，若str=str1,则称str是回文串，如″abcba″.

输入：a = ″abcba″ n = 5

输出：YES

14.单身情歌。“抓不住爱情的我 总是眼睁睁看它溜走……”，现在来练习一下发现爱的能力，给你一个字符串a，如果其中包含“LOVE”（love不区分大小写），则输出“LOVE”，否则输出“SINGLE”。

输入：a=″OurWorldIsFullOfLOVE″

输出：LOVE

15.时间就是金钱。给你两个时间st和et（00:00:00<=st <= et<=23:59:59），请你给出这两个时间间隔的秒数。如：st=″00:00:00″,et=″00:00:10″,则输出10。

输入：st=″00:00:00″ et= ″00:00:52″（et前面没有标点符号？）

输出：52

16.365还是366？一年有多少天？现在给你一个年份year（year为四位数字的字符串，

如"2008","0012"），请输出这一年的天数。如year="2013",则输出365。

输入：year="2008"

输出：366

17.格式化时间。给你一个时间t。t是一个字典，共有六个字key(year,month,day,hour,minute,second)，其中每个值都是数字组成的字符串，请将其按照以下格式输出，格式:XXXX-XX-XX XX:XX:XX。

输入：t={"year": "2013", "month": "9", "day": "30", "hour": "16", "minute": "45", "second": "2"}

输出：2013-09-30 16:45:02

18.序列判断。给你一个整数组成的列表L，按照下列条件输出：若L是升序排列的，则输出"UP"；若L是降序排列的，则输出"DOWN"；若L无序，则输出"WRONG"。

输入：L=[1, 1, 3, 3, 4]

输出：UP

19.山峰的个数。十一假期，小P出去爬山,在爬山过程中，每隔10米，他都会记录当前点的海拔高度（以一个浮点数表示），这些值序列保存在一个由浮点数组成的列表h中。回到家中，小P想研究一下自己经过了几个山峰，请你帮他计算一下，输出结果。

例如：h=[0.9,1.2,1.22,1.1,1.6,0.99]，将这些高度顺序连线，会发现有两个山峰，故输出一个2（序列两端不算山峰）。

输入：h=[0.9,1.2,1.22,1.1,1.6,0.99]

输出：2

20.三角形形状。

给出一个三角形的三边长a、b和c（边长是浮点数），请你判断三角形的形状。若是锐角三角形，则输出R；若是直角三角形，则输出Z，；若是钝角三角形，则输出D，若三边长不能构成三角形，则输出W。

输入：a=3.0，b=5.0，c=4.0

输出：Z

21.简单题之勾股定理。

给定直角三角形两个直角边的边长a、b，请求出其斜边边长，结果保留小数点后三位小数。

如a=3，b=4，则输出5.000。

输入：a=3.0 b=4.0

输出：5.000

22.编程实现进制之间的转换。

23.定义函数求阶乘。要求对参数进行判断，对合法的参数求阶乘，参数零的阶乘返回1,并在参数为负数时输出-1。

24.定义一个函数，实现输入一句话，单词之间使用空格隔开，统计出其中各单词的词频数，并以keyword:count的格式存在一个dict中，返回这个dict。

注意 jkl; 不是一个单词，jkl 是一个单词，要把“; + ”等非字符符号去掉，即输入：“abc fjf jkl+ abc abc jkl;”结果是{"abc":3,"fjf":1,"jkl":2}

25.计算并返回x的平方根整数部分，其中x是非负整数。由于返回类型是整数，结果只保留整数的部分，小数部分将被舍去。

26.将整数列表中所有的偶数摘选出来并输出一个新的列表。

例如：nums = [1,2,3,4,5,6,7,8,9,10]，输出结果为[2,4,6,8,10]。请将以下列表中的偶数筛选出来：

num1=[11,16,26,7,31]；num2=[13,18,21,37,42]；num3=[31,17,46,12,77]。

27.给定一个整数n，返回n!结果尾数中零的数量。

示例1：输入：3，输出：0，解释：3!= 6,尾数中没有零。

示例2：输入：5，输出：1，解释：5!=120,尾数中有1个零。

28.编写一个Python程序，接受两个正整数作为输入，并计算它们的最大公约数(GCD)。最大公约数是两个或多个整数的共有因子中的最大值。例如，18和24的最大公约数是6。编写一个函数TEST_DO_NOT_CHANGE(a,b)，它接受两个正整数a和b作为参数，返回它们的最大公约数。

