《Python程式設計》題庫

1. 填空題
2. Python安裝擴展庫常用的是\_\_\_\_\_\_\_工具。（pip）
3. Python標準庫math中用來計算平方根的函數是\_\_\_\_\_\_\_\_\_\_。（sqrt）
4. Python程式檔副檔名主要有\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_兩種，其中後者常用於GUI程式。（py、pyw）
5. Python原始程式碼程式編譯後的檔副檔名為\_\_\_\_\_\_\_\_\_。（pyc）
6. 使用pip工具升級科學計算擴展庫numpy的完整命令是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（pip install --upgrade numpy）
7. 使用pip工具查看當前已安裝的Python擴展庫的完整命令是\_\_\_\_\_\_\_\_\_\_\_\_\_。（pip list）
8. 在IDLE交互模式中流覽上一條語句的快速鍵是\_\_\_\_\_\_\_\_\_\_。（Alt+P）
9. 使用pip工具查看當前已安裝Python擴展庫列表的完整命令是\_\_\_\_\_\_\_\_\_。（pip list）
10. 在Python中\_\_\_\_\_\_\_\_\_\_表示空類型。（None）
11. 清單、元組、字串是Python的\_\_\_\_\_\_\_\_\_(有序？無序)序列。（有序）
12. 查看變數類型的Python內置函數是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（type()）
13. 查看變數記憶體位址的Python內置函數是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（id()）
14. 以3為實部4為虛部，Python複數的表達形式為\_\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_。（3+4j、3+4J）
15. Python運算子中用來計算整商的是\_\_\_\_\_\_\_\_\_。（//）
16. Python運算子中用來計算集合並集的是\_\_\_\_\_\_\_。（|）
17. 使用運算子測試集合包含集合A是否為集合B的真子集的運算式可以寫作\_\_\_\_\_\_\_。（A<B）
18. 運算式[1, 2, 3]\*3的執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 1, 2, 3, 1, 2, 3]）
19. list(map(str, [1, 2, 3]))的執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[‘1’, ‘2’, ‘3’]）
20. 語句x = 3==3, 5執行結束後，變數x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（(True, 5））
21. 已知 x = 3，那麼執行語句 x += 6 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（9）
22. 已知 x = 3，並且id(x)的返回值為 496103280，那麼執行語句 x += 6 之後，運算式 id(x) == 496103280 的值為\_\_\_\_\_\_\_\_\_\_\_。（False）
23. 已知 x = 3，那麼執行語句 x \*= 6 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（18）
24. 為了提高Python代碼運行速度和進行適當的保密，可以將Python程式檔編譯為副檔名\_\_\_\_\_\_\_\_\_\_的檔。（pyc）
25. 運算式“[3] in [1, 2, 3, 4]”的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
26. 清單物件的sort()方法用來對清單元素進行原地排序，該函數返回值為 。（None）
27. 假設清單物件aList的值為[3, 4, 5, 6, 7, 9, 11, 13, 15, 17]，那麼切片aList[3:7]得到的值是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[6, 7, 9, 11]）
28. 使用清單推導式生成包含10個數位5的清單，語句可以寫為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[5 for i in range(10)]）
29. 假設有列表a = ['name', 'age', 'sex']和b = ['Dong', 38, 'Male']，請使用一個語句將這兩個清單的內容轉換為字典，並且以清單a中的元素為“鍵”，以清單b中的元素為“值”，這個語句可以寫為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（c = dict(zip(a, b))）
30. 任意長度的Python清單、元組和字串中最後一個元素的下標為\_\_\_\_\_\_\_\_。（-1）
31. Python語句''.join(list('hello world!'))執行的結果是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'hello world!'）
32. 轉義字元r’\n’的含義是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（回車換行）
33. Python語句list(range(1,10,3))執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 4, 7]）
34. 運算式 list(range(5)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 1, 2, 3, 4]）
35. \_\_\_\_\_\_\_\_\_\_\_\_\_\_命令既可以刪除清單中的一個元素，也可以刪除整個列表。（del）
36. 已知a = [1, 2, 3]和b = [1, 2, 4]，那麼id(a[1])==id(b[1])的執行結果為\_\_\_\_\_\_\_\_\_\_\_。（True）
37. 運算式 int('123', 16) 的值為\_\_\_\_\_\_\_\_\_。（291）
38. 運算式 int('123', 8) 的值為\_\_\_\_\_\_\_\_\_。（83）
39. 運算式 int('123') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（123）
40. 運算式 int('101',2) 的值為\_\_\_\_\_\_\_\_\_\_。（5）
41. 運算式 abs(-3) 的值為\_\_\_\_\_\_\_\_\_\_\_。（3）
42. 切片操作list(range(6))[::2]執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 2, 4]）
43. 使用切片操作在清單物件x的開始處增加一個元素3的代碼為\_\_\_\_\_\_\_\_\_。（x[0:0] = [3]）
44. 語句sorted([1, 2, 3], reverse=True) == reversed([1, 2, 3])執行結果為\_\_\_\_\_\_\_\_\_。（False）
45. 運算式 'ab' in 'acbed' 的值為\_\_\_\_\_\_\_\_。（False）
46. Python 3.x語句 print(1, 2, 3, sep=':') 的輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_。（1:2:3）
47. 運算式 sorted([111, 2, 33], key=lambda x: len(str(x))) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[2, 33, 111]）
48. 假設n為整數，那麼運算式 n&1 == n%2 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
49. 運算式 int(4\*\*0.5) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（2）
50. 達式 sorted([111, 2, 33], key=lambda x: -len(str(x))) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[111, 33, 2]）
51. Python內置函數\_\_\_\_\_\_\_\_\_可以返回清單、元組、字典、集合、字串以及range物件中元素個數。（len()）
52. Python內置函數\_\_\_\_\_\_\_\_\_\_\_\_用來返回序列中的最大元素。（max()）
53. Python內置函數\_\_\_\_\_\_\_\_\_\_\_\_用來返回序列中的最小元素。（min()）
54. Python內置函數\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_用來返回數值型序列中所有元素之和。（sum()）
55. 已知清單物件x = ['11', '2', '3']，則運算式 max(x) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'3'）
56. 運算式 min(['11', '2', '3']) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'11'）
57. 已知清單物件x = ['11', '2', '3']，則運算式max(x, key=len) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'11'）
58. 語句 x = (3,) 執行後x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（(3,)）
59. 語句 x = (3) 執行後x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（3）
60. 已知x=3和y=5，執行語句 x, y = y, x 後x的值是\_\_\_\_。（5）
61. 可以使用內置函數\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_查看包含當前作用域內所有全域變數和值的字典。（globals()）
62. 可以使用內置函數\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_查看包含當前作用域內所有區域變數和值的字典。（locals()(）
63. 字典中多個元素之間使用\_\_\_\_\_\_\_\_\_\_\_\_分隔開，每個元素的“鍵”與“值”之間使用\_\_\_\_\_\_\_\_分隔開。（逗號、冒號）
64. 字典物件的\_\_\_\_\_\_\_\_\_\_\_方法可以獲取指定“鍵”對應的“值”，並且可以在指定“鍵”不存在的時候返回指定值，如果不指定則返回None。（get()）
65. 字典物件的\_\_\_\_\_\_\_\_\_\_\_\_\_方法返回字典中的“鍵-值對”列表。（items()）
66. 字典物件的\_\_\_\_\_\_\_\_\_\_\_\_方法返回字典的“鍵”清單。（keys()）
67. 字典物件的\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法返回字典的“值”清單。（values()）
68. 已知 x = {1:2}，那麼執行語句 x[2] = 3之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1: 2, 2: 3}）
69. 運算式 {1, 2, 3, 4} - {3, 4, 5, 6}的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2}）
70. 運算式set([1, 1, 2, 3])的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3}）
71. 關鍵字\_\_\_\_\_\_\_\_\_\_用於測試一個物件是否是一個可反覆運算物件的元素。（in）
72. 使用列表推導式得到100以內所有能被13整除的數的代碼可以寫作\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[i for i in range(100) if i%13==0]）
73. 運算式 3<5>2 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
74. 已知 x = {'a':'b', 'c':'d'}，那麼運算式 'a' in x 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
75. 已知 x = {'a':'b', 'c':'d'}，那麼運算式 'b' in x 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
76. 已知 x = {'a':'b', 'c':'d'}，那麼運算式 'b' in x.values() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
77. 運算式 1<2<3 的值為\_\_\_\_\_\_\_\_\_。（True）
78. 運算式 3 or 5 的值為\_\_\_\_\_\_\_\_（3）
79. 運算式 0 or 5 的值為\_\_\_\_\_\_\_\_\_（5）
80. 運算式 3 and 5 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（5）
81. 運算式 3 and not 5 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
82. 運算式 3 | 5 的值為\_\_\_\_\_\_\_\_\_\_。（7）
83. 運算式 3 & 6 的值為\_\_\_\_\_\_\_\_\_。（2）
84. 運算式 3 \*\* 2 的值為\_\_\_\_\_\_\_\_\_。（9）
85. 運算式 3 \* 2的值為\_\_\_\_\_\_\_\_\_\_\_。（6）
86. 已知 x = [3, 5, 7]，那麼運算式 x[10:]的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[]）
87. 已知 x = [3, 5, 7]，那麼執行語句 x[len(x):] = [1, 2]之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[3, 5, 7, 1, 2]）
88. 已知 x = [3, 7, 5]，那麼執行語句 x.sort(reverse=True)之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[7, 5, 3]）
89. 已知 x = [3, 7, 5]，那麼執行語句 x = x.sort(reverse=True)之後，x的值為\_\_\_\_\_\_\_\_\_。（None）
90. 已知 x = [1, 11, 111]，那麼執行語句 x.sort(key=lambda x: len(str(x)), reverse=True) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[111, 11, 1]）
91. 運算式 list(zip([1,2], [3,4])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[(1, 3), (2, 4)]）
92. 已知 x = [1, 2, 3, 2, 3]，執行語句 x.pop() 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 2]）
93. 運算式 list(map(list,zip(\*[[1, 2, 3], [4, 5, 6]]))) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 4], [2, 5], [3, 6]]）
94. 運算式 [x for x in [1,2,3,4,5] if x<3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2]）
95. 運算式 [index for index, value in enumerate([3,5,7,3,7]) if value == max([3,5,7,3,7])] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[2, 4]）
96. 已知 x = [3,5,3,7]，那麼運算式 [x.index(i) for i in x if i==3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 0]）
97. 已知列表 x = [1, 2]，那麼運算式 list(enumerate(x)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[(0, 1), (1, 2)]）
