# 填空题

**Python题库**

1. Python安装扩展库常用的工具是\_\_\_\_\_\_\_\_\_\_和conda，其中后者需要安装Python集成开发环境Anaconda3之后才可以使用，而前者是Python官方推荐和标配的。

2.  Python程序文件扩展名主要有\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_两种，其中后者常用于GUI程序。  
3.  Python源代码程序伪编译后的文件扩展名为\_\_\_\_\_\_\_\_\_\_。  
4.  使用pip工具在线安装Excel文件操作扩展库openpyxl的完整命令是\_\_\_\_\_\_\_\_\_\_。  
5.  使用pip工具在线升级科学计算扩展库numpy的完整命令是\_\_\_\_\_\_\_\_\_\_。  
6.  使用pip工具把本机已安装的Python扩展库及版本信息输出到文本文件requirements.txt中的完整命令是\_\_\_\_\_\_\_\_\_\_。  
7.  使用pip工具查看当前已安装的Python扩展库（不含版本号）的完整命令是\_\_\_\_\_\_\_\_\_\_。

8.  表达式int('11111', 2)的值为\_\_\_\_\_\_\_\_\_\_。  
9.  表达式chr(ord('D')+2)的值为\_\_\_\_\_\_\_\_\_\_。  
10.  运算符%\_\_\_\_\_\_\_\_\_\_（可以/不可以）对浮点数进行求余数操作。  
11.  一个数字5\_\_\_\_\_\_\_\_\_\_（是/不是）合法的Python表达式。  
12.  在Python 3.x中，内置函数input()把用户的键盘输入一律作为字符串返回。  
13.  在Python中关键字\_\_\_\_\_\_\_\_\_\_表示空类型。  
14.  列表、元组、字符串是Python的\_\_\_\_\_\_\_\_\_\_(有序、无序？)序列。  
15.  查看变量类型的Python内置函数是\_\_\_\_\_\_\_\_\_\_。  
16.  查看对象内存地址的Python内置函数是\_\_\_\_\_\_\_\_\_\_。  
17.  以3为实部4为虚部，Python复数的表达形式为\_\_\_\_\_\_\_\_\_\_或\_\_\_\_\_\_\_\_\_\_。  
18.  Python运算符中用来计算整商的是\_\_\_\_\_\_\_\_\_\_。  
19.  Python运算符中用来计算集合并集的是\_\_\_\_\_\_\_\_\_\_。  
20.  Python运算符中用来计算集合差集的是\_\_\_\_\_\_\_\_\_\_。  
21.  Python运算符中用来计算集合交集的是\_\_\_\_\_\_\_\_\_\_。  
22.  使用运算符测试集合A是否为集合B的真子集的表达式可以写作\_\_\_\_\_\_\_\_\_\_。  
23.  \_\_\_\_\_\_\_\_\_\_命令既可以删除列表中的一个元素，也可以删除整个列表或其他任意类型的Python对象。  
24.  表达式int('123', 16)的值为\_\_\_\_\_\_\_\_\_\_。  
25.  表达式int('123', 8)的值为\_\_\_\_\_\_\_\_\_\_。  
26.  表达式int('123')的值为\_\_\_\_\_\_\_\_\_\_。  
27.  表达式int('101',2)的值为\_\_\_\_\_\_\_\_\_\_。  
28.  表达式abs(-3)的值为\_\_\_\_\_\_\_\_\_\_。  
29.  Python 3.x语句print(1, 2, 3, sep=':')的输出结果为\_\_\_\_\_\_\_\_\_\_。  
30.  Python 3.x语句print(1, 2, 3, sep=',')的输出结果为\_\_\_\_\_\_\_\_\_\_。  
31.  表达式int(4\*\*0.5)的值为\_\_\_\_\_\_\_\_\_\_。  
32.  表达式8 \*\* (1/3)的值为\_\_\_\_\_\_\_\_\_\_。  
33.  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。  
34.  Python内置函数\_\_\_\_\_\_\_\_\_\_用来返回序列中的最大元素。  
35.  Python内置函数\_\_\_\_\_\_\_\_\_\_用来返回序列中的最小元素。  
36.  Python内置函数\_\_\_\_\_\_\_\_\_\_用来返回数值型序列中所有元素之和。  
37.  已知x = 3，那么执行语句x += 6之后，x的值为\_\_\_\_\_\_\_\_\_\_。  
38.  表达式13 / 4的值为\_\_\_\_\_\_\_\_\_\_。  
39.  表达式13 // 4的值为\_\_\_\_\_\_\_\_\_\_。  
40.  表达式-13 // 4的值为\_\_\_\_\_\_\_\_\_\_。  
41.  表达式3 \*\* 2的值为\_\_\_\_\_\_\_\_\_\_。  
42.  表达式chr(ord('a')-32) 的值为\_\_\_\_\_\_\_\_\_\_。  
43.  表达式abs(3+4j) 的值为\_\_\_\_\_\_\_\_\_\_。  
44.  表达式16 \*\* 0.5的值为\_\_\_\_\_\_\_\_\_\_。  
45.  表达式type({3})的值为\_\_\_\_\_\_\_\_\_\_。  
46.  表达式type({3:3})的值为\_\_\_\_\_\_\_\_\_\_。  
47.  表达式isinstance('Hello world', str)的值为\_\_\_\_\_\_\_\_\_\_。  
48.  表达式isinstance('abc', int)的值为\_\_\_\_\_\_\_\_\_\_。  
49.  表达式isinstance(4j, (int, float, complex))的值为\_\_\_\_\_\_\_\_\_\_。  
50.  表达式isinstance('4', (int, float, complex))的值为\_\_\_\_\_\_\_\_\_\_。  
51.  表达式type(3) in (int, float, complex)的值为\_\_\_\_\_\_\_\_\_\_。  
52.  表达式type('3') in (int, float, complex)的值为\_\_\_\_\_\_\_\_\_\_。  
53.  表达式type(3) == int的值为\_\_\_\_\_\_\_\_\_\_。