29.编写一个Python程序，计算并输出给定底数x和指数n的幂次方结果。

幂次方的定义如下：x的0次幂是1；x的正整数n次幂是x乘以自己n次，即x^n=x*x* ...*x(n次)。

例如，2的3次幂是2^3 =2*2*2=8。

请编写一个函数TEST_DO_NOT_CHANGE(x,n)，其中x为底数，n为指数，返回x^n的结果。

30.输入一行字符input_str，分别统计出其中英文字母、空格、数字和其他字符的个数，并将结果依次保存到列表lst_rlt。

输出示例：[中英文字母个数，空格个数，数字个数，其他字符个数]。

程序分析：利用 while 或 for 语句, 只要输入的字符不为'\n'，就不退出程序。

31.给定一个字符串s，由若干单词组成，单词之间用空格隔开。返回字符串中最后一个单词的长度。如果不存在最后一个单词，请返回 0。

单词，是指仅由字母组成，不包含任何空格字符的最大子字符串。

示例 1：输入：s = "Hello World"；输出：5。（前双引号格式对吗？）

示例 2：输入：s = " "；输出：0。

32.将两个升序链表，合并为一个新的升序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例：

（1）输入：l1 = [1,2,4]， l2 = [1,3,4]，输出：[1,1,2,3,4,4]。

（2）输入：l1 = []，l2 = [0]，输出：[0]。

33.使用26个字母和10个数字，产生一个随机组合的6位字符串。

提示：

（1）random.choice('tomorrow')；

（2）random.randint(1,10)；

（3）random.shuffle([1,3,5,6,7])。

34.已知一个3*3矩阵A，A的元素依次为1—9的平方，求该矩阵主对角线元素之和。

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。

35.给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，那么目标值通过排序插入数组中，并返回目标值插入数组中的位置。可以假设数组中无重复元素。

36.取一个整数a从右端开始的4～7位。

37.打印出杨辉三角形前十行。

38.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），凡报出3的人退出圈子，问最后留下的是原来第几号的那位。

39.求0—7所能组成的奇数的个数。

程序分析：

（1）组成1位数是4个。1、3、5、7结尾。

（2）组成2位数是7*4个。第1位不能为0。

（3）组成3位数是784个。中间随意。

（4）组成4位数是788*4个。

40.写一个函数，求出所有的水仙花数。调用这个函数，打印出来所有水仙花数。

水仙花数（Narcissistic number）也被称为超完全数字不变数（pluperfect digital invariant, PPDI）、自恋数、自幂数、阿姆斯壮数或阿姆斯特朗数（Armstrong number），水仙花数是指一个3位数，它的每个位上的数字的3次幂之和等于它本身。例如：1^3 + 5^3+ 3^3 =153。

41.写一个函数，求出所有的四叶玫瑰数。调用这个函数，打印出来所有四叶玫瑰数。四叶玫瑰数是4位数的自幂数。自幂数是指一个n位数，它的每个位上的数字的n次幂之和等于它本身。当n为3时，有1^3+5^3+3^3=153，153即是n为3时的一个自幂数，3位数的自幂数被称为水仙花数。

42.随机生成一个100以内的整数，共有10次机会开始游戏。输入猜测的数字，如果猜小了，则提示“猜小了”；如果猜大了，则提示“猜大了”；如果猜对了，则提示“猜对了”；结束游戏10次机会用完还没猜对，提示“游戏结束，没有猜到”。

43.公鸡每只5元，母鸡每只3元，小鸡3只1元，现要求用100元钱买100只鸡(三种类型的鸡都要买)，问公鸡、母鸡、小鸡各买几只?

44.输入年月日，输出该日期是否为闰年，并且输出该日期是此年份的第几天。

闰年判断条件：能被4整除，并且不能被100整除；能被400整除。两个条件满足任意一个就为闰年。

45.打印九九乘法表，在控制台输出。

46.游戏开始，初始状态下用户和电脑都有100分，赢一局+10分，输一局-10分。

当用户为0分时，游戏结束，提示游戏结束，比赛输了。当用户为200分时，游戏结束，提示游戏结束，比赛赢了。每轮比赛都输出当前的分数。

5.2 答案

1.小蓝正在学习一门神奇的语言，这门语言中的单词都是由小写英文字母组成，有些单词很长，远远超过正常英文单词的长度。小蓝学了很长时间也记不住一些单词，他准备不再完全记忆这些单词，而是根据单词中哪个字母出现得最多来分辨单词。

现在，请你帮助小蓝，给了一个单词后，帮助他找到出现最多的字母和这个字母出现的次数。其实就是让你输入一段字符串后，得到当前字符串出现最多的字母和它的次数。

1.程序设计如下。

```
def analyse_words(words):
    word_dict = {}
    for i in words:
        if i in word_dict:
            word_dict[i] += 1
        else:
            word_dict[i] = 1
    max_key = max(word_dict,key=word_dict.get)
    print(max_key)
    print(word_dict[max_key])
analyse_words('helloworld')
```

