# 第2~3章

1. 元组中的元素只能是同一数据类型 （ ）
2. 可以用a={}创建空的集合 （ ）
3. 在Python中，运行代码'5'+5，会生成字符串'55'（ ）
4. 语句20==20.0返回结果是True （ ）
5. 可以将列表作为字典的键 （ ）
6. 所有写在“//”号之后的文本会自动被Python解释器忽略 （ ）
7. 列表中的pop方法可以返回特定位置的元素但是remove不可以（ ）
8. 通过isinstance(a, (int, str))可以检测变量a的类型是否在int, str中 （ ）
9. 列表、字典、NumPy数组都是可变对象 （ ）
10. tup=(1,[1,2,3],3);tup[1].append(4)不能执行（ ）
11. '5'+'3'的结果是’8’（ ）
12. 对于对象a=[1,2,3]和b=[1,2,3]，a is b和a==b输出的结果都为True （ ）
13. d={};d[tuple([1, 2, 3])] = 5语句不能执行（ ）
14. 函数柯里化的过程中，如果传入的参数数量不足，剩下的参数将默认为空值传入（ ）
15. 运行以下代码，4 > 3 > 2 > 1，将返回True（ ）
16. 下列这句代码是对的：test\_data = 1, （ ）
17. Python中不可以使用java中表示语句块的“{}”，以及分隔符“；” （ ）
18. (4,None,'foo')+(6,0)+('bar')不能执行 ( )
19. 设a = ’python’,可以执行a[0]=‘P’ ( )
20. None是NoneType类型的唯一实例 ( )
21. 像Java语言一样，Python语句也是以分号结尾的 ( )
22. 对于含有换行的多行字符串，可以使用三个单引号’’’或者三个双引号”””（ ）
23. python的异常处理，try块后有可能不使用except块 （ ）
24. 表达式“[3] in [1, 2, 3, 4]”是的值是True.（ ）
25. def choise(x,y,z=5): choise(5,z= 2,y = 3) 可以调用 ( )
26. num1=num2=num3=20 在内存中指向不同的变量地址（ ）
27. 使用pop( )方法删除列表中某一元素后，该元素从栈顶被推出且无法再访问。（ ）
28. 在Python中可以使用 id 作为变量名 （ ）
29. 列表的插入函数insert()指定的序号可以大于数组长度。 （ ）
30. List中的remove()函数会移除列表中指定的所有的元素。（ ）
31. 用 sort 函数可以产生一个排好序的序列副本。 （ ）
32. 执行程序：tup1=(1,2,3,4,5)；a , b , \*rest = tup1；type(rest)；结果为 tuple 。 （ ）
33. set([2,[2,3,4]])的输出结果是{2,[2,3,4]}。 ( )
34. type((1)),和type(())的输出都是tuple （ ）
35. 已经a=[1,2,3,4,5],b=[6,7,8,9],则a.append(b)的输出结果是[1,2,3,4,5,6,7,8,9] ( )
36. a = [1, 2, 3];b = a;c = list(a)。a is b（T），a is c（F）

# 第4章

1. ndarray包含的每一个元素均为相同类型。（ ）
2. import numpy as np, arr=np.arange(10),arr.astype(float),

