1、 Python安装扩展库常用的是\_工具。（pip）   
2、 Python标准库math中用来计算平方根的函数是\_\_\_\_。（sqrt）   
3、 Python程序文件扩展名主要有\_\_和两种，其中后者常用于GUI程序。（py、pyw）   
4、 Python源代码程序编译后的文件扩展名为\_\_\_。（pyc）   
5、 使用pip工具升级科学计算扩展库numpy的完整命令是\_\_\_\_\_\_\_。（pip install –upgrade numpy）   
6、 使用pip工具查看当前已安装的Python扩展库的完整命令是\_\_\_。（pip list）   
7、 在IDLE交互模式中浏览上一条语句的快捷键是\_\_\_\_。（Alt+P）   
8、 使用pip工具查看当前已安装Python扩展库列表的完整命令是\_\_\_。（pip list）   
9、 在Python中\_\_\_\_表示空类型。（None）   
10、 列表、元组、字符串是Python的\_\_\_(有序？无序)序列。（有序）   
11、 查看变量内存地址的是\_\_\_\_\_\_。（type()）   
12、 查看变量内存地址的Python内置函数是\_\_\_\_\_\_\_。（id()）   
13、 以3为实部4为虚部，Python复数的表达形式为\_或\_\_。（3+4j、3+4J）   
14、 Python运算符中用来计算整商的是\_\_\_。（//）   
15、 Python运算符中用来计算集合并集的是\_。（|）   
16、 使用运算符测试集合包含集合A是否为集合B的真子集的表达式可以写作\_。（A < B ）   
17、 表达式[1, 2, 3]\*3的执行结果为\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 1, 2, 3, 1, 2, 3]）   
18、 list(map(str, [1, 2, 3]))的执行结果为\_\_\_\_\_\_\_\_\_\_\_。（[‘1’, ‘2’, ‘3’]）   
19、 语句x = 3==3, 5执行结束后，变量x的值为\_\_\_。（(True, 5））   
20、 已知 x = 3，那么执行语句 x += 6 之后，x的值为\_\_\_\_\_。（9）   
21、 已知 x = 3，并且id(x)的返回值为 496103280，那么执行语句 x += 6 之后，表达式 id(x) == 496103280 的值为\_。（False）   
22、 已知 x = 3，那么执行语句 x \*= 6 之后，x的值为\_\_\_\_\_\_。（18）   
23、 为了提高Python代码运行速度和进行适当的保密，可以将Python程序文件编译为扩展名\_\_\_\_的文件。（pyc）   
24、 表达式“[3] in [1, 2, 3, 4]”的值为\_\_\_\_\_\_。（False）   
25、 列表对象的sort()方法用来对列表元素进行原地排序，该函数返回值为 。（None）   
26、 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\_\_\_\_\_\_\_\_\_\_\_\_。（[6, 7, 9, 11]）   
27、 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\_\_\_\_\_。（[5 for i in range(10)]）   
28、 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\_\_\_\_\_\_\_\_\_\_\_。（c = dict(zip(a, b))）   
29、 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\_\_。（-1）   
30、 Python语句”.join(list(‘hello world!’))执行的结果是\_\_\_\_\_\_\_\_\_\_。（’hello world!’）   
31、 转义字符’\n’的含义是\_\_\_\_\_\_\_\_\_。（回车换行）   
32、 Python语句list(range(1,10,3))执行结果为\_\_\_\_\_\_\_\_\_。（[1, 4, 7]）   
33、 表达式 list(range(5)) 的值为\_\_\_\_\_\_。（[0, 1, 2, 3, 4]）   
34、 \_\_\_\_命令既可以删除列表中的一个元素，也可以删除整个列表。（del）   
35、 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\_。（True）   
36、 表达式 int(‘123’, 16) 的值为\_\_\_。（291）   
37、 表达式 int(‘123’, 8) 的值为\_\_\_。（83）   
38、 表达式 int(‘123’) 的值为\_\_\_。（123）   
39、 表达式 int(‘101’,2) 的值为\_\_\_\_。（5）   
40、 表达式 abs(-3) 的值为\_。（3）   
41、 切片操作list(range(6))[::2]执行结果为\_\_\_\_\_\_。（[0, 2, 4]）   
42、 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\_\_\_。（x[0:0] = [3]）   
43、 语句sorted([1, 2, 3], reverse=True) == reversed([1, 2, 3])执行结果为\_\_\_。（False）   
44、 表达式 ‘ab’ in ‘acbed’ 的值为\_\_。（False）   
45、 Python 3.x语句 print(1, 2, 3, sep=’:’) 的输出结果为\_\_。表达式 int(4\*\*0.5) 的   
46、 表达式 sorted([111, 2, 33], key=lambda x: len(str(x))) 的值为\_\_\_\_\_\_。（[2, 33, 111]）   
47、 假设n为整数，那么表达式 n&1 == n%2 的值为\_\_\_。（True）   
48、 表达式 int(4\*\*0.5) 的值为\_\_。可以返回列表、元组、字典、集合、字符串   
49、 达式 sorted([111, 2, 33], key=lambda x: -len(str(x))) 的值为\_\_。（[111, 33, 2]）   
50、 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。（len()）   
51、 Python内置函数\_\_用来返回序列中的最大元素。（max()）   
52、 Python内置函数\_\_用来返回序列中的最小元素。（min()）   
53、 Python内置函数\_\_\_\_\_\_用来返回数值型序列中所有元素之和。（sum()）   
54、 已知列表对象x = [‘11’, ‘2’, ‘3’]，则表达式 max(x) 的值为\_。（’3’）   
55、 表达式 min([‘11’, ‘2’, ‘3’]) 的值为\_\_\_\_\_\_\_。（’11’）   
56、 已知列表对象x = [‘11’, ‘2’, ‘3’]，则表达式max(x, key=len) 的值为\_。（’11’）   
57、 语句 x = (3,) 执行后x的值为\_\_\_\_\_。（(3,)）   
58、 语句 x = (3) 执行后x的值为\_\_\_\_\_\_。（3）   
59、 已知x=3和y=5，执行语句 x, y = y, x 后x的值是\_\_。（5）   
60、 可以使用内置函数\_\_\_\_\_查看包含当前作用域内所有全局变量和值的字典。（globals()）   
61、 可以使用内置函数\_\_\_\_\_\_\_\_\_查看包含当前作用域内所有局部变量和值的字典。（locals()(）   
62、 字典中多个元素之间使用\_\_分隔开，每个元素的“键”与“值”之间使用\_\_分隔开。（逗号、冒号）   
63、 字典对象的\_方法可以获取指定“键”对应的“值”，并且可以在指定“键”不存在的时候返回指定值，如果不指定则返回None。（get()）   