2.给定一个只包括：(、)、{、}、[、]的字符串，判断字符串是否有效。

有效字符串需满足：

（1）左括号必须用相同类型的右括号闭合。

（2）左括号必须以正确的顺序闭合。

（3）空字符串可被认为是有效字符串。

示例 1：

输入：“()”

输出：True

示例 2：

输入：“()[]{}”

输出：True

示例 3：

输入：“(]”

输出：False

示例 4：

输入：“([)]”

输出：False

2.程序设计如下。

```
def valid_str(string):
```

```
    if len(string) % 2 == 1:
        return False
    while '()' in string or '[]' in string or '{}' in string:
        string = string.replace('()', '')
        string = string.replace('[]', '')
        string = string.replace('{}', '')
    return string == ''
print(valid_str('()'))  # True
print(valid_str('()[]{}'))  # True
print(valid_str('()[]{[()]}'))  # True
print(valid_str('()[]{[(}]}'))  # False
```

3.回文数是指正序（从左向右）和倒序（从右向左）都是一样的整数。例如，1221 是回文数，而 1222 不是回文数。设计程序，判断输入的数是否为回文数。

3.程序设计如下。

```
def is_palindrome(x):
    if x < 0 or x > 0 and x % 10 == 0:
        return False
    str_x = str(x)
    return str_x == str_x[::-1]
print(is_palindrome(121))  # True
print(is_palindrome(120))  # False
```

4.输入边长 n，打印对应边长的菱形。

4.程序设计如下。

```
def diamond(n):
    stack = []
    for i in range(1, 2 * n):
        if i <= n:
            p_str = ' ' * (n - i) + '*' * (2 * i - 1)
            if i != n:
                stack.append(p_str)
            print(p_str)
        else:
            print(stack.pop())
diamond(5)
```

5.split 是 python 字符串内置的一个非常有用的方法，它可以将一个字符串通过分隔符，切成我们想要的列表。比如现在我们有个字符串 life-is-short-you-need-python，每一个单词之间，使用分隔符“-”进行分割。

我们去调用字符串 split 的方法之后，传入我们的分隔符“-”，那我们就会得到一个列表，里面每个元素都是通过分隔符切出来的子字符串。

5.程序设计如下。

```
def split_s(string, sep="", num=0):
    split_words = []
    last_index = 0
    count = 0
    for index, char in enumerate(string):
        if count == num and num > 0:
            split_words.append(string[last_index:len(string)])
            break
        if char == sep:
            split_words.append(string[last_index:index])
            last_index = index + 1
            count += 1
        elif index + 1 == len(string):
            split_words.append(string[last_index:index + 1])
    return split_words
print(split_s("life-is-short-you-need-python",'-'))
# ['life', 'is', 'short', 'you', 'need', 'python']
print(split_s("life-is-short-you-need-python",'-',2))
# ['life', 'is', 'short-you-need-python']
```

6.美国数学家维纳（N.Wiener）11岁就上了大学。他曾在1935—1936年应邀来中国清华大学讲学。一次，他参加某个重要会议，年轻的脸孔引人注目。

有人询问他的年龄，他回答说：“我年龄的立方是个4位数。我年龄的4次方是个6位数。这10个数字正好包含了从0到9这10个数字，每个都恰好出现1次。”

请你推算一下，他当时到底有多年轻？

6.程序设计如下。

```
for i in range(10, 30):
    i3 = str(i ** 3)
    i4 = str(i ** 4)
    if len(i3) == 4 and len(i4) == 6:
        if len(set(i3 + i4)) == 10:
            print(i)
            print(i3 + i4)
        # 18
# 5832104976 舍去
```

7.给定一个正整数列表L，输出L内所有数字的乘积末尾0的个数（提示：不要直接相乘，数字很多，相乘得到的结果可能会很大）。

输入：L=[2,8,3,50]

输出：2

7.程序设计如下。

```
def find_2(x):
    tmp=x
    f2=0 #记录因子2的个数
    f5=0 #记录因子5的个数
    while x%2==0:
        f2+=1
        x=x/2
    while tmp%5==0:
        f5+=1
        tmp/=5
    return f2,f5
#进行计算
L=[2,8,3,50]
a2=0 #记录列表中因子2的个数
a5=0 #记录列表中因子5的个数
for i in L:
    t2,t5=find_2(i)
    a2+=t2
    a5+=t5
print(min(a2,a5)) #2
```

8.结尾非零数的奇偶性。给定一个正整数列表L，判断列表内所有数字乘积的最后一个非零数字的奇偶性。如果为奇数则输出1，如果为偶数则输出0。

输入：L=[2,8,3,50]

