# 第2~3章

1. 执行后可以查看Python的版本的是 （ ）
   1. import sys ; print(sys.version)
   2. import system; print(system.version)
   3. import system; print(system.Version)
   4. import sys; print(sys.Version)
2. 已知x=5，三元表达式'a' if x>=0 else 'b'的结果为（ ）

A.’a’ B.’b’ C.5 D.0

1. 下列哪种不是Python元组的定义方式? （ ）

A. (1,) B. (1) C. (1, 2) D. (1, 2, (3, 4))

1. 下列代码的输出是（ ）

color = ['red', 'blue', 'green', 'red', 'yellow']

color.append('black');color.insert(1,'red') ;

color.remove('red') ;print(color)

1. ['blue', 'green', 'yellow', 'black']
2. ['red', 'blue', 'green', 'red', 'yellow', 'black']
3. ['red', ‘red’, 'blue', 'green', 'red', 'yellow', 'black']
4. ['blue', 'green', 'red', 'yellow', 'black']
5. 对于列表test\_list = [5, 4, 3, 2, 1]而言，下列不能输出[1, 2, 3, 4, 5]的是（ ）
   1. A. test\_list[::-1] B. list(reversed(test\_list))
   2. C. sorted(test\_list) D. list[0:-5:1]
6. 对于以下二元操作符，其描述为a的b次方的是（ ）

A. a^b B. a\*\*b C. a\*b D. a//b

1. 存在一个列表为a = [1, 2, 3]，下列的操作中错误的是（ ）

A.a + 4 B. a + [4] C.a + a D. a \* 4

1. 对于下列代码

a=1; type(a); bool(a) 输出的结果为（ ）

A. int;True B. float;True C. int;False D. float;False

1. 设a=[2, 3, 7]; b=list(a)，执行a is b 和a == b得到的输出分别是（ ）

A. False,False B. False,True

C. True,False D. True,True

1. 下列代码的输出是（ ）

t = 2

def a(x):

return x \*\* t

print(a(a(3)))

A. 9 B. 27 C.81 D.243

1. 下面不可以作为字典的键的是 （ ）

A. tuple([1,2,3,4]) B.(1, 2, (1, 2, 3)) C.(1, 2, [1, 2]) D.1

1. 对于代码 a=[1,2,3]; b=a ; a.append(4); c=list(a) 以下输出结果为False的是 （ ）

A、a == c B、a is c C、a == b D、a is b

1. 下列代码a = '{0:.2f}{0:d}{1:s}'.format(int('10'), 5.0, 5) ; print(a) 执行结果为（ ）

A.10.00105.0 B.10.0055 C.105.05 D．会报错

1. 对于以下代码 a = “Hello world” ;a[0] = “h” ; print(a) 运行结果为（ ）

A. “hello world” B. “hHello world” C. “Hhello world” D. 报错

1. 设seq = [7, 2, 3, 7,5, 6, 0, 1]，执行seq[3:4] = [6, 3]之后，seq的值是（ ）

A. [7, 2, 3, 6, 3, 5, 6, 0, 1]

B. [7,2, 6, 3, 7，5, 6, 0, 1]

C. [7, 2, 3, 6, 5, 6, 0, 1]

D. [7,2, 6, 3, 5, 6, 0, 1]

1. 以下是不可变对象的是（ ）

A．列表 B.字典 C.Numpy数组 D.字符串

1. 代码

a=[1,2]; b=[3,4,5] ; c=[6,7,8,9]

len(tuple(zip(a,b,c)))的输出为（ ）

A、1 B、2 C、3 D、4

1. 设a=[1, 2, 3],执行b=a;a.append(3) 之后输出b的值是（ ）

A. [1, 2, 3] B. [1, 2] C. [1, 2, 3, 3] D. [1, 2, 3, 3, 3]

1. 下列代码的输出是（ ）

list = [11, 12, 13, 12, 15, 16, 13, 14] a = set(list)

[x for x in a if x > 11]

A. [12, 13, 12, 15, 16, 13, 14] B. [11, 12, 13, 12, 15, 16, 13, 14]

C. [12, 13, 15, 16, 14] D.[12, 13, 14, 15, 16]

1. 下面对列表a = [1, 2, 3, 4, 5, 6, 7, 8, 9]的操作，得到的结果正确的是（ ）

A. c = a[::-2], c为[1, 3, 5, 7, 9]

B. c = a[::3], c为[1, 4, 7]

C. c = a[:-1], c为[1, 2, 3, 4, 5, 6, 7, 8, 9]

D. c = a[-4:], c为[5, 6, 7, 8, 9]

1. 下面对于列表的操作方法的叙述，错误的是（ ）

A. 使用insert方法可以将元素插入到指定的列表位置。

B. 使用pop方法可以将特定位置的元素移除并返回。

C. 使用append方法可以将元素添加到列表的尾部。

D. 使用remove方法可以移除列表中所有符合要求的值。

1. 下面对字典d = {'a' : 'some value', 'b' : [1, 2, 3, 4]}的操作的叙述，错误的是（ ）。

A. 'a' in d, 返回True.

B. 'some value' in d, 返回True

C. print(d.pop('a')),输出结果为some value

D. d.update({'b': 1}),d改变为{'a': 'some value', 'b': 1}

1. 对于以下代码 a = (3, 4, (5, 6)); a[1] = ‘four’运行结果为（ ）

A. (‘four’, 4, (5, 6)) B. (3, 4, ‘four’, (5, 6))

C. TypeError D. (3, ‘four’, (5, 6))

1. 下面代码的结果是：

list = [lambda: i for i in range(10)]

result = list[0]()

print (result)

A. 0 B. 1 C. 9 D. 10

1. 下面代码的结果是（ ）

a = [['a', 'b', 'a', 'i'], ['i','b', 'b', 'c'],

['c', 'a','i', 'c'],['a','b','c','c','i','a']]

b = [i for i, x in enumerate(a) if x.index('i') > 1]

print(b)

A.[0,2,3]

B.[0,1,2,3]

C.[['a', 'b', 'a', 'i'],['c', 'a','i', 'c'],['a','b','c','c','i','a']]

D.[['a', 'b', 'a', 'i'], ['i','b', 'b', 'c'],['c', 'a','i', 'c'],['a','b','c','c','i','a']]

1. 执行下面代码的结果是( )

seq1=[('foo','bar'),('one','two')]

a,b=zip(\*seq1)

print(a)

A.['foo',''bar'] B.('foo','bar')

C.['foo','one'] D.('foo','one')

1. 以下代码的输出结果为( )

seq1=['a','b','c']

seq2=['A','B','C']

combine = zip(seq1,seq2)

first,last=zip(\*combine)

first

A. (‘a’, ‘A’ ) B.( ‘A’, ‘a’)

C.( ‘a’, ‘b’, ‘c’) D. (‘A’, ‘B’, ‘C’)

1. 下列哪种函式参数定义不合法?（ ）

　　A. def myfunc(\*args):

　　B. def myfunc(arg1=1):

　　C. def myfunc(\*args, a=1):

D. def myfunc(\*\*args,a=1):

1. a.union(b)和哪个是一样的（ ）

A．a | b B.a&b C.a||b D.a&&b

1. list=[5, 9, 12, 3, 10, 8, 16, 1], list[::-3]的结果是( )

A.[1, 8, 3, 9] B.[8, 16, 1] C.[1, 10, 9] D.[5, 3, 16]

1. 取集合a和集合b中的所有不重复元素的操作符是()

A. - B. | C.& D.^

1. 以下代码的运行结果为：( )

c = [1, 8, 2, 2, 3, 4, 7]

bisect.insort(c, 6)

c

A. [1, 2, 2, 3, 4, 6, 7, 8]

B. [1, 6, 8, 2, 2, 3, 4, 7]

C. [1, 8, 2, 2, 3, 4, 6, 7]

D. Error

1. 关于Python组合数据类型，以下描述错误的是：( )

A. Python的字符串、元组和列表类型都属于序列类型

B. 组合数据类型能够将多个相同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序、更容易

C. 组合数据类型可以分为3类：序列类型、集合类型和映射类型

D. 序列类型是二维元素向量，元素之间存在先后关系，通过序号访问

1. 下列代码的的运行结果为：( )

4>3>2>1

4>(3>2)>1

A. True True

B. True False

C. False False

D. False True

1. 下列代码的运行结果为：( )

tup1 =( 4 ,'#', 'for') + ( 1 ,[6 , 0] ) \* 2 + tuple ( 'bar' )

tup1

A. ( 4 , '#' , 'for' , 1 , [ 6 , 0 ] , 1 , [ 6 , 0 ] , 'b' , 'a' , 'r' )

B. ( 4 , '#' , 'f' , 'o' , 'r' , 1 , [ 6 , 0 ] , 1 , [ 6 , 0 ] , 'b' , 'a' , 'r' )

C. ( 4 , '#' , 'for' , 1 , [ 6 , 0 ] , 1 , [ 6 , 0 ] , 'bar' )

D. Error

1. a = [1, 2, 3, None, (), [], ]

print(len(a))

以上代码输出结果为( )

A. 4

B. 5

C. 6

D. 7

1. 以下不能创建一个字典的语句是：（）

A.d1 = {'a':[1,3,5],'b':128}

B.d1 = dict('a'=[1,2,3])

C.d1 = dict.fromkeys(['a','b','c'])

D. seq1 = ['a','b','c','d']

seq2 = [123,'second',True,[1,3,5]]

d1 = dict(zip(seq1,seq2))

1. 下列语句结果为： ( )

a = [1,2,3];b=[4,5,6];a+3\*b

A. [1,2,3,12,15,18]

B. [1,2,3,4,5,6,4,5,6,4,5,6]

C. [1,2,3,4,5,6,1,2,3,4,5,6]

D. 输出错误

1. 以下哪个结果为2（）

A) 5/2 B) 5\2 C) 5//2 D 5%2

# 第4章

1. 代码arr=np.arange(1,10).reshape(3,3);arr[1:2,:].shape 的输出是（ ）

A、(2, 3) B、(1, 2) C、(1, 3) D、(2, 2)

1. 代码arr=np.arange(1,10).reshape(3,3);arr[[0,-1,-2]] 的输出是（ ）
   * 1. array([[1, 2, 3], [7, 8, 9],[4, 5, 6]]) B、array([[7, 8, 9],[7, 8, 9],[4, 5, 6]]
     2. array([[1, 2, 3],[1, 2, 3], [4, 5, 6]]) D、array([[7, 8, 9],[1, 2, 3],[4, 5, 6]])
2. 计算numpy中元素个数的方法（ ）
   * 1. np.sqrt() B、np.size() C、np.identity() D、np.itemsize()
3. NumPy中没有以下哪个功能？（ ）

A.随机数生成 B.矩阵计算 C.生成DataFrame数据结构 D.傅里叶变换

1. 以下结果为TRUE的是（ ）

A. np.nan == np.nan B. np.inf > np.nan

C. 0.3 == 3 \* 0.1 D. 0 \* np.nan!=0

1. 已知arr=np.arange(9).reshape(3,3), 执行arr.cumsum(axis=0)得到的结果是（ ）

A.36 B.array([3,12,21])

C.array([[0,1,2],[3,5,7],[9,12,15]]) D.array([[0,1,3],[3,7,12],[6,13,21]])

1. 设bools=np.array([False,False,True,False]),执行bools.any()和bools.all()得到的输出分别是（ ）

A.True,False B.True,True

C.False,True D.False,False

1. 假设arr = np.array([[1, 2 ,3], [4, 5, 6]])，下列哪项可计算arr每一列的平均值？（ ）。

A，arr.mean() B，arr.mean(1) C，arr.mean(0) D，arr.mean(-1)

1. NumPy中的ndim属性表示的意思是（ ）

A.元素类型 B.秩 C.每一维度的数量 D.元素的值

1. 根据输出结果，下列①处的代码应该为（ ）

import numpy as np;

arr = np.arange(12).reshape(3, 4);

① ;

print(arr)。输出：

[[ 0 1 2 3]

[12 12 6 7]

[ 8 9 10 11]]

A. arr[1][:2] = 12 B. arr[1][1:2] = 12

C. arr[1,1:2] = 12 D. arr[1,2] = 12

1. a = [1, 2, 3]， c = np.array(a)，以下对a和c的操作，错误的是：（ ）。

A，c\*\*2 B，c + 1 C，a\*2 D，a + 1

1. numpy中向量转成矩阵使用( )

A、reshape B、reval C arange D、random

1. data1=[6,7.5,8,0,1];arr1=np.array(data1) 则arr1的输出结果是：( )

A.array([6. , 7.5, 8. , 0. , 1. ])

B.array([6 , 7, 8 , 0 , 1 ])