64、 字典对象的\_\_\_方法返回字典中的“键-值对”列表。（items()）   
65、 字典对象的\_\_方法返回字典的“键”列表。（keys()）   
66、 字典对象的\_\_\_\_方法返回字典的“值”列表。（values()）   
67、 已知 x = {1:2}，那么执行语句 x[2] = 3之后，x的值为\_\_\_\_\_\_。（{1: 2, 2: 3}）   
68、 表达式 {1, 2, 3, 4} - {3, 4, 5, 6}的值为\_\_\_\_\_\_\_\_。（{1, 2}）   
69、 表达式set([1, 1, 2, 3])的值为\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3}）   
70、 关键字\_\_\_\_用于测试一个对象是否是一个可迭代对象的元素。（in）   
71、 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[i for i in range(100) if i%13==0]）   
72、 表达式 3<5>2 的值为\_\_\_\_\_。（True）   
73、 已知 x = {‘a’:’b’, ‘c’:’d’}，那么表达式 ‘a’ in x 的值为\_\_\_\_。（True）   
74、 已知 x = {‘a’:’b’, ‘c’:’d’}，那么表达式 ‘b’ in x 的值为\_\_\_\_。（False）   
75、 已知 x = {‘a’:’b’, ‘c’:’d’}，那么表达式 ‘b’ in x.values() 的值为\_\_\_\_。（True）   
76、 表达式 1<2<3 的值为\_\_\_。（True）   
77、 表达式 3 or 5 的值为\_\_（3）   
78、 表达式 0 or 5 的值为\_\_\_（5）   
79、 表达式 3 and 5 的值为\_\_。（5）   
80、 表达式 3 and not 5 的值为\_\_\_\_。（False）   
81、 表达式 3 | 5 的值为\_\_\_\_。（7）   
82、 表达式 3 & 6 的值为\_\_\_。（2）   
83、 表达式 3 \*\* 2 的值为\_\_\_。（9）   
84、 表达式 3 \* 2的值为\_。（6）   
85、 已知 x = [3, 5, 7]，那么表达式 x[10:]的值为\_\_。（[]）   
86、 已知 x = [3, 5, 7]，那么执行语句 x[len(x):] = [1, 2]之后，x的值为\_\_\_\_。（[3, 5, 7, 1, 2]）   
87、 已知 x = [3, 7, 5]，那么执行语句 x.sort(reverse=True)之后，x的值为\_\_\_\_\_\_\_。（[7, 5, 3]）   
88、 已知 x = [3, 7, 5]，那么执行语句 x = x.sort(reverse=True)之后，x的值为\_\_\_。（None）   
89、 已知 x = [1, 11, 111]，那么执行语句 x.sort(key=lambda x: len(str(x)), reverse=True) 之后，x的值为\_\_\_\_\_\_\_\_。（[111, 11, 1]）   
90、 表达式 list(zip([1,2], [3,4])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[(1, 3), (2, 4)]）   
91、 已知 x = [1, 2, 3, 2, 3]，执行语句 x.pop() 之后，x的值为\_\_\_。（[1, 2, 3, 2]）   
92、 表达式 list(map(list,zip(\*[[1, 2, 3], [4, 5, 6]]))) 的值为\_\_\_\_\_\_。（[[1, 4], [2, 5], [3, 6]]）   
93、 表达式 [x for x in [1,2,3,4,5] if x<3] 的值为\_\_\_\_\_\_\_\_\_\_\_。（[1, 2]）  
94、 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\_\_\_\_\_\_\_\_。（[2, 4]）   
95、 已知 x = [3,5,3,7]，那么表达式 [x.index(i) for i in x if i==3] 的值为\_\_\_\_。（[0, 0]）   
96、 已知列表 x = [1, 2]，那么表达式 list(enumerate(x)) 的值为\_\_\_\_\_。（[(0, 1), (1, 2)]）   
97、 已知 vec = [[1,2], [3,4]]，则表达式 [col for row in vec for col in row] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 4]）   
98、 已知 vec = [[1,2], [3,4]]，则表达式 [[row[i] for row in vec] for i in range(len(vec[0]))] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 3], [2, 4]]）   
99、 已知 x = list(range(10))，则表达式 x[-4:] 的值为\_\_\_\_。（[6, 7, 8, 9]）   
100、 已知 path = r’c:\test.html’，那么表达式 path[:-4]+’htm’ 的值为\_\_\_\_。（’c:\test.htm’）   
101、 已知 x = [3, 5, 7]，那么执行语句 x[1:] = [2]之后，x的值为\_\_\_\_\_\_。（[3, 2]）   
102、 已知 x = [3, 5, 7]，那么执行语句 x[:3] = [2]之后，x的值为\_\_\_\_\_\_。（[ 2]）   
103、 已知x为非空列表，那么执行语句y = x[:]之后，id(x[0]) == id(y[0])的值为\_\_\_\_。（True）   
104、 已知 x = [1, 2, 3, 2, 3]，执行语句 x.remove(2) 之后，x的值为\_\_。（[1, 3, 2, 3]）   
105、 表达式 3<<2 的值为\_\_\_\_\_。（12）   
106、 表达式 65 >> 1 的值为\_\_\_。（32）   
107、 表达式 chr(ord(‘a’)^32) 的值为\_。（’A’）   
108、 表达式 chr(ord(‘a’)-32) 的值为\_。（’A’）   
109、 表达式 abs(3+4j) 的值为\_\_。（5.0）   
110、 表达式 callable(int) 的值为\_。（True）   
111、 表达式 list(str([1,2,3])) == [1,2,3] 的值为\_\_\_\_。（False）   
112、 表达式 str([1, 2, 3]) 的值为\_\_\_\_\_\_\_\_。（’[1, 2, 3]’）   
113、 表达式 str((1, 2, 3)) 的值为\_\_\_\_\_\_\_\_。（’(1, 2, 3)’）   
114、 Python中用于表示逻辑与、逻辑或、逻辑非运算的关键字分别是\_、\_\_\_、\_\_\_。（and、or、not）   
115、 Python 3.x语句 for i in range(3):print(i, end=’,’) 的输出结果为\_\_\_\_\_\_\_\_\_\_\_。（0,1,2,）   
116、 Python 3.x语句 print(1, 2, 3, sep=’,’) 的输出结果为\_\_\_\_\_\_。（1,2,3）   
117、 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。（会）   
118、 在循环语句中，\_\_\_\_语句的作用是提前结束本层循环。（break）   
119、 在循环语句中，\_语句的作用是提前进入下一次循环。（continue）   
120、 表达式 sum(range(1, 10, 2)) 的值为\_\_。（25）   
121、 表达式 sum(range(1, 10)) 的值为\_\_\_。（45）   