54.  Python 3.x的range()函数返回一个\_\_\_\_\_\_\_\_\_\_\_\_\_。  
55.  表达式“[3] in [1, 2, 3, 4]”的值为\_\_\_\_\_\_\_\_\_\_\_。  
56.  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\_\_\_\_\_\_\_\_。  
57.  列表对象的\_\_\_\_\_\_\_\_\_\_\_方法删除首次出现的指定元素，如果列表中不存在要删除的元素，则抛出异常。  
58.  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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
59.  在Python中，字典和集合都是用一对\_\_\_\_\_\_\_\_\_\_\_\_作为界定符，字典的每个元素有两部分组成，即\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_，其中\_\_\_\_\_\_\_\_不允许重复。  
60.  使用字典对象的\_\_\_\_\_\_\_\_\_\_方法可以返回字典的“键-值对”，使用字典对象的\_\_\_\_\_\_\_\_\_\_\_\_\_方法可以返回字典的“键”，使用字典对象的\_\_\_\_\_\_\_\_\_\_方法可以返回字典的“值”。  
61.  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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
62.  假设有一个列表a，现要求从列表a中每3个元素取1个，并且将取到的元素组成新的列表b，可以使用语句\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
63.  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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
64.  \_\_\_\_\_\_\_\_\_\_\_（可以、不可以）使用del命令来删除元组中的部分元素。  
65.  表达式[1, 2, 3]\*3的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
66.  表达式list(map(str, [1, 2, 3]))的值为\_\_\_\_\_\_\_\_\_\_。  
67.  语句x = 3==3, 5执行结束后，变量x的值为\_\_\_\_\_\_\_\_\_\_。  
68.  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\_\_\_\_\_\_\_\_\_\_。  
69.  已知x = 3，那么执行语句x \*= 6之后，x的值为\_\_\_\_\_\_\_\_\_\_。  
70.  表达式3 in [1, 2, 3, 4]的值为\_\_\_\_\_\_\_\_\_\_。  
71.  使用列表与整数的乘法生成包含10个数字5的列表，可以写为\_\_\_\_\_\_\_\_\_\_。  
72.  任意长度的非空列表、元组和字符串中最后一个元素的下标为\_\_\_\_\_\_\_\_\_\_。  
73.  表达式list(range(1, 10, 3))的值为\_\_\_\_\_\_\_\_\_\_。  
74.  表达式list(range(10, 1, -3))的值为\_\_\_\_\_\_\_\_\_\_。  
75.  表达式list(range(5))的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
76.  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\_\_\_\_\_\_\_\_\_\_。  
77.  切片操作list(range(6))[::2]执行结果为\_\_\_\_\_\_\_\_\_\_。  
78.  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\_\_\_\_\_\_\_\_\_\_。  
79.  语句sorted([1, 2, 3], reverse=True) == reversed([1, 2, 3])执行结果为\_\_\_\_\_\_\_\_\_\_。  
80.  表达式sorted([111, 2, 33], key=lambda x: len(str(x)))的值为\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。  
81.  表达式sorted([111, 2, 33], key=lambda x: -len(str(x)))的值为\_\_\_\_\_\_\_\_\_\_。  
82.  表达式max([111, 22, 3], key=str)的值为\_\_\_\_\_\_\_\_\_\_。  
83.  语句x = (3,)执行后x的值为\_\_\_\_\_\_\_\_\_\_。  
84.  语句x = (3)执行后x的值为\_\_\_\_\_\_\_\_\_\_。  
85.  已知x=3和y=5，执行语句 x, y = y, x 后x的值是\_\_\_\_\_\_\_\_\_\_。  
86.  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。  
87.  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。  
88.  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。