C.array([6 , 8, 8. , 0. , 1. ])

D.array([6 , 7.5, 8 , 0 , 1 ])

1. 下列代码的输出是( )

import numpy as np

data = np.random.randn(3, 4);data1 = data[1].copy();

data1[0:2] = 12;data1[2:4] = 13;print(data1)

A. [12. 12. 13. 13.] B. [12. 12. 12. 13.]

C.[[12. 12.] [13. 13.]] D.发生错误，没有输出

1. arr=np.arange(12).reshape(4,3)，切片操作arr[[1],:3]返回的结果是（ ）

A. array([[0,1, 2]]) B. array([[3, 4]])

C.array([[1, 2]]) D. array([[3, 4, 5]])

1. 对于一个numpy对象data = np.arange(12).reshape(4, 3)，下列代码可以得到[[ 7]

[10]]

切片的是（ ）

A.data[2: 4, 1: 2] B. data[2: 4][1: 2]

C.data[2: 4, 1: 3] D. data[2: 3, 1: 2]

1. 以下可以创建数组[1.,3.,5.]的是（ ）

A.numpy.arange(1,7) B.numpy.arange(1.,7)

C.numpy.arange(1.,7,2) D.numpy.arange(1.,5.,2)

1. 如果 x = np.array([[1., 2.], [4., 5.]])，则print(x\*x)的结果为（ ）

A. [[1.2.] [4. 5.]] B. [[ 1. 4.][16.25.]]

C. [[9., 12.] [24., 33.]] D. [[1. ,8.][8., 25.]]

1. 代码

arr1=np.arange(10) arr2=arr1[5:8] arr2[:]=123

arr1 的输出是（）

A、array([ 0, 1, 2, 3, 4, 123, 6, 7, 8, 9])

B、array([ 0, 1, 2, 3, 4, 123, 123, 123, 8, 9])

C、array([ 0, 1, 2, 3, 4, 5, 123, 7, 8, 9])

D、array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

1. 已知arr=np.arange(1,10).reshape(3,3),执行arr[:,:1]得到的是（ ）

A.array([[1]]) B.array([1,4,7])

C.array([1,2,3]) D.array([[1],[4],[7]])

1. 已知c= np.arange(24).reshape(3,4,2) 那么c.sum(axis = 0)所得的结果为（）

A、array([[12, 16],[44, 48],[76, 80]]) (列0，行1)

B、array([[ 1, 5, 9, 13],[17, 21, 25, 29],[33, 37, 41, 45]])

C、array([[24, 27], [30, 33],[36, 39],[42, 45]])

1. 下列对于ndarray的运算属性的叙述，错误的是：（ ）。

A,.shape表示ndarray对象的尺度，对于矩阵，表示它有几行几列。

B,.size表示ndarray对象元素的个数。

C,.type表示ndarray对象的元素类型。

D,.itemsize表示ndarray对象中每个元素的大小，以字节为单位。

1. 假设arr = np.array([[1,2 ,3], [4, 5, 6]])，下列关于arr的叙述，错误的是：（ ）。

A，执行arr[:1],arr，输出为array([[1, 2, 3],[4, 5, 6]])。

B，执行arr[0:10],arr，会报错。

C，执行arr[:][:1],arr输出结果与执行arr[:,:1],arr输出结果不相同。

D，执行arr[1,1] = 1,arr，输出结果为array([[1, 2, 3],[4, 1, 6]])。

1. 假设arr = np.array([[1,2 ,3], [4, 5, 6]])，下列关于arr的叙述，错误的是：（ ）。

A，arr[arr<5] = 0,arr，输出结果为array([[0, 0, 0],[0, 5, 6]])。

B，arr[arr<5]，输出结果为array([1, 2, 3, 4])。

C，arr[~arr<4]，输出结果为array([4, 5, 6])。

D，arr[[0],[0,1]]，输出结果为array([1, 2])。

1. 对于下面代码

arr = np.arange(5)

arr\_slice1 = arr[0:3]

arr\_slice1[0] = 6

arr\_slice2 = arr[0:3].copy()

arr\_slice2[0] = 0

arr

运行之后得到的结果为（）

A. [0,1,2,3,4] ; [0,1,2,3,4]

B. [6,1,2,3,4] ; [6,1,2,3,4]

C. [0,1,2,3,4] ; [6,1,2,3,4]

D. [6,1,2,3,4] ; [0,1,2,3,4]

1. 对于np.sort()方法，以下说法正确的是（ ）

A. 该方法不允许传递参数

B. 该方法是对原数组按位置排序

C. 该方法返回的是已经排序好的数组拷贝

D. 该方法不适用于多维数组

1. 下面代码的输出结果是 ( )

arr=np.arange(12).reshape(3,4)

arr[1,:].shape

A.(4,1) B.(1,) C.(4,) D.(1,4)

1. 下面代码的输出结果是（ ）

sample1 = [[1,1,1],[2,2,2],[3,3,3]]

sample2 = np.array([[1,1,1],[2,2,2],[3,3,3]])

arr1 = np.array(sample1)

arr2 = np.array(sample2)

sample1[1] = [-2,-2,-2]

sample2[1] = [-2,-2,-2]

arr1

arr2

A.array([[1,1,1],[-2,-2,-2],[3,3,3]])

array([[1,1,1],[-2,-2,-2],[3,3,3]])

B.array([[1,1,1],[2,2,2],[3,3,3]])

D.array([[1,1,1],[-2,-2,-2],[3,3,3]])

array([[1,1,1],[2,2,2],[3,3,3]])

1. 下面代码的输出结果是（ ）

a=np.array( [[[1,1],[2,2],[3,3]],

[[4,4],[5,5],[6,6]],

[[7,7],[8,8],[9,9]],

[[10,10],[11,11],[12,12]]])

a.mean(0)

A

[[[ 1 1]

[ 2 2]

[ 3 3]]

[[ 4 4]

[ 5 5]

[ 6 6]]

[[ 7 7]

[ 8 8]

[ 9 9]]

[[10 10]

[11 11]

[12 12]]]

B

[[5.5 5.5]

[6.5 6.5]

[7.5 7.5]]

C

[[ 2. 2.]

[ 5. 5.]

[ 8. 8.]

[11. 11.]]

D

[[ 1. 2. 3.]

[ 4. 5. 6.]

[ 7. 8. 9.]

[10. 11. 12.]]

1. 下面代码的输出结果是( )

arr1 = np.array(['Mike', 'Bob'])

arr2 = np.array(['Bob', 'Alice'])

print(np.union1d(arr1, arr2))

A.['Alice' 'Bob' 'Mike']

B.['Alice', 'Bob', 'Mike']

C.['Mike' 'Bob' 'Alice']

D.['Mike', 'Bob', 'Alice']

1. 输入np.abs(12+5j)的结果为？ （ ）

A、12 B、13 C、5 D、17

1. 假设x是长度为m的向量，y是长度为n的向量，则输入 X,Y=np.meshgrid(x,y) 最终生成的矩阵X和Y的维度是 ( )

A、m;n B、mn；nm C、mn;mn D、nm;nm

1. 以下程序的运行结果是（ ）

import numpy as np

x = np.array([1,2,3,4,5], dtype = np.int8)

print (x.itemsize)

A、1 B、8 C、56 D、64

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

A.[3,5,7,1,2] B. [3,5,7,1] C. [3,5,7] D.[1,2]

1. 计算NumPy中元素个数的方法（ ）

A、 np.sqrt( ) B、np.size( ) C、np.identity( )

1. arr = np.arange(32).reshape((8,4)) + 1；arr[[1,5,7,2],[0,3,1,2]]输出的结果是：

A． array([5,24,30,11])

B． array([4,23,29,10])

C． array([[ 5, 8, 6, 7], [21, 24, 22, 23], [29, 32, 30, 31],

[ 9, 12, 10, 11]])

1. 已知c= np.arange(24).reshape(3,4,2) 那么c.sum(axis = 0)所得的结果为（）

A、array([[12, 16],[44, 48],[76, 80]])

B、array([[ 1, 5, 9, 13],[17, 21, 25, 29],[33, 37, 41, 45]])

C、array([[24, 27], [30, 33],[36, 39],[42, 45]])

1. a = np.arange(20).reshape(4,5)

a[[0,2,3],[1,2,3]]

输出结果是（ ）

A. array([[ 1, 2, 3],

[11, 12, 13],

[16, 17, 18]])

B. array([ 1, 12,18])

1. 对array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])，array[:, :-1]的结果是

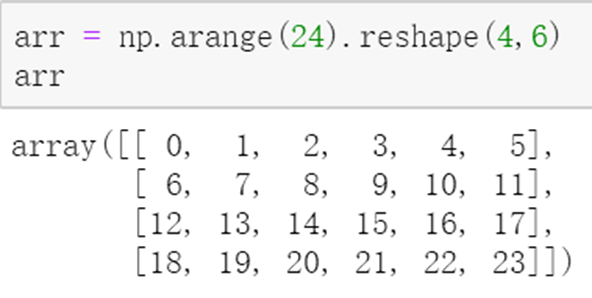
A.array([[1, 2, 3], [4 5, 6]])

B.array([[1, 2], [4, 5]])

C.array([[1, 2], [4, 5], [7, 8]])

D.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

1. 有下面4×6的二维数组：



则 arr [ [ 3 , -2 ] ] [ : , [ 1 , 0 , 3 ] ] 的输出结果为：

A.array ( [ [ 13, 12, 14 ]

[ 19, 18, 20 ] ] )

B.array ( [ [ 13, 12, 15]

[ 19, 18, 21] ] )

C.array ( [ [ 19, 18 ,20 ]

[ 13, 12, 14] ] )

D.array ( [ [ 19, 18, 21]

[ 13, 12, 15] ] )

1. arr=np.array([1.2, 2.4, 3.8, 4.5], dtype=np.int32);arr 输出的结果为（ ）

A. array([1., 2., 3., 4.])

B.[1., 2., 3., 4.]

C.[1., 2., 4., 5.]

D.array([1., 2., 4., 5.])

1. 执行以下代码的结果是( )

import numpy as np

arr = np.arrange(5)

arr\_slice = arr[2:4]

arr\_slice = 10086

arr

A.array([0, 1, 2, 3, 4])

B.array([0,1,10086,10086,4])

C.array([0,1,10086,10086,10086])

D.array([0,1,10086,4])

# 第5章

1. 下列哪个函数用来展示DataFrame的基础信息和它的统计数据（ ）

A.dataFrame.describe() B.dataFrame.head()

C.dataFrame.info D.dataFrame.values

1. 以下时间不能被pd.to\_datetime()函数转化为xxxx-xx-xx格式的是（ ）

A.'01 Jan 2010' B.'02-02-2011' C.'20120303' D.都可以

1. 对于一个Series对象而言，若其中某一行含有缺失值NaN，在使用命令sort\_value()之后，该行将会（ ）

A.自动被删除 B.移到最前面 C.移到最后面 D.位置不作改变

1. 下列不属于DataFrame构造函数的有效输入的是：（ ）

A， Series构成的字典 B，字典构成的字典

C，字典或者元组构成的列表 D，字典或者Series构成的元组

1. 对于一个DataFrame对象，里面有一列数据名称叫做year，下列哪一条语句不能成功筛选出year这一列（ ）

A. frame.year B. frame['year']

C. frame.T.iloc[0] D. frame.loc[‘year’]

1. 假设frame = pd.DataFrame([[1,2],[3,4]])，以下操作可以将frame的行索引，列索引都改变为‘a’， ‘b’的是：（ ）

A， frame.reindex(['a','b'])

B， frame.reindex(columns = ['a','b'])

C， frame.reindex(columns = ['a','b'], index = ['a','b'])

D， frame.loc[['a', 'b'], ['a', 'b']]

1. 假设df1，df2为两个DataFrame对象，则以下操作能运行的是：（ ）（多选）

A，df1 + df2 B，1 / df1 C，np.exp(df1) D，df1%10

1. 已知myarr= np.arange(12).reshape(3,4)，下列会报错的是（ ）

A．myarr[[0,1]] B. myarr[[0,1],[1]]

C. myarr[[0,1],[1,2]] D. myarr[[0,1],[1,2,3]]

1. obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])；obj.index=['a', 'd', 'b', 'e'] ；obj[’a’]的输出结果是（ ）

A．a -5.3 B.a 4.5 C.a 7.2 D.a 3.6

1. 下列代码的输出是（ ）

data2 = pd.Series(np.arange(5), index = ['a', 'b', 'c', 'd', 'e'])；data2['b': 'd'] = 2.；print(data2)