122、 表达式 ‘%c’%65 的值为\_\_\_。（’A’）   
123、 表达式 ‘%s’%65 的值为\_\_\_\_。（’65’）   
124、 表达式 ‘%d,%c’ % (65, 65) 的值为\_\_。（’65,A’）   
125、 表达式 ‘The first:{1}, the second is {0}’.format(65,97) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（’The first:97, the second is 65’）   
126、 表达式 ‘{0:#d},{0:#x},{0:#o}’.format(65) 的值为\_\_\_。（’65,0x41,0o101’）   
127、 表达式 isinstance(‘abcdefg’, str) 的值为\_\_。（True）   
128、 表达式 isinstance(‘abcdefg’, object) 的值为\_\_\_。（True）   
129、 表达式 isinstance(3, object) 的值为\_\_\_。（True）   
130、 表达式 ‘abcabcabc’.rindex(‘abc’) 的值为\_\_。（6）   
131、 表达式 ‘:’.join(‘abcdefg’.split(‘cd’)) 的值为\_\_\_\_。（’ab:efg’）   
132、 表达式 ‘Hello world. I like Python.’.rfind(‘python’) 的值为\_\_。（-1）   
133、 表达式 ‘abcabcabc’.count(‘abc’) 的值为\_\_\_。（3）   
134、 表达式 ‘apple.peach,banana,pear’.find(‘p’) 的值为\_\_\_\_。（1）   
135、 表达式 ‘apple.peach,banana,pear’.find(‘ppp’) 的值为\_\_。（-1）   
136、 表达式 ‘abcdefg’.split(‘d’) 的值为\_\_\_\_\_\_\_\_。（[‘abc’, ‘efg’]）   
137、 表达式 ‘:’.join(‘1,2,3,4,5’.split(‘,’)) 的值为\_\_\_\_\_\_\_\_。（’1:2:3:4:5’）   
138、 表达式 ‘,’.join(‘a b ccc\n\n\nddd ‘.split()) 的值为\_\_\_\_。（’a,b,ccc,ddd’）   
139、 表达式 ‘Hello world’.upper() 的值为\_。（’HELLO WORLD’）   
140、 表达式 ‘Hello world’.lower() 的值为\_\_\_。（’hello world’）   
141、 表达式 ‘Hello world’.lower().upper() 的值为\_。（’HELLO WORLD’）  
142、 表达式 ‘Hello world’.swapcase().swapcase() 的值为\_\_\_\_。（’Hello world’）   
143、 表达式 r’c:\windows\notepad.exe’.endswith(‘.exe’) 的值为\_\_\_。（True）   
144、 表达式 r’c:\windows\notepad.exe’.endswith((‘.jpg’, ‘.exe’)) 的值为\_。（True）   
145、 表达式 ‘C:\Windows\notepad.exe’.startswith(‘C:’) 的值为\_\_\_。（True）   
146、 表达式 len(‘Hello world!’.ljust(20)) 的值为\_\_\_。（20）   
147、 表达式 len(‘abcdefg’.ljust(3)) 的值为\_\_\_。（7）   
148、 表达式 len([i for i in range(10)]) 的值为\_\_\_\_。（10）   
149、 表达式 len(range(1,10)) 的值为\_\_\_。（9）   
150、 表达式 range(10)[-1] 的值为\_\_。（9）   
151、 表达式 range(10,20)[4] 的值为\_\_\_\_。（14）   
152、 表达式 round(3.4) 的值为\_。（3）   
153、 表达式 round(3.7) 的值为\_\_\_。（4）   
154、 表达式 ‘a’ + ‘b’ 的值为\_\_\_。（’ab’）   
155、 已知 x = ‘123’ 和 y = ‘456’，那么表达式 x + y 的值为\_\_\_\_。（’123456’）   
156、 表达式 ‘a’.join(‘abc’.partition(‘a’)) 的值为\_\_\_\_\_\_。（’aaabc’）   
157、 表达式 re.split(‘.+’, ‘alpha.beta…gamma..delta’) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。（[‘alpha’, ‘beta’, ‘gamma’, ‘delta’]）   
158、 已知 x = ‘a234b123c’，并且re模块已导入，则表达式 re.split(‘\d+’, x) 的值为\_\_\_\_\_\_\_。（[‘a’, ‘b’, ‘c’]）   
159、 表达式 ”.join(‘asdssfff’.split(‘sd’)) 的值为\_\_。（’assfff’）  
160、 表达式 ”.join(re.split(‘[sd]’,’asdssfff’)) 的值为\_\_\_\_\_。（’afff’）   
161、 假设re模块已导入，那么表达式 re.findall(‘(\d)\1+’, ‘33abcd112’) 的值为\_\_\_\_\_\_\_\_\_。（[‘3’, ‘1’]）   
162、 语句 print(re.match(‘abc’, ‘defg’)) 输出结果为\_\_\_。（None）   
163、 表达式 ‘Hello world!’[-4] 的值为\_\_\_\_\_\_。（’r’）   
164、 表达式 ‘Hello world!’[-4:] 的值为\_\_\_\_\_\_。（’rld!’）   
165、 表达式 ‘test.py’.endswith((‘.py’, ‘.pyw’)) 的值为\_\_\_\_。（True）   
166、 已知 x = (3)， 那么表达式 x \* 3 的值为\_\_\_\_。（9）   
167、 已知 x = (3,)，那么表达式 x \* 3 的值为\_\_\_。（(3, 3, 3)）   
168、 表达式 len(‘abc’.ljust(20)) 的值为\_\_\_\_。（20）   
169、 代码 print(re.match(‘^[a-zA-Z]+$’,’abcDEFG000’)) 的输出结果为\_\_\_。（None）   
170、 当在字符串前加上小写字母或大写字母表示原始字符串，不对其中的任何字符进行转义。（r、R）   
171、 在设计正则表达式时，字符\_紧随任何其他限定符(\*、+、?、{n}、{n,}、{n,m})之后时，匹配模式是“非贪心的”，匹配搜索到的、尽可能短的字符串。（?）   
172、 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\_\_\_\_\_\_\_\_\_\_\_。（’a1bbbb1c1d1e’）   
173、 假设列表对象x = [1, 1, 1]，那么表达式id(x[0]) == id(x[2])的值为\_\_\_。（True）   
174、 已知列表 x = list(range(10))，那么执行语句 del x[::2]之后，x的值为\_\_\_。（[1, 3, 5, 7, 9]）   
175、 已知列表 x = [1, 2, 3, 4]，那么执行语句 del x[1] 之后x的值为\_\_。（[1, 3, 4]）   
176、 表达式 [1] \* 2 的值为\_\_\_\_\_\_\_。（[1, 1]）   
177、 表达式 [1, 2] \* 2 的值为\_\_\_\_。（[1, 2, 1, 2]）   
178、 已知列表 x = [1, 2, 3]，那么执行语句 x.insert(1, 4) 只有，x的值为\_。（[1, 4, 2, 3]）   