print(arr),输出结果为[0123456789].（ ）

1. 嵌套序列能被转换为多维数组（ ）
2. NumPy中的random函数中的randn()可以产生N(0,1)的数（ ）
3. 使用np.unique函数可以对NumPy数组进行去重。（ ）
4. 两个大小相等数组之间的算术运算，都是“点对点”的运算。（ ）
5. 可以通过numpy中的astype方法显式地转换数组的数据类型。（ ）
6. np.range()是python内建函数的range的数组版（ ）
7. arr.mean(1)计算的是每一列的平均值，arr.sum(0)计算行轴向的累和（ ）
8. 数组的切片就是将原数组的数据复制，对切片的修改不会影响原数组（ ）
9. 数组生成函数array将输入数据转化为ndarray，默认复制所有的输入数据.（ ）
10. 运用arr.ndim可以查看数组arr的秩。（ )
11. 同尺寸数组之间直接的比较会产生一个布尔值。（ ）
12. data=np.random.randn(2,3)，则data.ndim的输出结果是3（ ）
13. NumPy中的zeros方法和empty方法本质是一样的，都是创建一个全0的数组 （ ）
14. np.sort方法对数组按位置进行排序。（ ）
15. 用NumPy数组对象的reshape方法变换维度时会改变原数组。 （ ）
16. bool = np.array([[True,True,False],[True,True,False]])，则(bool>0).any()返回True。（ )
17. 使用布尔值索引选择数据时，生成的是数据的视图( )
18. Numpy数组的修改元素操作可以使用tuple。 ( )
19. 运行np.ones((3,4))，empty会创建一个全1数组。( )
20. 对一个array数组进行reshape操作，作用对象array的数组形状会发生改变。( )
21. 下面代码会报错。arr=[1,2,3];np.sqrt(arr,arr)( )
22. numpy的方法dot和\*计算方式没有差别。( )
23. numpy的mean()函数只能对一维数组使用( )
24. 如果a.shape=（2，3，4），那么a.mean(1).shape=（2，3）( )
25. 输入np.empty(10)的结果为：array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]) ( )
26. 随机数生成器中的随机数种子相同可以产生不同的随机数组。（ ）
27. 对Numpy 数组切片的修改并会影响原数组。（ )
28. 一个数组对象的itemsize，返回的值是由数组的大小决定的（ ）
29. 已知x=np.array([3,5,1,9,6,3]),那么表达式np.argmax(x)的值为3（ ）
30. ar = np.arange(9).reshape(3,3) ；ar[:2,0] 会生成一维数组（ ）
31. x=np.array([[0,1,2,3,4],[,5,6,7,8]])；x.shape。输出结果是（2，5）（ ）
32. 若结果中的值在原始数组中是连续的，flatten方法不会生成底层数值的副本。( )
33. arr=[4,-1,9,16];np.sqrt(arr);arr 以上代码输出的结果为array([2,nan,3,4]) ( )
34. 与列表不同，数组中要求所有元素的dtype是一样的，如果传入的参数类型和数组类型不一样，需要按照已有的类型进行转换。( )
35. np.zeros()生成的矩阵元素属性是Int ( )