A、a 0.0 b 2.0 c 2.0 d 2.0 e 4.0

B、a 1.0 b 2.0 c 2.0 d 2.0 e 5.0

C、a 0.0 b 2.0 c 2.0 d 3.0 e 4.0

D、a 1.0 b 2.0 c 2.0 d 2.0 e 4.0

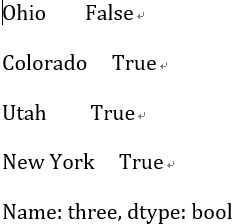
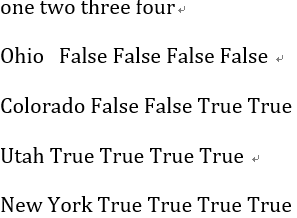
1. DataFrame构造函数中的有效输入不包括（ ）

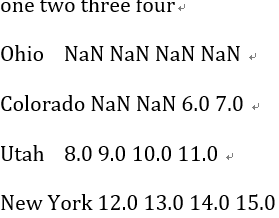
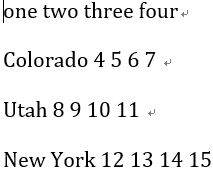
A.数据、列表或元组组成的字典 B.原有的DataFrame

C.多个元组 D.Series构成的字典

1. 对于代码

data=pd.DataFrame(np.arange(16).reshape(4,4),index=['Ohio','Colorado','Utah','New York'],columns=['one','two','three','four'])，data[data['three']>=5]的输出为（ ）

A、 B、

C、 D、

1. 下列对pandas中的索引对象的叙述，错误的是：（ ）

A、索引对象是不可变的，因此用户是无法修改索引对象的。

B、可以用'pop' in frame.columns查看pop是否为DataFrame对象frame中的列。

C、在构造Series或者DataFrame时，所使用的任意数组或者标签序列都可以在内部转化为索引对象。

D、与python集合一样，pandas索引也不可以包含重复标签。

1. 执行frame = pd.DataFrame([[1,2,3],[3,4,6]]),frame.drop(0, axis = 1, inplace = False),frame，输出结果为：（ ）

A.  B.  C.  D. 

1. 对于一个新的DataFrame对象，删除名字为’score’的这一列的操作是（ ）

A. frame.drop('score')

B. frame.T.drop('score').T

C. del frame('score', axis = 1)

D. del frame('score')

1. 已知df=pd.DataFrame([[1,np.nan],[2,3],[np.nan,np.nan],[4,5]],

index=['a','b','c','d'],columns=['one','two']),

执行df.sum()得到的结果是（ ）

A. a 1.0

b 5.0

c 0.0

d 9.0

B. one 7.0

two 8.0

C. 15.0

D. one 7.0

two 15.0

1. 以下哪个嵌套字典能生成该DataFrame对象（ ）

Num Val

2019 1 2

2020 3 4

A. {‘2019’:{‘Num’:1, ‘Val’:2}, ‘2020’:{‘Num’:3, ‘Val’:4}}

B. {‘Num’:{2019:1, 2020:3}, ‘Val’:{2019:2, 2020:4}}

C. {{1,2}:{2019:’Num’, 2019:’Val’}, {3,4}:{2020:’Num’, 2020:’Val’}}

D. 不能通过嵌套字典产生DataFrame对象

1. 关于DataFrame排序，以下方法是按照列标签的降序排序的是（ ）

A. frame.sort\_index(axis=0,ascending=True)

B. frame.sort\_index(axis=0,ascending=False)

C. frame.sort\_index(axis=1,ascending=False)

D. frame.sort\_inxex(axis=1,ascending=True)

1. 下面代码，返回的结果为（ ）

data = np.arange(16).reshape(4,4)

df1 = pd.DataFrame(data)

df2 = pd.DataFrame(df1)

df3 = pd.DataFrame(data)

df4 = pd.DataFrame(data)

print(df1 is df2,df1.index is df2.index,df3.index is df4.index)

A.True Ture True

B.False True False

C.True False True

D.False False False

1. numpy中Series的默认数据类型是（ ）

A.float64 B.int64 C.object D.bool

1. 下列说法错误的是（ ）

A.pandas中的索引对象是不可变的，因此用户无法修改索引对象。

B.在DataFrame中，reindex可改变行索引、列索引、也可同时改变二者。

C.在reindex方法的参数中，method=‘ffill’为后向填充，‘bfill’为前向填充。

D.普通的python切片是不包含尾部的，Series切片包含尾部。

1. 下列哪种方法是计算两个索引的交集？( )

A.difference B.intersection C.union D.append

1. obj=pd.Series(np.arange(4),index=['x1','x2','x3','x4'])

obj['x1':'x3']的输出结果是（ ）

A.

x1 0

x2 1

dtype: int32

B.

x1 0

x2 1

x3 2

dtype: int32

C.

x1 0

x3 2

dtype: int32

1. 下面语句能输出DataFrame第一列前三行的数据的是（ ）

data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],

'year': [2000, 2001, 2002, 2001, 2002, 2003],

'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}

frame = pd.DataFrame(data)

A. print(frame.state[:2])

B. print(frame.iloc[0:2,[0]])

C. print(frame.loc[0:2,'state'])

D. print(frame.iloc[0:2,0])

1. 已知

import pandas as pd

s2=pd.Series([10.5,20,38,40])

s2.index=['a','b','c','d']

下列不正确的排序方法是：（ ）

A.s2.sort\_index(ascending=False)

B.s2.sort\_index(axis=1)

C.s2.sort\_index(axis=0)

1. pandas中，用于表示缺失值的是( )

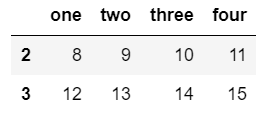
[空] B.NaN C.Null D.None

1. 有下面DataFrame，

data = pd.DataFrame( np.arange(16).reshape((4,4)),

columns = ['one', 'two', 'three', 'four'])

下列代码中能输出结果

的是（ ）

A. data[ three >5 ] B. data[ ' three ' > 5]

C. data[ data['three'] >6] D. data [2 : 3 ]

1. aSer = pd.Series([1,3,5], index = ['a', 'b', 'c']),以下哪一项不能获取第0个和第2个元素（）

A.aSer[[0,2]]

B.aSer[['a','c']]

C.aSer[['0','c']]

D.以上都可以

1. obj2a = pd.Series([4,7,-5,1],index=['d','a','c','f'])，obja=pd.Series([4,7,-5,1],index=['a','b','c','d'])

以下表述错误的是（ ）

A. 执行：obja['a':'h']

输出：

a 4

b 7

c -5

d 1

dtype: int64

B. 执行obj2a['a':'h']

输出：

a 7

c -5

d 1

dtype: int64

C. 执行：obja['a':'d']

输出：

a 4

b 7

c -5

d 1

dtype: int64

D. 执行obj2a['a':'f']

输出：

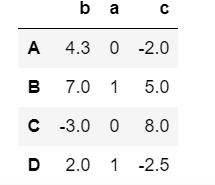
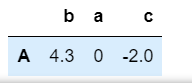
a 7

c -5

d 1

dtype: int64

1. 有如下DataFrame—frame:

则要得到

代码不可以是（ ）(多选)

A. frame[:’A’]

B. frame[:1]

C. frame[0]

D. frame.iloc[0,:]

E. frame.iloc[:1,:]

# 第6章

1. Response对象的json方法返回的是（ ）

A、JSON对象 B、字典 C、DataFrame D、Series

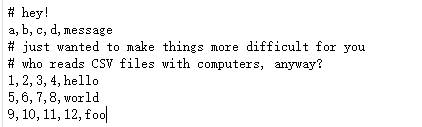
1. 以下表示输出的数据结构是 列->值 的形式的式子是（ ）

A. data.to\_json(orient='columns')

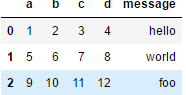
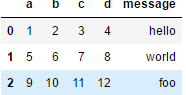
B. data.to\_json(orient='records')

C. data.to\_json(orient='index')

D. data.to\_json(orient='split')

1. 对于文件example.csv  执行下列代码后结果为（ ）

import pandas as pd;pd.read\_csv('example.csv', skiprows=[0, 1, 2, 3], names = ['a' , 'b', 'c', 'd','e'])

A. ​​ ​​  B. C. ​​ ​​  D.报错

1. 以下对read\_csv/read\_table函数参数叙述错误的是（ ）。

A、skiprows表示从文件开头处起，需要跳过的行数或者行号列表。

B、na\_values指定需要用NA替换的值序列。

C、sep或delimiter指定用于分隔每行字段的字符序列或者正则表达式。

D、names 是结果的列名列表，它不可以和header参数一起用。

1. 以下对于csv文件ex2.csv的读取和操作，错误的是：（ ）

A、pd.read\_csv('ex2.csv',header=None)。

B、pd.read\_csv('ex2.csv',header=1)

C、pd.read\_csv('ex2.csv',names=['a', 'b', 'c', 'd', 'index'], header = None)

D、pd.read\_csv('ex2.csv',names=['a', 'b', 'c', 'd', 'index'], header = [0,1])

1. 以下函数中，可以从Excel的XLS或XLSX文件中读取表格数据的是（ ）

A.read\_csv B.read\_sql C.read\_html D.read\_excel

1. 以下对read\_csv/read\_table函数参数叙述错误的是（ ）。

A、skiprows表示从文件开头处起，需要跳过的行数或者行号列表。

B、na\_values指定需要用NA替换的值序列。

C、sep或delimiter指定用于分隔每行字段的字符序列或者正则表达式。

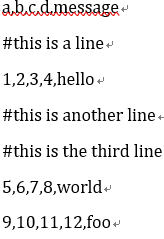
D、names 是结果的列名列表，它不可以和header参数一起用。

1. 在Python中，我们处理大文件时，可以通过以下哪条语句对pandas的显示设置进行调整（ ）

A. pd.max\_rows =10 B. pd.options.display.max\_rows=10

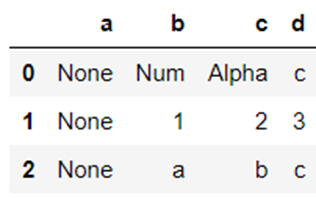
C. pd.rows = 10 D. pd.display = 10

1. 下面关于read\_csv/read\_table函数参数的说法，错误的是（ ）
   1. skiprows指需要跳过的行数或行号列表，skiprows=1表明要跳过第一行
   2. na\_values是需要用NA替换的值序列
   3. chunksize是用于迭代的块的大小
   4. encoding是Unicode文本编码
2. 假设ex.csv文件为同目录下的csv文件，文件内容为：

下列哪一段语句可以得到以下的结果（ ）

* 1. f = pd.read\_csv('ex1.csv', skiprows = [1, 3, 4], index\_col = 'message')；print(f)
  2. f = pd.read\_csv('ex1.csv', skiprows = [1, 3, 4], index = 'message')；print(f)
  3. f = pd.read\_csv('ex1.csv', skiprows = [2, 4, 5], index\_col = 'message')；print(f)
  4. f = pd.read\_csv('ex1.csv', skiprows = [2, 4, 5], index = 'message')；print(f)

1. 以下不是HDF5格式的优势的是（ ）
   1. HDF5支持多种压缩器的即时压缩
   2. 能更高效地存储重复模式数据
   3. 可以将HDF5看作一种数据库
   4. 可以高效地分块读写，从而处理无法直接写入内存的较大的数据集
2. 如何使得下图名为data的dataFrame中的第二行第二个数字1变为空值。( )



A. data[1,1]=Null

B. data[1,1]=None

C. data['b',1]=None

D. data['b'][1]=None

1. pandas中的read\_table函数的默认分隔符为( )

A. \n B. \t

C. \a D. \v

1. 以下哪个函数参数可以用于跳过某些指定行打印输出？( )

A. skiprows B. delimiter C. nrows D. squeeze

1. 对于一些比较大的文件，我们想要一次读取一小部分，或者每次迭代一小部分。如果只是想要读取前5行（不读取整个文件），以下哪个选项是最佳选择：（ ）

A. pd.read\_csv('../examples/ex6.csv', chunksize=5)

B. pd.read\_csv('../examples/ex6.csv', nrows=5)

C. pd.read\_csv('../examples/ex6.csv', rows=5)

D. pd.read\_csv('../examples/ex6.csv', size=5)

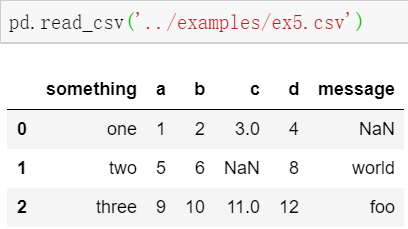
1. 下列参数用于描述行终止符的是:( )