179、 已知列表 x = [1, 2, 3]，那么执行语句 x.insert(0, 4) 只有，x的值为\_。（[4, 1, 2, 3]）   
180、 已知列表 x = [1, 2, 3]，那么执行语句 x.pop(0) 之后，x的值为\_\_\_。（[2, 3]）   
181、 已知 x = [[1]] \* 3，那么执行语句 x[0][0] = 5之后，变量x的值为\_\_\_\_\_\_。（[[5], [5], [5]]）   
182、 表达式 list(map(lambda x: x+5, [1, 2, 3, 4, 5])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_。（[6, 7, 8, 9, 10]）   
183、 表达式 {1, 2, 3, 4, 5} ^ {4, 5, 6, 7} 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 6, 7}）   
184、 表达式 5 if 5>6 else (6 if 3>2 else 5) 的值为\_\_\_。（6）   
185、 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 2, 4, 5, 6]）   
186、 表达式 len(range(1, 10)) 的值为\_。（9）   
187、 表达式 len(‘中国’.encode(‘utf-8’)) 的值为\_。（6）   
188、 表达式 len(‘中国’.encode(‘gbk’)) 的值为\_\_。（4）   
189、 表达式 chr(ord(‘A’)+2) 的值为\_\_\_\_。（’C’）   
190、 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\_。（False）   
191、 表达式 sorted([13, 1, 237, 89, 100], key=lambda x: len(str(x))) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 13, 89, 237, 100]）   
192、 Python中定义函数的关键字是\_\_\_\_\_\_\_。（def）   
193、 在函数内部可以通过关键字\_\_\_\_\_\_来定义全局变量。（global）   
194、 如果函数中没有return语句或者return语句不带任何返回值，那么该函数的返回值为\_\_\_\_\_\_\_。（None）   
195、 表达式 sum(range(10)) 的值为\_\_\_\_\_\_。（45）   
196、 表达式 sum(range(1, 10, 2)) 的值为\_\_。（25）   
197、 表达式 ‘abcab’.replace(‘a’,’yy’) 的值为\_。（’yybcyyb’）   
198、 已知 table = ”.maketrans(‘abcw’, ‘xyzc’)，那么表达式 ‘Hellow world’.translate(table) 的值为\_\_\_\_\_\_\_\_\_\_\_\_。（’Helloc corld’）   
199、 表达式 ‘hello world, hellow every one’.replace(‘hello’, ‘hi’) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（’hi world, hiw every one’）   
200、 已知字符串 x = ‘hello world’，那么执行语句 x.replace(‘hello’, ‘hi’) 之后，x的值为\_\_。（’hello world’）   
201、 正则表达式元字符\_\_用来表示该符号前面的字符或子模式1次或多次出现。（+）   
202、 已知 x = ‘a b c d’，那么表达式 ‘,’.join(x.split()) 的值为\_。（’a,b,c,d’）   
203、 正则表达式元字符\_\_用来表示该符号前面的字符或子模式0次或多次出现。（\*）   
204、 表达式 ‘abcab’.strip(‘ab’) 的值为\_\_\_\_。（’c’）   
205、 表达式 [str(i) for i in range(3)] 的值为\_\_。（[‘0’, ‘1’, ‘2’]）   
206、 表达式 ‘abc.txt’.endswith((‘.txt’, ‘.doc’, ‘.jpg’)) 的值为\_。（True）   
207、 表达式 list(filter(None, [0,1,2,3,0,0])) 的值为\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）   
208、 表达式 list(filter(lambda x:x>2, [0,1,2,3,0,0])) 的值为\_\_\_。（[3]）  
209、 表达式 list(range(50, 60, 3)) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。（[50, 53, 56, 59]）   
210、 表达式 list(filter(lambda x: x%2==0, range(10))) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 2, 4, 6, 8]）   
211、 表达式 list(filter(lambda x: len(x)>3, [‘a’, ‘b’, ‘abcd’])) 的值为\_。（[‘abcd’]）   
212、 Python使用\_\_\_\_\_\_\_关键字来定义类。（class）   
213、 表达式 isinstance(‘abc’, str) 的值为\_\_\_\_\_\_。（True）   
214、 表达式 isinstance(‘abc’, int) 的值为\_\_\_\_\_。（False）   
215、 表达式 isinstance(4j, (int, float, complex)) 的值为\_\_\_。（True）   
216、 表达式 isinstance(‘4’, (int, float, complex)) 的值为\_\_\_。（False）   
217、 表达式 type(3) in (int, float, complex) 的值为\_\_。（True）   
218、 表达式 type(3.0) in (int, float, complex) 的值为\_\_。（True）   
219、 表达式 type(3+4j) in (int, float, complex) 的值为\_\_。（True）   
220、 表达式 type(‘3’) in (int, float, complex) 的值为\_\_。（False）   
221、 表达式 type(3) == int 的值为\_\_\_\_。（True）   
222、 代码 print(1,2,3,sep=’:’) 的执行结果为\_\_\_\_\_\_\_\_。（1:2:3）   
223、 代码 for i in range(3):print(i, end=’,’) 的执行结果为\_\_\_\_\_\_\_。（0,1,2,）   
224、 表达式 eval(”’import(‘math’).sqrt(9)”’) 的值为\_\_\_\_。（3.0）   
225、 表达式 eval(”’import(‘math’).sqrt(3\*\*2+4\*\*2)”’) 的值为\_\_\_。（5.0）   
226、 表达式 eval(‘3+5’) 的值为\_\_\_\_\_\_\_。（8）   
227、 表达式 eval(‘[1, 2, 3]’) 的值为\_\_\_\_\_\_\_\_。（[1, 2, 3]）   
228、 假设math标准库已导入，那么表达式 eval(‘math.sqrt(4)’) 的值为\_\_\_。（2.0）   
229、 已知x为非空列表，那么表达式 random.choice(x) in x 的值为\_。（True）  
230、 表达式 ‘abc10’.isalnum() 的值为\_\_\_\_。（True）   