# 第5章

1. pd.isnull(frame)与pd.notnull(frame)方法可以判断DataFrame对象frame中的元素是否缺失，返回布尔值数组。（ ）
2. 针对DataFrame在行上的标签索引iloc是左闭右开区间,loc是左闭右闭区间。（ ）
3. 在DataFrame中，当A是有效的Python变量名时，frame[‘A’]与frame.A返回的结果是相同的（ ）
4. pandas中的Series数据结构可以认为是一个长度固定且有序的字典（ ）
5. pandas.to\_datetime(date).year可以实现从时间序列（data）中提取年份（ ）
6. 设有s = pd.Series(np.arange(3))，则s.loc[:1]，s.iloc[:2]，s[:2]输出的结果不一样（ ）
7. 从DataFrame中选取的列是数据的视图，而不是拷贝。（ ）
8. df1 = pd.DataFrame([[5, 2, 3], [4, 5, 6],[7,8,9]], index=['A', 'B','C'], columns=['C1', 'C2', 'C3']) 其中df1.loc[2:1]=8 （ ）
9. 包含Series的字典可以用于构造DataFrame（ ）
10. obj = pd.Series([1,-4,np.nan,5,np.nan]),obj.sort\_values()，此操作会将obj中所有的缺失值都放到它的尾部。（ ）
11. 如果嵌套字典被赋值给DataFrame，pandas会将字典的键作为列，内部字典的键作为行索引（ ）
12. pd.DataFrame(np.arange(9).reshape((3,3))))可以建立3列3行的DataFrame。（ ）
13. 将Series赋值给DataFrame的一列时，Series的索引会按照DataFrame的索引重新排列，并在空缺的地方填充缺失值。（ ）
14. 在pandas中，通过isnull和notnull函数检查缺失数据，返回的结果是一样的（ ）
15. ser=pd.Series(np.arange(3.))，ser.loc[:1]和ser.iloc[:1]的结果是一样的（ ）
16. pandas中如果我们不为Series对象指定索引，程序会报错 （）
17. ['c':'d']是无效的索引 ()
18. 对DataFrame中选取的列进行修改不会映射到DataFrame中。（）
19. DataFrame的 values属性会把包含在DataFrame中的数据以NumPy数组的形式返回（ ）
20. 对于索引值为[‘a’,’b’]的Series对象obj，可以使用obj[1]获取值 （）
21. 4 in pd.Series([4,7,-5,3])返回的结果是True。 ( )
22. obj=pd.Series(np.arange(4.),index=["a","b","c","d"]);对于obj的切片，obj["b":"c"],其输出值为 b 1.0。( )
23. 已知obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])，输入obj2['a':'f']，会正确执行。 ( )
24. states = ('California', 'Ohio', 'Oregon', 'Texas')和states = ['California', 'Ohio', 'Oregon', 'Texas']都可以执行的。( )
25. 从DataFrame中选取的列是数据的拷贝，而不是视图。( )
26. Series中的索引值可以重复；DateFrame每列需是相同的值的类型( )
27. a1是一个Series，a2是一个DataFrame，他们都属于Pandas，但是a1.values和a2.values返回的数据类型不一样( )
28. 对pandas对象进行切片时：按实际索引进行切片，遵循左闭右闭原则；按默认索引进行切片，遵循左闭右开原则；切片范围超出索引值的时候，也可以进行切片。（ ）
29. del在DataFrame中只能够删除列不能够删除行。（ ）
30. 已知有一个DataFrame对象，可以直接用切片索引它的行列 （ ）

# 第6章

1. 用pandas读取文件时不指定列名时，会报错（ ）
2. pd.read\_csv('examples/ex2.csv', header=0)，header = 0表示取文件中的第一行作为列名。（ ）
3. 使用read\_csv方法读取文件时，可以通过指定chunksize进行分块读入文件( )
4. csv产生的数据都是字符串类型的，它不会做任何其它类型的转换（ ）
5. 将pandas数据写入到Excel格式中必须直接调用ExcelWriter（ ）
6. 使用json.loads方法可以将JSON字符串转化为python形式，使用json.dumps可以将python对象转化为JSON。（ ）
7. DataFrame中的to\_csv方法只能导出逗号分隔的文件（ ）
8. JSON对象中的键可以是int形式（ ）
9. 当要读入的文件没有列名时，可以设置关键字header=None（ ）
10. 读取的字段由多种不同数量的空格分开时，我们可以通过向seq传入正则表达式来进行读取（ )
11. CSV文件在写入的时候data1=data.to\_csv('/home/zhf/3.csv',sep='|')，其中sep参数用于制定分隔符（ ）
12. 使用read\_csv（）函数必须从头开始逐行读取（ ）
13. 将pandas数据写入到Excel格式中必须直接调用ExcelWriter（ ）
14. pandas中的函数read\_html在默认的情况下会解析一个html页面中的所有标签，返回的结果是DataFrame对象的列表 （ ）
15. 语句options.display.max\_rows可以调整pandas的最大显示行数，设置显示最开始的行数.( )
16. 使用read\_csv()方法读入文件的时候，默认的分隔符是”,”。 （ ）
17. DataFrame中的to\_csv方法只能导出逗号分隔的文件（ ）
18. 语句options.display.max\_rows可以调整pandas的最大显示行数，设置显示最开始的行数。 （ ）
19. na\_values可以接受一组用于表示缺失值的字符串 （ ）
20. 在字典中，每列不可以指定不同的缺失值标识。（ ）
21. 当要读入的文件没有列名时，可以设置关键字header=None （ ）
22. 使用read\_table方法读取文件时，可以传入seq参数指定分隔符 （ ）
23. pd.read\_csv('test.csv',skiprows=1)没有语法错误。( )
24. 在read\_csv的参数中，skiprows传入的行号必须为列表形式( )
25. 读入文件时，行和列的标签不可以禁止写入。( )
26. 当读取的txt文件中列名的数量比数据的列数少一个，因此read\_table推断第一列应当作为DataFrame的索引。（ ）
27. panda.read\_json只能自动将JSON数据集按指定次序转换为Series（ ）
28. pd.read\_csv('examples/ex6.csv', nrows=5) 和