A. delimiter B. lineterminator C. quotechar D. quoting

1. ~~下面哪些解析函数不是合法的（ ）~~//作废

~~A.to\_csv B.to\_json C.read\_fwf~~

1. 有以下文件

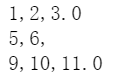
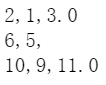


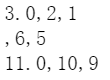
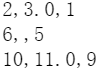
执行代码：

import sys

data.to\_csv(sys.stdout, index=False, header=False,columns=['b', 'c', 'a'])

控制台的打印的文本结果为：( )

A. B. 

C.  D. 

1. 读入文件’File.csv'。自定义列名为'a','b','c'.( )

A.pd.read\_csv('File.csv','a','b','c')

B.pd.read\_csv('File.csv',names=['a','b','c'])

C.pd.read\_csv('File.csv',column=['a','b','c'])

D.pd.read\_csv('File.csv',['a','b','c'])

1. 以下使用pandas读取csv文件的操作，会报错的是（ ）

A.pd.read\_csv('../examples/ex1.csv',header=None)

B.pd.read\_csv('../examples/ex1.csv',names=None)

C.pd.read\_csv('../examples/ex1.csv',header=['a','b','c','d','e'])

D.pd.read\_csv('../examples/ex1.csv',names=['a','b','c','d','e'])

1. 以下对于csv文件examples/ex2.csv的读取和操作，错误的是：（ ）

A.pd.read\_csv(r'examples/ex2.csv',header=None)。

B.pd.read\_csv(r'examples/ex2.csv',

header=1)

C.pd.read\_csv(r'examples/ex2.csv',names=['a', 'b', 'c', 'd', 'index'], header = None)

D.pd.read\_csv(r'examples/ex2.csv',names=['a', 'b', 'c', 'd', 'index'], header = [0,1])

1. 以下函数中，可以从txt文件中读取数据的是（ ）

A.read\_csv B.read\_sql C.read\_html D.read\_excel

# 第7章

1. 在replace()方法的使用中，下列不正确的是（ ）

A. replace(-2, 999) B. replace([-1, -2], np.nan)

C. replace([-2, -4], [np.nan, 999]) D. replace((1, 2), 999)

1. DataFrame的（ ）方法返回一个布尔值的Series,这个Series反应的是每一行是否存在重复。

A.duplicated B.fillna C.replace D.drop\_duplicates

1. 已知代码import numpy as np;import pandas as pd; string\_data = pd.Series(['aardvark',1, np.nan, 'avocado'])，下列将string\_data中的值1替换为NaN的操作正确的是：（ ）

A， string\_data[1] = None B， string\_data[1] = np.nan

C， string\_data[1] = NA D，string\_data[1] = nan

1. 以下返回的是data的视图的是（ ）

A. data.dropna() B. data[data.notnull()]

C. data.iloc[[0,2,4]] D. data[1:3]

1. data=pd.DataFrame([[2,2,2,NA],[1.,NA,NA,NA],[NA,NA,NA,NA],[NA,1,1,NA]])，下列哪个方法能删掉值全为NA的列？（ ）

A.data.dropna() B.data.dropna(how='all')

C.data.dropna(axis=1,how='all') D.data.dropna(axis=1,how='any')

1. data = pd.Series([1., np.nan, 2., np.nan, -1000., 3.])，以下对于data的操作，输出结果相同的是：（ ）。

①data.replace(np.nan, 0) ②data.fillna(0) ③data.replace([np.nan, -1000], [0, -1000])

④data.replace({np.nan: 0})

A，① B，①② C，①②③ D，①②③④

1. 对于data = pd.Series([1., NA, 5., NA, 6.])，使用fillna(data.mean())之后，原来NA处的代码处应该填入什么数字？（ ）

A.5.， 6. B.1.， 5. C.4.， 4. D.2.4.，2.4

1. 以下不能替换对象中的值的是（ ）

A、map B、fillna C、inplace D、replace

1. 下列哪个函数可以判断数据中是否存在空值(例如df['首映地点'])（ ）

A.df['首映地点'].dropna() B.df['首映地点'].isnull()

C.df['首映地点'].fillna() D.以上都不对

1. 假设字符串val = 'a,b,guido'，下列叙述错误的是：（ ）。

A，val.index(',')返回1，val.find(',')返回1.

B，'|'.join(val)输出结果为'a|b|guido'。

C，val.count(',')返回2. D，val.rfind(',')返回3.

1. 对于pandas中的分箱操作，下列说法错误的是（ ）
   1. pd.value\_counts()函数是输出分箱结果中每个区间所对应的计数
   2. 可以通过向labels选项传递一个列表或数组来传入自定义的箱名
   3. qcut是基于样本分位数进行分箱的函数
   4. qcut函数可以传入整数代替显式的箱边来获取等长的箱，而cut函数却不能
2. 以下哪个数据清洗方法能将数据按照元素进行转换（ ）

A.duplicated B.map C.fillna D.dropna

1. 怎么查看一份数据中某一列的数据格式？（例如df["年份"]）（ ）

A.df["年份"].dtype() B.df["年份"].astype()

C.df["年份"].values D.以上都不对

1. 对于一些Python内建字符串的方法，以下说法错误的是（ ）
   1. split方法可以将字符串按照传入的子字符进行拆分
   2. find方法用于检测子字符串
   3. count方法用于返回某个特定子字符串在字符串中出现的次数
   4. index方法在字符串中没有找到时返回-1
2. 下列代码的输出是（ ）

import pandas as pd

data = pd.DataFrame({'one': [3, 4] \* 2 + [4],

'two': [1, 2, 3, 4, 5]})

print(data)

A. one two

0 3 1

1 4 2

2 3 3

3 4 4

4 4 5

B. one two

0 3 1

1 4 2

2 3 3

3 4 4

C.

0 3 1

1 4 2

2 3 3

3 4 4

4 4 5

D.

0 3 1

1 4 2

2 3 3

3 4 4

1. 下列代码代表什么意思？（ ）

import pandas as pd

data = pd.DataFrame({'k1': ['one', 'two'] \* 3 + ['two'],

'k2': [1, 1, 2, 3, 3, 4, 4]})

print(data[data.duplicated(keep='last') == True])

A.将data中的值筛选一遍，将没有重复的值输出

B.将data中第一个重复的值输出

C.将data中最后一个筛选的值输出

D.将data中除了最后一个重复的值之外的重复值进行输出

1. string\_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])，string\_data.notnull()的结果是（ ）

A. 0 aardvark

1 artichoke

2 np.nan

3 avocado

B. 0 aardvark

1 artichoke

2 avocado

C. 0 True

1 True

2 False

3 True

D. 0 True

1 True

2 True

1. 代码data=pd.DataFrame({'k1':['one','two']\*3+['two'],

'k2':[1,1,2,3,3,4,4]})

data.drop\_duplicates()的输出为( )

A、

k1 k2

0 one 1

1 two 1

2 one 2

3 two 3

4 one 3

5 two 4

B、

k1 k2

0 one 1

1 two 1

2 one 2

3 two 3

4 one 3

6 two 4

C、

k1 k2

0 one 1

1 two 1

D、

k1 k2

0 one 1

2 one 2

3 two 3

5 two 4

1. 已知df=pd.DataFrame([[1,None,None],[4,None,6],[7,8,9]])，执行df.dropna(thresh=2)得到的结果是（ ）

A.

0 1 2

0 1 NaN NaN

1 4 NaN 6.0

2 7 8.0 9.0

B.

0 1 2

1 4 NaN 6.0

2 7 8.0 9.0

C.

0 1 2

2 7 8.0 9.0

D.

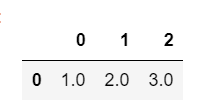
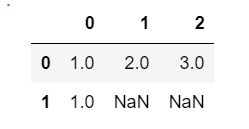
0 1 2

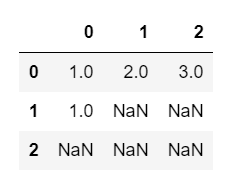
0 1 NaN NaN

2 7 8.0 9.0

1. data=pd.DataFrame([[1.0,2.0,3.0],[1.0,NA,NA],[NA,NA,NA]])

则cleaned=data.dropna()的输出结果是：( )

A. B. 

C. 

1. 下列代码输出为：（ ）

import numpy as np

import pandas as pd

a = None

b = np.nan

c = ' '

d = pd.Series([a,b,c,])

d.isnull()

A， 

B， 

C， 

D, 

1. 假设data = pd.DataFrame([[1., 6.5, 3.], [1., NA, NA],

[NA, NA, NA], [NA, 6.5, 3.]])

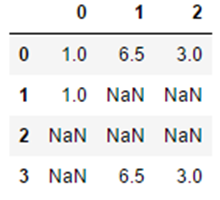
data.dropna() ①

data.dropna(how='all') ②

data.dropna(axis=1, how='all') ③

data.dropna(axis=1) ④

以上①②③④4个操作，哪个操作后得到的结果如下图：（ ）。



A，① B，② C，③ D，④

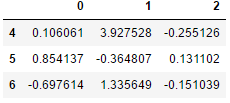
1. 以下代码的输出结果为（ ）。

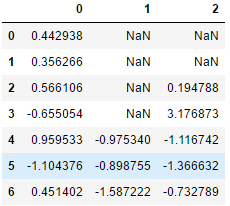
df = pd.DataFrame(np.random.randn(7, 3))

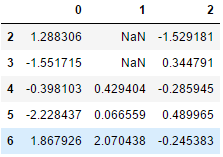
df.iloc[:4, 1] = NA

df.iloc[:2, 2] = NA

df.dropna(thresh=3)

A，

B，

C，

D，

1. 以下对于duplicated方法和drop\_duplicates方法的叙述，错误的是：（ ）（不定项选择题）

A， duplicated返回的是一个布尔值Series，反映每一行是否与之前出现过的行相同。

B， 对于DataFrame对象data，data.drop\_duplicates的结果与data[data.duplicated()]相同。

C， duplicated方法和drop\_duplicates方法默认都是保留第一个观测到的值。

D，在duplicated方法和drop\_duplicates方法中传入参数keep = last将会保留最后一个观测值到的与前面行重复的行。

1. data = pd.Series([1., np.nan, 2., np.nan, -1000., 3.])，以下对于data的操作，输出结果相同的是：（ ）。

①data.replace(np.nan, 0)

②data.fillna(0)

③data.replace([np.nan, -1000], [0, -1000])

④data.replace({np.nan: 0})

A，① B，①② C，①②③ D，①②③④

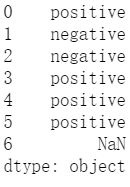
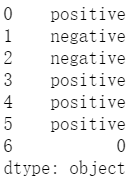
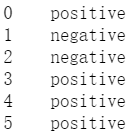
1. 运行下列代码

ser = pd.Series([1,-1,-1,1,1,1,0])

match = {1:'positive',-1:'negative'}

ser.map(match)

输出的结果为（ )

1. 
2. 
3. 报错
4. 
5. 下面代码实现删除所有重复值的是( )

A. data.drop\_duplicates(keep=False)

B. data.drop\_duplicates(keep='first')

C. data.drop\_duplicates(keep='False')

D. data.drop\_duplicates(keep='last')

1. 下列哪个方法返回的不是boolean Series（ ）

A.data.duplicated() B data.isnull()

C.data>data.median() D.data[data > data.median()]

1. 下列方法能将字符串中所有非重叠匹配模式以迭代器形式返回的是（ ）

A.findall B.finditer C.match D.search

1. 以下关于离散化和分箱说法不正确的是（ ）

A. 使用pandas中的cut可以将数据列表分箱成左闭右开的一定数量区间

B．使用pandas中的cut时，可以通过向labels选项传递列表或数组来传入箱

C. 使用pandas中的qcut可以获得等长的箱子

1. 下列说法正确的是（ ）

A：df.fillna的method为bfill时，以na值的下面一个值填充。为fill时，以na值的上一个值填充

B：df.fillna的limit参数的作用是只填充na值个数小于等于limit的行

C：要将data[‘food’]列中字符串转换为小写的方法是：data[‘food’].lower()

D：drop\_duplicates返回的是DataFrame，内容是duplicated返回数组中为False的部分，且默认为保留第一个观测到的值。

1. 已知val='a,c, emmm'。val.find(',')的结果是什么（ ）

A.1 B.-1 C.ValueError D.2

# 第8章

1. 下列哪个方法可以改变索引的层次？（ ）

A.reindex() B.swaplevel() C.change\_level() D.reset\_index()

1. 用哪个Numpy库的函数可以将多个NumPy数组进行堆叠（ ）

A.concatenate B.concat C.merge D.join

1. 默认情况下，用merge方法连接两张表做的是（ ）。

A. 内连接 B. 外连接 C. 左连接 D.右连接

1. 假设有DataFrame 对象frame，下列操作会改变frame的是（ ）

(1)frame.index.names = ['a']

(2)frame.stack() (3)frame. swaplevel(0,1)

A，（1） B，（2） C，（3） D，（1）（2）（3）

1. 对于堆叠与拆堆操作，以下说法错误的是（ ）
   1. 如果层级中的所有值并未包含于每个子分组中时，拆分可能会引入缺失值
   2. 默认情况下，堆叠会过滤出缺失值
   3. 在DataFrame中拆堆时，被拆堆的层级会变为结果中最低的层级
   4. 堆叠和拆堆操作是不可逆的
2. 下列哪个方法可以对dataframe数据进行转置。（ ）

A. swaplevel() B.reset\_index() C.dataframe.T D.以上都不是

1. 默认情况下，用merge方法连接两张表做的是（ ）。

A. 内连接 B. 外连接 C. 左连接 D.右连接

1. 假设有双层索引DataFrame对象data，则下列代码的输出结果是：（ ）。

data1 = data.unstack().stack();data is data1

A. True B.False C. 报错 D.不能确定

1. 下列哪个方法可以取消dataframe层次化索引。( )

A. swaplevel() B.reset\_index() C.dataframe.T D. set\_index()

1. 对于两个Series对象a和b。

a = pd.Series([np.nan,2.5,0.0,3.5,4.5,np.nan]);

b = pd.Series([0.,np.nan,2.,np.nan,np.nan,5.])