输出：0

8.程序设计如下。

```
L=[2,8,3,50]
def find_last_isodd(x): #求解末尾第一个非0的数字
    if x%10!=0:
        return x%10
    else:
        while x%10==0:
            x//=10
            if x%10!=0:
                return x%10
                break
ji=1
for i in L:
    ji*=i
```

```
if find_last_isodd(ji)%2==0:
    print(0)
else:
    print(1)
```

9. 给定一个整数a，计算出a在二进制表示下1的个数，并将其输出。

输入：a = 7

输出：3

9.程序设计如下。

```
a=7
def erjinzhi(x):
    he=0
    while x>0:
        if x%2==1:
            he+=1
        x//=2
    return he
print(erjinzhi(a))
```

10.大小写转换。给定一个字符串a，将a中的大写字母转换成小写字母，其他字符不变，并输出。

输入：a = "KDJIskos234k,.;djfeiJ"

输出：kdjiskos234k,.;djfeij

```
a = "KDJIskos234k,.;djfeiJ"
print(a.lower()) #输出kdjiskos234k,.;djfeij
```

11.判断三角形。给三个整数a、b、c，判断能否以它们为三个边长构成三角形。若能，则输出YES，否则输出NO。

输入：a=5，b=5，c=5

输出：YES

11.程序设计如下。

```
a = 5
b = 5
c = 3
L=list()
L.append(a)
L.append(b)
L.append(c)
L.sort() #默认从小到大排序
if L[0]+L[1]>L[2]:
    print('YES')
else:
```

```
    print('NO')
#YES
```

12.公约数的个数。给两个正整数a、b，输出它们公约数的个数。

输入：a = 24 ，b = 36

输出：6

```
def find_yueshu(x,y):
    he=0
    for i in range(1,min(x,y)+1):
        if x%i==0 and y%i==0:
            he+=1
    return he
print(find_yueshu(24,36)) #6
```

13.回文子串。给一个字符串a和一个正整数n，判断a中是否存在长度为n的回文子串。如果存在，则输出YES，否则输出NO。 回文串的定义：记串str逆序之后的字符串是str1，若str=str1,则称str是回文串，如"abcba".

输入：a = "abcba" n = 5

输出：YES

```
a="abcbe"
n=5
flag=0
for i in range(len(a)):
    if i+n>len(a): #索引超过最大值，直接提前退出
        break
    str_tmp=a[i:i+n]
    str_tmp_reverse=str_tmp[::-1] #字符串翻转
    if str_tmp==str_tmp_reverse:
        flag=1
        break
if flag==1:
    print('YES')
else:
    print('NO')
#输出YES
```

14.单身情歌。“抓不住爱情的我 总是眼睁睁看它溜走……”，现在来练习一下发现爱的能力，给你一个字符串a，如果其中包含“LOVE”（love不区分大小写），则输出“LOVE”，否则输出“SINGLE”。

输入：a="OurWorldIsFullOfLOVE"

输出：LOVE

```
a = "OurWorldIsFullOfLOVE"
```

```
b=a.lower()
flag=0 #判断开关
for i in range(len(a)):
    if i+4>len(a): #防止超出范围
        break
    tmp=b[i:i+4]
    if tmp==’love’:
        flag=1
if flag:
    print(’LOVE’)
else:
    print(’SINGLE’)
```

15.时间就是金钱。给你两个时间st和et（00:00:00<=st <= et<=23:59:59），请你给出这两个时间间隔的秒数。 如：st="00:00:00", et="00:00:10", 则输出10。

输入：st = "00:00:00" et = "00:00:52"

输出：52

```
st = "00:00:00"
et = "00:00:52"
st_time=int(st[0:2])*60*60+int(st[3:5])*60+int(st[6:8]) #强制类型转换
et_time=int(et[0:2])*60*60+int(et[3:5])*60+int(et[6:8])
print(et_time-st_time) #进行做差
#输出52
```

16.365还是366？一年有多少天？现在给你一个年份year（year为四位数字的字符串，如"2008","0012"），请输出这一年的天数。如year="2013",则输出365。

输入：year = "2008"

输出：366

```
def Year_day(x):
    if (x%4==0 and x%100!=0) or x%400==0:
        return 366
    else:
        return 365
year=’2008’
year_int=int(year)
print(Year_day(year_int))
```

17.格式化时间。给你一个时间t。t是一个字典，共有六个字key(year,month,day,hour,minute,second)，其中每个值都是数字组成的字符串，请将其按照以下格式输出，格式:XXXX-XX-XX XX:XX:XX。

输入：t = {"year": "2013", "month": "9", "day": "30", "hour": "16", "minute": "45", "second": "2"}