chunker =pd.read\_csv('../examples/ex6.csv', chunksize=1000)

chunker.get\_chunk(5) 输出结果一样( )

1. 使用read\_table函数时，可以使用一个正则表达式作为分隔符。（ ）
2. read\_csv中的参数header默认为None （ ）
3. 为了分块读入文件，可以使用chunksize作为每一块的行数 （ ）
4. HDF不是数据库，它最适合用作”一次写多次读“的数据集 ( )
5. Pd.read\_csv()方法中，参数header和names同时设置最终以names作为基准设置列标签 ( )
6. 当要读入的文件没有列名时，可以设置关键字header=0 （ ）

# 第7章

1. 使用cut方法需要制定具体的边界。（ ）
2. data = pd.Series(['a', 'bb', 'cc', 'ab',9])，data.str.replace('a','d')其中9会被处理为Nan （ ）
3. 默认情况下，fillna方法可以传入指定值来补全缺失值，从而修改了原对象（ ）
4. 在删除dataframe的某一行时,设置关键词inplace=True可以覆盖原有的数据。（ ）
5. 对于DataFrame对象，dropna在默认情况下会删除所有值均为NA的行（ ）
6. data.dropna(how='all')可以删除data中所有包含NA的行（ ）
7. map和replace都可以用于修改一个对象中的子集的值（ ）
8. 处理缺失数据时可以用dropna()方法直接过滤缺失值（ ）
9. data是Series，操作data[data.notnull()]与data.dropna()是等价的。（ ）
10. 使用pandas的cut操作返回的是一个DataFrame对象。（ ）
11. pandas的索引中不可以包含重复标签（ ）
12. Python中in关键字、index方法、find方法都可以检查子字符串（ ）
13. fillna()方法可以传入字典，实现对不同列的缺失值用不同的值进行填充（ ）
14. DataFrame中的列索引是自动从1开始的（ ）
15. pandas中如果我们不为Series对象指定索引，程序会报错（ ）
16. NA和NaN的意思都是一样的 （ ）
17. data.index[0]='aaa'可以将data中第0行的索引变为aaa （ ）
18. 通过index方法查找指定字符的位置，如果字符不存在会返回-1。（ ）
19. drop\_duplicates函数删除重复行，默认保留第一个观测到的值。 ( )
20. 对于存放数值型数据的DataFrame对象，调用describe方法可以返回各个列的统计数据 ( )
21. Python内建的None值在对象数组中也被当作NA处理。（ ）
22. 使用rename方法对DataFrame对象df进行重命名将会返回一个新的对象。（ ）
23. pandas中用cut函数进行分箱得到的分组区间只能是前开后闭（ ）
24. pandas中的cut函数可以通过传入参数实现两端均封闭。( )
25. 在DataFrame对象中，使用fillna来补全缺失值时，返回的是一个新对象。（ ）
26. dropna 默认情况下会删除包含缺失值的列。（ ）
27. 语句np.sign(data)根据数据中的值的正负分别生成0和1的值 ( )
28. dropna（）函数的axis和how参数的默认值分别为0，any( )
29. fillna方法的参数可以传入字典为不同列设置不同的填充值 （ ）