运行下行代码 pd.Series(np.where(pd.isnull(a),b,a))，下面哪个选项输出的结果与该行代码输出的结果相同（ ）

A.a.combine\_first(b) B.pd.Series(np.where(pd.isnull(b),a,b))

C.pd.Series(np.where(pd.isnull(a),a,b)) D.pd.Series(np.where(pd.isnull(b),b,a))

1. 对于concat()方法，下列的说法错误的是？（ ）

A. concat()默认情况下是在axis=0轴向进行连接的。

B. concat()可以通过join\_axes方法来指定用于连接的轴。

C. concat()在对Series对象在axis=1时进行连接得到的还是一个Series对象。

D. concat()之后得到的对象如果是存在多级索引时可以进行unstack()操作。

1. 以下merge函数的参数how的取值范围，其中是将两个表共有的键进行连接操作的是（ ）

A.inner B.left C.outer D.right

1. 对于DataFrame

frame = pd.DataFrame({'a':range(7),'b':range(7,0,-1),

'c':['one','one','one','two','two',

'two','two'],

'd':[0,1,2,0,1,2,3]})

代码frame.set\_index(['c','d'])的输出为（ ）

A、

a b c d

0 0 7 one 0

1 1 6 one 1

2 2 5 one 2

3 3 4 two 0

4 4 3 two 1

5 5 2 two 2

6 6 1 two 3

B、

a b

c d

one 0 0 7

1 1 6

2 2 5

two 0 3 4

1 4 3

2 5 2

3 6 1

C

c d a b

0 one 0 0 7

1 one 1 1 6

2 one 2 2 5

3 two 0 3 4

4 two 1 4 3

5 two 2 5 2

6 two 3 6 1

D、

a b c d

c d

one 0 0 7 one 0

1 1 6 one 1

2 2 5 one 2

two 0 3 4 two 0

1 4 3 two 1

2 5 2 two 2

3 6 1 two 3

1. 设有data=pd.Series(range(7),index=[['a','a','a','b','b','c','c'],[1,2,3,1,2,1,2]])，执行data.loc[['a','c']]得到的结果是（ ）

A. a 1 0

2 1

3 2

b 1 3

2 4

c 1 5

2 6

B. a 1 0

c 1 5

C. a 1 0

2 1

c 1 5

2 6

D. a 1 0

2 1

3 2

c 1 5

2 6

1. 列关于DataFrame中的set\_index函数的叙述，错误的是：（ ）。

A， 使用set\_index函数会生成一个新的DataFrame。

B， 使用set\_index方法，默认情况下，被用作索引的列将会自动从DataFrame中移除。

C， 可以改变参数drop的值为True，将被用做索引的列留在DataFrame中。

D，set\_index的反操作reset\_index将分层索引的索引层级移动到列中

1. 下列对pandas中的concat方法的叙述，错误的是：（ ）

A， concat方法中的参数join\_axes可以用来指定用于连接其他轴向的轴。

B， 可以用concat方法中的参数keys通过在连接轴向上创建一个多层索引来区分结果中拼接在一起的各部分。

C， 默认情况下，concat方法是沿着axis = 1轴向生效的。

D，concat方法可以用于连接多个DataFrame。

1. 对于concat函数的参数，以下说法错误的是（ ）

objs参数用于指定需要连接的对象列表或字典

keys参数用于指定多层索引的层级

axis参数用于指定连接的轴向，默认为0轴

join参数用于指定连接的方式

1. 以下使用merge函数连接左右DataFrame(两者都有两个列key1和key2)的操作正确的是( )

pd.merge(left,right,left\_on='key1');

pd.merge(left,right,right\_on='key1');

pd.merge(left,right,left\_on='key1',right\_index=True);

pd.merge(left,right,left\_on='key1',right\_on='key2');

1. 对于下列代码

data = pd.Series(np.random.randn(6),

index=[['a','a','b','b','b','c'],[1,2,1,2,3,2]])

data['a':'b']

下列选项中输出结果不同的是( )

A.data[['a','b']]

B.data.loc[['a','b']]

C.data.loc[['a','b'],:3]

D.data[:2]

1. 已知result是Series，数据如下

state number

Ohio one 0

two 1

three 2

Colorado one 3

two 4

three 5

下列语句结果不同于其他三项的是（ ）

A.result.unstack()

B.result.unstack(0)

C.result.unstack('number')

D.result.unstack(1)

1. 下列关于merge方法说法正确的是：（）

A. 默认情况下，merge做的是外连接

B. 有指定在哪一列上进行连接时，即连接键信息没有指定时，merge会自动将重叠的列名作为连接的键

C. merge方法的内连接，结果是两张表的并集，外连接是两张表的交集

D．以上都不对

1. 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']])

frame.index.names = ['key1', 'key2']

frame.columns.names = ['state', 'color']

下列代码可以正确输出的是：（）

A. frame['state']

B. frame['key1']

C. frame['Green']

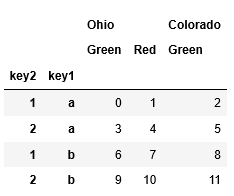
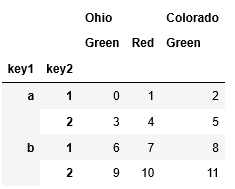
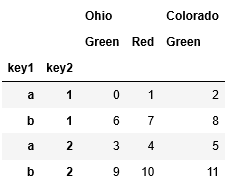
D. frame['Colorado']

1. 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']])

frame.index.names = ['key1', 'key2']

frame.sort\_index(level=0)的结果是（ ）

1. 
2. 
3. 
4. 下列关于concat函数的参数说法正确的是：（ ）

A.axis的轴向默认是0沿着行方向

B.objs不是必选参数

C.join默认是‘inner’

1. 取消DataFrame的多层索引可以使用哪个方法（ ）

A swaplevel（） B.reset\_index Cconcat()

1. 选出下面正确的一项（ ）

A.DataFrame的set\_index函数会在原来的DataFrame上进行修改

B.一般情况下，merge是进行外连接

C.用sort\_index()只能在单一层级上对数据进行排序

1. 执行下列代码的效果是：( )

import pandas as pd

import numpy as np

df = pd.DataFrame(np.arange(72).reshape((8,9)))

df.sub(df.mean(axis=1), axis=0)

A.输出各行的均值。

B.输出每列原来的数值减去每行数据的平均值。（如第一列的数据减去第一行的均值）

C.输出每行原来的数值减去所有数据的平均值。

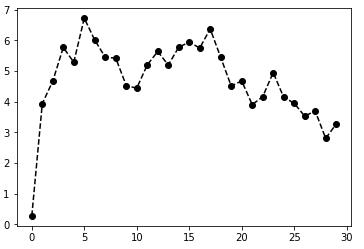
D.输出每行原来的数值减去该行数据的平均值。

# 第9章

1. 在创建了子图之后，接连输入两条绘图命令plt.plot()，会出现什么情况？（ ）
   1. matplotlib会在最后一个图片和子图上展示两条命令的图像。
   2. matplotlib会在最后一个子图上展示最后一条的图像，在倒数第二个子图上展示倒数第二条的图像。
   3. 报错。
   4. matplotlib只会在最后一个图片和子图上展示最后一条命令的图像。
2. 在图标中添加图形时，需要用到的命令是？（ ）

A.add\_subplot() B.add\_patch() C.figure() D.set\_title()

1. 下面这个图符合下列哪个语句的样式？（ ）



* 1. plt.plot(np.random.randn(30).cumsum(), 'ko--')
  2. plt.plot(np.random.randn(30).cumsum(), 'ko-')
  3. plt.plot(np.random.randn(30).cumsum(), 'k--')
  4. plt.plot(np.random.randn(30).cumsum(), 'k-')

1. 下面在matplotlib中不表示颜色的是（ ）

A.b C.k D.o D.r

1. 以下哪个参数用于旋转Series绘图的刻度标签（ ）

A、rotation B、rot C、xticks D、yticks

1. 下面在pandas中用于绘制密度图的是（ ）

A.line（） B.barh() C.hist() D.density

1. 以下哪种方法用于绘制水平柱状图（ ）

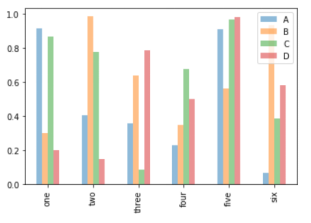
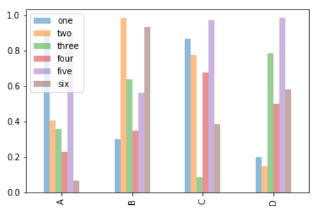
A、line B、bar C、barh D、hist

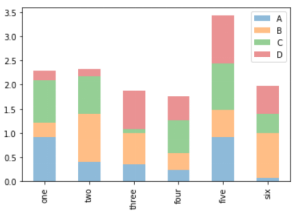
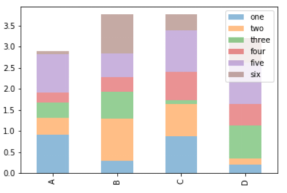
1. 一下哪种方法绘制的是直方图（ ）

A.scatter B.line C.density D.hist

1. df=pd.DataFrame(np.random.rand(6,4),index=['one','two','three','four','five','six'],

columns=['A','B','C','D'])代码df.plot.bar(alpha=0.5,stacked=True)的结果为（ ）

A、 B、

C、 D、

1. seaborn中的哪个函数可以绘制散点图（ ）

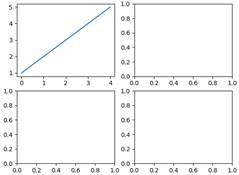
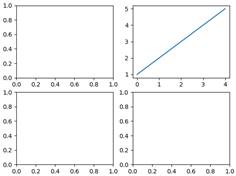
distplot B.regplot C.barplot D.pairplot

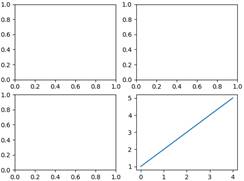
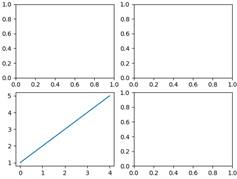
1. 以下哪个函数用于为每条线生成一个区分的图例（ ）

A.plt.plot B.plt.legend C.plt.figure D.plt.subplots

1. 对于seaborn，以下说法错误的是（ ）
   1. seaborn可以提高matplotlib图表的可读性和美观性
   2. seaborn中的绘图函数使用一个data参数来传入数据
   3. seaborn中的distplot用于绘制直方图和连续密度估计
   4. seaborn不适用与在绘图前需要聚合或汇总的数据
2. 关于Series和DataFrame的plot方法的参数，以下说法错误的是（ ）
   1. alpha用于传入绘图所用的子图对象
   2. sort\_columns用于将各列按字母顺序绘制
   3. legend用于生成子图图例
   4. xlim，ylim分别用于设置x轴范围和y轴范围
3. 以下代码的输出结果为（ ）。

fig, axes = plt.subplots(2,2) plt.plot([1,2,3,4,5])