98. 已知 vec = [[1,2], [3,4]]，則運算式 [col for row in vec for col in row] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 4]）
99. 已知 vec = [[1,2], [3,4]]，則運算式 [[row[i] for row in vec] for i in range(len(vec[0]))] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 3], [2, 4]]）
100. 已知 x = list(range(10))，則運算式 x[-4:] 的值為\_\_\_\_\_\_\_\_\_\_。（[6, 7, 8, 9]）
101. 已知 path = r'c:\test.html'，那麼運算式 path[:-4]+'htm' 的值為\_\_\_\_\_\_\_\_\_\_。（'c:\\test.htm'）
102. 已知 x = [3, 5, 7]，那麼執行語句 x[1:] = [2]之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[3, 2]）
103. 已知 x = [3, 5, 7]，那麼執行語句 x[:3] = [2]之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[ 2]）
104. 已知x為非空列表，那麼執行語句y = x[:]之後，id(x[0]) == id(y[0])的值為\_\_\_\_\_\_\_\_\_\_。（True）
105. 已知 x = [1, 2, 3, 2, 3]，執行語句 x.remove(2) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 3, 2, 3]）
106. 運算式 3<<2 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（12）
107. 運算式 65 >> 1 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（32）
108. 運算式 chr(ord('a')^32) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'A'）
109. 運算式 chr(ord('a')-32) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'A'）
110. 運算式 abs(3+4j) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（5.0）
111. 運算式 callable(int) 的值為\_\_\_\_\_\_\_\_\_\_\_。（True）
112. 運算式 list(str([1,2,3])) == [1,2,3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
113. 運算式 str([1, 2, 3]) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'[1, 2, 3]'）
114. 運算式 str((1, 2, 3)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'(1, 2, 3)'）
115. Python中用於表示邏輯與、邏輯或、邏輯非運算的關鍵字分別是\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_。（and、or、not）
116. Python 3.x語句 for i in range(3):print(i, end=',') 的輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0,1,2,）
117. Python 3.x語句 print(1, 2, 3, sep=',') 的輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1,2,3）
118. 對於帶有else子句的for迴圈和while迴圈，當迴圈因迴圈條件不成立而自然結束時\_\_\_\_\_\_\_\_(會？不會？)執行else中的代碼。（會）
119. 在迴圈語句中，\_\_\_\_\_\_\_\_\_\_語句的作用是提前結束本層迴圈。（break）
120. 在迴圈語句中，\_\_\_\_\_\_\_語句的作用是提前進入下一次迴圈。（continue）
121. 運算式 sum(range(1, 10, 2)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（25）
122. 運算式 sum(range(1, 10)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（45）
123. 運算式 '%c'%65 的值為\_\_\_\_\_\_\_\_\_。（'A'）
124. 運算式 '%s'%65 的值為\_\_\_\_\_\_\_\_\_\_。（'65'）
125. 運算式 '%d,%c' % (65, 65) 的值為\_\_\_\_\_\_\_\_。（'65,A'）
126. 運算式 'The first:{1}, the second is {0}'.format(65,97) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'The first:97, the second is 65'）
127. 運算式 '{0:#d},{0:#x},{0:#o}'.format(65) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（'65,0x41,0o101'）
128. 運算式 isinstance('abcdefg', str) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
129. 運算式 isinstance('abcdefg', object) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
130. 運算式 isinstance(3, object) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
131. 運算式 'abcabcabc'.rindex('abc') 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（6）
132. 運算式 ':'.join('abcdefg'.split('cd')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'ab:efg'）
133. 運算式 'Hello world. I like Python.'.rfind('python') 的值為\_\_\_\_\_\_\_\_。（-1）
134. 運算式 'abcabcabc'.count('abc') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（3）
135. 運算式 'apple.peach,banana,pear'.find('p') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1）
136. 運算式 'apple.peach,banana,pear'.find('ppp') 的值為\_\_\_\_\_\_\_\_。（-1）
137. 運算式 'abcdefg'.split('d') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['abc', 'efg']）
138. 運算式 ':'.join('1,2,3,4,5'.split(',')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'1:2:3:4:5'）
139. 運算式 ','.join('a b ccc\n\n\nddd '.split()) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'a,b,ccc,ddd'）
140. 運算式 'Hello world'.upper() 的值為\_\_\_\_\_\_\_\_\_\_\_。（'HELLO WORLD'）
141. 運算式 'Hello world'.lower() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（'hello world'）
142. 運算式 'Hello world'.lower().upper() 的值為\_\_\_\_\_\_\_\_\_\_\_。（'HELLO WORLD'）
143. 運算式 'Hello world'.swapcase().swapcase() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'Hello world'）
144. 運算式 r'c:\windows\notepad.exe'.endswith('.exe') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
145. 運算式 r'c:\windows\notepad.exe'.endswith(('.jpg', '.exe')) 的值為\_\_\_\_\_\_\_。（True）
146. 運算式 'C:\\Windows\\notepad.exe'.startswith('C:') 的值為\_\_\_\_\_\_\_\_\_。（True）
147. 運算式 len('Hello world!'.ljust(20)) 的值為\_\_\_\_\_\_\_\_\_。（20）
148. 運算式 len('abcdefg'.ljust(3)) 的值為\_\_\_\_\_\_\_\_\_。（7）
149. 運算式 len([i for i in range(10)]) 的值為\_\_\_\_\_\_\_\_\_\_。（10）
150. 運算式 len(range(1,10)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（9）
151. 運算式 range(10)[-1] 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（9）
152. 運算式 range(10,20)[4] 的值為\_\_\_\_\_\_\_\_\_\_。（14）
153. 運算式 round(3.4) 的值為\_\_\_\_\_\_\_\_\_\_\_。（3）
154. 運算式 round(3.7) 的值為\_\_\_\_\_\_\_\_\_。（4）
155. 運算式 'a' + 'b' 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（'ab'）
156. 已知 x = '123' 和 y = '456'，那麼運算式 x + y 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'123456'）
157. 運算式 'a'.join('abc'.partition('a')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'aaabc'）
158. 運算式 re.split('\.+', 'alpha.beta...gamma..delta') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['alpha', 'beta', 'gamma', 'delta']）
159. 已知 x = 'a234b123c'，並且re模組已導入，則運算式 re.split('\d+', x) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['a', 'b', 'c']）
160. 運算式 ''.join('asdssfff'.split('sd')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（'assfff'）
161. 運算式 ''.join(re.split('[sd]','asdssfff')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'afff'）
162. 假設re模組已導入，那麼運算式 re.findall('(\d)\\1+', '33abcd112') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['3', '1']）
163. 語句 print(re.match('abc', 'defg')) 輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_\_。（None）
164. 運算式 'Hello world!'[-4] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'r'）
165. 運算式 'Hello world!'[-4:] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'rld!'）
166. 運算式 'test.py'.endswith(('.py', '.pyw')) 的值為\_\_\_\_\_\_\_\_\_\_。（True）
167. 已知 x = (3)， 那麼運算式 x \* 3 的值為\_\_\_\_\_\_\_\_\_\_。（9）
168. 已知 x = (3,)，那麼運算式 x \* 3 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（(3, 3, 3)）
169. 運算式 len('abc'.ljust(20)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（20）
170. 代碼 print(re.match('^[a-zA-Z]+$','abcDEFG000')) 的輸出結果為\_\_\_\_\_\_\_\_\_。（None）
171. 當在字串前加上小寫字母\_\_\_\_\_或大寫字母\_\_\_\_\_表示原始字串，不對其中的任何字元進行轉義。（r、R）
172. 在設計規則運算式時，字元\_\_\_\_\_\_\_緊隨任何其他限定詞(\*、+、?、{n}、{n,}、{n,m})之後時，匹配模式是“非貪心的”，匹配搜索到的、盡可能短的字串。（?）
173. 假設規則運算式模組re已導入，那麼運算式 re.sub('\d+', '1', 'a12345bbbb67c890d0e') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'a1bbbb1c1d1e'）
174. 假設清單物件x = [1, 1, 1]，那麼運算式id(x[0]) == id(x[2])的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
175. 已知列表 x = list(range(10))，那麼執行語句 del x[::2]之後，x的值為\_\_\_\_\_\_\_\_\_。（[1, 3, 5, 7, 9]）
176. 已知列表 x = [1, 2, 3, 4]，那麼執行語句 del x[1] 之後x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 3, 4]）
177. 運算式 [1] \* 2 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 1]）
178. 運算式 [1, 2] \* 2 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 1, 2]）
179. 已知列表 x = [1, 2, 3]，那麼執行語句 x.insert(1, 4) 只有，x的值為\_\_\_\_\_\_\_\_\_\_\_。（[1, 4, 2, 3]）
180. 已知列表 x = [1, 2, 3]，那麼執行語句 x.insert(0, 4) 只有，x的值為\_\_\_\_\_\_\_\_\_\_\_。（[4, 1, 2, 3]）
181. 已知列表 x = [1, 2, 3]，那麼執行語句 x.pop(0) 之後，x的值為\_\_\_\_\_\_\_\_\_。（[2, 3]）
182. 已知 x = [[1]] \* 3，那麼執行語句 x[0][0] = 5之後，變數x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[5], [5], [5]]）
183. 運算式 list(map(lambda x: x+5, [1, 2, 3, 4, 5])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[6, 7, 8, 9, 10]）
184. 運算式 {1, 2, 3, 4, 5} ^ {4, 5, 6, 7} 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 6, 7}）
185. 運算式 5 if 5>6 else (6 if 3>2 else 5) 的值為\_\_\_\_\_\_\_\_\_。（6）