输出：2013-09-30 16:45:02

```
t = {"year": "2013", "month": "9", "day": "30", "hour": "16","minute": "45", "second": "2"}
for key in t:
    if key!='year':
        if int(t[key])<10:
            t[key]='0'+t[key]
    if key=='year':
        if int(t[key])<1000 and int(t[key])>=100:
            t[key]='0'+t[key]
        elif int(t[key])<100 and int(t[key])>=10:
            t[key]='00'+t[key]
        elif int(t[key])<10 and int(t[key])>=0:
            t[key]='000'+t[key]
print(t['year']+'-'+t['month']+'-'+t['day']+'
'+t['hour']+':'+t['minute']+':'+t['second'])
#输出2013-09-30 16:45:02
```

18.序列判断。给你一个整数组成的列表L，按照下列条件输出：若L是升序排列的，则输出"UP"；若L是降序排列的，则输出"DOWN"；若L无序，则输出"WRONG"。

输入：L = [1, 1, 3, 3, 4]

输出：UP

```
L = [1, 1, 3, 3, 4]
L1=sorted(L,reverse=False) #升序
L2=sorted(L,reverse=True)  #降序
if L==L1:
    print('UP')
elif L==L2:
    print('DOWN')
else:
    print('WRONG')
```

19.山峰的个数。十一假期，小P出去爬山,在爬山的过程中，每隔10米，他都会记录当前点的海拔高度（以一个浮点数表示），这些值序列保存在一个由浮点数组成的列表h中。回到家中，小P想研究一下自己经过了几个山峰，请你帮他计算一下，输出结果。

例如：h=[0.9,1.2,1.22,1.1,1.6,0.99]，将这些高度顺序连线，会发现有两个山峰，故输出一个2（序列两端不算山峰）。

输入：h=[0.9,1.2,1.22,1.1,1.6,0.99]

输出：2

```
h = [0.9, 1.2, 1.22, 1.1, 1.6, 0.99]
he=0
```

```
for i in range(1,len(h)-1):
    if h[i]>h[i-1] and h[i]>h[i+1]:
        he+=1
print(he) #2
```

20.三角形形状。

给出一个三角形的三边长a、b和c（边长是浮点数），请你判断三角形的形状。若是锐角三角形，则输出R；若是直角三角形，则输出Z，；若是钝角三角形，则输出D，若三边长不能构成三角形，则输出W。

输入：a=3.0，b=5.0，c=4.0

输出：Z

```
a = 3.0
b = 5.0
c = 4.0
min_v=min(a,b,c)
max_v=max(a,b,c)
mid_v=a+b+c-min_v-max_v
if min_v+mid_v<=max_v:
    print('W')
else:
    if pow(min_v,2)+pow(mid_v,2)>pow(max_v,2): #锐角
        print('R')
    elif pow(min_v,2)+pow(mid_v,2)==pow(max_v,2): #直角
        print('Z')
    else: #此外为钝角
        print('D')
```

21.简单题之勾股定理。

给定直角三角形两个直角边的边长a、b，请求出其斜边边长，结果保留小数点后三位小数。

如a=3，b=4，则输出5.000。

输入：a=3.0 b=4.0（b之前不加标点符号吗？）

输出：5.000

```
a=3.0
b=5.0
c=pow(a**2+b**2,0.5)
c='%.3f'%c
print(c)
```

22.编程实现进制之间的转换。

```
print(int('0b1111011',2))#二进制转换成十进制：v = “0b1111011”
print(bin(18))#十进制转换成二进制：v = 18
```

```
print(int('011',8))#八进制转换成十进制：v = “011”
print(oct(30))#十进制转换成八进制：v = 30
print(int('0x12',16))#十六进制转换成十进制：v = “0x12”
print(hex(87))##十进制转换成十六进制：v = 8
```

23.定义函数求阶乘。要求对参数进行判断，对合法的参数求阶乘，参数零的阶乘返回1,并在参数为负数时输出-1。

```
def TEST_DO_NOT_CHANGE(num):
    rlt = None
    ##########start下面可以改动
    a=1
    for i in range(1,num+1):
        a*=i
    return a
print(TEST_DO_NOT_CHANGE(4))
```

24.定义一个函数，实现输入一句话，单词之间使用空格隔开，统计出其中各单词的词频数，并以keyword:count的格式存在一个dict中，返回这个dict。

注意 jkl; 不是一个单词，jkl 是一个单词，要把“; + ”等非字符符号去掉，即输入：“abc fjf jkl+ abc abc jkl;”结果是{"abc":3,"fjf":1,"jkl":2}

```
def TEST_DO_NOT_CHANGE(str_line):
    word_dict = {}
    ##########
    import re
    tempstrlist=re.findall('[A-Za-z]+',str_line)
    for  i in  tempstrlist:
        if  word_dict.get(i):
            word_dict[i]+=1
        else:
            word_dict[i] = 1
    return word_dict
print(TEST_DO_NOT_CHANGE("abc fjf jkl+ abc abc jkl;"))
```

