1. Python是一种跨平台、开源、免费的高级动态编程语言。
2. Python 3.x完全兼容Python 2.x。
3. 已知 x = 3，那么赋值语句 x = 'abcedfg' 是无法正常执行的。
4. Python变量使用前必须先声明，并且一旦声明就不能再当前作用域内改变其类型。
5. Python采用的是基于值得自动内存管理方式。
6. Python不允许使用关键字作为变量名，允许使用内置函数名作为变量名，但这会改变函数名的含义。
7. 在Python中可以使用if作为变量名。
8. 在Python 3.x中可以使用中文作为变量名。
9. Python变量名必须以字母或下划线开头，并且区分字母大小写。
10. 不管输入什么，Python 3.x中input()函数的返回值总是字符串。
11. Python使用缩进来体现代码之间的逻辑关系。
12. 在Python中可以使用 id 作为变量名，尽管不建议这样做。
13. Python关键字不可以作为变量名。
14. 执行语句 from math import sin 之后，可以直接使用 sin() 函数，例如 sin(3)。
15. Python变量名区分大小写，所以student和Student不是同一个变量。
16. 如果只需要math模块中的sin()函数，建议使用from math import sin来导入，而不要使用import math导入整个模块。
17. 在Python 3.x中，使用内置函数input()接收用户输入时，不论用户输入的什么格式，一律按字符串进行返回。
18. Python支持使用字典的“键”作为下标来访问字典中的值。
19. 列表可以作为字典的“键”。
20. 元组可以作为字典的“键”。
21. 字典的“键”必须是不可变的。
22. 已知x为非空列表，那么表达式 sorted(x, reverse=True) == list(reversed(x)) 的值一定是True。
23. 已知x为非空列表，那么x.sort(reverse=True)和x.reverse()的作用是等价的。
24. Python集合中的元素不允许重复。
25. Python字典中的“键”不允许重复。
26. Python字典中的“值”不允许重复。
27. Python字典中的“键”可以是元组。
28. Python列表中所有元素必须为相同类型的数据。
29. Python列表、元组、字符串都属于有序序列。
30. 列表对象的append()方法属于原地操作，用于在列表尾部追加一个元素。
31. 使用Python列表的方法insert()为列表插入元素时会改变列表中插入位置之后元素的索引。
32. 假设x为列表对象，那么x.pop()和x.pop(-1)的作用是一样的。
33. 使用del命令或者列表对象的remove()方法删除列表中元素时会影响列表中部分元素的索引。
34. 已知列表 x = [1, 2, 3]，那么执行语句 x = 3 之后，变量x的地址不变。
35. 使用列表对象的remove()方法可以删除列表中首次出现的指定元素，如果列中不存在要删除的指定元素则抛出异常。
36. 元组是不可变的，不支持列表对象的inset()、remove()等方法，也不支持del命令删除其中的元素，但可以使用del命令删除整个元组对象。
37. Python字典和集合属于无序序列。
38. 无法删除集合中指定位置的元素，只能删除特定值的元素。
39. 元组的访问速度比列表要快一些，如果定义了一系列常量值，并且主要用途仅仅是对其进行遍历二不需要进行任何修改，建议使用元组而不使用列表。
40. 当以指定“键”为下标给字典对象赋值时，若该“键”存在则表示修改该“键”对应的“值”，若不存在则表示为字典对象添加一个新的“键-值对”。
41. 只能通过切片访问列表中的元素，不能使用切片修改列表中的元素。
42. 只能通过切片访问元组中的元素，不能使用切片修改元组中的元素。
43. Python集合不支持使用下标访问其中的元素。
44. 删除列表中重复元素最简单的方法是将其转换为集合后再重新转换为列表。
45. 已知列表x中包含超过5个以上的元素，那么语句 x = x[:5]+x[5:] 的作用是将列表x中的元素循环左移5位。
46. 对于生成器对象x = (3 for i in range(5))，连续两次执行list(x)的结果是一样的。
47. 对于大量列表的连接，extend()方法比运算符+具有更高的效率。
48. 列表对象的extend()方法属于原地操作，调用前后列表对象的地址不变。
49. 对于数字n，如果表达式 0 not in [n%d for d in range(2, n)] 的值为True则说明n是素数。
50. 创建只包含一个元素的元组时，必须在元素后面加一个逗号，例如(3,)。
51. 表达式 list('[1, 2, 3]') 的值是[1, 2, 3]。
52. 同一个列表对象中的元素类型可以各不相同。
53. 已知x为非空列表，那么执行语句x[0] = 3之后，列表对象x的内存地址不变。
54. 列表可以作为集合的元素。
55. 集合可以作为列表的元素。
56. 元组可以作为集合的元素。
57. 集合可以作为元组的元素。
58. 字典可以作为集合的元素。
59. 列表对象的pop()方法默认删除并返回最后一个元素，如果列表已空则抛出异常。
60. 表达式 {1, 2} \* 2 的值为 {1, 2, 1, 2}。