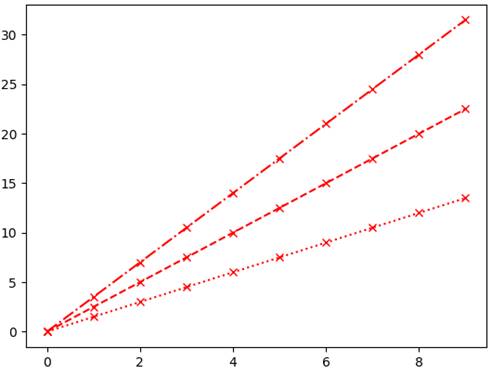
A、 B，

C，D，

1. 若要在散点图中比较直观地展示点的密集程度，最好改变下列哪一个值？（ ）

A.alpha B.style C.kind D.lopy

1. 下列代码的①②③处应该填入下列代码中的哪几个，才能得到下图（ ）。



import numpy as np

import matplotlib.pyplot as plt

a = np.arange(10)

plt.plot(a, a\*1.5, ①, a, a\*2.5,②, a, a\*3.5, ③)

plt.show()

A，'r:x'，'r--x'，'r-.x'

B，'r:x'，'r-.x'，'r--x'

C，'r-.x'，'r--x'，'r:x'

D，'r-.x'，'r:x'，'r--x'

1. 下列format\_string，表示绿色虚线的是（ ）。

A，'g:' B，'g-.' C，'g..' D，'g--'

1. 对于DataFrame对象df，df.plot()与下列哪条语句是等价的。（ ）。

A，df.bar()

B，df.plot().line()

C，df.line()

D，df.plot.line()

1. plot中可以使用哪个函数作柱状图（ ）。

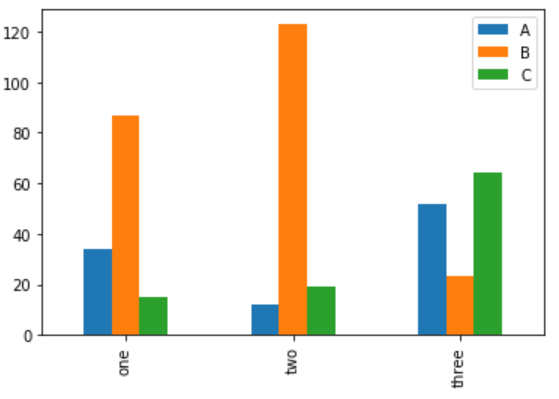
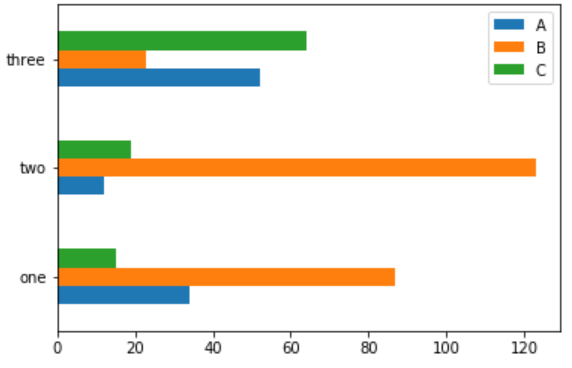
A，Bar() B，barh() C，hist() D，Barh

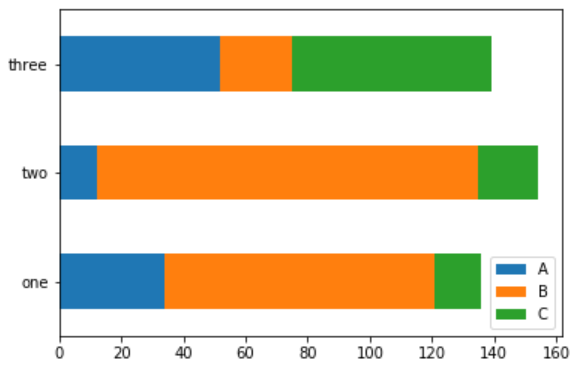
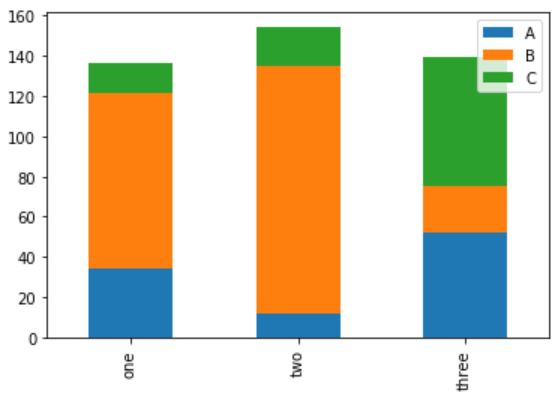
1. 运行下列代码

df = pd.DataFrame({'A':[34,12,52],'B':[87,123,23],'C':[15,19,64]},index=['one','two','three'])

df.plot.bar()

得到的可视化结果为（）

A. B. 

C. D. 

1. 运行下列代码

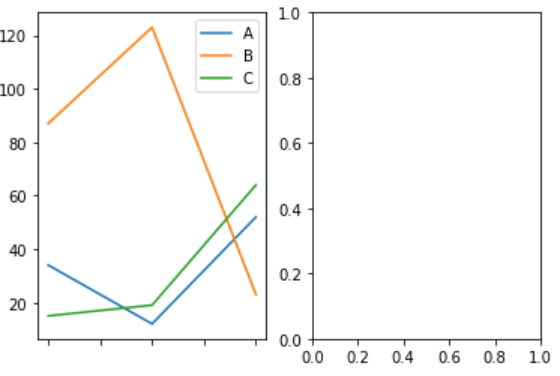
fig,axes=plt.subplots(1,2)

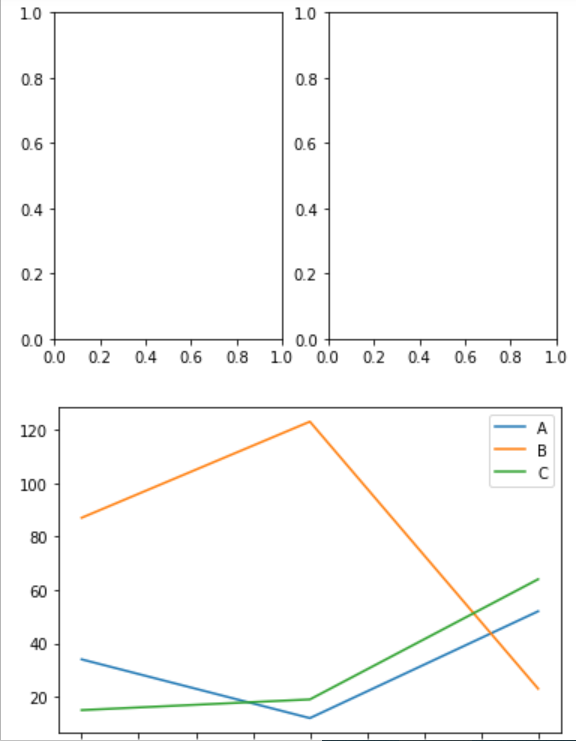
df = pd.DataFrame({'A':[34,12,52],'B':[87,123,23],'C':[15,19,64]},

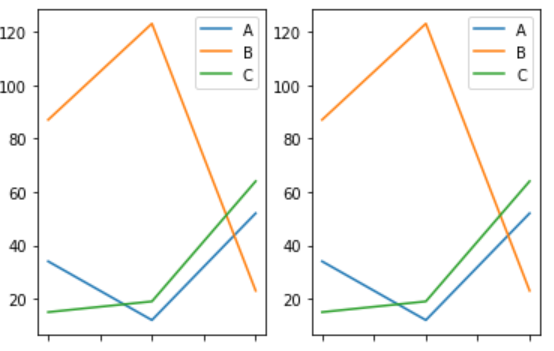
index=['one','two','three'])

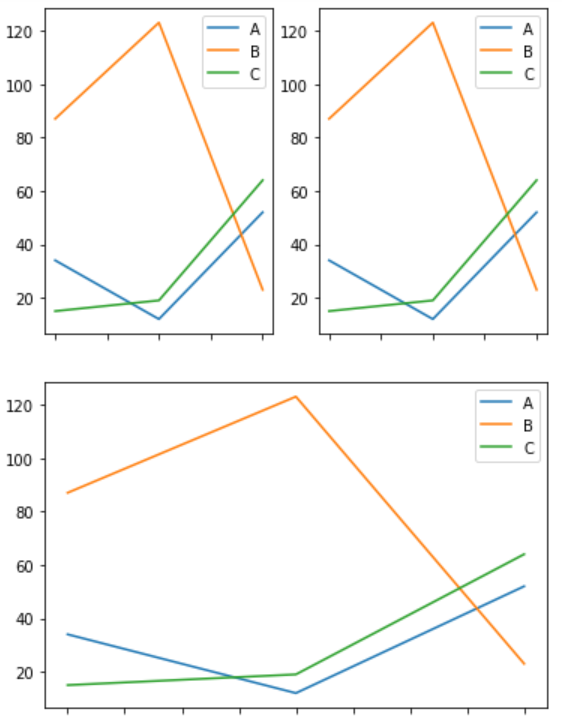
df.plot()

输出的结果为（ ）

A. 

B. 

C. 

D. 

1. 通过fig,axes=plt.subplots(2,2)生成的子图，不可以通过以下方式调用：

A.axes[1]

B.axes[0,0]

C.fig[0,0]

D.fig.axes[0]

1. 下列关于可视化的说法，错误的是（）

A.plt.legend用于为每一条线生成一个区分的图例

B.plt.savefig方法可以将图片写入到所有的文件型对象中

C.默认情况下，Series和DataFrame的plot()方法绘制的是条形图

D.seaborn中的绘图函数使用data参数来传入数据

1. 下列说法错误的是（）

A.使用jupyter notebook画复杂图表必须将绘图命令放在单个单元格里

B.matplotlib会自动检查标签是否重叠

C.Series.plot.grid方法展示轴网络默认是打开的

D.Series和DataFrame的索引可以用来绘制轴刻度

1. 下面说法正确的是：（）

A. ax1=fig.add\_subplot(2,2,0)是可以运行的

B. 绘制图形时，其中使用到的alpha参数用于调节图像透明的

C. plot.bar()和plot.barg()函数分别用于绘制水平的和垂直的柱状图

1. 下面哪一条语句可以为DataFrame生成堆积柱状图？( )

df.plot.bar(stacked=True, alpha=0.5)

df.plot.bar(stack=True, alpha=0.5)

df.plot.bar(stacked=False, alpha=0.5)

df.plot.bar(stack=False, alpha=0.5)

1. 下哪个说法是正确的？（ ）

A.add()方法可以往图表中添加图形

B.如果想取消图例中的元素，可以选择不要传入label参数或传入label = '\_nolegend'

C.注释只可以用于添加文本

D.只能将绘制的图片保存为格式为.svg的文件

1. 以下哪条执行语句输出结果和plt.plot(np.random.randn(30).cumsum(), 'ko--')不同。( )

A. plt.plot(np.random.randn(30).cumsum(), color='k', linestyle='dashed', marker='o')

B. plt.plot(np.random.randn(30).cumsum(), color='k', '--o')

C. plt.plot(np.random.randn(30).cumsum(), 'ok--')

D. plt.plot(np.random.randn(30).cumsum(), 'o--k')

# 第10章

1. 以下不可以作为Groupby的参数的（ ）

A. 函数 B.字典 C.series D.具体的值

1. groupby没有以下哪个方法？（ ）

A. count() B.peak\_to\_peak C.prod() D.std()

1. 假设有如下两组数据：

df\_USA=df[df.产地=="美国"] df\_CN=df[df.产地=="中国大陆"]

下列哪个函数能把两组数据合成一组数据 ( )

A.df\_CN.append(df\_USA) B.df\_CN.groupby(df\_USA) C.df\_CN.cut(df\_USA)

1. 下列可以用于DataFrame中分组的是（ ）。

①DataFrame中的一个或者一组列 ②字典或者Series

③函数 ④索引级别

A，① B，①② C，①②③ D，①②③④

1. pivot\_table的默认聚合类型是（ ）。

A， 分组平均值 B， 各个分组的中位数

C， 各个分组的方差 D，各个分组的值的类和

1. 下面方法可以计算分组中的非NA值的乘积的是（ ）

A，std B，count C，prod D，median

1. groupby没有以下哪个方法？（ ）

A. count() B.std() C.prod() D.get\_max()

1. 下列对于分组与聚合的叙述，正确的是（ ）。

A、只能用一个键对数据进行分组。

B、分组键可以是正确长度的任何数组。

C、默认情况下，一个DataFrame中所有的列都可以聚合。

D、mean方法可以计算GroupBy对象中各个分组中所有数值的均值。

1. 假设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)})

以下那条语句可以根据key1标签计算data1列的均值。（ ）

* + - 1. df['data1'].mean()
      2. df['data1'].groupby(df['key1']).mean()
      3. df['data1'].groupby('key1').mean()
      4. df['data1'].median()