25.计算并返回x的平方根整数部分，其中x是非负整数。由于返回类型是整数，结果只保留整数的部分，小数部分将被舍去。

```
def TEST_DO_NOT_CHANGE(x):
    rlt = None
    return int(x**0.5)
print(TEST_DO_NOT_CHANGE(3))
```

26.将整数列表中所有的偶数摘选出来并输出一个新的列表。

例如：nums = [1,2,3,4,5,6,7,8,9,10]，输出结果为[2,4,6,8,10]。请将以下列表中的偶数筛选出来：

num1=[11,16,26,7,31]；num2=[13,18,21,37,42]；num3=[31,17,46,12,77]。

```
def TEST_DO_NOT_CHANGE(nums):
    a=[]
    for n in nums:
        if n%2==0:
            a.append(n)
    return a
num1 = [11,16,26,7,31]
num2 = [13,18,21,37,42]
num3 = [31,17,46,12,77]
print(TEST_DO_NOT_CHANGE(num1))
print(TEST_DO_NOT_CHANGE(num2))
print(TEST_DO_NOT_CHANGE(num3))
```

27.给定一个整数 n，返回 n! 结果尾数中零的数量。

示例1：输入：3，输出：0，解释：3!= 6,尾数中没有零。

示例2：输入：5，输出：1，解释：5!=120,尾数中有1个零。

```
def TEST_DO_NOT_CHANGE(n):
    factorial = None
    num = 0
    for i in range(1, n+1):
        while i % 5 == 0:
            num += 1
            i = int(i / 5)
    return num
print(TEST_DO_NOT_CHANGE(3))
print(TEST_DO_NOT_CHANGE(5))
```

28.编写一个Python程序，接受两个正整数作为输入，并计算它们的最大公约数（GCD）。最大公约数是两个或多个整数的共有因子中的最大值。例如，18 和 24 的最大公约数是 6。编写一个函数TEST_DO_NOT_CHANGE(a,b)，它接受两个正整数a和b作为参数，返回它们的最大公约数。

```
def TEST_DO_NOT_CHANGE(a, b):
    x=None
    a1,b1 = a,b
    t=1
    for i in range(2,min(a,b)):
        while(a % i == 0 and b % i == 0):
            t=t*i
            a=a/i
            b=b/i
```

```
    return t
print(TEST_DO_NOT_CHANGE(18,24))
```

29.编写一个Python程序，计算并输出给定底数x和指数n的幂次方结果。

幂次方的定义如下：x的0次幂是1；x的正整数n次幂是x乘以自己n次，即x^n=x*x* ...*x(n次)。

例如，2的3次幂是2^3 =2*2*2=8。

请编写一个函数TEST_DO_NOT_CHANGE(x,n)，其中x为底数，n为指数，返回x^n的结果。

```
def TEST_DO_NOT_CHANGE(x, n):
    if n == 0:
        return 1
    elif n < 0:
        x = 1 / x
        n = -n
    ans = 1.0
    while n > 0:
        if n & 1:
            ans *= x
        x *= x
        n >>= 1
    return ans
print(TEST_DO_NOT_CHANGE(2, 3))
```

30.输入一行字符input_str，分别统计出其中英文字母、空格、数字和其他字符的个数，并将结果依次保存到列表lst_rlt。

输出示例：[中英文字母个数，空格个数，数字个数，其他字符个数]。

程序分析：利用 while 或 for 语句,只要输入的字符不为'\n'，就不退出程序。

```
def TEST_DO_NOT_CHANGE(input_str):
    lst_rlt = []
    ##########start下面可以改动
    letter = 0
    space = 0
    digit = 0
    other = 0
    for i in input_str:
        if i.isalpha():
            letter += 1
        elif i.isspace():  # 判断是否是空格
            space += 1
        elif i.isdigit():
            digit += 1  # 判断是否是数字
```

```
        else:
            other += 1
        # 判断是否是字母
    lst_rlt.append(letter)
    lst_rlt.append(space)
    lst_rlt.append(digit)
    lst_rlt.append(other)
    return lst_rlt
print(TEST_DO_NOT_CHANGE('a12 sdD3  d@@#'))
```

31.给定一个字符串s，由若干单词组成，单词之间用空格隔开。返回字符串中最后一个单词的长度。如果不存在最后一个单词，请返回 0。

单词，是指仅由字母组成，不包含任何空格字符的最大子字符串。

示例 1：输入：s = "Hello World"；输出：5。

示例 2：输入：s = " "；输出：0。

```
def TEST_DO_NOT_CHANGE(s):
    print(s)
    factorial = None
    ##########start下面可以改动
    s = s.strip()
    if not s:
        return 0
    else:
        return len(s.split(' ')[-1])
print(TEST_DO_NOT_CHANGE("Hello World"))
print(TEST_DO_NOT_CHANGE(" "))
```