89. 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。

90. 表达式{1:'a',2:'b',3:'c'}.get[4:’d’]的值为\_\_\_\_\_\_\_\_\_\_。

91. 已知x={1:2},那么执行语句x[2]=3之后，x的值为\_\_\_\_\_\_\_\_\_\_

92. 表达式{1,2,3,4}-{3,4,5,6}的值为\_\_\_\_\_\_\_\_\_\_。

93. 表达式set([1,1,2,3])的值为\_\_\_\_\_\_\_\_\_\_。

94. 使用列表推导式得到100以内所有能被13整除的整数的代码可以写作\_\_\_\_\_\_\_\_\_\_。

95. 已知x=[3,5,7],那么表达式x[10:]的值为\_\_\_\_\_\_\_\_\_\_。

96. 已知x=[3,5,7],那么执行语句x[len(x):]=[1,2]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

97. 已知x=[3,7,5]，那么执行语句x.sort(reverse=True)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

98. 已知x=[3,7,5]，那么执行语句x=x.sort(reverse=True)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

99. 已知x=[3,7,5]，那么执行语句

x.sort(key=lambda x:len(str(x)),reverse=True)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

100. 表达式list(zip([1,2],[3,4]))的值为\_\_\_\_\_\_\_\_\_\_。

101. 已知x=[1,2,3,2,3]，执行语句x.pop()已知之后，x的值为\_\_\_\_\_\_\_\_\_\_。

102. 已知x=[1,2,3,2,3]，执行语句x.pop(0)已知之后，x的值为\_\_\_\_\_\_\_\_\_\_。

103. 表达式list(map(list,zip(\*[[1,2,3],[4,5,6]])))的值为\_\_\_\_\_\_\_\_\_\_。

104. 表达式[x for x in [1,2,3,4,5] if x<3]的值为\_\_\_\_\_\_\_\_\_\_。

105. 表达式[x for x in [1,2,3,4,5] if x%2==1]的值为\_\_\_\_\_\_\_\_\_\_。

106. 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\_\_\_\_\_\_\_\_\_\_。

107. 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\_\_\_\_\_\_\_\_\_\_。

108. 已知列表x=[1,2],那么表达式list(enumerate(x))的值为\_\_\_\_\_\_\_\_\_\_。

109. 已知vec=[[1,2],[3,4]]，则表达式[col for row in vec for col in row]的值为\_\_\_\_\_\_\_\_\_\_。

110. 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\_\_\_\_\_\_\_\_\_\_。

111. 已知x=list(range(10)),则表达式x[-4:]的值为\_\_\_\_\_\_\_\_\_\_。

112. 已知x=[3,5,7]，那么执行语句x[1:]=[2]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

113. 已知x=[3,5,7]，那么执行语句x[:3]=[2]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

114. 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\_\_\_\_\_\_\_\_\_\_。

115. 已知x=[1,2,3,2,3],执行语句x.remove(2)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

116. 表达式len([i+5 for i in range(10)])的值为\_\_\_\_\_\_\_\_\_\_。

117. 表达式len(range(1,10))的值为\_\_\_\_\_\_\_\_\_\_。

118. 表达式range(10)[-1]的值为\_\_\_\_\_\_\_\_\_\_。

119. range(10,20)[4]的值为\_\_\_\_\_\_\_\_\_\_。

120. 表达式round(3.4)的值为\_\_\_\_\_\_\_\_\_\_。

121. 表达式round(3.7)的值为\_\_\_\_\_\_\_\_\_\_。

122. 已知x=(3),那么表达式x\*3的值为\_\_\_\_\_\_\_\_\_\_。

123. 已知x=(3,),那么表达式x\*3的值为\_\_\_\_\_\_\_\_\_\_。

124. 假设列表对象x=[1,1,1],那么表达式id(x[0])==id(x[2])的值为\_\_\_\_\_\_\_\_\_\_。

125. 已知列表x=list(range(10)),那么执行语句del x[::2]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