231、 表达式 ‘abc10’.isalpha() 的值为\_\_\_\_\_\_。（False）   
232、 表达式 ‘abc10’.isdigit() 的值为\_\_\_\_\_\_\_\_。（False）   
233、 表达式 [1,2,3].count(4) 的值为\_\_\_\_\_。（0）   
234、 Python标准库random中的\_方法作用是从序列中随机选择1个元素。（choice()）   
235、 表达式 ‘C:\windows\notepad.exe’.endswith(‘.exe’) 的值为\_\_。（True）   
236、 Python标准库random中的sample(seq, k)方法作用是从序列中选择\_\_(重复？不重复？)的k个元素。（不重复）   
237、 random模块中\_\_\_方法的作用是将列表中的元素随机乱序。（shuffle()）   
238、 Python关键字elif表示\_\_和\_\_\_两个单词的缩写。（else、if）   
239、 执行代码 x, y, z = sorted([1, 3, 2]) 之后，变量y的值为\_\_。（2）   
240、 已知 x = {1:2, 2:3}，那么表达式 x.get(3, 4) 的值为\_\_。（4）   
241、 已知 x = {1:2, 2:3}，那么表达式 x.get(2, 4) 的值为\_\_。（3）   
242、 表达式 {1, 2, 3} | {3, 4, 5} 的值为\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 4, 5}）  
243、 表达式 {1, 2, 3} | {2, 3, 4} 的值为\_\_\_\_\_\_\_\_\_\_\_\_。（{1, 2, 3, 4}）   
244、 表达式 {1, 2, 3} & {3, 4, 5} 的值为\_\_\_\_。（{3}）   
245、 表达式 {1, 2, 3} & {2, 3, 4} 的值为\_。（{2, 3}）   
246、 表达式 {1, 2, 3} - {3, 4, 5} 的值为\_\_\_。（{1, 2}）   
247、 表达式 {1, 2, 3} < {3, 4, 5} 的值为\_\_\_。（False）   
248、 表达式 {1, 2, 3} < {1, 2, 4} 的值为\_。（False）   
249、 表达式 ‘%s’%[1,2,3] 的值为\_\_\_。（’[1, 2, 3]’）   
250、 在Python定义类时，与运算符“\*\*”对应的特殊方法名为 \_\_\_。（pow()）   
251、 在Python中定义类时，与运算符“//”对应的特殊方法名为\_\_\_\_\_。（floordiv()）   
252、 对文件进行写入操作之后，\_\_\_\_\_方法用来在不关闭文件对象的情况下将缓冲区内容写入文件。（flush()）   
253、 Python内置函数\_\_\_用来打开或创建文件并返回文件对象。（open()）   
254、 使用上下文管理关键字\_\_\_\_可以自动管理文件对象，不论何种原因结束该关键字中的语句块，都能保证文件被正确关闭。（with）   
255、 Python标准库os中用来列出指定文件夹中的文件和子文件夹列表的方式是\_\_。（listdir()）   
256、 Python标准库os.path中用来判断指定文件是否存在的方法是\_\_\_\_。（exists()）   
257、 Python标准库os.path中用来判断指定路径是否为文件的方法是\_\_\_\_\_。（isfile()）   
258、 Python标准库os.path中用来判断指定路径是否为文件夹的方法是\_\_\_\_。（isdir()）   
259、 Python标准库os.path中用来分割指定路径中的文件扩展名的方法是\_\_\_\_。（splitext()）   
260、 Python内建异常类的基类是\_\_\_\_\_\_。（BaseException）   
261、 Python扩展库\_\_\_支持Excel 2007或更高版本文件的读写操作。（openpyxl）  
262、 Python标准库\_\_中提供了计算MD5摘要的方法md5()。（hashlib）   
263、 表达式 len(‘SDIBT’) 的值为\_\_\_\_。（5）   
264、 表达式 ‘Hello world!’.count(‘l’) 的值为\_。（3）   
265、 表达式 (1, 2, 3)+(4, 5) 的值为\_\_\_\_\_\_\_\_。（(1, 2, 3, 4, 5)）   
266、 表达式 dict(zip([1, 2], [3, 4])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（{1: 3, 2: 4}）   
267、 已知 x = ‘abcdefg’，则表达式 x[3:] + x[:3] 的值为\_\_\_\_\_\_\_\_\_。（’defgabc’）   
268、 一直 g = lambda x, y=3, z=5: x\*y\*z，则语句 print(g(1)) 的输出结果为\_。（15）   
269、 表达式 list(map(lambda x: len(x), [‘a’, ‘bb’, ‘ccc’])) 的值为\_\_\_\_。（[1, 2, 3]）   
270、 语句 x, y, z = [1, 2, 3] 执行后，变量y的值为\_\_\_\_。（2）   
271、 Python标准库\_\_\_\_对Socket进行了二次封装，支持Socket接口的访问，大幅度简化了网络程序的开发。（socket）   
272、 Python扩展库\_\_\_\_中封装了Windows底层几乎所有API函数。（pywin32）   
273、 线程对象的\_方法用来阻塞当前线程，指定线程运行结束或超时后继续运行当前线程。（join()）   
274、 Python用来访问和操作内置数据库SQLite的标准库是\_\_\_。（sqlite3）   
275、 用于删除数据库表test中所有name字段值为’10001’的记录的SQL语句为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（delete from test where name=’10001’）   
276、 Python扩展库\_\_\_\_\_\_完美封装了图形库OpenGL的功能。（pyopengl）   
277、 Python扩展库\_\_\_\_和\_\_\_\_提供了图像处理功能。（PIL、pillow）   
278、 已知 x = [[1,3,3], [2,3,1]]，那么表达式 sorted(x, key=lambda item:item[0]+item[2]) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[2, 3, 1], [1, 3, 3]]）   
279、 已知 x = [[1,3,3], [2,3,1]]，那么表达式 sorted(x, key=lambda item:(item[1],item[2])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[2, 3, 1], [1, 3, 3]]）   
280、 已知 x = [[1,3,3], [2,3,1]]，那么表达式 sorted(x, key=lambda item:(item[1], -item[2])) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 3, 3], [2, 3, 1]]）   
281、 已知 x = {1, 2, 3}，那么执行语句 x.add(3) 之后，x的值为\_\_\_\_。（{1, 2, 3}）   
282、 已知 x = {1:1}，那么执行语句 x[2] = 2之后，len(x)的值为\_\_。（2）   
283、 已知 x = {1:1, 2:2}，那么执行语句 x[2] = 4之后，len(x)的值为\_\_。（2）   
284、 假设已从标准库functools导入reduce()函数，那么表达式 reduce(lambda x, y: x-y, [1, 2, 3]) 的值为\_\_。（-4）   
