《Python程序设计》填空题1-280题

2017-03-23 董付国 [Python小屋](https://mp.weixin.qq.com/s?__biz=MzI4MzM2MDgyMQ==&mid=100000021&idx=1&sn=7b07c818ed8f673f856bd933bdf5c69a&chksm=6b8aaa4f5cfd235936ad60fc0b8b624331261dba8f863522099cb99d782048eb2d7df57133bb&scene=18&key=4795db0e4d196de292d986044e5a955c2b81dfaf9e5a75871f6b0236bd9d3e398efbea52e7ac7266fdda9320541c1a10758bb3284eba6e2bd7d746fa5a80b4414eb3ef6d2bceb059173e94153f0611dc&ascene=1&uin=NjA1NDc1MjQ0&devicetype=Windows+8&version=6204014f&pass_ticket=jFSZYF3GKrLHG1fi2d1BZZBbXgoIp%2FaKywgj%2FkyYeBQEPT1m%2BNR%2BRfGiWbBFIPBA&winzoom=1##)

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）

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、查看变量类型的Python内置函数是\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（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））

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）

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）

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))）

32、Python语句list(range(1,10,3))执行结果为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。（[1, 4, 7]）

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）

36、表达式 int('123', 16) 的值为\_\_\_\_\_\_\_\_\_。（291）

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）

45、Python 3.x语句 print(1, 2, 3, sep=':') 的输出结果为\_\_\_\_\_\_\_\_\_\_\_\_。（1:2:3）

47、假设n为整数，那么表达式 n&1 == n%2 的值为\_\_\_\_\_\_\_\_\_\_\_\_\_。（True）

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()(）

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()）

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）

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）

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）

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]）

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）

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)'）

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）

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'）

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<f>\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'）

