1. Python标准库math中用来计算平方根的函数是\_\_\_\_\_\_\_\_\_\_。
2. 在Python中\_\_\_\_\_\_\_\_\_\_表示空类型。
3. 查看变量类型的Python内置函数是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
4. 查看变量内存地址的Python内置函数是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
5. Python运算符中用来计算整商的是\_\_\_\_\_\_\_\_\_。
6. \_\_\_\_\_\_\_\_\_\_\_\_\_\_命令既可以删除列表中的一个元素，也可以删除整个列表。
7. 表达式 abs(-3) 的值为\_\_\_\_\_\_\_\_\_\_\_。
8. Python 3.x语句 print(1, 2, 3, sep=':') 的输出结果为\_\_\_\_\_\_\_\_\_\_\_\_。
9. 表达式 int(4\*\*0.5) 的值为\_\_\_\_\_\_\_\_\_\_\_\_。
10. 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。
11. Python内置函数\_\_\_\_\_\_\_\_\_\_\_\_用来返回序列中的最大元素。
12. Python内置函数\_\_\_\_\_\_\_\_\_\_\_\_用来返回序列中的最小元素。
13. Python内置函数\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_用来返回数值型序列中所有元素之和。
14. 已知 x = 3，那么执行语句 x += 6 之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
15. 表达式type({3})的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
16. 表达式[1, 2, 3]\*3的执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
17. list(map(str, [1, 2, 3]))的执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
18. 表达式[3] in [1, 2, 3, 4]的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
19. 假设列表对象aList的值为[3, 4, 5, 6, 7, 9, 11, 13, 15, 17]，那么切片aList[3:7]得到的值是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
20. 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
21. 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\_\_\_\_\_\_\_\_。
22. Python语句list(range(1,10,3))执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
23. 表达式 list(range(5)) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
24. 使用切片操作在列表对象x的开始处增加一个元素3的代码为\_\_\_\_\_\_\_\_\_。
25. 语句 x = (3,) 执行后x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
26. 语句 x = (3) 执行后x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
27. 已知x=3和y=5，执行语句 x, y = y, x 后x的值是\_\_\_\_。
28. 已知 x = {1:2}，那么执行语句 x[2] = 3之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
29. 表达式set([1, 1, 2, 3])的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
30. 使用列表推导式得到100以内所有能被13整除的数的代码可以写作\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
31. 已知 x = [3, 5, 7]，那么表达式 x[10:]的值为\_\_\_\_\_\_\_\_\_\_\_\_。
32. 已知 x = [3, 7, 5]，那么执行语句 x.sort(reverse=True)之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
33. 表达式 list(zip([1,2], [3,4])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
34. 已知 x = [1, 2, 3, 2, 3]，执行语句 x.pop() 之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。
35. 表达式 [x for x in [1,2,3,4,5] if x<3] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
36. 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
37. 已知 x = list(range(10))，则表达式 x[-4:] 的值为\_\_\_\_\_\_\_\_\_\_。
38. 已知 x = [3, 5, 7]，那么执行语句 x[1:] = [2]之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
39. 已知 x = [3, 5, 7]，那么执行语句 x[:3] = [2]之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
40. 已知 x = [1, 2, 3, 2, 3]，执行语句 x.remove(2) 之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_。
41. 表达式 len([i for i in range(10)]) 的值为\_\_\_\_\_\_\_\_\_\_。
42. 已知 x = (3,)，那么表达式 x \* 3 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。
43. 已知列表 x = list(range(10))，那么执行语句 del x[::2]之后，x的值为\_\_\_\_\_\_\_\_\_。
44. 已知列表 x = [1, 2, 3, 4]，那么执行语句 del x[1] 之后x的值为\_\_\_\_\_\_\_\_\_\_\_\_。
45. 已知列表 x = [1, 2, 3]，那么执行语句 x.insert(1, 4) 只有，x的值为\_\_\_\_\_\_\_\_\_\_\_。
46. 表达式 list(map(lambda x: x+5, [1, 2, 3, 4, 5])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
47. 已知 x = [1, 2, 3]，那么执行语句 x[len(x)-1:] = [4, 5, 6]之后，变量x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
48. 已知x是一个列表对象，那么执行语句 y = x[:] 之后表达式 id(x) == id(y) 的值为\_\_\_\_\_\_\_\_\_\_\_。
49. 表达式 sorted([13, 1, 237, 89, 100], key=lambda x: len(str(x))) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
50. 已知 x = {1:2, 2:3}，那么表达式 x.get(3, 4) 的值为\_\_\_\_。
51. 已知 x = {1:2, 2:3}，那么表达式 x.get(2, 4) 的值为\_\_\_\_\_\_。