285、 假设已从标准库functools导入reduce()函数，那么表达式 reduce(lambda x, y: x+y, [1, 2, 3]) 的值为\_\_。（6）   
286、 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\_\_\_\_。（6、10）   
287、 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\_\_。（[1, 2, 3]）   
288、 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\_\_。（[1, 2]）   
289、 已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 id(x) == id(y) 的值为\_。（False）   
290、 已知列表 x = [1, 2]，执行语句 y = x 后，表达式 id(x) == id(y) 的值为\_。（True）   
291、 已知列表 x = [1, 2]，执行语句 y = x 后，表达式 x is y 的值为\_。（True）   
292、 已知列表 x = [1, 2]，执行语句 y = x[:] 后，表达式 x is not y 的值为\_。（True）   
293、 表达式 sorted(random.sample(range(5), 5)) 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[0, 1, 2, 3, 4]）   
294、 表达式 [i for i in range(10) if i>8] 的值为\_\_\_\_\_\_。（[9]）   
295、 已知有列表 x = [[1, 2, 3], [4, 5, 6]]，那么表达式 [[row[i] for row in x] for i in range(len(x[0]))] 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[[1, 4], [2, 5], [3, 6]]）   
296、 执行语句 x,y,z = map(str, range(3)) 之后，变量y的值为\_。（’1’）   
297、 已知列表 x = [1, 2]，那么执行语句 x.extend([3]) 之后， x的值为\_\_。（[1, 2, 3]）   
298、 已知列表 x = [1, 2]，那么执行语句 x.append([3]) 之后，x的值为\_\_\_。（[1, 2, [3]]）   
299、 表达式 ‘aaasdf’.lstrip(‘as’) 的值为\_\_\_\_\_\_。（’df’）   
300、 表达式 ‘aaasdf’.lstrip(‘af’) 的值为\_\_\_\_\_\_。（’sdf’）   
301、 表达式 ‘aaasdf’.strip(‘af’) 的值为\_\_\_\_。（’sd’）   
302、 表达式 ‘aaasdf’.rstrip(‘af’) 的值为\_\_\_\_\_。（’aaasd’）   
303、 已知 f = lambda x: x+5，那么表达式 f(3) 的值为\_\_。（8）   
304、 表达式 print(0b10101) 的值为\_\_。（21）   
305、 表达式 ‘\x41’ == ‘A’ 的值为\_\_\_。（True）   
306、 已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[:3] 之后，x的值为\_\_\_\_。（[4, 5]）   
307、 表达式 sorted([‘abc’, ‘acd’, ‘ade’], key=lambda x:(x[0],x[2])) 的值为\_\_\_。（[‘abc’, ‘acd’, ‘ade’]）   
308、 已知 x = range(1,4) 和 y = range(4,7)，那么表达式 sum([i\*j for i,j in zip(x,y)]) 的值为\_\_\_\_。（32）   
309、 表达式 [5 for i in range(3)] 的值为\_\_\_\_\_。（[5, 5, 5]）   
310、 表达式 {1, 2, 3} == {1, 3, 2} 的值为\_。（True）   
311、 表达式 [1, 2, 3] == [1, 3, 2] 的值为\_\_。（False）   
312、 已知 x = [1, 2, 1]，那么表达式 id(x[0]) == id(x[2]) 的值为\_\_\_\_\_。（True）   
313、 表达式 3 not in [1, 2, 3]的值为\_\_\_\_。（False）   
314、 已知 x = [1, 2]，那么执行语句 x[0:0] = [3, 3]之后，x的值为\_。（[3, 3, 1, 2]）   
315、 已知 x = [1, 2]，那么执行语句 x[0:1] = [3, 3]之后，x的值为\_。（[3, 3, 2]）   
316、 已知 x = [1, 2, 3, 4, 5]，那么执行语句 del x[1:3] 之后，x的值为\_\_。（[1, 4, 5]）   
317、 已知 x = [[1, 2, 3,], [4, 5, 6]]，那么表达式 sum([i\*j for i,j in zip(\*x)]) 的值为\_\_。（32）   
318、 已知列表 x = [1, 2, 3] 和 y = [4, 5, 6]，那么表达式 [(i,j) for i, j in zip(x,y) if i==3] 的值为\_\_。（[(3, 6)]）   
319、 已知列表 x = [1.0, 2.0, 3.0]，那么表达式 sum(x)/len(x) 的值为\_。（2.0）   
320、 表达式 ‘abc’ in (‘abcdefg’) 的值为\_\_\_\_。（True）   
321、 表达式 ‘abc’ in [‘abcdefg’] 的值为\_\_\_\_。（False）   
322、 已知 x = {1:2, 2:3, 3:4}，那么表达式 sum(x) 的值为\_。（6）   
323、 已知 x = {1:2, 2:3, 3:4}，那么表达式 sum(x.values()) 的值为\_\_\_。（9）   
324、 已知 x = [3, 2, 3, 3, 4]，那么表达式 [index for index, value in enumerate(x) if value==3] 的值为\_\_。（[0, 2, 3]）   
325、 表达式 1234%1000//100 的值为\_。（2）   
326、 正则表达式模块re的\_\_\_\_方法用来编译正则表达式对象。（compile()）   
327、 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。（match()）   
328、 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。（search()）   
329、 表达式 re.search(r’\w\*?(?P\b\w+\b)\s+(?P=f)\w\*?’, ‘Beautiful is is better than ugly.’).group(0) 的值为\_。（’is is’）   
330、 已知 g = lambda x, y=3, z=5: x+y+z，那么表达式 g(2) 的值为\_\_。（10）   
331、 假设有Python程序文件abc.py，其中只有一条语句print(name)，那么直接运行该程序时得到的结果为\_\_\_。（main）   
332、 表达式 3 in {1, 2, 3} 的值为\_\_\_。（True）   
333、 表达式 ‘ac’ in ‘abce’ 的值为\_\_\_。（False）   
334、 表达式 not 3 的值为\_\_\_\_\_\_。（False）   
335、 表达式 3 // 5 的值为\_\_\_\_\_。（0）   
336、 表达式 [1, 2] + [3] 的值为\_\_\_\_\_\_\_\_。（[1, 2, 3]）   
337、 表达式 (1,) + (2,) 的值为\_\_\_。（(1, 2)）   