126. 已知列表x=[1,2,3,4],那么执行语句del x[1]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

127. 表达式[1,2]\*2的值为\_\_\_\_\_\_\_\_\_\_。

128. 已知列表x=[1,2,3],那么执行语句x.insert(1,4)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

129. 已知列表x＝[1,2,3]，那么执行语句x.insert(0,4）之后，x的值为\_\_\_\_\_\_\_\_\_\_。

130. 已知x=[[1]]\*3，那么执行语句x[0][0]=5之后，变量x的值为\_\_\_\_\_\_\_\_\_\_。

131. 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\_\_\_\_\_\_\_\_\_\_。

132. 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\_\_\_\_\_\_\_\_\_\_。

133. 表达式sorted([13,1,237,89,100],key=Iambda x:Ien(str(x)))的值为\_\_\_\_\_\_\_\_\_\_。

134. 已知x={1:2,2:3}，那么表达式x.get(3,4)的值为\_\_\_\_\_\_\_\_\_\_。

135. 已知x={1:2,2:3}，那么表达式x.get(2,4)的值为\_\_\_\_\_\_\_\_\_\_。

136. 表达式{1,2,3}|{3,4,5}的值为

137. 表达式{1,2,3}&{3,4,5}的值为\_\_\_\_\_\_\_\_\_\_。

138. 表达式{1,2,3}-{3,4,5}的值为\_\_\_\_\_\_\_\_\_\_。

139. 表达式{1,2,3}<{3,4,5}的值为\_\_\_\_\_\_\_\_\_\_。

140. 表达式{1,2,3}<{1,2,4}的值为\_\_\_\_\_\_\_\_\_\_。

141. 表达式[1,2,3].count(4)的值为\_\_\_\_\_\_\_\_\_\_。

142. Python标准库random中\_\_\_\_\_\_\_\_\_\_函数的作用是从序列中随机选择1个元素

143. Python标准库random中sample(seq,k）函数从序列中选择\_\_\_\_\_\_\_\_\_\_ (重复、不重复？)的k个元素

144. random模块中\_\_\_\_\_\_\_\_\_\_函数的作用是将列表中的元素随机乱序。

145. 执行代码x,y,z=sorted([1,3,2]）之后，变量y的值为\_\_\_\_\_\_\_\_\_\_。

146. 表达式(1,2,3)+(4,5)的值为\_\_\_\_\_\_\_\_\_\_。

147. 表达式dict(zip([1,2],[3,4]))的值为\_\_\_\_\_\_\_\_\_\_。

148. 语句x,y,z=[1,2,3]执行后，变量y的值为\_\_\_\_\_\_\_\_\_\_。

149. 已知x={1,2,3}，那么执行语句x.add(3)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

150. 已知x={1:1}，那么执行语句x[2]=4之后，Ien(x）的值为\_\_\_\_\_\_\_\_\_\_。

151. 已知x={1:1,2:2}，那么执行语句x[2]=4之后，len(x)的值为\_\_\_\_\_\_\_\_\_\_。

152. 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\_\_\_\_\_\_\_\_\_\_。

153. 已知列表x=[1,2]，那么连续执行命令v=xl:l和v.append(3）之后，x的值为\_\_\_\_\_\_\_\_\_\_。

154. 已知列表x=[1,2]，执行语句y=x[:]后，表达式id(x)==id(y）的值为\_\_\_\_\_\_\_\_\_\_

155. 已知列表x=[1,2]，执行语句y=x后，表达式id(x)==id(y）的值为\_\_\_\_\_\_\_\_\_\_。

156. 已知列表x=[1,2]，执行语句y=x后，表达式x is y的值为\_\_\_\_\_\_\_\_\_\_。

157. 已知列表x=[1,2]，执行语句y=x[:]后，表达式x is not y的值为\_\_\_\_\_\_\_\_\_\_。

158. 表达式[I for I in range(10) if i>8]的值为\_\_\_\_\_\_\_\_\_\_。

159. 已知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]))]值为\_\_\_\_\_\_\_\_\_\_。

160. 执行语句x,y,z=map(str,range(3))之后，变量y的值为\_\_\_\_\_\_\_\_\_\_。

161. 已知列表x=[1,2]，那么执行语句x.extend([3])之后，x的值为\_\_\_\_\_\_\_\_\_\_。

162. 已知列表x=[1,2]，那么执行语句x.append([3])之后，x的值为\_\_\_\_\_\_\_\_\_\_。

163. 执行语句x,y,z='123'之后，y的值为\_\_\_\_\_\_\_\_\_\_。

164. 表达式print(0b10101）的值为\_\_\_\_\_\_\_\_\_\_。

165. 已知x=[1,2,3,4,5]，那么执行语句del x[:3]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