# 第8章

1. 默认情况下，concat方法是沿着axis=0的轴向生效的（ ）
2. NumPy中的concatenate函数默认情况下是进行横向拼接（ ）
3. merge函数会在没有指定连接键的情况下，自动将重叠列名作为连接的键（ ）
4. 如果不是所有的级别值都能在各分组中找到，则unstack操作会引入缺失数据（ ）
5. swaplevel接收两个层级序号或名称，将原对象的层级进行变更。（ ）
6. 分层索引中“内部层级”是不可以选择的 （ ）
7. 在DataFrame中拆堆时，被拆堆的层级会变为结果中最低的层级（ ）
8. DataFrame的join方法，能更为方便地实现按索引合并。它还可用于合并多个带有相同或相似索引的DataFrame对象，但要求没有重叠的列（ ）
9. 一个分层索引的Series对象使用unstack方法会修改原对象（ ）
10. 使用merge函数进行列-列连接时，传递的DataFrame索引对象会被丢弃。（ ）
11. ~~在使用pandas.melt的时候，需要指明作为分组目标的列（ ）~~ 作废
12. combine\_first可用于拼接重复数据，用一个对象中的值填充另一个对象中的缺失值（ ）
13. 在数据透视表中，加入margins=False,可以在下方显示一些总和数据。（ ）
14. 以列表（或者数组）作为Series的索引可以构造分层索引。（ ）
15. 在分级索引中，可以通过level选项并通过改变轴向来进行聚合。 （）
16. unstack操作的只能是最内层，stack也是如此 （ ）
17. 默认情况下，merge函数会联合左连接和右连接的效果。（ ）
18. 使用merge函数进行列-列连接时，传递的DataFrame索引对象会被丢弃。（ ）
19. 通过给出层级排序函数sort\_index函数中的level选项，可以指明对这一个层级的索引排序（ ）
20. concat函数对两个含有重叠索引的Series进行连接后，重叠索引对应的值为连接前两个重叠索引对应的值之和（ ）
21. DataFrame的join方法进行连接键上的左连接，完全保留左边DataFrame的行索引（ ）
22. 若一个具有二重索引的数据，使用data.loc(axis=0)[0,1]命令，获得的数据保留二重索引。（ ）
23. DataFrame的多对多连接是行的笛卡尔积（ )
24. groupby()后，使用agg()可以对分组结果调用自定义的函数；可以对一列使用多个函数；但不能对不同的列使用不同的函数( )
25. 对两个Series数组调用concat方法，一般情况下，如果设定axis=0，会生成一个Series，如果设定axis=1，会生成一个DataFrame （ ）
26. DataFrame的set\_index函数会生成一个新的DataFrame。（ ）
27. stack(堆叠)会将行中的数据透视到列中（ ）
28. swaplevel接收两个层级序号或层级名称，返回一个进行了层级变更的新对象（但是原数据是不变的）( )
29. concat函数中，join默认的连接方式是'inner'（ ）
30. concat函数中，如果传入了keys和/或levels参数，则该参数不能用于多层索引的层级名称（ ）
31. 在concat方法中传入参数ignore\_index参数将忽略轴向索引 （ ）
32. merge函数中on参数所需要连接的列名并不必须在两边的DataFrame对象中都出现。( )
33. frame=pd.DataFrame(np.arange(12).reshape((4, 3)),index=[['a', 'a', 'b', 'b'], [1, 2, 1, 2]],columns=[['Ohio', 'Ohio', 'Colorado'],['Green', 'Red', 'Green']]),对frame进行排序，frame.reset\_index(1)和frame.sort\_index(level=1)返回的结果相同。 （ ）
34. sort\_index只能在单一层级上对数据进行排序 （ ）