186. 已知 x = [1, 2, 3]，那麼執行語句 x[len(x)-1:] = [4, 5, 6]之後，變數x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 4, 5, 6]）
187. 運算式 len(range(1, 10)) 的值為\_\_\_\_\_\_\_\_\_\_\_。（9）
188. 運算式 len('中國'.encode('utf-8')) 的值為\_\_\_\_\_\_\_\_\_\_\_。（6）
189. 運算式 len('中國'.encode('gbk')) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（4）
190. 運算式 chr(ord('A')+2) 的值為\_\_\_\_\_\_\_\_\_\_。（'C'）
191. 已知x是一個清單物件，那麼執行語句 y = x[:] 之後運算式 id(x) == id(y) 的值為\_\_\_\_\_\_\_\_\_\_\_。（False）
192. 運算式 sorted([13, 1, 237, 89, 100], key=lambda x: len(str(x))) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 13, 89, 237, 100]）
193. Python中定義函數的關鍵字是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（def）
194. 在函數內部可以通過關鍵字\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_來定義全域變數。（global）
195. 如果函數中沒有return語句或者return語句不帶任何返回值，那麼該函數的返回值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（None）
196. 運算式 sum(range(10)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（45）
197. 運算式 sum(range(1, 10, 2)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（25）
198. 運算式 'abcab'.replace('a','yy') 的值為\_\_\_\_\_\_\_\_\_\_\_。（'yybcyyb'）
199. 已知 table = ''.maketrans('abcw', 'xyzc')，那麼運算式 'Hellow world'.translate(table) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'Helloc corld'）
200. 運算式 'hello world, hellow every one'.replace('hello', 'hi') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'hi world, hiw every one'）
201. 已知字串 x = 'hello world'，那麼執行語句 x.replace('hello', 'hi') 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（'hello world'）
202. 規則運算式元字元\_\_\_\_\_\_\_\_用來表示該符號前面的字元或子模式1次或多次出現。（+）
203. 已知 x = 'a b c d'，那麼運算式 ','.join(x.split()) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'a,b,c,d'）
204. 規則運算式元字元\_\_\_\_\_\_\_\_用來表示該符號前面的字元或子模式0次或多次出現。（\*）
205. 運算式 'abcab'.strip('ab') 的值為\_\_\_\_\_\_\_\_\_\_。（'c'）
206. 運算式 [str(i) for i in range(3)] 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（['0', '1', '2']）
207. 運算式 'abc.txt'.endswith(('.txt', '.doc', '.jpg')) 的值為\_\_\_\_\_\_\_\_\_\_\_。（True）
208. 運算式 list(filter(None, [0,1,2,3,0,0])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
209. 運算式 list(filter(lambda x:x>2, [0,1,2,3,0,0])) 的值為\_\_\_\_\_\_\_\_\_。（[3]）
210. 運算式 list(range(50, 60, 3)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[50, 53, 56, 59]）
211. 運算式 list(filter(lambda x: x%2==0, range(10))) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 2, 4, 6, 8]）
212. 運算式 list(filter(lambda x: len(x)>3, ['a', 'b', 'abcd'])) 的值為\_\_\_\_\_\_\_\_\_\_\_。（['abcd']）
213. Python使用\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_關鍵字來定義類。（class）
214. 運算式 isinstance('abc', str) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
215. 運算式 isinstance('abc', int) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
216. 運算式 isinstance(4j, (int, float, complex)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
217. 運算式 isinstance('4', (int, float, complex)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
218. 運算式 type(3) in (int, float, complex) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
219. 運算式 type(3.0) in (int, float, complex) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
220. 運算式 type(3+4j) in (int, float, complex) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
221. 運算式 type('3') in (int, float, complex) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（False）
222. 運算式 type(3) == int 的值為\_\_\_\_\_\_\_\_\_\_。（True）
223. 代碼 print(1,2,3,sep=':') 的執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1:2:3）
224. 代碼 for i in range(3):print(i, end=',') 的執行結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0,1,2,）
225. 運算式 eval('''\_\_import\_\_('math').sqrt(9)''') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（3.0）
226. 運算式 eval('''\_\_import\_\_('math').sqrt(3\*\*2+4\*\*2)''') 的值為\_\_\_\_\_\_\_\_\_。（5.0）
227. 運算式 eval('3+5') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（8）
228. 運算式 eval('[1, 2, 3]') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
229. 假設math標準庫已導入，那麼運算式 eval('math.sqrt(4)') 的值為\_\_\_\_\_\_\_\_\_。（2.0）
230. 已知x為非空列表，那麼運算式 random.choice(x) in x 的值為\_\_\_\_\_\_\_\_\_\_\_。（True）
231. 運算式 'abc10'.isalnum() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
232. 運算式 'abc10'.isalpha() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
233. 運算式 'abc10'.isdigit() 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
234. 運算式 [1,2,3].count(4) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0）
235. Python標準庫random中的\_\_\_\_\_\_\_\_\_\_\_方法作用是從序列中隨機選擇1個元素。（choice()）
236. 運算式 'C:\\windows\\notepad.exe'.endswith('.exe') 的值為\_\_\_\_\_\_\_\_。（True）
237. Python標準庫random中的sample(seq, k)方法作用是從序列中選擇\_\_\_\_\_\_\_\_(重複？不重複？)的k個元素。（不重複）
238. random模組中\_\_\_\_\_\_\_\_\_\_\_\_\_方法的作用是將清單中的元素隨機亂序。（shuffle()）
239. Python關鍵字elif表示\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_兩個單詞的縮寫。（else、if）
240. 執行代碼 x, y, z = sorted([1, 3, 2]) 之後，變數y的值為\_\_\_\_\_\_。（2）
241. 已知 x = {1:2, 2:3}，那麼運算式 x.get(3, 4) 的值為\_\_\_\_。（4）
242. 已知 x = {1:2, 2:3}，那麼運算式 x.get(2, 4) 的值為\_\_\_\_\_\_。（3）
243. 運算式 {1, 2, 3} | {3, 4, 5} 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 4, 5}）
244. 運算式 {1, 2, 3} | {2, 3, 4} 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 4}）
245. 運算式 {1, 2, 3} & {3, 4, 5} 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{3}）
246. 運算式 {1, 2, 3} & {2, 3, 4} 的值為\_\_\_\_\_\_\_\_\_\_\_。（{2, 3}）
247. 運算式 {1, 2, 3} - {3, 4, 5} 的值為\_\_\_\_\_\_\_\_\_。（{1, 2}）
248. 運算式 {1, 2, 3} < {3, 4, 5} 的值為\_\_\_\_\_\_\_\_\_。（False）
249. 運算式 {1, 2, 3} < {1, 2, 4} 的值為\_\_\_\_\_\_\_\_\_\_\_。（False）
250. 運算式 '%s'%[1,2,3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（'[1, 2, 3]'）
251. 在Python定義類時，與運算子“\*\*”對應的特殊方法名為 \_\_\_\_\_\_\_\_\_\_\_\_\_。（\_\_pow\_\_()）
252. 在Python中定義類時，與運算子“//”對應的特殊方法名為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（\_\_floordiv\_\_()）
253. 對檔進行寫入操作之後，\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法用來在不關閉檔物件的情況下將緩衝區內容寫入檔。（flush()）
254. Python內置函數\_\_\_\_\_\_\_\_\_\_\_\_\_用來打開或創建檔並返回檔物件。（open()）
255. 使用上下文管理關鍵字\_\_\_\_\_\_\_\_\_\_\_\_\_\_可以自動管理檔物件，不論何種原因結束該關鍵字中的語句塊，都能保證檔被正確關閉。（with）
256. Python標準庫os中用來列出指定資料夾中的檔和子資料夾清單的方式是\_\_\_\_\_\_\_\_\_\_\_\_。（listdir()）
257. Python標準庫os.path中用來判斷指定檔是否存在的方法是\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（exists()）
258. Python標準庫os.path中用來判斷指定路徑是否為檔的方法是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（isfile()）
259. Python標準庫os.path中用來判斷指定路徑是否為資料夾的方法是\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（isdir()）
260. Python標準庫os.path中用來分割指定路徑中的檔副檔名的方法是\_\_\_\_\_\_\_\_\_\_。（splitext()）
261. Python內建異常類的基類是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（BaseException）
262. Python擴展庫\_\_\_\_\_\_\_\_\_\_\_\_\_支援Excel 2007或更高版本檔的讀寫操作。（openpyxl）
263. Python標準庫\_\_\_\_\_\_\_\_\_\_\_\_中提供了計算MD5摘要的方法md5()。（hashlib）
264. 運算式 len('SDIBT') 的值為\_\_\_\_\_\_\_\_\_\_。（5）
265. 運算式 'Hello world!'.count('l') 的值為\_\_\_\_\_\_\_\_\_\_\_。（3）
266. 運算式 (1, 2, 3)+(4, 5) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（(1, 2, 3, 4, 5)）
267. 運算式 dict(zip([1, 2], [3, 4])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1: 3, 2: 4}）
268. 已知 x = 'abcdefg'，則運算式 x[3:] + x[:3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'defgabc'）
269. 一直 g = lambda x, y=3, z=5: x\*y\*z，則語句 print(g(1)) 的輸出結果為\_\_\_\_\_\_\_。（15）
270. 運算式 list(map(lambda x: len(x), ['a', 'bb', 'ccc'])) 的值為\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
271. 語句 x, y, z = [1, 2, 3] 執行後，變數y的值為\_\_\_\_\_\_\_\_\_\_。（2）
272. Python標準庫\_\_\_\_\_\_\_\_\_\_對Socket進行了二次封裝，支援Socket介面的訪問，大幅度簡化了網路程式的開發。（socket）
273. Python擴展庫\_\_\_\_\_\_\_\_\_\_\_\_\_\_中封裝了Windows底層幾乎所有API函數。（pywin32）
274. 執行緒物件的\_\_\_\_\_\_\_\_\_\_\_方法用來阻塞當前執行緒，指定執行緒運行結束或超時後繼續運行當前執行緒。（join()）
275. Python用來訪問和操作內置資料庫SQLite的標準庫是\_\_\_\_\_\_\_\_\_\_\_\_\_。（sqlite3）
276. 用於刪除資料庫表test中所有name欄位值為’10001’的記錄的SQL語句為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（delete from test where name=’10001’）