32.将两个升序链表，合并为一个新的升序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例：

（1）输入：l1 = [1,2,4]，l2 = [1,3,4]，输出：[1,1,2,3,4,4]。

（2）输入：l1 = []，l2 = [0]，输出：[0]。

```
def TEST_DO_NOT_CHANGE( l1, l2):
    factorial = None
    ##########start下面可以改动
    i,j,l3 = 0,0,[]
    while i<len(l1) and j<len(l2):
        if l1[i] <= l2[j]:
            l3.append(l1[i])
            i += 1
        else:
```

```
            l3.append(l2[j])
            j += 1
    if len(l1) == i:
        while j<len(l2):
            l3.append(l2[j])
            j += 1
    else:
        while i<len(l1):
            l3.append(l1[i])
            i += 1
    return l3
l1 = [1,2,4]
l2 = [1,3,4]
print(TEST_DO_NOT_CHANGE( l1, l2))
l1 = []
l2 = [0]
print(TEST_DO_NOT_CHANGE( l1, l2))
```

33.使用26个字母和10个数字，产生一个随机组合的6位字符串。

提示：

（1）random.choice('tomorrow')；

（2）random.randint(1,10)；

（3）random.shuffle([1,3,5,6,7])。

```
import random
def TEST_DO_NOT_CHANGE():
    chars_ = [chr(i) for i in range(97,123)]
    nums_ = [str(i) for i in range(0,10)]
    random_str = ""
    random_str = ''.join([random.choice(chars_+nums_) for i in range(6)])
    return random_str
print(TEST_DO_NOT_CHANGE())
```

34.已知一个3*3矩阵A，A的元素依次为1—9的平方，求该矩阵主对角线元素之和。

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。

```
      |1^2 2^2 3^2|
A=    |4^2 5^2 6^2|
      |7^2 8^2 9^2|
def TEST_DO_NOT_CHANGE():
    rlt_sum=0
    A = [[1**2, 2**2, 3**2],
```

```
        [4**2, 5**2, 6**2],
        [7**2, 8**2, 9**2],
        ]
    ##########start下面可以改动
    sum=0
    for i in range(3):
        sum+=A[i][i]
    return sum
print(TEST_DO_NOT_CHANGE())
```

35.给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，那么目标值通过排序插入数组中，并返回目标值插入数组中的位置。可以假设数组中无重复元素。

```
def TEST_DO_NOT_CHANGE(nums, target):
    factorial = None
    for i in nums:
        if i>=target:
            return nums.index(i)
    if nums[0]>=target:
        return 0
    else:
        return len(nums)
nums=[1,2,3,4,5,6,7,8]
target=9
print(TEST_DO_NOT_CHANGE(nums, target))
```

36.取一个整数a从右端开始的4~7位。

```
a=int(input('输入一个数字:'))
b=0                 #     0
b=~b                #     1
b=b<<4              # 10000
b=~b                #  1111
c=a>>4
d=c&b
print('a:',bin(a))
print('b:',bin(b))
print('c:',bin(c))
print('d:',bin(d))
```

37.打印出杨辉三角形前十行。

```
def generate(numRows):
```

```
    r = [[1]]
    for i in range(1,numRows):
        r.append(list(map(lambda x,y:x+y, [0]+r[-1],r[-1]+[0])))
    return r[:numRows]
a=generate(10)
for i in a:
    print(i)
```

38.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。

```
if __name__ == '__main__':
    nmax = 50
    n = int(input('请输入总人数:'))
    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 += 1
        if i == n: i = 0
    i = 0
    while num[i] == 0: i += 1
    print(num[i])
```

39.求0—7所能组成的奇数的个数。

程序分析：

（1）组成1位数是4个。1、3、5、7结尾。

（2）组成2位数是7*4个。第1位不能为0。

（3）组成3位数是784个。中间随意。

（4）组成4位数是788*4个。

```
if __name__ == '__main__':
    sum = 4
    s = 4
    for j in range(2,9):
```

```
        print (sum)
        if j <= 2:
            s *= 7
        else:
            s *= 8
        sum += s
    print('sum = %d' % sum)
```

40.写一个函数，求出所有的水仙花数。调用这个函数，打印出来所有水仙花数。

水仙花数（Narcissistic number）也被称为超完全数字不变数（pluperfect digital invariant, PPDI）、自恋数、自幂数、阿姆斯壮数或阿姆斯特朗数（Armstrong number），水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身。例如：1^3 + 5^3+ 3^3 = 153。

```
def find_numbers():
    for num in range(100, 1000):
        sum_of_cubes = sum(int(digit) ** 3 for digit in str(num))
        if sum_of_cubes == num:
            print(num)
find_numbers()
```