# 第9章

1. fig.add\_subplot(2, 3, 5)，第3个参数的含义是在创建的6个子图中，它是第五个（ ）
2. DataFrame的plot方法可以在同一个子图中将每一列绘制为不同的折线，并自动生成图例（ ）
3. matplotlib和seaborn都是可以用于绘图的包。（ ）
4. 在图表中添加图形时，需要生成patch对象shp，再调用add\_patch(shp)将它加入到子图（ ）
5. fig.add\_subplot返回的是Axes Subplot对象（ ）
6. matplotlib并不检查标签是否重叠（ ）
7. regplot是seaborn中的绘制折线图的方法（ ）
8. 对于折线图，后续的点默认是线性内插（ ）
9. 运行代码fig.add\_subplot（2，2，0）生成的是2×2图片中的第一个子图（ ）
10. 使用分面网格是利用多种分组变量对数据进行可视化的方式（ ）
11. 绘制图表时可以通过传递label参数添加图例来区分不同的绘图元素 （ ）
12. DataFrame对象调用plot方法时，可以通过传递subplots=True来使得对象的每一列绘制在独立的子图中（ ）
13. 假设有一个figure对象fig，则fig.add\_subplot(224)和fig.add\_subplot(2,2,4)执行的结果相同。（ ）
14. plt.plot([2,3, 6, 7,8], 'g--o')与plt.plot([2,3, 6, 7,8], 'go--')生成的图形不一样。（ ）
15. pandas对象的plot.barh方法绘制的是垂直的柱状图（ ）
16. matplotlib在设置x轴刻度的时候，区间长度必须一致。 （ ）
17. matplotlib绘制的图位于Figure对象中，可以使用plt.figure()调用 （ ）
18. 通过使用shareX、shareY可以使得子图分别拥有相同的x轴和y轴 （ ）
19. set\_xticks在数据范围内设定刻度的位置，而不设置刻度的标签。（ ）
20. plt.savefig函数可以将图片保存到文件中，并且只能以fig格式存储（ ）
21. 在绘制直方图的参数中，bins的默认参数是10。 （ ）
22. 假设有一个figure对象fig，则fig.add\_subplot(224)和fig.add\_subplot(2,2,4)执行的结果相同。（ ）
23. 默认情况下，Series和DataFrame的plot()方法绘制的是条形图（ ）
24. fig=plt.figure()随后fig.plot()没有问题。（ ）
25. plot函数中'b'代表黑色 ( )
26. 代码plt.xlim（[0, 10]）的目的是将x轴的范围设置为0到10。 （ ）
27. 使用value\_counts:s.value\_counts().plot.bar(）可以有效地对Series值进行可视化。（ ）
28. plt.subplots方法可以创建一个新的Figure，并返回一个含有已创建的subplot对象的NumPy数组。（ ）
29. matplotlib会自动检查标签是否重叠（ ）
30. ax.scatter(np.arange(10), np.arange(10))可以绘制一个x和y取值都在1-10之间的散点图。 （ ）
31. DataFrame直接进行绘图时，默认把columns当成x轴（ ）

# 第10章

1. 调用groupby方法生成一个GroupBy对象可以直接进行输出。（ ）
2. quantile（分位数）是Series的方法所以也可以用来聚合。（ ）
3. 在没有特定指明对DataFrame对象的哪一列进行计算时，默认情况下会对所 有的数值列进行聚合（ ）
4. 聚合数据必须返回行索引，并且自己创建的聚合函数是可以直接使用的（ ）
5. 使用groupby方法分组时分组键可以是多种形式的，而且键的类型也不一定完全相同（ ）
6. 分层索引的数据集可以在某个指定的层级上进行聚合（ ）
7. 若传入一组函数或函数名，得到的DataFrame的列就会以相应的函数命名（ ）
8. 聚合只是分组运算中的一种，它是数据转换的一个特例。也就是说，它只是接受能够将一维数组简化为标量值的函数。（ ）
9. 对多个分组键的GroupBy对象进行迭代，生成的元组序列中的第一个元素是键值的元组（ ）
10. Crosstab的前两个参数可以是数组、Series、或者数组的列表（ ）
11. 对于groupby中的size方法，分组键中的任何缺失值都会被排除在结果之外（ ）
12. apply方法是将对象拆分为多块，在每一块都会调用传递的函数（ ）
13. 使用字典构造传给groupby的数组时，字典的键必须与对象的长度一致（ ）
14. 默认情况下，groupby在axis = 1的轴向上分组（ ）
15. 分组键中有缺失值会导致程序报错 （ ）
16. 分组信息不能以非数组形式存在 （ ）
17. 在没有特定指明对DataFrame对象的哪一列进行计算时，默认情况下会对所有的数值列进行聚合（ ）
18. 已知df = pd.DataFrame({'key1' : ['a', 'a', 'b', 'b', 'a'],