277. Python擴展庫\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_完美封裝了圖形庫OpenGL的功能。（pyopengl）
278. Python擴展庫\_\_\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_\_\_提供了影像處理功能。（PIL、pillow）
279. 已知 x = [[1,3,3], [2,3,1]]，那麼運算式 sorted(x, key=lambda item:item[0]+item[2]) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[2, 3, 1], [1, 3, 3]]）
280. 已知 x = [[1,3,3], [2,3,1]]，那麼運算式 sorted(x, key=lambda item:(item[1],item[2])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[2, 3, 1], [1, 3, 3]]）
281. 已知 x = [[1,3,3], [2,3,1]]，那麼運算式 sorted(x, key=lambda item:(item[1], -item[2])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 3, 3], [2, 3, 1]]）
282. 已知 x = {1, 2, 3}，那麼執行語句 x.add(3) 之後，x的值為\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3}）
283. 已知 x = {1:1}，那麼執行語句 x[2] = 2之後，len(x)的值為\_\_\_\_\_\_\_\_\_\_\_\_。（2）
284. 已知 x = {1:1, 2:2}，那麼執行語句 x[2] = 4之後，len(x)的值為\_\_\_\_\_\_\_\_\_\_\_\_。（2）
285. 假設已從標準庫functools導入reduce()函數，那麼運算式 reduce(lambda x, y: x-y, [1, 2, 3]) 的值為\_\_\_\_。（-4）
286. 假設已從標準庫functools導入reduce()函數，那麼運算式 reduce(lambda x, y: x+y, [1, 2, 3]) 的值為\_\_\_\_。（6）
287. 已知有函式定義 def demo(\*p):return sum(p)，那麼運算式 demo(1, 2, 3) 的值為\_\_\_\_\_\_、運算式 demo(1, 2, 3, 4) 的值為\_\_\_\_\_\_\_\_\_。（6、10）
288. 已知列表 x = [1, 2]，那麼連續執行命令 y = x和 y.append(3) 之後，x的值為\_\_\_\_\_\_\_\_。（[1, 2, 3]）
289. 已知列表 x = [1, 2]，那麼連續執行命令 y = x[:] 和 y.append(3) 之後，x的值為\_\_\_\_\_\_\_\_。（[1, 2]）
290. 已知列表 x = [1, 2]，執行語句 y = x[:] 後，運算式 id(x) == id(y) 的值為\_\_\_\_\_\_\_。（False）
291. 已知列表 x = [1, 2]，執行語句 y = x 後，運算式 id(x) == id(y) 的值為\_\_\_\_\_\_\_。（True）
292. 已知列表 x = [1, 2]，執行語句 y = x 後，運算式 x is y 的值為\_\_\_\_\_\_\_。（True）
293. 已知列表 x = [1, 2]，執行語句 y = x[:] 後，運算式 x is not y 的值為\_\_\_\_\_\_\_。（True）
294. 運算式 sorted(random.sample(range(5), 5)) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 1, 2, 3, 4]）
295. 運算式 [i for i in range(10) if i>8] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[9]）
296. 已知有列表 x = [[1, 2, 3], [4, 5, 6]]，那麼運算式 [[row[i] for row in x] for i in range(len(x[0]))] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 4], [2, 5], [3, 6]]）
297. 執行語句 x,y,z = map(str, range(3)) 之後，變數y的值為\_\_\_\_\_\_\_\_\_\_\_。（'1'）
298. 已知列表 x = [1, 2]，那麼執行語句 x.extend([3]) 之後， x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
299. 已知列表 x = [1, 2]，那麼執行語句 x.append([3]) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, [3]]）
300. 運算式 'aaasdf'.lstrip('as') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'df'）
301. 運算式 'aaasdf'.lstrip('af') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'sdf'）
302. 運算式 'aaasdf'.strip('af') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'sd'）
303. 運算式 'aaasdf'.rstrip('af') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'aaasd'）
304. 已知 f = lambda x: x+5，那麼運算式 f(3) 的值為\_\_\_\_\_\_\_\_。（8）
305. 運算式 print(0b10101) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（21）
306. 運算式 '\x41' == 'A' 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
307. 已知 x = [1, 2, 3, 4, 5]，那麼執行語句 del x[:3] 之後，x的值為\_\_\_\_\_\_\_\_\_\_。（[4, 5]）
308. 運算式 sorted(['abc', 'acd', 'ade'], key=lambda x:(x[0],x[2])) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（['abc', 'acd', 'ade']）
309. 已知 x = range(1,4) 和 y = range(4,7)，那麼運算式 sum([i\*j for i,j in zip(x,y)]) 的值為\_\_\_\_\_\_\_\_\_\_。（32）
310. 運算式 [5 for i in range(3)] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[5, 5, 5]）
311. 運算式 {1, 2, 3} == {1, 3, 2} 的值為\_\_\_\_\_\_\_\_\_\_\_。（True）
312. 運算式 [1, 2, 3] == [1, 3, 2] 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（False）
313. 已知 x = [1, 2, 1]，那麼運算式 id(x[0]) == id(x[2]) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
314. 運算式 3 not in [1, 2, 3]的值為\_\_\_\_\_\_\_\_\_\_。（False）
315. 已知 x = [1, 2]，那麼執行語句 x[0:0] = [3, 3]之後，x的值為\_\_\_\_\_\_\_\_\_\_\_。（[3, 3, 1, 2]）
316. 已知 x = [1, 2]，那麼執行語句 x[0:1] = [3, 3]之後，x的值為\_\_\_\_\_\_\_\_\_\_\_。（[3, 3, 2]）
317. 已知 x = [1, 2, 3, 4, 5]，那麼執行語句 del x[1:3] 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 4, 5]）
318. 已知 x = [[1, 2, 3,], [4, 5, 6]]，那麼運算式 sum([i\*j for i,j in zip(\*x)]) 的值為\_\_\_\_\_\_。（32）
319. 已知列表 x = [1, 2, 3] 和 y = [4, 5, 6]，那麼運算式 [(i,j) for i, j in zip(x,y) if i==3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[(3, 6)]）
320. 已知列表 x = [1.0, 2.0, 3.0]，那麼運算式 sum(x)/len(x) 的值為\_\_\_\_\_\_\_\_\_\_\_。（2.0）
321. 運算式 'abc' in ('abcdefg') 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
322. 運算式 'abc' in ['abcdefg'] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
323. 已知 x = {1:2, 2:3, 3:4}，那麼運算式 sum(x) 的值為\_\_\_\_\_\_\_\_\_\_\_。（6）
324. 已知 x = {1:2, 2:3, 3:4}，那麼運算式 sum(x.values()) 的值為\_\_\_\_\_\_\_\_\_。（9）
325. 已知 x = [3, 2, 3, 3, 4]，那麼運算式 [index for index, value in enumerate(x) if value==3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 2, 3]）
326. 運算式 1234%1000//100 的值為\_\_\_\_\_\_\_\_\_\_\_。（2）
327. 規則運算式模組re的\_\_\_\_\_\_\_\_\_\_方法用來編譯規則運算式物件。（compile()）
328. 規則運算式模組re的\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法用來在字串開始處進行指定模式的匹配。（match()）
329. 規則運算式模組re的\_\_\_\_\_\_\_\_\_\_\_\_\_\_方法用來在整個字串中進行指定模式的匹配。（search()）
330. 運算式 re.search(r'\w\*?(?P<f>\b\w+\b)\s+(?P=f)\w\*?', 'Beautiful is is better than ugly.').group(0) 的值為\_\_\_\_\_\_\_\_\_\_\_。（'is is'）
331. 已知 g = lambda x, y=3, z=5: x+y+z，那麼運算式 g(2) 的值為\_\_\_\_\_\_\_\_。（10）
332. 假設有Python程式檔abc.py，其中只有一條語句print(\_\_name\_\_)，那麼直接運行該程式時得到的結果為\_\_\_\_\_\_\_\_\_\_\_\_\_。（\_\_main\_\_）
333. 運算式 3 in {1, 2, 3} 的值為\_\_\_\_\_\_\_\_\_。（True）
334. 運算式 'ac' in 'abce' 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
335. 運算式 not 3 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
336. 運算式 3 // 5 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0）
337. 運算式 [1, 2] + [3] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
338. 運算式 (1,) + (2,) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（(1, 2)）
339. 運算式 (1) + (2) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（3）
340. 已知 x, y = map(int, ['1', '2'])，那麼運算式 x + y 的值為\_\_\_\_\_\_\_。（3）
341. 已知列表 x = list(range(5))，那麼執行語句 x.remove(3) 之後，運算式 x.index(4) 的值為\_\_\_\_\_\_\_\_。（3）
342. 已知列表 x = [1, 3, 2]，那麼執行語句 x.reverse() 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（[2, 3, 1]）
343. 已知列表 x = [1, 3, 2]，那麼執行語句 x = x.reverse() 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（None）
344. 已知x為非空列表，那麼運算式 x.reverse() == list(reversed(x)) 的值為\_\_\_\_\_\_\_\_\_\_。（False）
345. 已知x為非空列表，那麼運算式 x.sort() == sorted(x) 的值為\_\_\_\_\_\_\_\_\_\_。（False）
346. 已知列表 x = [1, 3, 2]，那麼執行語句 y = list(reversed(x)) 之後，x的值為\_\_\_\_\_\_\_\_\_\_。（[1, 3, 2]）
347. 已知列表 x = [1, 3, 2]，那麼執行語句 y = list(reversed(x)) 之後，y的值為\_\_\_\_\_\_\_\_\_\_。（[2, 3, 1]）
348. 運算式 'Beautiful is better than ugly.'.startswith('Be', 5) 的值為\_\_\_\_\_\_\_\_\_。（False）
349. 已知清單x中包含超過5個以上的元素，那麼運算式 x == x[:5]+x[5:] 的值為\_\_\_\_\_\_\_\_。（True）
350. 已知字典 x = {i:str(i+3) for i in range(3)}，那麼運算式 sum(x) 的值為\_\_\_\_\_\_。（3）
351. 已知字典 x = {i:str(i+3) for i in range(3)}，那麼運算式 ''.join(x.values()) 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（'345'）
352. 已知字典 x = {i:str(i+3) for i in range(3)}，那麼運算式 sum(item[0] for item in x.items()) 的值為\_\_\_\_\_\_\_\_\_\_\_。（3）
353. 已知字典 x = {i:str(i+3) for i in range(3)}，那麼運算式 ''.join([item[1] for item in x.items()]) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（'345'）
354. 已知列表 x = [1, 3, 2]，那麼運算式 [value for index, value in enumerate(x) if index==2] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[2]）
355. 已知列表 x = [1, 3, 2]，那麼執行語句 a, b, c = sorted(x) 之後，b的值為\_\_\_\_。（2）
356. 已知列表 x = [1, 3, 2]，那麼執行語句 a, b, c = map(str,sorted(x)) 之後，c的值為\_\_\_\_\_\_。（'3'）
357. 運算式 set([1,2,3]) == {1, 2, 3} 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
358. 運算式 set([1,2, 2,3]) == {1, 2, 3} 的值為\_\_\_\_\_\_\_\_\_\_\_\_。（True）
359. 運算式 '%c'%65 == str(65) 的值為\_\_\_\_\_\_\_\_\_\_\_。（False）
360. 運算式 '%s'%65 == str(65) 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
361. 運算式 chr(ord('b')^32) 的值為\_\_\_\_\_\_\_\_\_。（'B'）