338、 表达式 (1) + (2) 的值为\_\_。（3）   
339、 已知 x, y = map(int, [‘1’, ‘2’])，那么表达式 x + y 的值为\_。（3）   
340、 已知列表 x = list(range(5))，那么执行语句 x.remove(3) 之后，表达式 x.index(4) 的值为\_\_。（3）   
341、 已知列表 x = [1, 3, 2]，那么执行语句 x.reverse() 之后，x的值为\_\_。（[2, 3, 1]）   
342、 已知列表 x = [1, 3, 2]，那么执行语句 x = x.reverse() 之后，x的值为\_\_。（None）   
343、 已知x为非空列表，那么表达式 x.reverse() == list(reversed(x)) 的值为\_\_\_\_。（False）   
344、 已知x为非空列表，那么表达式 x.sort() == sorted(x) 的值为\_\_\_\_。（False）   
345、 已知列表 x = [1, 3, 2]，那么执行语句 y = list(reversed(x)) 之后，x的值为\_\_\_\_。（[1, 3, 2]）   
346、 已知列表 x = [1, 3, 2]，那么执行语句 y = list(reversed(x)) 之后，y的值为\_\_\_\_。（[2, 3, 1]）   
347、 表达式 ‘Beautiful is better than ugly.’.startswith(‘Be’, 5) 的值为\_\_\_。（False）   
348、 已知列表x中包含超过5个以上的元素，那么表达式 x == x[:5]+x[5:] 的值为\_\_。（True）   
349、 已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 sum(x) 的值为\_\_。（3）   
350、 已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 ”.join(x.values()) 的值为\_\_。（’345’）   
351、 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\_。（3）   
352、 已知字典 x = {i:str(i+3) for i in range(3)}，那么表达式 ”.join([item[1] for item in x.items()]) 的值为\_\_\_。（’345’）   
353、 已知列表 x = [1, 3, 2]，那么表达式 [value for index, value in enumerate(x) if index==2] 的值为\_\_\_\_\_。（[2]）   
354、 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\_\_。（2）   
355、 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\_\_。（’3’）   
356、 表达式 set([1,2,3]) == {1, 2, 3} 的值为\_\_。（True）   
357、 表达式 set([1,2, 2,3]) == {1, 2, 3} 的值为\_\_。（True）   
358、 表达式 ‘%c’%65 == str(65) 的值为\_。（False）   
359、 表达式 ‘%s’%65 == str(65) 的值为\_\_\_。（True）   
360、 表达式 chr(ord(‘b’)^32) 的值为\_\_\_。（’B’）   
361、 表达式 ‘abc’ in ‘abdcefg’ 的值为\_\_\_。（False）   
362、 已知函数定义 def func(\*p):return sum(p)，那么表达式 func(1,2,3) 的值为\_\_。（6）   
363、 已知函数定义 def func(\*p):return sum(p)，那么表达式 func(1,2,3, 4) 的值为\_\_。（10）   
364、 已知函数定义 def func(\*\*p):return sum(p.values())，那么表达式 func(x=1, y=2, z=3) 的值为\_\_。（6）   
365、 已知函数定义 def func(\*\*p):return ”.join(sorted(p))，那么表达式 func(x=1, y=2, z=3)的值为\_\_\_\_。（’xyz’）   
366、 已知x为整数变量，那么表达式 int(hex(x), 16) == x 的值为\_\_\_。（True）   
367、 已知 f = lambda x: 5，那么表达式 f(3)的值为\_\_\_。（5）   
368、 已知 x, y = 3, 5，那么执行x, y = y, x 之后，x的值为\_\_\_\_。（5）   
369、 已知 x = ‘abcd’ 和 y = ‘abcde’，那么表达式 [i==j for i,j in zip(x,y)] 的值为\_\_\_\_\_\_。（[True, True, True, True]）   
370、 表达式16\*\*0.5的值为\_\_\_\_\_\_\_\_\_\_。（4.0）   
371、 表达式type({3})的值为\_\_\_\_\_\_\_\_\_\_。（set）   
372、 表达式isinstance(‘Hello world’, str)的值为\_\_\_\_\_\_\_\_\_\_。（True）   
373、 已知x = list(range(20))，那么表达式x[-1]的值为\_\_\_\_\_\_\_\_\_\_。（19）   
374、 已知x = 3+4j和y = 5+6j，那么表达式x+y的值为\_\_\_\_\_\_\_\_\_\_。（8+10j）   
375、 已知x = [3]，那么执行x += [5]之后x的值为\_\_\_\_\_\_\_\_\_\_。（[3, 5]）   
376、 已知x = [3, 3, 4]，那么表达式id(x[0])==id(x[1])的值为\_\_\_\_\_\_\_\_。（True）   
377、 表达式int(‘11’, 2)的值为\_\_\_\_\_\_\_\_\_\_\_\_。（3）   
378、 表达式int(‘11’, 8)的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。（9）   
379、 表达式int(bin(54321), 2)的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。（54321）   
380、 表达式chr(ord(‘A’)+1)的值为\_\_\_\_\_\_\_\_\_\_。（’B’）   
381、 表达式int(str(34)) == 34的值为\_\_\_\_\_\_\_\_\_。（True）   
382、 表达式list(str([3, 4])) == [3, 4]的值为\_\_\_\_\_\_\_。（False）   
383、 表达式{1, 2, 3, 4, 5, 6} ^ {5, 6, 7, 8}的值为\_\_\_\_\_\_\_\_。（{1, 2, 3, 4, 7, 8}）   
384、 表达式15 // 4的值为\_\_\_\_\_\_\_\_。（3）   
385、 表达式sorted({‘a’:3, ‘b’:9, ‘c’:78})的值为\_\_\_\_\_\_\_。（[‘a’, ‘b’, ‘c’]）   
386、 表达式sorted({‘a’:3, ‘b’:9, ‘c’:78}.values())的值为\_\_\_。（[3, 9, 78]）   
387、 已知x = [3, 2, 4, 1]，那么执行语句x = x.sort()之后，x的值为\_\_。（None）   
388、 表达式list(filter(lambda x: x>5, range(10)))的值为\_\_\_\_\_\_\_。（[6, 7, 8, 9]）   
389、 已知x = list(range(20))，那么语句print(x[100:200])的输出结果为\_\_\_\_\_。（[]）   