61. 假设re模块已成功导入，并且有 pattern = re.compile('^'+'\.'.join([r'\d{1,3}' for i in range(4)])+'$')，那么表达式 pattern.match('192.168.1.103') 的值为None。
62. 已知x = (1, 2, 3, 4)，那么执行x[0] = 5之后，x的值为(5, 2, 3, 4)。
63. 已知x = 3，那么执行x += 6语句前后x的内存地址是不变的。
64. 内置函数len()返回指定序列的元素个数，适用于列表、元组、字符串、字典、集合以及range、zip等迭代对象。
65. 已知x和y是两个等长的整数列表，那么表达式sum((i\*j for i, j in zip(x, y)))的作用是计算这两个列表所表示的向量的内积。
66. 已知x和y是两个等长的整数列表，那么表达式[i+j for i,j in zip(x,y)]的作用时计算这两个列表所表示的向量的和。
67. 表达式int('1'\*64, 2)与sum(2\*\*i for i in range(64))的计算结果是一样的，但是前者更快一些。
68. 已知x = list(range(20))，那么语句del x[::2]可以正常执行。
69. 已知x = list(range(20))，那么语句x[::2] = []可以正常执行。
70. 已知x = list(range(20))，那么语句print(x[100:200])无法正常执行。
71. 已知x是个列表对象，那么执行语句y = x之后，对y所做的任何操作都会同样作用到x上。
72. 已知x是个列表对象，那么执行语句y = x[:]之后，对y所做的任何操作都会同样作用到x上。
73. 表达式(i\*\*2 for i in range(100))的结果是个元组。
74. 在Python中元组的值是不可变的，因此，已知x = ([1], [2])，那么语句x[0].append(3)是无法正常执行的。
75. Python内置的字典dict中元素是按添加的顺序依次进行存储的。
76. Python内置的集合set中元素顺序是按元素的哈希值进行存储的，并不是按先后顺序。
77. 已知x = {1:1, 2:2}，那么语句x[3] =3无法正常执行。
78. 已知列表x = [1, 2, 3, 4]，那么表达式x.find(5)的值应为-1。
79. 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。
80. 已知x是一个列表，那么x = x[3:] + x[:3]可以实现把列表x中的所有元素循环左移3位。
81. 如果仅仅是用于控制循环次数，那么使用for i in range(20)和for i in range(20, 40)的作用是等价的。
82. 在循环中continue语句的作用是跳出当前循环。
83. 在编写多层循环时，为了提高运行效率，应尽量减少内循环中不必要的计算。
84. 带有else子句的循环如果因为执行了break语句而退出的话，则会执行else子句中的代码。
85. 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。
86. 在条件表达式中不允许使用赋值运算符“=”，会提示语法错误。
87. Python运算符%不仅可以用来求余数，还可以用来格式化字符串。
88. 如果需要连接大量字符串成为一个字符串，那么使用字符串对象的join()方法比运算符+具有更高的效率。
89. 正则表达式模块re的match()方法是从字符串的开始匹配特定模式，而search()方法是在整个字符串中寻找模式，这两个方法如果匹配成功则返回match对象，匹配失败则返回空值None。
90. 已知x为非空字符串，那么表达式 ''.join(x.split()) == x 的值一定为True。
91. 已知x为非空字符串，那么表达式 ','.join(x.split(',')) == x 的值一定为True。
92. 当作为条件表达式时，[]与None等价。
93. 当作为条件表达式时，{}与None等价。
94. 当作为条件表达式时，空值、空字符串、空列表、空元组、空字典、空集合、空迭代对象以及任意形式的数字0都等价于False。
95. 正则表达式对象的match()方法可以在字符串的指定位置开始进行指定模式的匹配。
96. 使用正则表达式对字符串进行分割时，可以指定多个分隔符，而字符串对象的split()方法无法做到这一点。
97. 正则表达式元字符“^”一般用来表示从字符串开始处进行匹配，用在一对方括号中的时候则表示反向匹配，不匹配方括号中的字符。
98. 正则表达式元字符“\s”用来匹配任意空白字符。
99. 正则表达式 元字符“\d”用来匹配任意数字字符。
100. 已知x和y是两个字符串，那么表达式sum((1 for i,j in zip(x,y) if i==j))可以用来计算两个字符串中对应位置字符相等的个数。
101. 已知x = 'hellow world.'.encode()，那么表达式x.decode('gbk')的值为'hellow world.'。
102. 正则表达式'^http'只能匹配所有以'http'开头的字符串。
103. 正则表达式'^\d{18}|\d{15}$'只能检查给定字符串是否为18位或15位数字字符，并不能保证一定是合法的身份证号。
104. 正则表达式'[^abc]'可以一个匹配任意除'a'、'b'、'c'之外的字符。