362. 運算式 'abc' in 'abdcefg' 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
363. 已知函式定義 def func(\*p):return sum(p)，那麼運算式 func(1,2,3) 的值為\_\_\_\_\_\_。（6）
364. 已知函式定義 def func(\*p):return sum(p)，那麼運算式 func(1,2,3, 4) 的值為\_\_\_\_\_\_。（10）
365. 已知函式定義 def func(\*\*p):return sum(p.values())，那麼運算式 func(x=1, y=2, z=3) 的值為\_\_\_\_\_\_\_\_。（6）
366. 已知函式定義 def func(\*\*p):return ''.join(sorted(p))，那麼運算式 func(x=1, y=2, z=3)的值為\_\_\_\_\_\_\_\_\_\_。（'xyz'）
367. 已知x為整數變數，那麼運算式 int(hex(x), 16) == x 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
368. 已知 f = lambda x: 5，那麼運算式 f(3)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（5）
369. 已知 x, y = 3, 5，那麼執行x, y = y, x 之後，x的值為\_\_\_\_\_\_\_\_\_\_。（5）
370. 已知 x = 'abcd' 和 y = 'abcde'，那麼運算式 [i==j for i,j in zip(x,y)] 的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[True, True, True, True]）
371. 運算式16\*\*0.5的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（4.0）
372. 運算式type({3})的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（set）
373. 運算式isinstance('Hello world', str)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
374. 已知x = list(range(20))，那麼運算式x[-1]的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（19）
375. 已知x = 3+4j和y = 5+6j，那麼運算式x+y的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（8+10j）
376. 已知x = [3]，那麼執行x += [5]之後x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[3, 5]）
377. 已知x = [3, 3, 4]，那麼運算式id(x[0])==id(x[1])的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
378. 運算式int('11', 2)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（3）
379. 運算式int('11', 8)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（9）
380. 運算式int(bin(54321), 2)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（54321）
381. 運算式chr(ord('A')+1)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'B'）
382. 運算式int(str(34)) == 34的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
383. 運算式list(str([3, 4])) == [3, 4]的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
384. 運算式{1, 2, 3, 4, 5, 6} ^ {5, 6, 7, 8}的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 4, 7, 8}）
385. 運算式15 // 4的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（3）
386. 運算式sorted({'a':3, 'b':9, 'c':78})的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['a', 'b', 'c']）
387. 運算式sorted({'a':3, 'b':9, 'c':78}.values())的值為\_\_\_\_\_\_\_\_\_\_\_\_\_。（[3, 9, 78]）
388. 已知x = [3, 2, 4, 1]，那麼執行語句x = x.sort()之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_。（None）
389. 運算式list(filter(lambda x: x>5, range(10)))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[6, 7, 8, 9]）
390. 已知x = list(range(20))，那麼語句print(x[100:200])的輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[]）
391. 已知x = list(range(20))，那麼執行語句x[:18] = []後列表x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[18, 19]）
392. 已知x = [1, 2, 3]，那麼連續執行y = x[:]和y.append(4)這兩條語句之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
393. 已知x = [1, 2, 3]，那麼連續執行y = x和y.append(4)這兩條語句之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 4]）
394. 已知x = [1, 2, 3]，那麼連續執行y = [1, 2, 3]和y.append(4)這兩條語句之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）
395. 已知x = [[]] \* 3，那麼執行語句x[0].append(1)之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1], [1], [1]]）
396. 已知x = [[] for i in range(3)]，那麼執行語句x[0].append(1)之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1], [], []]）
397. 已知x = ([1], [2])，那麼執行語句x[0].append(3)後x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（([1, 3], [2])）
398. 已知x = {1:1, 2:2}，那麼執行語句x.update({2:3, 3:3})之後，運算式sorted(x.items())的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[(1, 1), (2, 3), (3, 3)]）
399. 已知x = {1:1, 2:2}，那麼執行語句x[3] = 3之後，運算式sorted(x.items())的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[(1, 1), (2, 2), (3, 3)]）
400. 運算式type({}) == dict的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
401. 運算式type({}) == set的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
402. 已知x = [1, 2, 3]，那麼運算式not (set(x\*100)-set(x))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）
403. 已知x = [1, 2, 3]，那麼運算式not (set(x\*100)&set(x))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（False）
404. 運算式{'x': 1, \*\*{'y': 2}}的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{'x': 1, 'y': 2}）
405. 運算式{\*range(4), 4, \*(5, 6, 7)}的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{0, 1, 2, 3, 4, 5, 6, 7}）
406. 在Python中，不論類的名字是什麼，構造方法的名字都是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（\_\_init\_\_）
407. 如果在設計一個類時實現了\_\_contains\_\_ ()方法，那麼該類的物件會自動支援\_\_\_\_\_\_\_\_\_\_\_\_\_運算子。（in）
408. 已知函式定義def demo(x, y, op):return eval(str(x)+op+str(y))，那麼運算式demo(3, 5, '+')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（8）
409. 已知函式定義def demo(x, y, op):return eval(str(x)+op+str(y))，那麼運算式demo(3, 5, '\*')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（15）
410. 已知函式定義def demo(x, y, op):return eval(str(x)+op+str(y))，那麼運算式demo(3, 5, '-')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（-2）
411. 字串編碼格式UTF8使用\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_個位元組表示一個漢字。（3）
412. 字串編碼格式GBK使用\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_個位元組表示一個漢字。（2）
413. 已知字串編碼格式utf8使用3個位元組表示一個漢字、1個位元組表示英語字母，那麼運算式len('abc你好')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（5）
414. 已知字串編碼格式utf8使用3個位元組表示一個漢字、1個位元組表示英語字母，那麼運算式len('abc你好'.encode())的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（9）
415. 已知字串編碼格式gbk使用2個位元組表示一個漢字、1個位元組表示英語字母，那麼運算式len('abc你好'.encode('gbk'))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（7）
416. 已知ord('A')的值為65並且hex(65)的值為'0x41'，那麼運算式'\x41b'的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'Ab'）
417. 已知formatter = 'good {0}'.format，那麼運算式list(map(formatter, ['morning']))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['good morning']）
418. 已知x = 'hello world.'，那麼運算式x.find('x')和x.rfind('x')的值都為\_\_\_\_\_\_\_\_\_\_\_\_\_。（-1）
419. 運算式':'.join('hello world.'.split())的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'hello:world.'）
420. 運算式':'.join('a b c d'.split(maxsplit=2))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'a:b:c d'）
421. 已知x = 'hello world'，那麼運算式x.replace('l', 'g')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'heggo worgd'）
422. 假設已成功導入Python標準庫string，那麼運算式len(string.digits)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（10）
423. 運算式'aaaassddf'.strip('af')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'ssdd'）
424. 運算式len('aaaassddf'.strip('afds'))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0）
425. 運算式len('hello world'[100:])的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0）
426. 運算式chr(ord('a')^32^32)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'a'）
427. 運算式chr(ord('a')^32)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'A'）
428. 已知x = 'aa b ccc dddd'，那麼運算式''.join([v for i,v in enumerate(x[:-1]) if v==x[i+1]])的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'accddd'）
429. 已知當前資料夾中有純英文文字檔readme.txt，請填空完成功能把readme.txt檔中的所有內容複製到dst.txt中，with open('readme.txt') as src, open('dst.txt', \_\_\_\_\_\_\_\_\_\_\_\_) as dst:dst.write(src.read())。（'w'）
430. 假設規則運算式模組re已正確導入，那麼運算式''.join(re.findall('\d+', 'abcd1234'))的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'1234'）
431. 假設規則運算式模組re已正確導入，那麼運算式re.findall('\d+?', 'abcd1234')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（['1', '2', '3', '4']）
432. 假設規則運算式模組re已正確導入，那麼運算式re.sub('(.\s)\\1+', '\\1','a a a a a bb')的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（'a bb'）
433. Python標準庫\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_提供了對SQLite資料庫的訪問介面。（sqlite3）
434. 已知f = lambda n: len(bin(n)[bin(n).rfind('1')+1:])，那麼運算式f(6)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1）
435. 已知f = lambda n: len(bin(n)[bin(n).rfind('1')+1:])，那麼運算式f(7)的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（0）
436. 已知 x = [1,2,3,4,5]，那麼執行語句 x[::2] = range(3) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 2, 1, 4, 2]）