390、 已知x = list(range(20))，那么执行语句x[:18] = []后列表x的值为\_\_\_\_。（[18, 19]）   
391、 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\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）   
392、 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\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3, 4]）   
393、 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\_\_\_\_\_\_\_\_\_\_。（[1, 2, 3]）   
394、 已知x = [[]] \* 3，那么执行语句x[0].append(1)之后，x的值为\_\_\_\_\_\_\_\_\_\_。（[[1], [1], [1]]）   
395、 已知x = [[] for i in range(3)]，那么执行语句x[0].append(1)之后，x的值为\_\_\_\_\_\_\_。（[[1], [], []]）   
396、 已知x = ([1], [2])，那么执行语句x[0].append(3)后x的值为\_\_\_\_\_\_。（([1, 3], [2])）   
397、 已知x = {1:1, 2:2}，那么执行语句x.update({2:3, 3:3})之后，表达式sorted(x.items())的值为\_\_\_\_\_\_\_\_\_\_。（[(1, 1), (2, 3), (3, 3)]）   
398、 已知x = {1:1, 2:2}，那么执行语句x[3] = 3之后，表达式sorted(x.items())的值为\_\_\_\_\_\_\_\_\_\_。（[(1, 1), (2, 2), (3, 3)]）   
399、 表达式type({}) == dict的值为\_\_\_\_\_\_\_。（True）   
400、 表达式type({}) == set的值为\_\_\_\_\_。（False）   
401、 已知x = [1, 2, 3]，那么表达式not (set(x\*100)-set(x))的值为\_\_\_\_\_\_。（True）   
402、 已知x = [1, 2, 3]，那么表达式not (set(x\*100)&set(x))的值为\_\_\_\_\_\_。（False）   
403、 表达式{‘x’: 1, \*\*{‘y’: 2}}的值为\_\_\_\_\_\_\_\_\_。（{‘x’: 1, ‘y’: 2}）   
404、 表达式{range(4), 4, (5, 6, 7)}的值为\_\_\_\_\_\_\_\_。（{0, 1, 2, 3, 4, 5, 6, 7}）   
405、 在Python中，不论类的名字是什么，构造方法的名字都是\_\_\_\_\_\_。（init）   
406、 如果在设计一个类时实现了contains ()方法，那么该类的对象会自动支持\_\_\_运算符。（in）   
407、 已知函数定义def demo(x, y, op):return eval(str(x)+op+str(y))，那么表达式demo(3, 5, ‘+’)的值为\_\_\_\_\_\_\_\_\_\_\_\_。（8）   
408、 已知函数定义def demo(x, y, op):return eval(str(x)+op+str(y))，那么表达式demo(3, 5, ‘\*’)的值为\_\_\_\_\_\_\_\_\_\_\_\_。（15）   
409、 已知函数定义def demo(x, y, op):return eval(str(x)+op+str(y))，那么表达式demo(3, 5, ‘-‘)的值为\_\_\_\_\_\_\_\_\_\_\_\_。（-2）   
410、 字符串编码格式UTF8使用\_\_\_\_\_\_\_\_\_\_个字节表示一个汉字。（3）   
411、 字符串编码格式GBK使用\_\_\_\_\_\_\_\_个字节表示一个汉字。（2）   
412、 已知字符串编码格式utf8使用3个字节表示一个汉字、1个字节表示英语字母，那么表达式len(‘abc你好’)的值为\_\_\_\_\_。（5）   
413、 已知字符串编码格式utf8使用3个字节表示一个汉字、1个字节表示英语字母，那么表达式len(‘abc你好’.encode())的值为\_\_\_\_\_。（9）   
414、 已知字符串编码格式gbk使用2个字节表示一个汉字、1个字节表示英语字母，那么表达式len(‘abc你好’.encode(‘gbk’))的值为\_\_\_\_\_。（7）   
415、 已知ord(‘A’)的值为65并且hex(65)的值为’0x41’，那么表达式’\x41b’的值为\_\_\_\_\_\_。（’Ab’）   
416、 已知formatter = ‘good {0}’.format，那么表达式list(map(formatter, [‘morning’]))的值为\_\_\_\_\_\_\_\_。（[‘good morning’]）   
417、 已知x = ‘hello world.’，那么表达式x.find(‘x’)和x.rfind(‘x’)的值都为\_\_\_。（-1）   
418、 表达式’:’.join(‘hello world.’.split())的值为\_\_\_\_\_\_\_\_\_。（’hello:world.’）   
419、 表达式’:’.join(‘a b c d’.split(maxsplit=2))的值为\_\_\_\_\_\_\_。（’a:b:c d’）   
420、 已知x = ‘hello world’，那么表达式x.replace(‘l’, ‘g’)的值为\_\_\_\_\_。（’heggo worgd’）   
421、 假设已成功导入Python标准库string，那么表达式len(string.digits)的值为\_\_\_\_\_。（10）   
422、 表达式’aaaassddf’.strip(‘af’)的值为\_\_\_\_\_\_\_\_。（’ssdd’）   
423、 表达式len(‘aaaassddf’.strip(‘afds’))的值为\_\_\_\_\_\_。（0）   
424、 表达式len(‘hello world’[100:])的值为\_\_\_\_\_\_\_\_。（0）   
425、 表达式chr(ord(‘a’)^32^32)的值为\_\_\_\_\_\_\_。（’a’）   
426、 表达式chr(ord(‘a’)^32)的值为\_\_\_\_\_\_\_。（’A’）   
427、 已知x = ‘aa b ccc dddd’，那么表达式”.join([v for i,v in enumerate(x[:-1]) if v==x[i+1]])的值为\_\_\_\_\_\_\_。（’accddd’）   
428、 已知当前文件夹中有纯英文文本文件readme.txt，请填空完成功能把readme.txt文件中的所有内容复制到dst.txt中，with open(‘readme.txt’) as src, open(‘dst.txt’, \_\_) as dst:dst.write(src.read())。（’w’）   
429、 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\_\_\_\_\_\_\_\_\_\_\_。（’1234’）   
430、 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\_\_\_\_\_\_\_\_\_\_\_。（[‘1’, ‘2’, ‘3’, ‘4’]）   
431、 假设正则表达式模块re已正确导入，那么表达式re.sub(‘(.\s)\1+’, ‘\1’,’a a a a a bb’)的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（’a bb’）   
432、 Python标准库\_\_\_\_\_\_\_\_\_\_\_提供了对SQLite数据库的访问接口。（sqlite3）