52. 表达式 [1,2,3].count(4) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
53. 执行代码 x, y, z = sorted([1, 3, 2]) 之后，变量y的值为\_\_\_\_\_\_。
54. 表达式 dict(zip([1, 2], [3, 4])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
55. 已知 x = {1:1}，那么执行语句 x[2] = 2之后，len(x)的值为\_\_\_\_\_\_\_\_\_\_\_\_。
56. 已知 x = {1:1, 2:2}，那么执行语句 x[2] = 4之后，len(x)的值为\_\_\_\_\_\_\_\_\_\_\_\_。
57. 已知有函数定义 def demo(\*p):return sum(p)，那么表达式 demo(1, 2, 3) 的值为\_\_\_\_\_\_、表达式 demo(1, 2, 3, 4) 的值为\_\_\_\_\_\_\_\_\_。
58. 已知列表 x = [1, 2]，那么连续执行命令 y = x和 y.append(3) 之后，x的值为\_\_\_\_\_\_\_\_。
59. 已知列表 x = [1, 2]，那么连续执行命令 y = x[:] 和 y.append(3) 之后，x的值为\_\_\_\_\_\_\_\_。
60. 已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 id(x) == id(y) 的值为\_\_\_\_\_\_\_。
61. 已知列表 x = [1, 2]，执行语句 y = x 后，表达式 id(x) == id(y) 的值为\_\_\_\_\_\_\_。
62. 已知列表 x = [1, 2]，执行语句 y = x 后，表达式 x is y 的值为\_\_\_\_\_\_\_。
63. 已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 x is not y 的值为\_\_\_\_\_\_\_。
64. 表达式 [i for i in range(10) if i>8] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
65. 执行语句 x,y,z = map(str, range(3)) 之后，变量y的值为\_\_\_\_\_\_\_\_\_\_\_。
66. 已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[:3] 之后，x的值为\_\_\_\_\_\_\_\_\_\_。
67. 已知 x = [1, 2]，那么执行语句 x[0:0] = [3, 3]之后，x的值为\_\_\_\_\_\_\_\_\_\_\_。
68. 已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[1:3] 之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_。
69. 已知 x = {1:2, 2:3, 3:4}，那么表达式 sum(x.values()) 的值为\_\_\_\_\_\_\_\_\_。
70. 已知列表 x = [1, 3, 2]，那么执行语句 x.reverse() 之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_。
71. 已知列表 x = [1, 3, 2]，那么执行语句 y = list(reversed(x)) 之后，x的值为\_\_\_\_\_\_\_\_\_\_。
72. 已知列表 x = [1, 3, 2]，那么执行语句 y = list(reversed(x)) 之后，y的值为\_\_\_\_\_\_\_\_\_\_。
73. 已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 sum(x) 的值为\_\_\_\_\_\_。
74. 已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 sum(item[0] for item in x.items()) 的值为\_\_\_\_\_\_\_\_\_\_\_。
75. 已知列表 x = [1, 3, 2]，那么执行语句 a, b, c = sorted(x) 之后，b的值为\_\_\_\_。
76. 已知列表 x = [1, 3, 2]，那么执行语句 a, b, c = map(str,sorted(x)) 之后，c的值为\_\_\_\_\_\_。
77. 表达式 'abc' in 'abdcefg' 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。
78. 已知 x, y = 3, 5，那么执行x, y = y, x 之后，x的值为\_\_\_\_\_\_\_\_\_\_。
79. 已知 x = 'abcd' 和 y = 'abcde'，那么表达式 [i==j for i,j in zip(x,y)] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
80. 已知x = list(range(20))，那么表达式x[-1]的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
81. 已知x = 3+4j和y = 5+6j，那么表达式x+y的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
82. 已知x = [3]，那么执行x += [5]之后x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
83. 已知x = [3, 2, 4, 1]，那么执行语句x = x.sort()之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_。
84. 表达式list(filter(lambda x: x>5, range(10)))的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
85. 已知x = list(range(20))，那么语句print(x[100:200])的输出结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
86. 已知x = list(range(20))，那么执行语句x[:18] = []后列表x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
87. 已知x = [1, 2, 3]，那么连续执行y = x[:]和y.append(4)这两条语句之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
88. 已知x = [1, 2, 3]，那么连续执行y = x和y.append(4)这两条语句之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
89. 已知x = [1, 2, 3]，那么连续执行y = [1, 2, 3]和y.append(4)这两条语句之后，x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
90. 已知x = ([1], [2])，那么执行语句x[0].append(3)后x的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
91. 表达式 True\*3 的值为\_\_\_\_\_\_\_\_\_\_。
92. 表达式 False+1 的值为\_\_\_\_\_\_\_\_\_\_。
93. 表达式 'ab' in 'acbed' 的值为\_\_\_\_\_\_\_\_。
94. Python 3.x语句 for i in range(3):print(i, end=',') 的输出结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
95. Python 3.x语句 print(1, 2, 3, sep=',') 的输出结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
96. 对于带有else子句的for循环和while循环，当循环因循环条件不成立而自然结束时\_\_\_\_\_\_\_\_(会？不会？)执行else中的代码。