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'）

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]）

12、Python采用的是基于值得自动内存管理方式。（对）

14、Python不允许使用关键字作为变量名，允许使用内置函数名作为变量名，但这会改变函数名的含义。（对）

30、调用函数时，在实参前面加一个型号\*表示序列解包。（对）

31、放在一对三引号之间的内容将被认为是注释。（错）

32、Python支持使用字典的“键”作为下标来访问字典中的值。（对）

34、元组可以作为字典的“键”。（对）

35、字典的“键”必须是不可变的。（对）

36、尽管可以使用import语句一次导入任意多个标准库或扩展库，但是仍建议每次只导入一个标准库或扩展库。（对）

37、为了让代码更加紧凑，编写Python程序时应尽量避免加入空格和空行。（错）

38、在Python 3.5中运算符+不仅可以实现数值的相加、字符串连接，还可以实现列表、元组的合并和集合的并集运算。（错）

39、已知x为非空列表，那么表达式 sorted(x, reverse=True) == list(reversed(x)) 的值一定是True。（错）

40、已知x为非空列表，那么x.sort(reverse=True)和x.reverse()的作用是等价的。（错）

41、生成器推导式比列表推导式具有更高的效率，推荐使用。（对）

47、Python集合中的元素可以是列表。（错）

51、Python列表、元组、字符串都属于有序序列。（对）

52、在Python 3.x中语句 print(\*[1,2,3]) 不能正确执行。（错）

53、已知A和B是两个集合，并且表达式A<B的值为False，那么表达式A>B的值一定为True。（错）

54、列表对象的append()方法属于原地操作，用于在列表尾部追加一个元素。（对）

55、对于列表而言，在尾部追加元素比在中间位置插入元素速度更快一些，尤其是对于包含大量元素的列表。（对）

56、假设有非空列表x，那么x.append(3)、x = x+[3]与x.insert(0,3)在执行时间上基本没有太大区别。（错）

57、使用Python列表的方法insert()为列表插入元素时会改变列表中插入位置之后元素的索引。（对）

58、假设x为列表对象，那么x.pop()和x.pop(-1)的作用是一样的。（对）

59、使用del命令或者列表对象的remove()方法删除列表中元素时会影响列表中部分元素的索引。（对）

65、如果仅仅是用于控制循环次数，那么使用for i in range(20)和for i in range(20, 40)的作用是等价的。（对）

66、使用列表对象的remove()方法可以删除列表中首次出现的指定元素，如果列中不存在要删除的指定元素则抛出异常。（对）

67、元组是不可变的，不支持列表对象的inset()、remove()等方法，也不支持del命令删除其中的元素，但可以使用del命令删除整个元组对象。（对）

70、元组的访问速度比列表要快一些，如果定义了一系列常量值，并且主要用途仅仅是对其进行遍历二不需要进行任何修改，建议使用元组而不使用列表。（对）

73、只能对列表进行切片操作，不能对元组和字符串进行切片操作。（错）

78、使用print()函数无法将信息写入文件。（错）

80、相同内容的字符串使用不同的编码格式进行编码得到的结果并不完全相同。（对）

81、删除列表中重复元素最简单的方法是将其转换为集合后再重新转换为列表。（对）

89、Python字符串方法replace()对字符串进行原地修改。（错）

93、列表对象的extend()方法属于原地操作，调用前后列表对象的地址不变。（对）

97、编写函数时，一般建议先对参数进行合法性检查，然后再编写正常的功能代码。（对）

100、定义Python函数时，如果函数中没有return语句，则默认返回空值None。（对）

101、如果在函数中有语句 return 3，那么该函数一定会返回整数3。（错）

104、不同作用域中的同名变量之间互相不影响，也就是说，在不同的作用域内可以定义同名的变量。（对）

107、在函数内部，既可以使用global来声明使用外部全局变量，也可以使用global直接定义全局变量。（对）

109、对于数字n，如果表达式 0 not in [n%d for d in range(2, n)] 的值为True则说明n是素数。（对）

111、在函数内部直接修改形参的值并不影响外部实参的值。（对）

112、在函数内部没有任何方法可以影响实参的值。（错）

113、调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值。（错）

114、创建只包含一个元素的元组时，必须在元素后面加一个逗号，例如(3,)。（对）

115、在同一个作用域内，局部变量会隐藏同名的全局变量。（对）

116、形参可以看做是函数内部的局部变量，函数运行结束之后形参就不可访问了。（对）

117、假设已导入random标准库，那么表达式 max([random.randint(1, 10) for i in range(10)]) 的值一定是10。（错）

118、Python标准库random的方法randint(m,n)用来生成一个[m,n]区间上的随机整数。（对）

119、Python中一切内容都可以称为对象。（对）

120、栈和队列的都具有先入后出的特点。（错）

121、在一个软件的设计与开发中，所有类名、函数名、变量名都应该遵循统一的风格和规范。（对）

122、定义类时所有实例方法的第一个参数用来表示对象本身，在类的外部通过对象名来调用实例方法时不需要为该参数传值。（对）

123、在面向对象程序设计中，函数和方法是完全一样的，都必须为所有参数进行传值。（错）

124、Python中没有严格意义上的私有成员。（对）

125、在Python中定义类时，运算符重载是通过重写特殊方法实现的。例如，在类中实现了\_\_mul\_\_()方法即可支持该类对象的\*\*运算符。（错）

126、在IDLE交互模式下，一个下划线“\_”表示解释器中最后一次显示的内容或最后一次语句正确执行的输出结果。（对）

127、对于Python类中的私有成员，可以通过“对象名.\_类名\_\_私有成员名”的方式来访问。（对）

128、运算符 / 在Python 2.x和Python 3.x中具有相同的功能。（错）

129、运算符“-”可以用于集合的差集运算。（对）

130、如果定义类时没有编写析构函数，Python将提供一个默认的析构函数进行必要的资源清理工作。（对）

131、已知seq为长度大于10的列表，并且已导入random模块，那么[random.choice(seq) for i in range(10)]和random.sample(seq,10)等价。（错）