166. 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\_\_\_\_\_\_\_\_\_\_。

167. 表达式[5 for I in range(3)]的值为\_\_\_\_\_\_\_\_\_\_。

168. 表达式{1,2,3}=={1,3,2}的值为\_\_\_\_\_\_\_\_\_\_。

169. 表达式[1,2,3]==[1,3,2]的值为 \_\_\_\_\_\_\_\_\_\_。

170. 已知x=[1,2,1].)，那么表达式id(x[0])==id(x[2)的值为\_\_\_\_\_\_\_\_\_\_。

171. 表达式3 not in[1,2,3]的植为\_\_\_\_\_\_\_\_\_\_。

172. 已知x=[1,2]，那么执行语句x[0:0]＝[3,3］之后，x的值为\_\_\_\_\_\_\_\_\_\_。

173. 已知x=[1,2]．那么执行语句x[0:1]＝[3,3］之后，x的值为\_\_\_\_\_\_\_\_\_\_。

174. 已知x=[1,2,3,4,5]，那么执行语句del x[1:3]之后，x的值为\_\_\_\_\_\_\_\_\_\_。

175. 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\_\_\_\_\_\_\_\_\_\_。

176. 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\_\_\_\_\_\_\_\_\_\_。

177. 已知列表x=[1.0,2.0,3.0］，那么表达式sum(x)/len(x)的值为\_\_\_\_\_\_\_\_\_\_。

178. 已知x={1:2,2:3,3:4}，那么表达式sum(x)的值为\_\_\_\_\_\_\_\_\_\_。

179. 已知x={1:2,2:3,3:4}．那么表达式sum(x.values())的值为\_\_\_\_\_\_\_\_\_\_。

180. 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\_\_\_\_\_\_\_\_\_\_。

181. 表达式1234%1000//100的值为\_\_\_\_\_\_\_\_\_\_。

182. 表达式3//5的值为\_\_\_\_\_\_\_\_\_\_。

183. 表达式[1,2]＋[3]的值为\_\_\_\_\_\_\_\_\_\_。

184. 表达式(1,)+(2,)的值为\_\_\_\_\_\_\_\_\_\_。

185. 表达式(1)＋(2)的值为\_\_\_\_\_\_\_\_\_\_。

186. 已知x,y=map(int,['1",'2"]),那么表达式x+y的值为\_\_\_\_\_\_\_\_\_\_。

187. 已知x,y=map(str,range(1,3))．那么表达式x+y的值为\_\_\_\_\_\_\_\_\_\_。

188. 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\_\_\_\_\_\_\_\_\_\_。

189. 已知列表x＝[1,3,2]，那么执行语句x.reverse()之后，x的值为\_\_\_\_\_\_\_\_\_\_。

190 . 已知列表x=[1,3,2]，那么执行语句x=x.reverse()之后，x的值为\_\_\_\_\_\_\_\_\_\_。

191. 已知x为非空列表，那么表达式x.reverse()==Iist(reversed(x))的值为\_\_\_\_\_\_\_\_\_\_。

192. 已知x为非空列表，那么表达式x.sort()==sorted(x)的值为\_\_\_\_\_\_\_\_\_\_。

193. 已知列表x＝[1,3,2]．那么执行语句y=Iist(reversed(x))之后，x的值为\_\_\_\_\_\_\_\_\_\_。

194. 已知列表x＝[1,3,2]，那么执行语句y=Iist(reversed(x))之后，x的值为\_\_\_\_\_\_\_\_\_\_。

195. 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\_\_\_\_\_\_\_\_\_\_。

196. 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\_\_\_\_\_\_\_\_\_\_。

197. 已知l列表x=[1,3,2]．那么执行语句a,b,c=sorted(x)之后，b的值为\_\_\_\_\_\_\_\_\_\_

198. 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\_\_\_\_\_\_\_\_\_\_。

199. 表达式set([1,2,3])=={1,2,3}的值为\_\_\_\_\_\_\_\_\_\_。

200. 表达式set([1,2,2,3])=={i,2,3}的值为\_\_\_\_\_\_\_\_\_\_。

201. 表达式'%c'%65==Str(65)的值为\_\_\_\_\_\_\_\_\_\_。

202. 表达式'%s'%65==str(65)的值为\_\_\_\_\_\_\_\_\_\_。

203. 表达式chr(ord('b")^32)的值为\_\_\_\_\_\_\_\_\_\_。

204. 表达式'abc'in'abdcefg'的值为\_\_\_\_\_\_\_\_\_\_。

205. 已知x='abcd'和y='abcde'，那么表达式[i==j for i,j in zip(x,y)]的值为\_\_\_\_\_\_\_\_\_\_。