97. 在循环语句中，\_\_\_\_\_\_\_\_\_\_语句的作用是提前结束本层循环。
98. 在循环语句中，\_\_\_\_\_\_\_语句的作用是提前进入下一次循环。
99. 表达式 5 if 5>6 else (6 if 3>2 else 5) 的值为\_\_\_\_\_\_\_\_\_。
100. 表达式 'abc' in ('abcdefg') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
101. 表达式 'abc' in ['abcdefg'] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
102. Python语句''.join(list('hello world!'))执行的结果是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
103. 表达式 'abcabcabc'.rindex('abc') 的值为\_\_\_\_\_\_\_\_\_\_\_\_。
104. 表达式 ':'.join('abcdefg'.split('cd')) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
105. 表达式 'Hello world. I like Python.'.rfind('python') 的值为\_\_\_\_\_\_\_\_。
106. 表达式 'abcabcabc'.count('abc') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。
107. 表达式 'apple.peach,banana,pear'.find('p') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
108. 表达式 'apple.peach,banana,pear'.find('ppp') 的值为\_\_\_\_\_\_\_\_。
109. 表达式 'abcdefg'.split('d') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
110. 表达式 ':'.join('1,2,3,4,5'.split(',')) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
111. 已知 x = '123' 和 y = '456'，那么表达式 x + y 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
112. 表达式 ''.join('asdssfff'.split('sd')) 的值为\_\_\_\_\_\_\_\_\_\_\_\_。
113. 表达式 ''.join(re.split('[sd]','asdssfff')) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
114. 假设re模块已导入，那么表达式 re.findall('(\d)\\1+', '33abcd112') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
115. 语句 print(re.match('abc', 'defg')) 输出结果为\_\_\_\_\_\_\_\_\_\_\_\_\_。
116. 表达式 'Hello world!'[-4] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
117. 表达式 'Hello world!'[-4:] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
118. 表达式 len('abc'.ljust(20)) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
119. 假设正则表达式模块re已导入，那么表达式 re.sub('\d+', '1', 'a12345bbbb67c890d0e') 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
120. 表达式 [str(i) for i in range(3)] 的值为\_\_\_\_\_\_\_\_\_\_\_\_。
121. 表达式 'abc.txt'.endswith(('.txt', '.doc', '.jpg')) 的值为\_\_\_\_\_\_\_\_\_\_\_。
122. 代码 print(1,2,3,sep=':') 的执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
123. 代码 for i in range(3):print(i, end=',') 的执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
124. 表达式 'Hello world!'.count('l') 的值为\_\_\_\_\_\_\_\_\_\_\_。
125. 已知 x = 'abcdefg'，则表达式 x[3:] + x[:3] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
126. 已知x = 'hello world.'，那么表达式x.find('x')和x.rfind('x')的值都为\_\_\_\_\_\_\_\_\_\_\_\_\_。
127. 表达式':'.join('hello world.'.split())的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
128. 已知x = 'hello world'，那么表达式x.replace('l', 'g')的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
129. 假设正则表达式模块re已正确导入，那么表达式''.join(re.findall('\d+', 'abcd1234'))的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
130. 假设正则表达式模块re已正确导入，那么表达式re.findall('\d+?', 'abcd1234')的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。
131. 正则表达式模块re的\_\_\_\_\_\_\_\_\_\_方法用来编译正则表达式对象。
132. 正则表达式模块re的\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法用来在字符串开始处进行指定模式的匹配。
133. 正则表达式模块re的\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法用来在整个字符串中进行指定模式的匹配。
134. 表达式 'Beautiful is better than ugly.'.startswith('Be', 5) 的值为\_\_\_\_\_\_\_\_\_。
135. 表达式 list(filter(lambda x:x>2, [0,1,2,3,0,0])) 的值为\_\_\_\_\_\_\_\_\_。
136. 表达式 list(filter(lambda x: len(x)>3, ['a', 'b', 'abcd'])) 的值为\_\_\_\_\_\_\_\_\_\_\_。
137. 已知 g = lambda x, y=3, z=5: x\*y\*z，则语句 print(g(1)) 的输出结果为\_\_\_\_\_\_\_。
138. 表达式 list(map(lambda x: len(x), ['a', 'bb', 'ccc'])) 的值为\_\_\_\_\_\_\_\_\_\_。
139. 已知 f = lambda x: x+5，那么表达式 f(3) 的值为\_\_\_\_\_\_\_\_。
140. 已知 g = lambda x, y=3, z=5: x+y+z，那么表达式 g(2) 的值为\_\_\_\_\_\_\_\_。
141. 已知 f = lambda x: 5，那么表达式 f(3)的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。
142. Python使用\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_关键字来定义类。