132、在派生类中可以通过“基类名.方法名()”的方式来调用基类中的方法。（对）

133、Python支持多继承，如果父类中有相同的方法名，而在子类中调用时没有指定父类名，则Python解释器将从左向右按顺序进行搜索。（对）

134、对文件进行读写操作之后必须显式关闭文件以确保所有内容都得到保存。（对）

137、在try...except...else结构中，如果try块的语句引发了异常则会执行else块中的代码。（错）

139、异常处理结构中的finally块中代码仍然有可能出错从而再次引发异常。（对）

142、在Python中定义类时实例方法的第一个参数名称必须是self。（错）

143、在Python中定义类时实例方法的第一个参数名称不管是什么，都表示对象自身。（对）

153、表达式 list('[1, 2, 3]') 的值是[1, 2, 3]。（错）

154、在函数内部没有任何声明的情况下直接为某个变量赋值，这个变量一定是函数内部的局部变量。（对）

159、二进制文件不能使用记事本程序打开。（错）

160、使用普通文本编辑器软件也可以正常查看二进制文件的内容。（错）

161、二进制文件也可以使用记事本或其他文本编辑器打开，但是一般来说无法正常查看其中的内容。（对）

168、带有else子句的异常处理结构，如果不发生异常则执行else子句中的代码。（对）

169、异常处理结构也不是万能的，处理异常的代码也有引发异常的可能。（对）

170、在异常处理结构中，不论是否发生异常，finally子句中的代码总是会执行的。（对）

171、在Python中定义函数时不需要声明函数参数的类型。（对）

172、在Python中定义函数时不需要声明函数的返回值类型。（对）

173、在函数中没有任何办法可以通过形参来影响实参的值。（错）

174、已知 x = 3，那么执行语句 x+=6 之后，x的内存地址不变。（错）

175、已知x为非空字符串，那么表达式 ''.join(x.split()) == x 的值一定为True。（错）

176、已知x为非空字符串，那么表达式 ','.join(x.split(',')) == x 的值一定为True。（对）

177、在Python中可以使用 for 作为变量名。（错）

178、在Python中可以使用 id 作为变量名，尽管不建议这样做。（对）

179、Python关键字不可以作为变量名。（对）

180、一个数字5也是合法的Python表达式。（对）

181、同一个列表对象中的元素类型可以各不相同。（对）

182、同一个列表对象中所有元素必须为相同类型。（错）

183、已知x为非空列表，那么执行语句x[0] = 3之后，列表对象x的内存地址不变。（对）

191、可以使用del删除集合中的部分元素。（错）

194、当作为条件表达式时，[]与None等价。（对）

195、表达式 [] == None 的值为True。（错）

196、当作为条件表达式时，{}与None等价。（对）

197、表达式 {}==None 的值为True。（错）

198、表达式 pow(3,2) == 3\*\*2 的值为True。（对）

199、当作为条件表达式时，空值、空字符串、空列表、空元组、空字典、空集合、空迭代对象以及任意形式的数字0都等价于False。（对）

200、在定义函数时，某个参数名字前面带有一个\*符号表示可变长度参数，可以接收任意多个普通实参并存放于一个元组之中。（对）

201、在定义函数时，某个参数名字前面带有两个\*符号表示可变长度参数，可以接收任意多个关键参数并将其存放于一个字典之中。（对）

202、定义函数时，带有默认值的参数必须出现在参数列表的最右端，任何一个带有默认值的参数右边不允许出现没有默认值的参数。（对）

203、在调用函数时，可以通过关键参数的形式进行传值，从而避免必须记住函数形参顺序的麻烦。（对）

204、在调用函数时，必须牢记函数形参顺序才能正确传值。（错）

205、调用函数时传递的实参个数必须与函数形参个数相等才行。（错）

206、正则表达式对象的match()方法可以在字符串的指定位置开始进行指定模式的匹配。（对）

207、使用正则表达式对字符串进行分割时，可以指定多个分隔符，而字符串对象的split()方法无法做到这一点。（对）

208、在编写函数时，建议首先对形参进行类型检查和数值范围检查之后再编写功能代码，或者使用异常处理结构，尽量避免代码抛出异常而导致程序崩溃。（对）

210、列表对象的pop()方法默认删除并返回最后一个元素，如果列表已空则抛出异常。（对）

211、在Python中定义类时，如果某个成员名称前有2个下划线则表示是私有成员。（对）

212、在类定义的外部没有任何办法可以访问对象的私有成员。（错）

213、可以使用py2exe或pyinstaller等扩展库把Python源程序打包成为exe文件，从而脱离Python环境在Windows平台上运行。（对）

216、一般来说，Python扩展库没有通用于所有版本Python的，安装时应选择与已安装Python的版本对应的扩展库。

218、Python变量名区分大小写，所以student和Student不是同一个变量。（对）

219、正则表达式元字符“^”一般用来表示从字符串开始处进行匹配，用在一对方括号中的时候则表示反向匹配，不匹配方括号中的字符。（对）

220、正则表达式元字符“\s”用来匹配任意空白字符。（对）

221、正则表达式 元字符“\d”用来匹配任意数字字符。（对）

222、lambda表达式中可以使用任意复杂的表达式，但是必须只编写一个表达式。（对）

223、Python类的构造函数是\_\_init\_\_()。（对）

224、定义类时，在一个方法前面使用@classmethod进行修饰，则该方法属于类方法。（对）

225、定义类时，在一个方法前面使用@staticmethod进行休息，则该方法属于静态方法。（对）

226、通过对象不能调用类方法和静态方法。（错）

227、在Python中可以为自定义类的对象动态增加新成员。（对）

228、Python类不支持多继承。（错）

229、属性可以像数据成员一样进行访问，但赋值时具有方法的优点，可以对新值进行检查。（对）

230、文件对象是可以迭代的。（对）

231、文件对象的tell()方法用来返回文件指针的当前位置。（对）

232、以写模式打开的文件无法进读操作。（对）

233、假设已成功导入os和sys标准库，那么表达式 os.path.dirname(sys.executable) 的值为Python安装目录。（对）

234、只可以动态为对象增加数据成员，而不能为对象动态增加成员方法。（错）

235、Python字典支持双向索引。（错）

236、Python集合支持双向索引。（错）

237、Python元组支持双向索引。（对）

238、假设re模块已成功导入，并且有 pattern = re.compile('^'+'\.'.join([r'\d{1,3}' for i in range(4)])+'$')，那么表达式 pattern.match('192.168.1.103') 的值为None。（错）

239、假设random模块已导入，那么表达式 random.sample(range(10), 20) 的作用是生成20个不重复的整数。（错）

240、假设random模块已导入，那么表达式 random.sample(range(10), 7) 的作用是生成7个不重复的整数。（对）