206. 已知x=Iist(range(20))。那么表达式x[-1]的值为\_\_\_\_\_\_\_\_\_\_。

207. 已知x=3+4j和v=5+6j，那么表达式x+y的值为\_\_\_\_\_\_\_\_\_\_。

208. 表达式int('11',2)的值为\_\_\_\_\_\_\_\_\_\_。

209. 表达式chr(ord('A')+1)的值为\_\_\_\_\_\_\_\_\_\_。

210. 表达式int(str(34))==34的值为\_\_\_\_\_\_\_\_\_\_。

211. 表达式Iist(str([3,4]))==[3,4]的值为\_\_\_\_\_\_\_\_\_\_。

212. 表达式15//4的值为\_\_\_\_\_\_\_\_\_\_。

213. 表达式sorted({'a':3,'b':9,'c':75})的值为\_\_\_\_\_\_\_\_\_\_。

214. 表达式sorted({'a':3,'b':9,'c':75}.vaIues())的值为\_\_\_\_\_\_\_\_\_\_。

215. 已知x=[3,2,4,1]．那么执行语句x=x.sort()之后，x的值为\_\_\_\_\_\_\_\_\_\_。

216. 已知x=list(range(20))，那么语句print(x[100:200])的输出结果为\_\_\_\_\_\_\_\_\_\_。

217. 已知x=Iist(range(20))，那么执行语句x[:18]=[]后列表x的值为\_\_\_\_\_\_\_\_\_\_。

218. 已知x=([1],[2])，那么执行语句x[0].append(3)后x的值为\_\_\_\_\_\_\_\_\_\_。

219. 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\_\_\_\_\_\_\_\_\_\_。

220. 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\_\_\_\_\_\_\_\_\_\_。

221. 已知x=[1,2,3]，那么表达式not(set(x\*100)-set(x))的值为\_\_\_\_\_\_\_\_\_\_。

222. 表达式[1,2,3]>[1,3,2]的值为＿\_\_\_\_\_\_\_\_\_\_。

223. 已知x=[1,2,3]，那么表达式set(x\*100)==set(x)的值为\_\_\_\_\_\_\_\_\_\_。

224. 已知x=[1,2,3,4,5]，那么执行语句x[::2]=range(3)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

225. 已知x＝[1,2,3,4,5]，那么执行语句x[1::2]=sorted(x[i::2]，reverse=True)之后，x的值为\_\_\_\_\_\_\_\_\_\_。

226. 表达式type({})＝dict的值为\_\_\_\_\_\_\_\_\_\_。

227. 表达式type({})==Set的值为\_\_\_\_\_\_\_\_\_\_。

228. Python提供了两种基本的循环结构：\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_。

# 判断题

1  Python是一种跨平台、开源、免费的高级动态编程语言。  
2  Python 3.x完全兼容Python 2.x。  
3  在Windows平台上编写的Python程序无法在Unix平台运行。  
4  不可以在同一台计算机上安装多个Python版本。  
5  pip命令支持使用扩展名为whl的文件离线安装Python扩展库。  
6  下载whl文件进行离线安装扩展库时，因为whl文件的名字都比较长，可以改成短一些的名字再使用pip进行安装。  
7  用来安装Python扩展库的pip命令应该在命令提示符环境下运行，如果安装了多个版本的Python，最好切换至相应版本的Python安装目录下运行。