105. 正则表达式'python|perl'或'p(ython|erl)'都可以匹配'python'或'perl'。
106. 调用函数时，在实参前面加一个型号\*表示序列解包。
107. 函数是代码复用的一种方式。
108. 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。
109. 编写函数时，一般建议先对参数进行合法性检查，然后再编写正常的功能代码。
110. 一个函数如果带有默认值参数，那么必须所有参数都设置默认值。
111. 定义Python函数时必须指定函数返回值类型。
112. 定义Python函数时，如果函数中没有return语句，则默认返回空值None。
113. 函数中必须包含return语句。
114. 不同作用域中的同名变量之间互相不影响，也就是说，在不同的作用域内可以定义同名的变量。
115. 函数内部定义的局部变量当函数调用结束后被自动删除。
116. 在函数内部，既可以使用global来声明使用外部全局变量，也可以使用global直接定义全局变量。
117. 在函数内部没有办法定义全局变量。
118. 在函数内部直接修改形参的值并不影响外部实参的值。
119. 调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值。
120. 在同一个作用域内，局部变量会隐藏同名的全局变量。
121. 形参可以看做是函数内部的局部变量，函数运行结束之后形参就不可访问了。
122. 在函数内部没有任何声明的情况下直接为某个变量赋值，这个变量一定是函数内部的局部变量。
123. 在Python中定义函数时不需要声明函数参数的类型。
124. 在Python中定义函数时不需要声明函数的返回值类型。
125. 在函数中没有任何办法可以通过形参来影响实参的值。
126. 已知 x = 3，那么执行语句 x+=6 之后，x的内存地址不变。
127. 在定义函数时，某个参数名字前面带有一个\*符号表示可变长度参数，可以接收任意多个普通实参并存放于一个元组之中。
128. 在定义函数时，某个参数名字前面带有两个\*符号表示可变长度参数，可以接收任意多个关键参数并将其存放于一个字典之中。
129. 定义函数时，带有默认值的参数必须出现在参数列表的最右端，任何一个带有默认值的参数右边不允许出现没有默认值的参数。
130. 在调用函数时，可以通过关键参数的形式进行传值，从而避免必须记住函数形参顺序的麻烦。
131. lambda表达式中可以使用任意复杂的表达式，但是必须只编写一个表达式。
132. g = lambda x: 3不是一个合法的赋值表达式。
133. 语句pass仅起到占位符的作用，并不会做任何操作。
134. 在Python中定义类时，如果某个成员名称前有2个下划线则表示是私有成员。
135. 在类定义的外部没有任何办法可以访问对象的私有成员。
136. Python中一切内容都可以称为对象。
137. 定义类时所有实例方法的第一个参数用来表示对象本身，在类的外部通过对象名来调用实例方法时不需要为该参数传值。
138. 面向对象程序设计中，函数和方法是完全一样的，都必须为所有参数进行传值。
139. Python中没有严格意义上的私有成员。
140. 在IDLE交互模式下，一个下划线“\_”表示解释器中最后一次显示的内容或最后一次语句正确执行的输出结果。
141. 对于Python类中的私有成员，可以通过“对象名.\_类名\_\_私有成员名”的方式来访问。
142. 如果定义类时没有编写析构函数，Python将提供一个默认的析构函数进行必要的资源清理工作。
143. Python支持多继承，如果父类中有相同的方法名，而在子类中调用时没有指定父类名，则Python解释器将从左向右按顺序进行搜索。
144. 在Python中定义类时实例方法的第一个参数名称不管是什么，都表示对象自身。
145. Python类的构造函数是\_\_init\_\_()。
146. 定义类时，在一个方法前面使用@classmethod进行修饰，则该方法属于类方法。
147. 定义类时，在一个方法前面使用@staticmethod进行休息，则该方法属于静态方法。
148. Python类不支持多继承。
149. 函数和对象方法是一样的，内部实现和外部调用都没有任何区别。错
150. 扩展库os中的方法remove()可以删除带有只读属性的文件。
151. 使用内置函数open()且以”w”模式打开的文件，文件指针默认指向文件尾。
152. 使用内置函数open()打开文件时，只要文件路径正确就总是可以正确打开的。
153. 对文件进行读写操作之后必须显式关闭文件以确保所有内容都得到保存。
154. 二进制文件不能使用记事本程序打开。
155. 使用普通文本编辑器软件也可以正常查看二进制文件的内容。
156. 二进制文件也可以使用记事本或其他文本编辑器打开，但是一般来说无法正常查看其中的内容。
157. Python标准库os中的方法exists()可以用来测试给定路径的文件是否存在。
158. 文件对象的tell()方法用来返回文件指针的当前位置。
159. 以读模式打开文件时，文件指针指向文件开始处。
160. 以追加模式打开文件时，文件指针指向文件尾。
161. 二进制文件也可以使用记事本程序打开，只是无法正确阅读和理解其中的内容。
162. 文本文件是可以迭代的，可以使用for line in fp类似的语句遍历文件对象fp中的每一行。