41.写一个函数，求出所有的四叶玫瑰数。调用这个函数，打印出来所有四叶玫瑰数。四叶玫瑰数是 4 位数的自幂数。自幂数是指一个 n 位数，它的每个位上的数字的 n 次幂之和等于它本身。当 n 为 3 时，有 1^3 + 5^3 + 3^3 = 153，153 即是 n 为 3 时的一个自幂数，3 位数的自幂数被称为水仙花数。

```
def find_four_leaf_rose_numbers():
    for num in range(1000, 10000):
        sum_of_powers = sum(int(digit) ** 4 for digit in str(num))
        if num == sum_of_powers:
            print(num)
find_four_leaf_rose_numbers()
```

42.随机生成一个100以内的整数，共有10次机会开始游戏。输入猜测的数字，如果猜小了，则提示“猜小了”；如果猜大了，则提示“猜大了”；如果猜对了，则提示“猜对了”；结束游戏10次机会用完还没猜对，提示“游戏结束，没有猜到”。

```
import random
def guess_number_game():
    target_number = random.randint(1, 100)
    attempts = 10
    print("猜数字游戏开始!我想了一个 1 到 100 之间的整数。你有 10 次机会猜对它。")
    for attempt in range(attempts):
        try:
```

```
            guess = int(input(f"机会 {attempt + 1} ：请输入你的猜测："))
            if guess < target_number:
                print("猜小了")
            elif guess > target_number:
                print("猜大了")
            else:
                print("猜对了！恭喜你！")
                return
        except ValueError:
            print("请输入有效的整数！")
    print(f"游戏结束，没有猜到。正确答案是 {target_number}。")
guess_number_game()
```

43.公鸡每只5元，母鸡每只3元，小鸡3只1元，现要求用100元钱买100只鸡(三种类型的鸡都要买)，问公鸡、母鸡、小鸡各买几只?

```
def buy_chickens():
    for rooster in range(1, 20):
        for hen in range(1, 33):
            chick = 100 - rooster - hen
            if 5 * rooster + 3 * hen + chick / 3 == 100:
                print(f"公鸡：{rooster} 只，母鸡：{hen} 只，小鸡：{chick} 只")
buy_chickens()
```

44.输入年月日，输出该日期是否为闰年，并且输出该日期是此年份的第几天。

闰年判断条件：能被4整除，并且不能被100整除；能被400整除。两个条件满足任意一个就为闰年。

```
def is_leap_year(year):
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
def day_of_year(year, month, day):
    days_per_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    if is_leap_year(year) and month > 2:
        days_per_month[1] = 29
    return sum(days_per_month[:month - 1]) + day
date_input = input("请输入日期（格式为 YYYY-MM-DD）: ")
year, month, day = map(int, date_input.split('-'))
if is_leap_year(year):
    print(f"{year} 是闰年")
else:
    print(f"{year} 不是闰年")
print(f"该日期是此年份的第 {day_of_year(year, month, day)} 天")
```

45.打印九九乘法表，在控制台输出。

```
for i in range(1, 10):
    for j in range(1, i + 1):
        print(f"{j}x{i}={i * j}", end="\t")
    print()
```

46.游戏开始，初始状态下用户和电脑都有 100 分，赢一局+10 分，输一局-10 分。

当用户为 0 分时，游戏结束，提示游戏结束，比赛输了。当用户为 200 分时，游戏结束，提示游戏结束，比赛赢了。每轮比赛都输出当前的分数。

```
import random
def rock_paper_scissors():
    user_score = 100
    computer_score = 100
    choices = ["剪刀", "石头", "布"]
    while user_score > 0 and user_score < 200 and computer_score > 0:
        print(f"\n 当 前 分 数 - 你 的 分 数 ： {user_score} ， 电 脑 的 分 数 ：
{computer_score}")
        user_choice = input("请选择 剪刀、石头、布: ")
        if user_choice not in choices:
            print("无效选择，请输入 剪刀、石头 或 布。")
            continue
        computer_choice = random.choice(choices)
        print(f"电脑选择了：{computer_choice}")
        if user_choice == computer_choice:
            print("平局！")
        elif (user_choice == "剪刀" and computer_choice == "布") or \
                (user_choice == "石头" and computer_choice == "剪刀") or \
                (user_choice == "布" and computer_choice == "石头"):
            print("你赢了！")
            user_score += 10
            computer_score -= 10
        else:
            print("你输了！")
            user_score -= 10
            computer_score += 10
    if user_score <= 0:
        print("\n游戏结束，比赛输了。")
    elif user_score >= 200:
        print("\n游戏结束，赢得比赛!")
rock_paper_scissors()
```