8  已知x = 3，那么赋值语句x = 'abcedfg'是无法正常执行的。  
9  Python变量使用前必须先声明，并且一旦声明就不能在当前作用域内改变其类型了。  
10  Python不允许使用关键字作为变量名，但是允许使用内置函数名作为变量名，不过这会改变函数名的含义，所以不建议这样做。  
11  在Python中可以使用if作为变量名。  
12  在Python 3.x中可以使用中文作为变量名。  
13  Python变量名必须以字母或下划线开头，并且区分字母大小写。  
14  加法运算符可以用来连接字符串并生成新字符串。  
15  x = 9999\*\*9999这样的语句在Python中无法运行，因为数字太大了超出了整型变量的表示范围。  
16  3+4j不是合法的Python表达式。  
17  0o12f是合法的八进制数字。  
18  只有Python扩展库才需要导入以后才能使用其中的对象，Python标准库不需要导入即可使用其中的所有对象。  
19  在Python中0xad是合法的十六进制数字表示形式。  
20  4j 是合法Python数字类型。  
21  Python使用缩进来体现代码之间的逻辑关系，对缩进的要求非常严格。  
22  Python代码的注释只有一种方式，那就是使用#符号。  
23  放在一对三引号之间的任何内容将被认为是注释。  
24  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。  
25  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。  
26  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、元组的连接和集合的并集运算。  
27  在Python中可以使用 for 作为变量名。  
28  在Python中可以使用 id 作为变量名，但是不建议这样做。  
29  一个数字5也是合法的Python表达式。  
30  执行语句from math import sin之后，可以直接使用sin()函数，例如 sin(3)。  
31  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。  
32  Python变量名区分大小写，所以student和Student不是同一个变量。  
33  在Python 3.x中reduce()是内置函数。  
34  如果只需要math模块中的sin()函数，建议使用from math import sin来导入，而不要使用import math导入整个模块。  
35  表达式pow(3, 2) == 3 \*\* 2的值为True。  
36  已知x = 3，那么执行语句x+=6之后，x的内存地址不变。  
37  安装Python扩展库时只能使用pip工具在线安装，如果安装不成功就没有别的办法了。

38 列表、元组、字符串属于有序序列，而字典和集合属于无序序列。

39、同一个集合中的元素都是唯一的，不会存在重复的元素。

40、同一个列表中的元素都是唯一的，不允许存在相同的元素。

41、列表、元组、字符串支持双向索引，-1表示最后一个元素的下标。

42、集合支持双向索引，-1表示最后一个元素的下标。

43、python支持使用字典的“键”作为下标来访问字典中的值。

44、列表可以作为字典的“键”。

45、元组可以作为字典的“键”。

46、字典的“键”必须是不可变的。

47、对于关键字in而言，集合的测试速度比列表快很多。

48、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。

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

50、生成器推导式比列表推导式具有更高的效率，推荐使用。

51、python集合可以包含相同的元素。

52、python字典中的“键”不允许重复。

53、python字典中的“值”不允许重复。

54、python集合中的元素可以是元组。

55、python集合中的元素可以是列表。

56、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。

57、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。

58、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。

59、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。

60、使用python列表的方法inset()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。

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

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

63、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。

64、元组是不可变的，不支持列表对象的insert()、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。

65、无法删除集合中指定位置的元素，只能删除特定值的元素。

66、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遍历而不需要进行任何修改，建议使用元组而不使用列表。

67、当以指定”键“为下标给字典对象赋值时，若该”键“存在则表示修改该”键“对应的”值“，若不存在则表示为字典对象添加一个新的”键：值对“。

68、假设x是含有5个元素的列表，那么切片操作x[10:]是无法执行的，会抛出异常。

69、假设x是含有5个元素的列表，那么使用print(x[10])是无法执行的，会抛出异常。

70、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。

71、只能通过切片访问列表中的元素，不能使用切片修改列表中的元素。

72、只能通过切片访问元组中的元素，不能使用切片修改元组中的元素。

73、python集合不支持使用下标访问其中的元素。

74、已知列表x中包含超过5个以上的元素，那么语句x=x[5:]+x[:5]可以实现将列表x中的元素循环左移5位。

75、对于生成器对象x=(3 for i in range(5))，连续两次执行list(x)的结果是一样的。

76、对于大量列表的连接，extend()方法比运算符+具有更高的效率。

77、表达式{1,3,2}>{1,2,3}的值为True。

78、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。

79、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。

80、表达式list('[1,2,3]')的值是[1,2,3].