1. groupby没有以下哪个方法？（ ）

A. count() B.std() C.prod() D.get\_max()

1. 关于agg方法，以下说法错误的是（ ）
   1. agg方法不能实现将不同的函数应用到不同的列
   2. agg方法可以传入包含多个函数的列表
   3. 通过向agg方法传入（name,function)元组的列表来实现对结果列名的修改
   4. agg方法可以传入自定义函数
2. 关于pivot\_table的选项，以下说法错误的是（ ）
   1. pivot\_table默认的聚合类型为平均值
   2. dropna选项用于在结果表中替换缺失值的值
   3. pivot\_table默认情况下聚合所有数值型的列
   4. margins用于添加行/列小计和总计
3. 假设有DataFrame对象df，key1是df中的列，下列代码不可以运行的是（ ）
   * + 1. means = df['data1'].groupby(df).mean()
       2. grouped = df.groupby(df.dtypes, axis=1)
       3. df.groupby(['key1']).mean()
       4. pieces = dict(list(df.groupby('key1')))
4. 已知DataFrame

people=pd.DataFrame(np.random.randn(5,5),

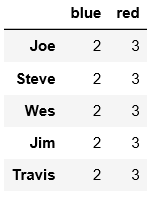
columns=['a','b','c','d','e'],

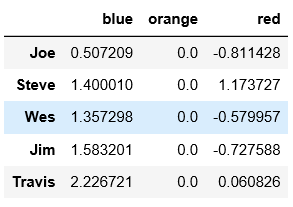
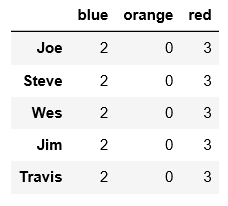
index=['Joe','Steve','Wes','Jim','Travis'])

mapping={'a':'red','b':'red','c':'blue',

'd':'blue','e':'red','f':'orange'}

则代码people.groupby(mapping,axis=1).sum()的输出可能为（ ）

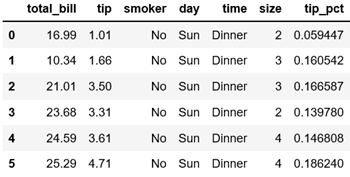
A、B、

C、D、

1. pandas.crosstable函数计算的是分组中的（ ）

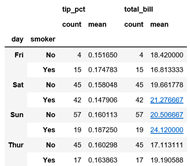
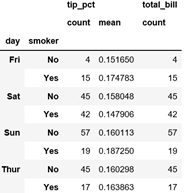
A、累积和 B、算术平均值 C、频率 D、算术中位数

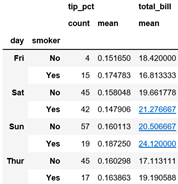
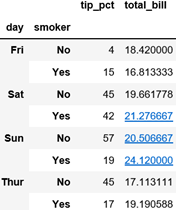
1. 有数据集tips，列出了其前6行的数据



执行grouped=tips.groupby(['day','smoker']),

grouped['tip\_pct','total\_bill'].agg(['count','mean'])后得到的输出是（ ）

A. B. 

C. D.

1. 下面代码的输出结果为：

def fun(data):

return data.max() // 2

data = pd.DataFrame({'key':['a', 'a', 'b', 'b'],'data1':[2, 2, 4, 4],'data2':range(4)})

data.groupby('key').agg(fun).max().mean()

A.0.5 B. 1 C.1.5 D. 2

1. 下列选项中错误的是（ ）

A、grouby可以在任意轴向上分组

B、默认情况下，groupby在axis=1的轴向上分组。

C、分组信息不一以数组形式存在

D、cut返回的Categorical对象可以直接传递给groupby

1. 对于一个GroupBy对象，如果你要将min函数应用到size列上，将std函数应用到tip列上，应该执行下列哪个语句？

A. grouped.agg([('size',np.min),('tip','std')])

B. grouped.agg({'size': np.min, 'tip': std})

C. grouped.agg({'size': np.min, 'tip': 'std'})

D. grouped.agg([('size',np.min),('tip',std)])

# 第11章

1. 下面哪个时间序列频率字符串表示“每月第二个星期天”（ ）

A，Q-SUN B，W-SUN C，WOM-2SUN D，BQ-SUN

1. 若start = datetime(2011, 1, 7)，则start + timedelta(12, 12, 12)的结果是（ ）

A，datetime.datetime(2011, 1, 19, 0, 0, 12, 12)

B，datetime.datetime(2011, 1, 19, 0, 12, 12)

C，datetime.datetime(2011, 1, 19, 12, 12)

D，datetime.datetime(2011, 1, 19, 12, 0, 12)

1. 下列基础的时间序列频率表示每分钟的是（ ）

A，D B，T C，M D，BM

1. 下列对于date\_range方法的叙述，正确的是（ ）

A、使用date\_range方法的时候，必须要传入一个起始日期和一个结尾日期。

B、默认情况下，date\_range生成的是每日的时间戳。

C、若执行语句index = pd.date\_range('2012-04-01', '2012-06-01')，则index中包含'2012-06-01'而不包含'2012-04-01'。

D、使用date\_range方法将生成一个datetime对象。

1. 1stamp = datetime(2011, 1, 3); stamp.strftime(?)，问号处应该填入下列哪条语句，才可以使输出结果为'2011-01-03'（ ）

A，'%Y-%M-%d' B，'%y-%m-%d' C，'%D' D，'%F'

1. 对于下列代码from datetime import timedelta; start = datetime(2019,1,1,0,0); end = start + timedelta(12,60); end.strftime('%Y-%m-%d/%H:%M:%S')，其运行结果为（ ）

A. '2019-01-13/00:01:00' B. '2031-01-1/00:01:00'

C. '2019-01-13/01:00:00' D. '2031-01-1/01:00:00'

1. pandas中的哪个函数或方法可以将字符串列表转换为日期索引（ ）

A.strftime B.srtptime C.to\_datetime D.timedelta

1. 已知now = datetime(2000,1,1)，运行MonthEnd().rollforward(now+MonthEnd())得到的结果为（ ）

A. Timestamp('2000-02-31 00:00:00')

B. Timestamp('2000-01-31 00:00:00')

C. Timestamp('2000-01-01 00:00:00')

D. Timestamp('2000-02-01 00:00:00')

1. from dateutil.parser import parse;parse('6/12/2011', ？)，问好处应填入以下哪条语句，才使得输出结果为datetime.datetime(2011, 12, 6, 0, 0)。（ ）

A， dayfirst=True B， dayfirst=False C， daylast=True D， daylast=False

1. 已知区间p=pd.Period('2007',freq='A-JUN')则代码p.asfreq('M','end')输出为（ ）

A、Period('2006-07', 'M') B、Period('2007-06', 'M')

C、Period('2007-07', 'M') C、Period('2008-06', 'M')

1. 下列哪个语句可以输出datetime.datetime(2011, 1, 3, 0, 0)（ ）。

A，import datetime.datetime;datetime(2011, 1, 3, 0, 0)

B，import dateutil.parser;parse('2011-01-03')

C，import pandas;pandas.to\_datetime('2011-01-03')

D，import pandas;pandas.to\_datetime('2011-01-03').to\_pydatetime()

1. 已知now = datetime(2011,11,17)，则代码offset=MonthEnd();offset.rollforward(now)的输出为（ ）

A、Timestamp('2011-10-31 00:00:00')

B、Timestamp('2011-12-17 00:00:00')

C、Timestamp('2011-11-30 00:00:00')

D、Timestamp('2011-10-17 00:00:00')

1. 存在一个时间序列对象p = pd.Period(2017, freq='A-DEC')，p + 2的结果为（ ）

A. Period(2018, freq=’A-FEB’) B. Period(2019, freq=’A-DEC’)

C. Period(2019, freq=’A-NOV’) D. Period(2018, freq=’A-FEB’)

1. 假设ts = pd.Series(np.random.randn(6), index=[datetime(2011, 1, 2), datetime(2011, 1, 5),datetime(2011, 1, 7), datetime(2011, 1, 8), datetime(2011, 1, 10), datetime(2011, 1, 12)])则以下语句输出结果相同的是（ ）。
   1. ts['1/10/2011']② ts['20110110']③ ts[datetime(2011, 1, 10)]④ ts[datetime(2011, 01, 10)]

A，①④ B，①③ C，②④ D，③④。

1. 以下哪个时间频率表示工作日的月初日期（ ）

A、M B、BM C、MS D、BMS

1. 在datetime对象中，“2019年5月20日15时20分”应该是下列哪一个格式？( )

A.%Y-%M-%D-%H-%m

B. %Y-%m-%D-%H-%M

C. %Y-%M-%D-%H-%M

D.%Y-%m-%d-%H-%M

1. 在时间序列中，输出从2010年1月1号0点开始的，每隔4个小时一次，一共十条记录的语句是？（ ）

A. pd.date\_range('2010-01-01', periods = 10, freq = '4h')

B. pd.date\_range('2010-01-01', periods = 10, freq = 'D-4h')

C. pd.date\_range('2010-01-01', '2010-01-02', freq = '4h')

D. pd.date\_range('2010-01-01', freq = '4h')

1. 下面是datetime(2019, 1, 1).strftime('%Y-%m-%d')的结果的是 （ ）

A.2019-01-03 B.19-01-03 C.2019-1-3 D.19-1-3

1. 设有dates=pd.DatetimeIndex(['1/1/2000', '1/2/2000', '1/2/2000','1/2/2000', '1/3/2000'])，ts=pd.Series(np.arange(5),index=dates)，执行ts['1/2/2000']后得到的输出是（ ）

Ａ．６

Ｂ．［１，２，３］

Ｃ．2000-01-02 1

2000-01-02 2

2000-01-02 3

Ｄ．2000-01-02 1

1. ts=pd.Series(np.arange(4),index=pd.date\_range('1/1/2019',periods=4,freq='M'))，执行ts.shift(1,freq='D')后得到的结果是（ ）

A. 2019-01-31 0

2019-02-28 1

2019-03-31 2

2019-04-30 3

B. 2019-01-31 NaN

2019-02-28 0.0

2019-03-31 1.0

2019-04-30 2.0

C. 2019-01-30 0

2019-02-27 1

2019-03-30 2

2019-04-29 3

D. 2019-02-01 0

2019-03-01 1

2019-04-01 2

2019-05-01 3

1. dates=[datetime(2019,7,10),datetime(2019,7,14),datetime(2019,7,16),datetime(2019,7,18),datetime(2019,7,20)]

s=pd.Series(range(1,6),index=dates)

则ts+ts[::2]的输出结果是：（ ）

A.

2019-07-10 2.0

2019-07-14 4.0

2019-07-16 6.0

2019-07-18 8.0

2019-07-20 10.0

B.

2019-07-10 2.0

2019-07-14 NaN

2019-07-16 6.0

2019-07-18 NaN

2019-07-20 10.0

C.

2019-07-10 1.0

2019-07-14 NaN

2019-07-16 3.0

2019-07-18 NaN

2019-07-20 5.0

1. dates=pd.DatetimeIndex(['7/10/2019','7/10/2019','7/11/2019','7/12/2019','7/12/2019'])

ts=pd.Series(np.arange(5),index=dates)

则ts['7/10/2019']的输出结果是：（ ）

A.2019-07-10 0

2019-07-10 1

B.2019-07-10 0

C.2019-07-10 1

1. 下列对于datatime模块中的数据类型的叙述，错误的是（ ）。

A， time将时间存储为小时，分钟，秒，微秒。

B， timedelta表示两个datetime值之间的差。

C， datetime表示存储日期和时间。

D， date表示使用农历日历存储的日历日期（年，月，日）。

1. 已知p = pd.Period(2007, freq='A-DEC')，则下列语句可以运行的是（ ）。

A，p + 1 B，p \* 2 C，p++ D，1/p

1. 下列语句输出的Datetimeindex是（ ）。

print(pd.date\_range('2019/7/12','2021/6/30',freq = 'WOM-1sun'))

A，2019/7/12-2021/6/30中的每个月的第一天的日期。

B，2019/7/12-2021/6/30中的每个月的第一个周一的日期。

C，2019/7/12-2021/6/30中的每个月的第一个周日的日期。

D，2019/7/12-2021/6/30中的每个月的最后一个周日的日期。

1. 下列方法中，不能将date = datetime(2000,1,1)转换为'2000-01-01'的是（ ）

A.date.strftime('%Y-%m-%d')

B.str(date).split(' ')[0]

C.str(date)[0:str(date).index(' ')]

D.str(date).strftime('%Y-%m-%d')

1. stamp = datetime(2020, 1, 1)

stamp.strftime(?)