437. 已知 x = [1,2,3,4,5]，那麼執行語句 x[::2] = map(lambda y:y!=5,range(3)) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[True, 2, True, 4, True]）
438. 已知 x = [1,2,3,4,5]，那麼執行語句 x[1::2] = sorted(x[1::2], reverse=True) 之後，x的值為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 4, 3, 2, 5]）
439. 運算式 True\*3 的值為\_\_\_\_\_\_\_\_\_\_。（3）
440. 運算式 False+1 的值為\_\_\_\_\_\_\_\_\_\_。（1）
441. 判斷題
442. Python是一種跨平臺、開源、免費的高級動態程式設計語言。（對）
443. Python 3.x完全相容Python 2.x。（錯）
444. Python 3.x和Python 2.x唯一的區別就是：print在Python 2.x中是輸出語句，而在Python 3.x中是輸出函數。（錯）
445. 在Windows平臺上編寫的Python程式無法在Unix平臺運行。（錯）
446. 不可以在同一台電腦上安裝多個Python版本。（錯）
447. 已知 x = 3，那麼設定陳述式 x = 'abcedfg' 是無法正常執行的。（錯）
448. 繼承自threading.Thread類的派生類中不能有普通的成員方法。（錯）
449. 擴展庫os中的方法remove()可以刪除帶有唯讀屬性的檔。（錯）
450. 使用內置函數open()且以”w”模式打開的檔，檔指標默認指向檔案結尾。（錯）
451. 使用內置函數open()打開檔時，只要檔路徑正確就總是可以正確打開的。（錯）
452. Python變數使用前必須先聲明，並且一旦聲明就不能再當前作用域內改變其類型。（錯）
453. Python採用的是基於值得自動記憶體管理方式。（對）
454. 在任何時刻相同的值在記憶體中都只保留一份（錯）
455. Python不允許使用關鍵字作為變數名，允許使用內置函數名作為變數名，但這會改變函數名的含義。（對）
456. 在Python中可以使用if作為變數名。（錯）
457. 在Python 3.x中可以使用中文作為變數名。（對）
458. Python變數名必須以字母或底線開頭，並且區分字母大小寫。（對）
459. 加法運算子可以用來連接字串並生成新字串。（對）
460. 9999\*\*9999這樣的命令在Python中無法運行。（錯）
461. 3+4j不是合法的Python運算式。（錯）
462. 0o12f是合法的八進位數字。（錯）
463. Python 2.x和Python 3.x中input()函數的返回值都是字串。（錯）
464. pip命令也支援副檔名為.whl的檔直接安裝Python擴展庫。（對）
465. 只有Python擴展庫才需要導入以後才能使用其中的物件，Python標準庫不需要導入即可使用其中的所有物件和方法。（錯）
466. 在Python中0xad是合法的十六進位數字表示形式。（對）
467. 3+4j 是合法Python數位類型。（對）
468. 在Python中0oa1是合法的八進位數字表示形式。（錯）
469. Python使用縮進來體現代碼之間的邏輯關係。（對）
470. Python代碼的注釋只有一種方式，那就是使用#符號。（錯）
471. 調用函數時，在實參前面加一個型號\*表示序列解包。（對）
472. 放在一對三引號之間的任何內容將被認為是注釋。（錯）
473. Python支援使用字典的“鍵”作為下標來訪問字典中的值。（對）
474. 清單可以作為字典的“鍵”。（錯）
475. 元組可以作為字典的“鍵”。（對）
476. 字典的“鍵”必須是不可變的。（對）
477. 儘管可以使用import語句一次導入任意多個標準庫或擴展庫，但是仍建議每次只導入一個標準庫或擴展庫。（對）
478. 為了讓代碼更加緊湊，編寫Python程式時應儘量避免加入空格和空行。（錯）
479. 在Python 3.5中運算子+不僅可以實現數值的相加、字元串連接，還可以實現列表、元組的合併和集合的並集運算。（錯）
480. 已知x為非空列表，那麼運算式 sorted(x, reverse=True) == list(reversed(x)) 的值一定是True。（錯）
481. 已知x為非空列表，那麼x.sort(reverse=True)和x.reverse()的作用是等價的。（錯）
482. 生成器推導式比列表推導式具有更高的效率，推薦使用。（對）
483. Python集合中的元素不允許重複。（對）
484. Python集合可以包含相同的元素。（錯）
485. Python字典中的“鍵”不允許重複。（對）
486. Python字典中的“值”不允許重複。（錯）
487. Python集合中的元素可以是元組。（對）
488. Python集合中的元素可以是清單。（錯）
489. Python字典中的“鍵”可以是列表。（錯）
490. Python字典中的“鍵”可以是元組。（對）
491. Python清單中所有元素必須為相同類型的資料。（錯）
492. Python清單、元組、字串都屬於有序序列。（對）
493. 在Python 3.x中語句 print(\*[1,2,3]) 不能正確執行。（錯）
494. 已知A和B是兩個集合，並且運算式A<B的值為False，那麼運算式A>B的值一定為True。（錯）
495. 清單物件的append()方法屬於原地操作，用於在清單尾部追加一個元素。（對）
496. 對於清單而言，在尾部追加元素比在中間位置插入元素速度更快一些，尤其是對於包含大量元素的清單。（對）
497. 假設有非空列表x，那麼x.append(3)、x = x+[3]與x.insert(0,3)在執行時間上基本沒有太大區別。（錯）
498. 使用Python清單的方法insert()為清單插入元素時會改變清單中插入位置之後元素的索引。（對）
499. 假設x為清單物件，那麼x.pop()和x.pop(-1)的作用是一樣的。（對）
500. 使用del命令或者清單物件的remove()方法刪除清單中元素時會影響清單中部分元素的索引。（對）
501. 帶有else子句的迴圈如果因為執行了break語句而退出的話，則會執行else子句中的代碼。（錯）
502. 對於帶有else子句的迴圈語句，如果是因為迴圈條件運算式不成立而自然結束迴圈，則執行else子句中的代碼。（對）
503. 已知列表 x = [1, 2, 3]，那麼執行語句 x = 3 之後，變數x的位址不變。（錯）
504. 在UTF-8編碼中一個漢字需要佔用3個位元組。（對）
505. 在GBK和CP936編碼中一個漢字需要2個位元組。（對）
506. 如果僅僅是用於控制迴圈次數，那麼使用for i in range(20)和for i in range(20, 40)的作用是等價的。（對）
507. 使用清單物件的remove()方法可以刪除列表中首次出現的指定元素，如果列中不存在要刪除的指定元素則拋出異常。（對）
508. 元組是不可變的，不支援清單物件的inset()、remove()等方法，也不支援del命令刪除其中的元素，但可以使用del命令刪除整個元組物件。（對）
509. Python字典和集合屬於無序序列。（對）
510. 無法刪除集合中指定位置的元素，只能刪除特定值的元素。（對）
511. 元組的存取速度比列表要快一些，如果定義了一系列常量值，並且主要用途僅僅是對其進行遍歷二不需要進行任何修改，建議使用元組而不使用列表。（對）
512. 當以指定“鍵”為下標給字典物件賦值時，若該“鍵”存在則表示修改該“鍵”對應的“值”，若不存在則表示為字典物件添加一個新的“鍵-值對”。（對）
513. 假設x是含有5個元素的清單，那麼切片操作x[10:]是無法執行的，會拋出異常。（錯）
514. 只能對列表進行切片操作，不能對元組和字串進行切片操作。（錯）
515. 只能通過切片訪問清單中的元素，不能使用切片修改清單中的元素。（錯）
516. 只能通過切片訪問元組中的元素，不能使用切片修改元組中的元素。（對）
517. 字串屬於Python有序序列，和列表、元組一樣都支援雙向索引。（對）
518. Python字典和集合支援雙向索引。（錯）
519. 使用print()函數無法將資訊寫入檔。（錯）
520. Python集合不支援使用下標訪問其中的元素。（對）
521. 相同內容的字串使用不同的編碼格式進行編碼得到的結果並不完全相同。（對）
522. 刪除清單中重複元素最簡單的方法是將其轉換為集合後再重新轉換為列表。（對）
523. 已知清單x中包含超過5個以上的元素，那麼語句 x = x[:5]+x[5:] 的作用是將清單x中的元素迴圈左移5位元。（錯）
524. 對於生成器對象x = (3 for i in range(5))，連續兩次執行list(x)的結果是一樣的。（錯）
525. 在迴圈中continue語句的作用是跳出當前迴圈。（錯）
526. 在編寫多層迴圈時，為了提高運行效率，應儘量減少內迴圈中不必要的計算。（對）
527. 在Python中，任意長的字串都遵守駐留機制。（錯）
528. Python運算子%不僅可以用來求餘數，還可以用來格式化字串。（對）
529. Python字串方法replace()對字串進行原地修改。（錯）
530. 如果需要連接大量字串成為一個字串，那麼使用字串物件的join()方法比運算子+具有更高的效率。（對）
531. 對於大量列表的連接，extend()方法比運算子+具有更高的效率。（對）
532. 運算式 {1, 3, 2} > {1, 2, 3} 的值為True。（錯）
533. 清單物件的extend()方法屬於原地操作，調用前後清單物件的位址不變。（對）
534. 規則運算式模組re的match()方法是從字串的開始匹配特定模式，而search()方法是在整個字串中尋找模式，這兩個方法如果匹配成功則返回match物件，匹配失敗則返回空值None。（對）
535. 函數是代碼複用的一種方式。（對）
536. 定義函數時，即使該函數不需要接收任何參數，也必須保留一對空的圓括號來表示這是一個函數。（對）
537. 編寫函數時，一般建議先對參數進行合法性檢查，然後再編寫正常的功能代碼。（對）
538. 一個函數如果帶有預設值參數，那麼必須所有參數都設置預設值。（錯）
539. 定義Python函數時必須指定函數返回數值型別。（錯）
540. 定義Python函數時，如果函數中沒有return語句，則默認返回空值None。（對）
541. 如果在函數中有語句 return 3，那麼該函數一定會返回整數3。（錯）
542. 函數中必須包含return語句。（錯）
543. 函數中的return語句一定能夠得到執行。（錯）
544. 不同作用域中的同名變數之間互相不影響，也就是說，在不同的作用域內可以定義同名的變數。（對）
545. 全域變數會增加不同函數之間的隱式耦合度，從而降低代碼可讀性，因此應儘量避免過多使用全域變數。（對）
546. 函數內部定義的區域變數當函式呼叫結束後被自動刪除。（對）
547. 在函數內部，既可以使用global來聲明使用外部全域變數，也可以使用global直接定義全域變數。（對）
548. 在函數內部沒有辦法定義全域變數。（錯）
549. 對於數字n，如果運算式 0 not in [n%d for d in range(2, n)] 的值為True則說明n是素數。（對）
550. 運算式 ‘a’+1的值為’b’。（錯）
551. 在函數內部直接修改形參的值並不影響外部實參的值。（對）
552. 在函數內部沒有任何方法可以影響實參的值。（錯）
553. 調用帶有預設值參數的函數時，不能為預設值參數傳遞任何值，必須使用函式定義時設置的預設值。（錯）
554. 創建只包含一個元素的元組時，必須在元素後面加一個逗號，例如(3,)。（對）
555. 在同一個作用域內，區域變數會隱藏同名的全域變數。（對）
556. 形參可以看做是函數內部的區域變數，函數運行結束之後形參就不可訪問了。（對）
557. 假設已導入random標準庫，那麼運算式 max([random.randint(1, 10) for i in range(10)]) 的值一定是10。（錯）
558. Python標準庫random的方法randint(m,n)用來生成一個[m,n]區間上的隨機整數。（對）
559. Python中一切內容都可以稱為物件。（對）
560. 棧和佇列的都具有先入後出的特點。（錯）
561. 在一個軟體的設計與開發中，所有類名、函數名、變數名都應該遵循統一的風格和規範。（對）
562. 定義類時所有實例方法的第一個參數用來表示物件本身，在類的外部通過物件名來調用實例方法時不需要為該參數傳值。（對）
563. 在物件導向程式設計中，函數和方法是完全一樣的，都必須為所有參數進行傳值。（錯）
564. Python中沒有嚴格意義上的私有成員。（對）
565. 在Python中定義類時，運算子重載是通過重寫特殊方法實現的。例如，在類中實現了\_\_mul\_\_()方法即可支援該類物件的\*\*運算子。（錯）
566. 在IDLE交互模式下，一個底線“\_”表示解譯器中最後一次顯示的內容或最後一次語句正確執行的輸出結果。（對）
567. 對於Python類中的私有成員，可以通過“物件名.\_類名\_\_私有成員名”的方式來訪問。（對）
568. 運算子 / 在Python 2.x和Python 3.x中具有相同的功能。（錯）
569. 運算子“-”可以用於集合的差集運算。（對）
570. 如果定義類時沒有編寫析構函數，Python將提供一個預設的析構函數進行必要的資源清理工作。（對）
571. 已知seq為長度大於10的列表，並且已導入random模組，那麼[random.choice(seq) for i in range(10)]和random.sample(seq,10)等價。（錯）
572. 在派生類中可以通過“基類名.方法名()”的方式來調用基類中的方法。（對）
573. Python支援多繼承，如果父類中有相同的方法名，而在子類中調用時沒有指定父類名，則Python解譯器將從左向右按順序進行搜索。（對）
574. 對檔進行讀寫操作之後必須顯式關閉檔以確保所有內容都得到保存。（對）
575. Python標準庫os中的方法startfile()可以啟動任何已關聯應用程式的檔，並自動調用關聯的程式。（對）
576. 程式中異常處理結構在大多數情況下是沒必要的。（錯）
577. 在try...except...else結構中，如果try塊的語句引發了異常則會執行else塊中的代碼。（錯）
578. Python標準庫threading中的Lock、RLock、Condition、Event、Semaphore物件都可以用來實現執行緒同步。（對）
579. 異常處理結構中的finally塊中代碼仍然有可能出錯從而再次引發異常。（對）
580. 在GUI設計中，核取方塊往往用來實現非互斥多選的功能，多個核取方塊之間的選擇互不影響。（對）
581. 在GUI設計中，選項按鈕用來實現使用者在多個選項中的互斥選擇，在同一組內多個選項中只能選擇一個，當選擇發生變化之後，之前選中的選項自動失效。（對）
582. 在Python中定義類時實例方法的第一個參數名稱必須是self。（錯）
583. 在Python中定義類時實例方法的第一個參數名稱不管是什麼，都表示對象自身。（對）
584. Python代碼可以內嵌在asp檔中。（對）
585. 無法配置IIS來支援Python程式的運行。（錯）
586. Python標準庫os中的方法startfile()可以用來打開外部程式或檔，系統會自動關聯相應的程式來打開或執行指定的檔。（對）
587. 在編寫應用程式時，應合理控制執行緒數量，執行緒並不是越多越好。（對）
588. 在多執行緒程式設計時，當某子執行緒的daemon屬性為False時，主執行緒結束時會檢測該子執行緒是否結束，如果該子執行緒尚未運行結束，則主執行緒會等待它完成後再退出。（對）
589. Python只能使用內置資料庫SQLite，無法訪問MS SQLServer、ACCESS或Oracle、MySQL等資料庫。（錯）
590. 使用OpenGL畫圖時，畫點是最基本的操作，具體生成的圖形由glBegin()函數指定的mode來決定。例如，mode值為GL\_TRIANGLES時表示將要繪製三角形。（對）
591. OpenGL採用的“狀態機”工作方式，一旦設置了某種狀態以後，除非顯式修改該狀態，否則該狀態將一直保持。（對）
592. 假設os模組已導入，那麼列表推導式 [filename for filename in os.listdir('C:\\Windows') if filename.endswith('.exe')] 的作用是列出C:\Windows資料夾中所有副檔名為.exe的文件。（對）
593. 運算式 list('[1, 2, 3]') 的值是[1, 2, 3]。（錯）
594. 在函數內部沒有任何聲明的情況下直接為某個變數賦值，這個變數一定是函數內部的區域變數。（對）
595. 定義類時如果實現了\_\_contains\_\_()方法，該類物件即可支援成員測試運算in。（對）
596. 定義類時如果實現了\_\_len\_\_()方法，該類物件即可支援內置函數len()。（對）
597. 定義類時實現了\_\_eq\_\_()方法，該類物件即可支援運算子==。（對）
598. 定義類時實現了\_\_pow\_\_()方法，該類物件即可支援運算子\*\*。（對）
599. 二進位檔案不能使用記事本程式打開。（錯）
600. 使用普通文字編輯器軟體也可以正常查看二進位檔案的內容。（錯）
601. 二進位檔案也可以使用記事本或其他文字編輯器打開，但是一般來說無法正常查看其中的內容。（對）
602. Python標準庫os中的方法isfile()可以用來測試給定的路徑是否為檔。（對）
603. Python標準庫os中的方法exists()可以用來測試給定路徑的檔是否存在。（對）
604. Python標準庫os中的方法isdir()可以用來測試給定的路徑是否為資料夾。（對）