81、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。

82、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。

83、表达式{1,2}\*2的值为{1,2,1,2}。

84、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。

85、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。

86、使用random模块的函数randint(1,100)获取随机数时，有可能会得到100。

87、已知x=[1,2,3,4]，那么执行x[0]=5之后，x的值为(5,2,3,4)。

88、已知x=3，那么执行x+=6语句前后x的内存地址是不变的。

89、内置函数len()返回指定序列的元素个数，适用于列表、元组、字符串、字典、集合以及range、zip等迭代对象。

90、已知x和y是两个等长的整数列表，那么表达式sum((i\*j for i,j in zip(x,y)))的作用是计算这两个列表所表示的向量的内积。

91、已知x和y是两个等长的整数列表，那么表达式[i+j for i,j in zip(x,y)]的作用是计算这两个列表所表示的向量的和。

92、表达式int['1'\*64,2]与sum(2\*\*i for i in range(64))的计算结果是一样的，但是前者更快一些。

93、已知x=list(range(20))，那么语句del x[::2]可以正常执行。

94、已知x=list(range(20))，那么语句x[::2]=[]可以正常执行。

95、已知x=list(range(20))，那么语句print(x[100:200])无法正常执行。

96、已知x是个列表对象，那么执行语句y=x之后，对y所做的任何操作都会同样作用到x上。

97、已知x是个列表对象，那么执行语句y=x[:]之后，对y所做的任何操作都会同样作用到x上。

98、在python中，变量不直接存储值，而是存储值的引用，也就是值在内存中的地址。

99、表达式(i\*\*2 for i in range(100))的结果是个元组。

100、在python中元组的值是不可变的，因此，已知x=([1],[2])，那么语句x[0].append(3)是无法正常执行的。

101、已知x={1:1,2:2}，那么语句x[3]=3无法正常执行。

102、已知列表x=[1,2,3,4]，那么表达式x.find(5)的值应为-1。

103、列表对象的排序方法sort()只能按元素从小到大排列，不支持别的排序方式。

104、表达式‘a’+1的值为‘b’

105、创建只包含一个元素的数组时，必须在元素后面加一个逗号，例如(3,)

# 问答题

1.  Python程序的\_\_name\_\_的作用是什么？

2. 为什么应尽量从列表的尾部进行元素的增加与删除操作？

3.  简单解释Python基于值的内存管理模式。

4. 简单解释运算符/和//的区别。

5. 分析逻辑运算符“or”的短路求值特性。

# 编程题

1.  编写程序，生成包含1000个0到100之间的随机整数，并统计每个元素的出现次数。  
2.  编写程序，用户输入一个列表和2个整数作为下标，然后使用切片获取并输出列表中介于2个下标之间的元素组成的子列表。例如用户输入[1, 2, 3, 4, 5, 6]和2,5，程序输出[3, 4, 5, 6]。

3.  设计一个字典，并编写程序，用户输入内容作为“键”，然后输出字典中对应的“值”，如果用户输入的“键”不存在，则输出“您输入的键不存在！”

4.  编写程序，生成包含20个随机数的列表，然后将前10个元素升序排列，后10个元素降序排列，并输出结果。

5. 编写程序，运行后用户输入4位整数作为年份，判断其是否为闰年。如果年份能被400整除，则为闰年；如果年份能被4整除但不能被100整除也为闰年。

6. 编写程序，生成一个包含50个随机整数的列表，然后删除其中所有奇数。（提示：从后向前删）

7. 编写程序，生成一个包含20个随机整数的列表，然后对其中偶数下标的元素进行降序排列，奇数下标的元素不变。（提示：使用切片）

8. 编写程序，用户从键盘输入小于1000的整数，对其进行因式分解。例如，10=2\*5，60=2\*2\*3\*5。

9. 编写程序，至少使用2种不同的方法计算100以内所有奇数的和。

10. 编写程序，输出所有由1，2，3，4这四个数字组成的素数，并且在每个素数中每个数字只使用一次。

11. 编写程序，实现分段函数计算，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 |

12. 编写程序，输入任意大的自然数，输出各位数字之和。

13. 编写程序，输入两个集合setA和setB，分别输出它们的交集、并集和差集。

14. 编写程序，输入一个自然数，输出它的二进制、八进制、十六进制表示形式。

15. 编写程序，输入一个包含若干整数的列表，输出一个新列表，要求新列表中只包含原列表中的偶数。

16. 编写程序，输入两个分别包含若干整数的列表lstA和lstB，输出一个字典，要求使用列表lstA中的元素作为键，列表lstB中的元素作为值，并且最终字典中的元素数量取决于lstA和lstB中元素最少的列表的数量。

17. 编写程序，输入一个包含若干整数的列表，输出新列表，要求新列表中的所有元素来自于输入的列表，并且降序排列。

18. 编写程序，输入一个包含若干整数的列表，输出列表中所有整数连乘的结果。

19. 编写程序，输入两个各包含2个整数的列表，分别表示城市中两个地点的坐标，输出两点之间的曼哈顿距离。

20. 编写程序，输入包含若干集合的列表，输出这些集合的并集。要求使用reduce()函数和lambda表达式完成。