'key2' : ['one', 'two', 'one', 'two', 'one'],

'data1' : np.random.randn(5),

'data2' : np.random.randn(5)})

执行df.groupby(['key1','key2'])['data2'].mean()可以获得data2列的均值并以DataFrame的形式返回结果（ ）

1. 若df是一个DataFrame对象df，则默认情况下，df中的所有的数值列都可以聚合。（ ）
2. 假设有DataFrame对象df，key1是df中的列，则df.groupby('key1').mean()等价于df.groupby(['key1']).mean()。（ ）
3. 使用自定义的聚合函数时需要将其传递给agg方法 （ ）
4. 使用含有原数据不存在的分组键的字典或Series进行分组会出现报错 （ ）
5. groupby的参数只可以是列名，不可以是index名。（ ）
6. cumcount函数不会对df进行排序。（ ）
7. 对从DataFrame创建的GroupBy对象用列表或数组进行索引时，返回的是分组的DataFrame( )
8. 返回的聚合数据都是有索引的，而且这个索引默认是多层级索引，取消这种模式，在调用groupby的时候设定as\_index=True即可。 （ ）
9. cut返回的Categorical对象不能直接传递给groupby （ ）
10. pandas提供了crosstab()函数根据一个DataFrame对象中的数据生成交叉表，返回新的DataFrame对象。 （ ）
11. apply函数除了只可以传入函数 ( )
12. 在用groupby方法对DataFrame对象进行分组处理时，group keys可以是字符串类型的列名、其他python对象（如：series对象）以及python的函数。 （ ）
13. Goupby 对象会生成包含组名和数据块的二位元组序列 （ ）
14. 由于quantile并不是显式的为GroupBy对象实现的，因此不能用于聚合 （ ）

# 第11章

1. datetime.strptime可以在已知格式的情况下转换日期。（ ）
2. 不能创建含有重复索引的时间序列（ ）
3. Timestamp与datetime都是时间类型，但是timestamp只使用datetime一半的存储空间。（ ）
4. 全部的datetime都可以转化为字符串。（ ）
5. 如果时间序列是含有重复索引的，进行索引得到的结果是切片（ ）
6. 不同索引的时间序列之间的算术运算在日期上自动对齐（ ）
7. pd.date\_range('2000-01-01','2000-01-03 23:59',freq='0.545H')是不能运行的。（ ）
8. 以时间戳作为索引的Series可以传递能解释为日期的字符串进行索引。（ ）
9. resample后使用ohlc聚合函数可以获得包含开端-峰值-谷值-结束四种聚合值列的DataFrame（ ）
10. 在date\_range的fre参数传入“1h10min”是无效的（ ）
11. 如果两个时间区间拥有相同的频率，则它们的差是它们之间的单位数。（ ）
12. 时间序列进行索引和选择时，传入的日期字符串格式必须与时间序列中的格式相同（ ）
13. 所有使用datetime对象的地方都可以使用Timestamp（ ）
14. 两个时区不同的时间序列联合的结果是UTC时间（ ）
15. 锚定偏置量构成的时间频率是均匀间隔的。（ ）
16. shift函数可以推移时间戳，但不能改变数据。（ ）
17. 当获得包含时间信息的时间戳时（如‘2019-07-12 12:56:31’），如果想要生成标准化为零点的时间戳（如‘2019-07-12’），可以将normalize设置为True。（ ）
18. 不可以使用不包含在时间序列中的时间戳进行切片。（ ）
19. 将高频率的数据聚合到低频率成为向上采样，将低频率数据转换到高频率数据称为向下采样。（ ）
20. 进行重新采样时，将W-WED转换到W-FRI属于向下采样（ ）
21. 运行代码pd.to\_datetime(['2000-1-1','31-12-2000','8-31-2000'])会报错（ ）
22. 当把datetime类型的数据作为索引时，数据类型变为DatetimeIndex。（ ）
23. timedelta可以查看days和seconds而不能查看minutes。（ ）
24. resample函数得loffset参数可以用shift实现（ ）
25. 向上采样中，目标频率是原频率得子区间（ ）
26. date\_range（）函数生产日期时间数据时，如果以6天为间隔可以设置参数freq为‘6D’。( )
27. value = '2011-01-03'