605. Python標準庫os中的方法listdir()返回包含指定路徑中所有檔和資料夾名稱的列表。（對）
606. Python擴展庫xlwt支持對Excel 2003或更低版本的Excel檔進行寫操作。（對）
607. Python擴展庫xlrd支持對Excel 2003或更低版本的Excel檔進行讀操作。（對）
608. 帶有else子句的異常處理結構，如果不發生異常則執行else子句中的代碼。（對）
609. 異常處理結構也不是萬能的，處理異常的代碼也有引發異常的可能。（對）
610. 在異常處理結構中，不論是否發生異常，finally子句中的代碼總是會執行的。（對）
611. 在Python中定義函數時不需要聲明函數參數的類型。（對）
612. 在Python中定義函數時不需要聲明函數的返回數值型別。（對）
613. 在函數中沒有任何辦法可以通過形參來影響實參的值。（錯）
614. 已知 x = 3，那麼執行語句 x+=6 之後，x的記憶體位址不變。（錯）
615. 已知x為非空字串，那麼運算式 ''.join(x.split()) == x 的值一定為True。（錯）
616. 已知x為非空字串，那麼運算式 ','.join(x.split(',')) == x 的值一定為True。（對）
617. 在Python中可以使用 for 作為變數名。（錯）
618. 在Python中可以使用 id 作為變數名，儘管不建議這樣做。（對）
619. Python關鍵字不可以作為變數名。（對）
620. 一個數字5也是合法的Python運算式。（對）
621. 同一個清單物件中的元素類型可以各不相同。（對）
622. 同一個清單物件中所有元素必須為相同類型。（錯）
623. 已知x為非空列表，那麼執行語句x[0] = 3之後，清單物件x的記憶體位址不變。（對）
624. 清單可以作為集合的元素。（錯）
625. 集合可以作為清單的元素。（對）
626. 元組可以作為集合的元素。（對）
627. 集合可以作為元組的元素。（對）
628. 字典可以作為集合的元素。（錯）
629. 集合可以作為字典的鍵。（錯）
630. 集合可以作為字典的值。（對）
631. 可以使用del刪除集合中的部分元素。（錯）
632. 標準庫os的rename()方法可以實現檔移動操作。（對）
633. 標準庫os的listdir()方法默認只能列出指定資料夾中當前層級的檔和資料夾清單，而不能列出其子資料夾中的檔。（對）
634. 當作為條件運算式時，[]與None等價。（對）
635. 運算式 [] == None 的值為True。（錯）
636. 當作為條件運算式時，{}與None等價。（對）
637. 運算式 {}==None 的值為True。（錯）
638. 運算式 pow(3,2) == 3\*\*2 的值為True。（對）
639. 當作為條件運算式時，空值、空字串、空清單、空元組、空字典、空集合、空反覆運算物件以及任意形式的數位0都等價於False。（對）
640. 在定義函數時，某個參數名字前面帶有一個\*符號表示可變長度參數，可以接收任意多個普通實參並存放於一個元組之中。（對）
641. 在定義函數時，某個參數名字前面帶有兩個\*符號表示可變長度參數，可以接收任意多個關鍵參數並將其存放於一個字典之中。（對）
642. 定義函數時，帶有預設值的參數必須出現在參數列表的最右端，任何一個帶有預設值的參數右邊不允許出現沒有預設值的參數。（對）
643. 在調用函數時，可以通過關鍵參數的形式進行傳值，從而避免必須記住函數形參順序的麻煩。（對）
644. 在調用函數時，必須牢記函數形參順序才能正確傳值。（錯）
645. 調用函數時傳遞的實參個數必須與函數形參個數相等才行。（錯）
646. 規則運算式物件的match()方法可以在字串的指定位置開始進行指定模式的匹配。（對）
647. 使用規則運算式對字串進行分割時，可以指定多個分隔符號，而字串物件的split()方法無法做到這一點。（對）
648. 在編寫函數時，建議首先對形參進行類型檢查和數值範圍檢查之後再編寫功能代碼，或者使用異常處理結構，儘量避免代碼拋出異常而導致程式崩潰。（對）
649. 執行語句 from math import sin 之後，可以直接使用 sin() 函數，例如 sin(3)。（對）
650. 清單物件的pop()方法預設刪除並返回最後一個元素，如果清單已空則拋出異常。（對）
651. 在Python中定義類時，如果某個成員名稱前有2個底線則表示是私有成員。（對）
652. 在類定義的外部沒有任何辦法可以訪問物件的私有成員。（錯）
653. 可以使用py2exe或pyinstaller等擴展庫把Python來源程式打包成為exe檔，從而脫離Python環境在Windows平臺上運行。（對）
654. Python程式只能在安裝了Python環境的電腦上以原始程式碼形式運行。（錯）
655. 不同版本的Python不能安裝到同一台電腦上。（錯）
656. 一般來說，Python擴展庫沒有通用於所有版本Python的，安裝時應選擇與已安裝Python的版本對應的擴展庫。
657. 運算式 {1, 2} \* 2 的值為 {1, 2, 1, 2}。（錯）
658. Python變數名區分大小寫，所以student和Student不是同一個變數。（對）
659. 規則運算式元字元“^”一般用來表示從字串開始處進行匹配，用在一對方括弧中的時候則表示反向匹配，不匹配方括號中的字元。（對）
660. 規則運算式元字元“\s”用來匹配任意空白字元。（對）
661. 規則運算式 元字元“\d”用來匹配任意數位字元。（對）
662. lambda運算式中可以使用任意複雜的運算式，但是必須只編寫一個運算式。（對）
663. Python類的構造函數是\_\_init\_\_()。（對）
664. 定義類時，在一個方法前面使用@classmethod進行修飾，則該方法屬於類方法。（對）
665. 定義類時，在一個方法前面使用@staticmethod進行休息，則該方法屬於靜態方法。（對）
666. 通過物件不能調用類方法和靜態方法。（錯）
667. 在Python中可以為自訂類的物件動態增加新成員。（對）
668. Python類不支持多繼承。（錯）
669. 屬性可以像資料成員一樣進行訪問，但賦值時具有方法的優點，可以對新值進行檢查。（對）
670. 檔物件是可以反覆運算的。（對）
671. 檔物件的tell()方法用來返回檔指針的當前位置。（對）
672. 以寫模式打開的檔無法進讀操作。（對）
673. 假設已成功導入os和sys標準庫，那麼運算式 os.path.dirname(sys.executable) 的值為Python安裝目錄。（對）
674. 只可以動態為物件增加資料成員，而不能為物件動態增加成員方法。（錯）
675. Python字典支援雙向索引。（錯）
676. Python集合支援雙向索引。（錯）
677. Python元組支援雙向索引。（對）
678. 假設re模組已成功導入，並且有 pattern = re.compile('^'+'\.'.join([r'\d{1,3}' for i in range(4)])+'$')，那麼運算式 pattern.match('192.168.1.103') 的值為None。（錯）
679. 假設random模組已導入，那麼運算式 random.sample(range(10), 20) 的作用是生成20個不重複的整數。（錯）
680. 假設random模組已導入，那麼運算式 random.sample(range(10), 7) 的作用是生成7個不重複的整數。（對）
681. 在Python 3.x中reduce()是內置函數。（錯）
682. 以讀模式打開檔時，檔指標指向檔開始處。（對）
683. 以追加模式打開檔時，檔指標指向檔案結尾。（對）
684. 已知x = (1, 2, 3, 4)，那麼執行x[0] = 5之後，x的值為(5, 2, 3, 4)。（錯）
685. 已知x = 3，那麼執行x += 6語句前後x的記憶體位址是不變的。（錯）
686. 成員測試運算子in作用于集合時比作用於清單快得多。（對）
687. 在Python 3.x中，使用內置函數input()接收使用者輸入時，不論用戶輸入的什麼格式，一律按字串進行返回。（對）
688. 安裝Python擴展庫時只能使用pip工具線上安裝，如果安裝不成功就沒有別的辦法了。（錯）
689. 使用random模組的函數randint(1, 100)獲取亂數時，有可能會得到100。（對）
690. 如果只需要math模組中的sin()函數，建議使用from math import sin來導入，而不要使用import math導入整個模組。（對）
691. 已知列表x = [1, 2, 3, 4]，那麼運算式x.find(5)的值應為-1。（錯）
692. 清單物件的排序方法sort()只能按元素從小到大排列，不支援別的排序方式。（錯）
693. g = lambda x: 3不是一個合法的賦值運算式。（錯）
694. 內置函數len()返回指定序列的元素個數，適用於清單、元組、字串、字典、集合以及range、zip等反覆運算物件。（對）
695. 已知x和y是兩個等長的整數清單，那麼運算式sum((i\*j for i, j in zip(x, y)))的作用是計算這兩個列表所表示的向量的內積。（對）
696. 已知x和y是兩個等長的整數清單，那麼運算式[i+j for i,j in zip(x,y)]的作用時計算這兩個列表所表示的向量的和。（對）
697. 運算式int('1'\*64, 2)與sum(2\*\*i for i in range(64))的計算結果是一樣的，但是前者更快一些。（對）
698. 已知x = list(range(20))，那麼語句del x[::2]可以正常執行。（對）
699. 已知x = list(range(20))，那麼語句x[::2] = []可以正常執行。（錯）
700. 已知x = list(range(20))，那麼語句print(x[100:200])無法正常執行。（錯）
701. 已知x是個清單物件，那麼執行語句y = x之後，對y所做的任何操作都會同樣作用到x上。（對）
702. 已知x是個清單物件，那麼執行語句y = x[:]之後，對y所做的任何操作都會同樣作用到x上。（錯）
703. 在Python中，變數不直接存儲值，而是存儲值的引用，也就是值在記憶體中的位址。（對）
704. 運算式(i\*\*2 for i in range(100))的結果是個元組。（錯）
705. 在Python中元組的值是不可變的，因此，已知x = ([1], [2])，那麼語句x[0].append(3)是無法正常執行的。（錯）
706. 包含yield語句的函數一般成為生成器函數，可以用來創建生成器物件。（對）
707. 在函數中yield語句的作用和return完全一樣。（錯）
708. Python內置的字典dict中元素是按添加的順序依次進行存儲的。（錯）
709. Python內置的集合set中元素順序是按元素的雜湊值進行存儲的，並不是按先後順序。（對）
710. 已知x = {1:1, 2:2}，那麼語句x[3] =3無法正常執行。（錯）
711. Python內置字典是無序的，如果需要一個可以記住元素插入順序的字典，可以使用collections.OrderedDict。（對）
712. 語句pass僅起到預留位置的作用，並不會做任何操作。（對）
713. 在條件運算式中不允許使用設定運算子“=”，會提示語法錯誤。（對）
714. 任何包含\_\_call\_\_()方法的類的物件都是可調用的。（對）
715. 在Python中函數和類都屬於可調用物件。（對）
716. 無法使用lambda運算式定義有名字的函數。（錯）
717. 已知x是一個列表，那麼x = x[3:] + x[:3]可以實現把清單x中的所有元素迴圈左移3位元。（對）
718. 已知x和y是兩個字串，那麼運算式sum((1 for i,j in zip(x,y) if i==j))可以用來計算兩個字串中對應位置字元相等的個數。（對）
719. 函數和物件方法是一樣的，內部實現和外部調用都沒有任何區別。（錯）
720. 在設計派生類時，基類的私有成員默認是不會繼承的。（對）
721. 如果在設計一個類時實現類\_\_len\_\_()方法，那麼該類的物件會自動支援Python內置函數len()。（對）
722. Python 3.x中字串物件的encode()方法預設使用utf8作為編碼方式。（對）
723. 已知x = 'hellow world.'.encode()，那麼運算式x.decode('gbk')的值為'hellow world.'。（對）
724. 已知x = 'Python是一種非常好的程式設計語言'.encode()，那麼運算式x.decode('gbk')的值為'Python是一種非常好的程式設計語言'。（錯）
725. 規則運算式'^http'只能匹配所有以'http'開頭的字串。（對）
726. 規則運算式'^\d{18}|\d{15}$'只能檢查給定字串是否為18位元或15位元數位字元，並不能保證一定是合法的身份證號。（對）
727. 二進位檔案也可以使用記事本程式打開，只是無法正確閱讀和理解其中的內容。（對）
728. 規則運算式'[^abc]'可以一個匹配任意除'a'、'b'、'c'之外的字元。（對）
729. 規則運算式'python|perl'或'p(ython|erl)'都可以匹配'python'或'perl'。（對）
730. 文字檔是可以反覆運算的，可以使用for line in fp類似的語句遍歷檔物件fp中的每一行。（對）
731. Python的主程序文件python.exe屬於二進位檔案。（對）
732. 使用記事本程式也可以打開二進位檔案，只不過無法正確識別其中的內容。（對）
733. 對字串資訊進行編碼以後，必須使用同樣的或者相容的編碼格式進行解碼才能還原本來的資訊。（對）
734. 使用pickle進行序列化得到的二進位檔案使用struct也可以正確地進行反序列化。（錯）
735. 已知當前資料夾中有一個檔readme.txt具有唯讀屬性，假設標準庫os已正確導入，那麼可以通過語句os.chmod('readme.txt', 0o777)來刪除該檔的唯讀屬性。（對）
736. Python標準庫os的函數remove()不能刪除具有唯讀屬性的檔。（對）
737. 位元組串b'hello world'和b'hello world.'的MD5值相差很小。（錯）
738. 由於異常處理結構try...except...finally...中finally裡的語句塊總是被執行的，所以把關閉檔的代碼放到finally塊裡肯定是萬無一失，一定能保證檔被正確關閉並且不會引發任何異常。（錯）
739. 使用TCP協議進行通信時，必須首先建立連接，然後進行資料傳輸，最後再關閉連接。（對）
740. TCP是可以提供良好服務品質的傳輸層協定，所以在任何場合都應該優先考慮使用。（錯）
741. 在4核CPU平臺上使用多執行緒程式設計技術可以很輕易地獲得400%的處理速度提升。（錯）
742. 多執行緒程式設計技術主要目的是為了提高電腦硬體的利用率，沒有別的作用了。（錯）
743. 簡答題
744. 簡單解釋Python基於值的自動記憶體管理方式？（Python採用的是基於值得記憶體管理方式，在Python中可以為不同變數賦值為相同值，這個值在記憶體中只有一份，多個變數指向同一個記憶體位址；Python具有自動記憶體管理功能，會自動跟蹤記憶體中所有的值，對於沒有任何變數指向的值，Python自動將其刪除。）
745. 寫出Python運算子&的兩種功能？（1）數位位元運算；2）集合交集運算。）
746. 在Python中導入模組中的物件有哪幾種方式？（1)import 模組名 [as 別名]；2)from 模組名 import 物件名[ as 別名]；3)from math import \*）
747. 解釋Python腳本程式的“\_\_name\_\_”變數及其作用？（每個Python腳本在運行時都有一個“\_\_name\_\_”屬性。如果腳本作為模組被導入，則其“\_\_name\_\_”屬性的值被自動設置為模組名；如果腳本獨立運行，則其“\_\_name\_\_”屬性值被自動設置為“\_\_main\_\_”。利用“\_\_name\_\_”屬性即可控制Python程式的運行方式。）
748. 為什麼應儘量從清單的尾部進行元素的增加與刪除操作？（當清單增加或刪除元素時，清單物件自動進行記憶體擴展或收縮，從而保證元素之間沒有縫隙，但這涉及到清單元素的移動，效率較低，應儘量從清單尾部進行元素的增加與刪除操作以提高處理速度。）
749. 分析邏輯運算子“or”的短路求值特性？（假設有運算式“運算式1 or 運算式2”，如果運算式1的值等價於True，那麼無論運算式2的值是什麼，整個運算式的值總是等價於True。因此，不需要再計算運算式2的值。）
750. 簡單解釋Python中短字串駐留機制？（對於短字串，將其賦值給多個不同的物件時，記憶體中只有一個副本，多個物件共用改副本。）
751. 異常和錯誤有什麼區別？（異常是指因為程式執行過程中出錯而在正常控制流以外採取的行為。嚴格來說，語法錯誤和邏輯錯誤不屬於異常，但有些語法錯誤往往會導致異常，例如由於大小寫拼寫錯誤而訪問不存在的物件，或者試圖訪問不存在的檔，等等。）
752. 使用pdb模組進行Python程式調試主要有哪幾種用法？（1）在交互模式下使用pdb模組提供的功能可以直接調試語句塊、運算式、函數等多種腳本。2）在程式中嵌入中斷點來實現調試功能。在程式中首先導入pdb模組，然後使用pdb.set\_trace()在需要的位置設置中斷點。如果程式中存在通過該方法調用顯式插入的中斷點，那麼在命令提示符環境下執行該程式或按兩下執行程式時將自動打開pdb調試環境，即使該程式當前不處於調試狀態。3）使用命令列偵錯工具。在命令列提示符下執行“python –m pdb 指令檔名”，則直接進入調試環境；當調試結束或程式正常結束以後，pdb將重啟該程式。）
753. 閱讀下面的代碼，並分析假設檔“D:\test.txt”不存在的情況下兩段代碼可能發生的問題。