如何才可以使输出'01/01/2019'（ ）。

A，'%c'。 B，'%p' C，'%X' D，'%x'

1. a = datetime(2011, 1, 3, 4)，想得到'11-01-03-04’的输出，需要输入以下哪个语句？（ ）

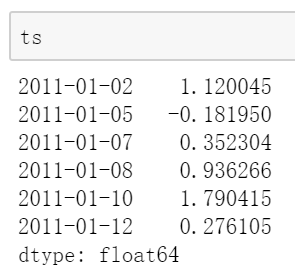
A a.strftime('%Y-%m-%d-%h')

B a.strftime('%y-%m-%d-%h')

C a.strftime('%Y-%m-%d-%H')

D a.strftime('%y-%m-%d-%H')

1. 已知



下列哪个选项不能得到

（ ）

A. ts['07/01/2011']

B. ts['20110107']

C. ts[ts.index[2]]

D. ts[2]

1. longer\_ts =pd.Series(np.random.randn(1000),

index=pd.date\_range('5/1/2001',periods=31))

下列语句中，会报错的是（ ）

A. longer\_ts['2001-05-25']

B. longer\_ts['20010525']

C. longer\_ts['200105']

D. longer\_ts['2001/05']

1. 下列说法中错误的是（ ）

A.datetime.srtptime可以将字符串转化成datetime格式

B.当日期出现在月份之前时，可以用dayfirst=True来说明

C.时间序列的切片原则是左闭右开

D.PeriodIndex类存储的是区间的序列

1. %F代表哪种格式的缩写( )

A %m/%d/%y B %y-%m-%d C %Y-%m-%d

1. timedelta中不包含以下哪个参数（ ）

A.months B.weeks C.days D.hours

1. 下列对于datatime模块中的数据类型叙述，错误的是（ ).

A.time将时间存储为小时，分钟，秒，微秒。

B.timedelta表示两个datetime值之间的差。

C.datetime表示存储日期与时间。

D.date表示使用农历日期日历存储的日历日期（年，月，日）。

1. 以下不是pandas.date\_range()的参数的是（ ）

A.after B.periods C.end D.freq

# 第12章

1. ~~假设cat\_s = pd.Series(['a', 'b', 'c', 'd'] \* 2).astype('category')，则以下语句的输出结果和cat\_s[cat\_s.isin(['a', 'b'])]一致的是（ ）。~~

~~A,cat\_s[cat\_s is 'a' or cat\_s is 'b']~~

~~B,cat\_s[cat\_s.isin('a', 'b')]~~

~~C,cat\_s3.cat.remove\_unused\_categories() D,cat\_s3.cat.remove\_uunsed\_categories()~~ 作废

1. 已知categories和codes，可通过下列哪个方法或函数快速构建Categorical对象（ ）

A,astype B,pd.Categorical C,from\_codes D,take

1. 假设df = pd.DataFrame({'key': ['a', 'b', 'c'] \* 4, 'value': np.arange(12.)})，g = df.groupby('key').value，则以下对g的操作，不能运行的是（ ）。

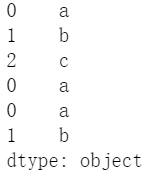
A，g = df.groupby('key').value

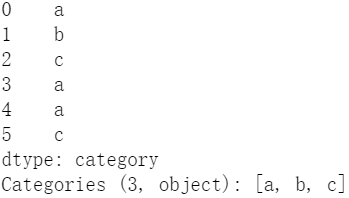
B，g.transform('mean')

C，g.transform(lambda x: x.rank(ascending=False))

D，g.transform(std)

1. 已知categories=['a','b','c']，codes=[0,1,2,0,0,1]，执行代码pd.Series(categories).take(pd.Series(codes))，其输出的结果为（ ）

A, B，

C, D，

1. 已知categories = ['a','b','c','d']，pd.Categorical.from\_codes(codes,categories)中传入以下哪个codes可以构造对象（ ）

A. codes = [1,2,1,3,2,0,0] B. codes = [0,0,2,3,1,2,1]

C. codes = [1,1,3,4,2,3,2] D. codes = [2,3,2,4,3,1,1]

1. 以下哪一个不是分类方法（ ）

A.add\_categorical B.as\_unordered

C.remove\_used\_categorical D.rename\_categorical

1. ~~df=pd.DataFrame({'key':['a','b','c']\*2,'value':np.arange(6.0)})~~

~~df1=df.groupby('key').value，则以下哪个语句的结果与其他三个不同（ ）~~

~~A、df1.rank()~~

~~B、df1.transform('rank')~~

~~B、df1.transform(lambda x: x.rank())~~

~~D、df['value'].groupby(df['key']).rank()~~ 作废

1. 如果一个Categorical对象的categories属性是[‘red’, ‘blue’, ‘green’, ‘white’]，codes属性是[0 1 2 3 2 1 3 0 2]，那么这个对象创建时传入的序列是（ ）

A.[ ‘red’, ‘blue’, ‘green’, ‘white’, ‘red’, ‘red’, ‘blue’, ‘green’, ‘white’]

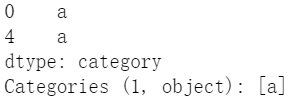
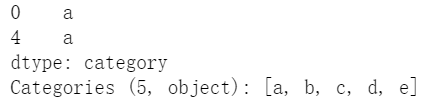
B.[ ‘red’, ‘blue’, ‘green’, ‘white’, ‘red’, ‘blue’, ‘green’, ‘white’, ‘white’]

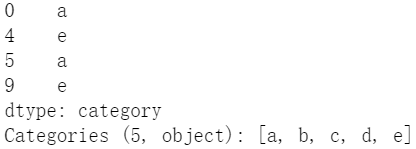
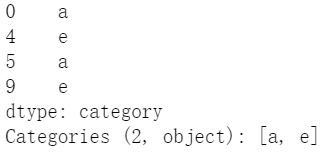
C.[ ‘red’, ‘blue’, ‘green’, ‘white’, ‘green’, ‘blue’, ‘white’, ‘red’, ‘green’]

* 1. [ ‘red’, ‘blue’, ‘green’, ‘white’, ‘green’, ‘white’, ‘white’, ‘red’, ‘green’]

1. 若一个Categorical对象是由序列['Apple', 'Orange', 'Pear', 'Orange', 'Apple']所创建，这个对象的codes属性是[0 1 2 1 0]，这个对象的categories属性是？（ ）
   * 1. Index(['Apple', 'Orange', 'Pear'], dtype='object')
     2. Index(['Orange', 'Apple', 'Pear'], dtype='object')
     3. Index(['Apple', 'Pear', 'Orange'], dtype='object')
     4. Index(['Orange', 'Pear','Apple'], dtype='object')
2. s = pd.Series(['a','b','c','d']\*2); cat\_s = s.astype('category'); cat\_s2 = cat\_s.cat.set\_categories(['a','b','c','d','e']); cat\_s3 = cat\_s2[cat\_s2.isin(['a','e'])]

cat\_s3.cat.remove\_unused\_categories()，输出的结果为（ ）

A, B，

C, D，

1. ~~对于 c = df['count'].astype('category').values,c.codes的执行结果是什么（ ）。~~

~~A，array([3,1,0,5,5,1,2,4], dtype=int8)~~

~~B，array([1,3,0,5,5,1,2,4], dtype=int8)~~

~~C，array([3,1,5,5,5,1,2,4], dtype=int8) D，array([3,1,0,4,5,1,2,4], dtype=int8)~~ 重复，作废

1. 使用from\_codes构造函数：如下

categories=['Lucy','Daisy','Lucky'];codes=[0,1,2,0,0,0]

my\_c=pd.Categorical.from\_codes(codes,categories)，则my\_c的输出结果是：( )

A.[Lucy, Daisy, Lucky, Lucy, Lucy, Lucy]

B.[Lucy, Daisy, Lucky, Daisy, Daisy, Daisy]

C.[Lucy, Daisy, Lucky, Lucky, Lucky, Lucky]

1. 若一个Categorical对象是由序列['Apple', 'Orange', 'Pear', 'Orange', 'Apple']所创建，这个对象的codes属性是[0 1 2 1 0]，这个对象的categories属性是？（ ）

A. Index(['Apple', 'Orange', 'Pear'], dtype='object')

B. Index(['Orange', 'Apple', 'Pear'], dtype='object')

C. Index(['Apple', 'Pear', 'Orange'], dtype='object')

D. Index(['Orange', 'Pear','Apple'], dtype='object')

1. 已知cat\_s = pd.Series(['a', 'b'] \* 2).astype('category')

cat\_s2 = cat\_s.cat.set\_categories(['a', 'b', 'c'])，cat\_s2返回的是（ ）

A.0 a

1 b

2 a

3 b

dtype: category

Categories (2, object): ['a', 'b']

B. 0 a

1 b

2 a

3 b

dtype: category

Categories (3, object): ['a', 'b', 'c']

C. 0 a

1 b

2 c

dtype: category

Categories (3, object): ['a', 'b', 'c']

D.0 a

1 b

2 a

3 b

4 a

5 b

6 c

dtype: category

Categories (5, object): ['a', 'b', 'c']

1. 已知分类转换

s=pd.Series(['a','b','c','d']\*2)

cat\_s=s.astype('category')

actual\_categories=['a','b','c','d','e']

cat\_s2=cat\_s.cat.set\_categories(actual\_categories)

则代码cat\_s2.value\_counts()的输出为（ ）

A、

a 2

b 2

c 2

d 2

e 0

dtype: int64

B、

d 2

c 2

b 2

a 2

e 0

dtype: int64

C、

a 2

b 2

c 2

d 2

dtype: int64

D、

d 2

c 2

b 2

a 2

dtype: int64

1. 设有数据集df=pd.DataFrame({'key':['a','b','c']\*2,'value':np.arange(6)})，执行df['value'].groupby(df['key']).transform('mean')后得到的结果是（ ）

A. key

a 1.5

b 2.5

c 3.5

B. key value

0 a 1.5

1 b 2.5

2 c 3.5

3 a 1.5

4 b 2.5

5 c 3.5

C. 0 1.5

1 2.5

2 3.5

3 1.5

4 2.5

5 3.5

D. key

a 3.0

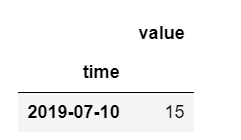
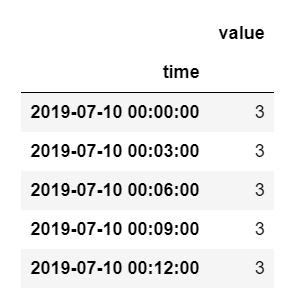
b 3.0

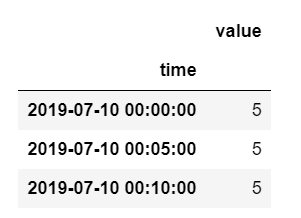
c 3.0

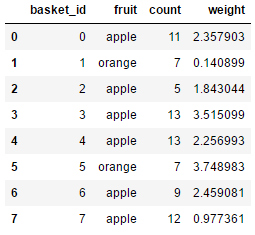
1. times=pd.date\_range('2019-07-10 00:00',freq='1min',periods=15)

df=pd.DataFrame({'time':times,'value':np.arange(15)})

则df.set\_index('time').resample('5min').count()的输出结果是：（ ）

A.B.

C.

1. 对于  c = df['count'].astype('category').values,

c.codes的执行结果是什么（ ）。

A，array([3, 1, 0, 5, 5, 1, 2, 4], dtype=int8)

B，array([1, 3, 0, 5, 5, 1, 2, 4], dtype=int8)

C，array([3, 1, 5, 5, 5, 1, 2, 4], dtype=int8)

D，array([3, 1, 0, 4, 5, 1, 2, 4], dtype=int8)

1. 下列说法正确的是 ( )

A.使用TimeGrouper的一个限制是时间必须是Series或DataFrame

B.agg不可以使用lambda表达式

C.transform可以使用‘mean’

D.apply可以传入‘mean’

1. 对于 s = pd.Series(['a', 'b', 'c']\*2); cat\_s = s.astype('category')，可以新增'd'类别的语句是（ ）

A.cat\_s.add\_categories('d')

B.s.add\_categories('d')

C.cat\_s.cat.add\_categories('d')

D.s.cat.add\_categories('d')

1. 已知

df =pd.DataFrame({'key': ['a', 'b', 'c'] \* 4,

'value': np.arange(12.)})

g =df.groupby('key').value

g.transform(lambdax: x.mean())

与下列代码的哪个输出结果相同 （ ）

A. g.agg(lambdax: x.mean())

B. g.apply(lambdax: x.mean())

C.上面两种都不与之匹配