datetime.strptime(value, '%Y-%m-%d')和datetime.strptime(value, '%Y/%m/%d')都可以正确运行 ( )

1. datatime的格式%Y显示的是四位的年份（ ）
2. ~~resample 的方法参数中，convention的默认值为‘start’（ ）~~ 作废
3. pandas的通用时间序列是不规则的，即时间序列的频率不是固定的 （ ）

# 第12章

1. 假设df是一个含有’fruit’列的Dataframe对象，则fruits\_cat=df['fruit'].astype('category')得到的fruits\_cat是一个Numpy数组。（ ）
2. 通过pandas.qcut分箱函数生成的Categories对象的类别已经指定了顺序（ ）
3. 一个分类数组可以包含任一不可变的值类型（ ）
4. 分类数据一定是字符串。（ ）
5. 将特定的大型数据集进行分析时，将数据转换为分类数据会大幅提高性能（ ）
6. 对于Series中的分类方法，rename\_categories和set\_categories都能用新类别替换现有类别，但前者不能改变类别的数量，而后者可以添加或删除类别（ ）
7. 对一个Categorical进行set\_categories操作，赋值给一个新的对象，不会影响到新对象的类别。（ ）
8. pipe方法的一个潜在用途是将操作的序列泛化成可复用的函数（ ）
9. pandas.qcut生成的结果的数据类型是pandas.Categorical（ ）
10. df2=df.copy(),df2['k']=v与df2=df.assign(k=v)是等价的（ ）
11. values = pd.Series(['apple', 'orange', 'apple','apple'] \* 2)，pd.unique(values)返回的结果是FALSE （ ）
12. transform可以产生一个标量值，并广播到各分组的尺寸数据中。（ ）
13. Categorical对象的codes属性对应的dtype为float64（ ）
14. transform函数不可以改变它的输入（ ）
15. 分类转换会自动指定类别的顺序。（ ）
16. Categorical对象比Series使用更少的内存，在对象上进行操作也更快。（ ）
17. resample是一种基于时间分段的分组操作。 （ ）
18. 使用TimeGrouper时，时间必须是Series或者DataFrame索引 （ ）
19. DataFrame.assign方法可以按指定修改原DataFrame（）
20. cat\_s=pd.Series(pd.Categorical.from\_codes([0,1,0,0,1,0],['a','b','c','d']))，可以执行cat\_s.remove\_unused\_categories()来去除所有没有出现在数据中的类别（ ）
21. 假设df是一个含有’fruit’列的Dataframe对象，则fruits\_cat = df['fruit'].astype('category')得到的fruits\_cat是一个Numpy数组。（ ）
22. 通过pandas.qcut分箱函数生成的Categories对象的类别已经指定了顺序（ ）
23. take函数得到的Series的索引是values的值。（ ）
24. ~~category的字符串类别输出时不带单引号。（ ）~~作废
25. pandas的TimeGrouper可以使用Grouper替代（ ）
26. remove\_categories将移除类别所对应的值。（ ）
27. Categorical对象比Series使用更少的内存，在对象上进行操作也明显更快。（ ）
28. g.transform('len') 可以运行。（ ）
29. 对一个Categorical进行set\_categories操作，赋值给一个新的对象，会影响到新对象的类别。（ ）
30. df['fruit'].astype('category')返回的是Numpy数组 （ ）
31. Categories对象的categories属性表示引用维度表的整数数组，codes属性表示维度表的观测值。 （ ）