代碼1：

>>> try:

fp = open(r'd:\test.txt')

print('Hello world!', file=fp)

finally:

fp.close()

代碼2：

>>> try:

fp = open(r'd:\test.txt', 'a+')

print('Hello world!', file=fp)

finally:

fp.close()

答：

假設文件“D:\test.txt”不存在，那麼第一段代碼會拋出異常，提示fp沒有定義；第二段代碼執行正常。原因是第二段代碼使用內置函數open()打開指定檔時如果不存在則會創建該檔，從而不會拋出異常。

1. 程式設計題
2. 編寫程式，在D盤根目錄下創建一個文字檔test.txt，並向其中寫入字串hello world。

答：

fp = open(r’D:\test.txt’, ‘a+’)

print(‘hello world’, file=fp)

fp.close()

1. 寫出下面代碼的優化版本，提高運行效率。

x = list(range(500))

for item in x:

t = 5\*\*5

print(item+t)

答：

x = list(range(500))

t = 5\*\*5

for item in x:

print(item+t)

1. 編寫程式，生成一個包含20個隨機整數的清單，然後對其中偶數下標的元素進行降冪排列，奇數下標的元素不變。（提示：使用切片。）

答：

import random

x = [random.randint(0,100) for i in range(20)]

print(x)

y = x[::2]

y.sort(reverse=True)

x[::2] = y

print(x)

1. 寫出下面代碼的執行結果。

def Join(List, sep=None):

return (sep or ',').join(List)

print(Join(['a', 'b', 'c']))

print(Join(['a', 'b', 'c'],':'))

答：

a,b,c

a:b:c

1. 寫出下面代碼的運行結果。

def Sum(a, b=3, c=5):

return sum([a, b, c])

print(Sum(a=8, c=2))

print(Sum(8))

print(Sum(8,2))

答：

13

16

15

1. 寫出下面代碼的運行結果。

def Sum(\*p):

return sum(p)

print(Sum(3, 5, 8))

print(Sum(8))

print(Sum(8, 2, 10))

答：

16

8

20

1. 編寫函數，判斷一個數位是否為素數，是則返回字串YES，否則返回字串NO。

答：

import math

def IsPrime(v):

n = int(math.sqrt(v)+1)

for i in range(2,n):

if v%i==0:

return 'No'

else:

return 'Yes'

1. 編寫函數，類比Python內置函數sorted()。

答：

def Sorted(v):

t = v[::]

r = []

while t:

tt = min(t)

r.append(tt)

t.remove(tt)

return r

1. 編寫程式，生成包含20個亂數的列表，然後將前10個元素昇冪排列，後10個元素降冪排列，並輸出結果。

答：

import random

x = [random.randint(0,100) for i in range(20)]

print(x)

y = x[0:10]

y.sort()

x[0:10] = y

y = x[10:20]

y.sort(reverse=True)

x[10:20] = y

print(x)

1. 編寫程式，運行後使用者輸入4位元整數作為年份，判斷其是否為閏年。如果年份能被400整除，則為閏年；如果年份能被4整除但不能被100整除也為閏年。

答：

x = input('Please input an integer of 4 digits meaning the year:')

x = eval(x)

if x%400==0 or (x%4==0 and not x%100==0):

print('Yes')

else:

print('No')

1. 編寫程式，實現分段函數計算，如下表所示。

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

答：

x = input('Please input x:')

x = eval(x)

if x<0 or x>=20:

print(0)

elif 0<=x<5:

print(x)

elif 5<=x<10:

print(3\*x-5)

elif 10<=x<20:

print(0.5\*x-2)

1. 閱讀下面的程式，判斷其是否可以正常運行，如果可以運行則寫出執行結果，如果不能運行則寫出理由。

class Test:

def \_\_init\_\_(self, value):

self.\_\_value = value

@property

def value(self):

return self.\_\_value

t = Test(3)

t.value = 5

print(t.value)

答：

不能運行。程式中定義的是唯讀屬性，不能修改屬性的值。

1. 下面代碼的功能是，隨機生成50個介於[1,20]之間的整數，然後統計每個整數出現頻率。請把缺少的代碼補全。

import random

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

r = dict()

for i in x:

r[i] = r.get(i, \_\_\_\_\_)+1

for k, v in r.items():

print(k, v)

答：

分別填寫randint、50、0

1. 假設有Python程式檔demo.py，代碼如下：

def main():

if \_\_name\_\_ == '\_\_main\_\_':

print(1)

else:

print(2)

main()

將該程式檔直接運行時輸出結果為\_\_\_\_\_\_\_\_\_，作為模組導入時得到結果\_\_\_\_\_\_\_\_\_\_\_-。（1、2）

1. 下面程式的執行結果是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1）

s = 0

for i in range(1,101):

s += i

else:

print(1)

1. 下面程式的執行結果是\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（1275）

s = 0

for i in range(1,101):

s += i

if i == 50:

print(s)

break

else:

print(1)

1. 下面的程式是否能夠正常執行，若不能，請解釋原因；若能，請分析其執行結果。

from random import randint

result = set()

while True:

result.add(randint(1,10))

if len(result)==20:

break

print(result)

答：無法正確執行，因為該程式的功能是從[1,10]區間中選擇20個不同的隨機整數，而該區間並沒有這麼多整數，所以程式閉環。

1. 下面的代碼是否能夠正確運行，若不能請解釋原因；若能，請分析其執行結果。

>>> x = list(range(20))

>>> for i in range(len(x)):

del x[i]

答：無法正確執行，因為刪除清單元素時會影響其他元素在清單中的索引，上面的代碼會拋出下標越界的異常。

1. 閱讀下面的代碼，解釋其功能。

>>> x = list(range(20))

>>> for index, value in enumerate(x):

if value == 3:

x[index] = 5

答：將清單x中值為3的元素修改為5。

1. 閱讀下面的代碼，解釋其功能。

>>> x = [range(3\*i, 3\*i+5) for i in range(2)]

>>> x = list(map(list, x))

>>> x = list(map(list, zip(\*x)))

答：首先生成一個包含列表的列表，然後模擬矩陣轉置。

1. 閱讀下面的代碼，解釋其功能。

>>> import string

>>> x = string.ascii\_letters + string.digits

>>> import random

>>> print(''.join(random.sample(x, 10)))

答：輸出由英文字母大小寫或數位組成的長度為10且不重複的隨機字串。

1. 閱讀下面的代碼，分析其執行結果。

def demo(\*p):

return sum(p)

print(demo(1,2,3,4,5))

print(demo(1,2,3))

答：輸出結果為

15

6

1. 閱讀下面的代碼，分析其執行結果。

def demo(a, b, c=3, d=100):

return sum((a,b,c,d))

print(demo(1, 2, 3, 4))

print(demo(1, 2, d=3))

答：輸出結果為

10

9

1. 下面的代碼輸出結果為\_\_\_\_\_\_\_\_\_\_\_\_\_。（3）

def demo():

x = 5

x = 3

demo()

print(x)

1. 下面函數的功能為\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（將序列迴圈左移k位，得到新序列並返回）

def demo(lst, k):

if k<len(lst):

return lst[k:]+lst[:k]

1. 編寫函數，求任意整數的二進位形式中最後連續0的個數。

def demo(n):

b\_n = bin(n)

index = b\_n.rfind('1') + 1

return len(b\_n[index:])

1. 有n個乒乓球運動員打淘汰賽，編寫函數計算至少需要多少場比賽才能決出冠軍，不允許直接使用n-1。

def demo(n):

if n == 1:

return 0

if n == 2:

return 1

m, c = divmod(n, 2)

return m + demo(c+m)

1. 使用迴圈和列表推導式兩種方法求解百錢買百雞問題。假設大雞5元一隻，中雞3元一隻，小雞1元三隻，現有100元錢想買100只雞，有多少種買法？
2. 迴圈

>>> for x in range(21):

for y in range(34):

z = 100-x-y

if z%3==0 and 5\*x + 3\*y + z//3 == 100:

print(x,y,z)

0 25 75

4 18 78

8 11 81

12 4 84

1. 列表推導式

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

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