CS 7646 Python Midterm Questions

Instructions to TAs:

Please cut and paste questions here, at most one question per page. Use the "courier new" font, 11pt for all text. Please insert a page break after each question.

Please post only those questions that scored 85% or higher according to the rubric.

What is the output of the joined Dataframe?

Code:

```
import numpy as np
import pandas as pd

df = pd.DataFrame([52, 46, 50, 51], columns = ['AAPL'], index = ['01-01', '01-02', '01-03', '01-04'])

df = df.join(pd.DataFrame([83, 88, 86, 90], columns = ['SPY'], index = ['01-01', '01-02', '01-04', '01-05']))

print df
```

Select one answer:

a)
	AAPL	SPY
01-01	52	83
01-02	46	88
01-03	50	86
01-04	51	90

b)
	AAPL	SPY
01-01	52	83
01-02	46	88
01-03	50	NaN
01-04	51	86

c)
	AAPL	SPY
01-01	52	83
01-02	46	88
01-03	50	NaN
01-04	51	86
01-05	NaN	90

d)
	AAPL	SPY
01-01	52	83
01-02	46	88
01-04	51	86

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame([52, 46, 50, 51], columns = ['APPL'], index = ['01-01', '01-02', '01-03', '01-04'])
>>> df = df.join(pd.DataFrame([83, 88, 86, 90], columns = ['SPY'], index = ['01-01', '01-02', '01-04', '01-05']))
>>> print df
      APPL  SPY
01-01   52   83
```

```
01-02     46    88
01-03     50   NaN
01-04     51    86
```

Which answer is the output of the following code?

```python
import pandas as pd
import numpy as np

d = {"SPY" : [86.80, 86.70, 87.28, 84.67, 85.01],
     "AAPL": [90.36, 94.18, 92.62, 90.62, 92.30],
     "HNZ" : [33.95, 33.82, 33.38, 32.59, 31.99],
     "XOM" : [74.48, 74.47, 73.26, 71.39, 85.13],
     "GLD" : [86.23, 84.48, 85.13, 82.75, 84.46]}

df = pd.DataFrame(d)

normed = df/df.ix[0]
normed['AAPL'] = np.nan
normed.fillna(value='0')
print normed[0:2]
```

a)
```
  AAPL        GLD        HNZ        SPY        XOM
1  NaN   0.979705   0.996171   0.998848   0.999866
2  NaN   0.987243   0.983211   1.005530   0.983620
```

b)
```
  AAPL        GLD        HNZ        SPY        XOM
1  94.18      84.48      33.82      86.70      74.47
2  92.62      85.13      33.38      87.28      73.26
```

c)
```
  AAPL        GLD        HNZ        SPY        XOM
0   0     1.000000   1.000000   1.000000   1.000000
1   0     0.979705   0.996171   0.998848   0.999866
```

d)
```
  AAPL        GLD        HNZ        SPY        XOM
0  NaN   1.000000   1.000000   1.000000   1.000000
1  NaN   0.979705   0.996171   0.998848   0.999866
```

answer: d

Python transcript:

```python
>>> import pandas as pd
>>> import numpy as np
>>>
>>>
>>> d = {"SPY" : [86.80, 86.70, 87.28, 84.67, 85.01],
...        "AAPL": [90.36, 94.18, 92.62, 90.62, 92.30],
...        "HNZ" : [33.95, 33.82, 33.38, 32.59, 31.99],
...        "XOM" : [74.48, 74.47, 73.26, 71.39, 85.13],
...        "GLD" : [86.23, 84.48, 85.13, 82.75, 84.46]}
>>>
>>> df = pd.DataFrame(d)
>>> normed = df/df.ix[0]
```

```
>>> normed['AAPL'] = np.nan
>>> normed.fillna(value='0')
 AAPL      GLD       HNZ       SPY       XOM
0    0  1.000000  1.000000  1.000000  1.000000
1    0  0.979705  0.996171  0.998848  0.999866
2    0  0.987243  0.983211  1.005530  0.983620
3    0  0.959643  0.959941  0.975461  0.958512
4    0  0.979474  0.942268  0.979378  1.142991
>>>
>>> print normed[0:2]
   AAPL      GLD       HNZ       SPY       XOM
0   NaN  1.000000  1.000000  1.000000  1.000000
1   NaN  0.979705  0.996171  0.998848  0.999866
```

What will be the output of the following code snippet, specifically the final print statement?

Code:

```
import numpy as np
array = np.ones((2,3,4))
array = array * 2
print array.sum(axis=None)
```

Select one answer:
a) 24
b) [[6.   6.   6.   6.]
[6.   6.   6.   6.]]
c) [[8.   8.   8.]
[8.   8.   8.]]
d) 48

Correct answer: d)

Python Transcript:

```
>>> import numpy as np

>>> array = np.ones((2,3,4))

>>> array = array * 2

>>>print array.sum(axis=None)
48.0
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import pandas as pd
import numpy as np

x = True
df1 = pd.DataFrame([[1,0,-5],[2,0,-1],[1,4,0]])
___A___
df1[f(df1)] = "!!!"
print df1
```

```
Output:

     0    1    2
0    1    0   !!!
1   !!!   0   -1
2    1   !!!   0
```

Select one answer:

```
        a) f = lambda x: x>1
        b) f = lambda x: np.abs(x)>=0
        c) f = lambda x: np.abs(x)>1
        d) f = lambda x: x>0
```

Correct answer: c)

```
Python Transcript
>>> import pandas as pd
>>> import numpy as np
>>>
>>> x = True
>>> df1 = pd.DataFrame([[1,0,-5],[2,0,-1],[1,4,0]])
>>> f = lambda x: np.abs(x)>1
>>> df1[f(df1)] = "!!!"
>>> print df1
     0    1    2
0    1    0   !!!
1   !!!   0   -1
2    1   !!!   0
```

Question 5   + Answer & Validation

How should section _A_ be filled in to complete code that will cause the following
output:

Code:

```
import numpy as np

a = np.ones((3,3))
print a
b = a
b[0,1] = 2

print b
print a

b  = _A_   # what should be filled instead of _A_ for below ouputs to be true ?

b[0,1] = 3
print b
print a
```

Output:

```
[[ 1.   1.   1.]
```

```
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

[[ 1.  3.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]

Select one answer:
a) a[0,0] = 2
b) a.copy()
c) b[1,1] = 3
d) b[1,0] = 3

Correct answer: b)

Python transcript:

>>> import numpy as np
>>> a = np.ones((3,3))
>>> print a
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
>>> b = a
>>> b[0,1] = 2
>>> print b
[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
>>> print a
[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
>>> b = a.copy()
>>> print b
[[ 1.  3.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
>>> print a
[[ 1.  2.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
```

Question 6   + Answer & Validation

What is the output of the following code?

```
import numpy as np

a = np.array([2]*4)

b = np.array([1, 2, 3, 4, 5, 6, 7])
b[1:] * a[-1]
```

Select one answer:
a) array([4, 6, 8, 10, 12, 14])
b) array([1, 2, 3, 4, 5, 6, 7])
c) array([2, 4, 6, 8, 10, 12, 14])
d) array([3, 9, 12, 15, 18])

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>>
>>> a = np.array([2]*4)
>>>
>>> b = np.array([1, 2, 3, 4, 5, 6, 7])
>>> b[1:] * a[-1]
array([ 4,  6,  8, 10, 12, 14])
```


Question 7 + Answer & Validation

1. What is the output of the following code

Code:

```
j = [8, 7, 6, 5, 4, 3, 2, 1]
print [x/j[-2] for x in j[1:-1]]
```

Select one answer:
a) [4.0, 3.5, 3.0, 2.5, 2.0, 1.5, 1.0, 0.5]
b) [3.5, 3.0, 2.5, 2.0, 1.5, 1.0, 0.5]
c) [3, 3, 2, 2, 1, 1]
d) [3, 3, 2, 2, 1, 1, 0]

Correct answer: c)

Python transcript:

```
>>> j = [8, 7, 6, 5, 4, 3, 2, 1]
>>> print [x/j[-2] for x in j[1:-1]]
[3, 3, 2, 2, 1, 1]
```


Question 8 + Answer & Validation

What is the output of the following code?

Code:
```
import pandas as pd
import numpy as np

syms=['IBM', 'AAPL', 'HNZ', 'XOM', 'GLD']
prices =  pd.DataFrame(np.random.rand(10, len(syms)),columns=syms)
print prices.tail(1).values
```

Select one answer:
a:)
```
        IBM        AAPL        HNZ        XOM        GLD
5  0.596553   0.654170   0.020114   0.889447   0.024598
6  0.557738   0.478691   0.011598   0.812025   0.668150
7  0.231585   0.993491   0.302910   0.261203   0.659507
8  0.715009   0.244946   0.644569   0.415497   0.827711
9  0.282840   0.814612   0.542779   0.325938   0.387805
```

b:)
```
        IBM        AAPL        HNZ        XOM        GLD
0  0.021792   0.41759   0.628591   0.834644   0.118658
```

c:)
```
[[ 0.28284041  0.81461207  0.54277906  0.32593834  0.3878054 ]]
```

d:)
```
        IBM       AAPL       HNZ        XOM        GLD
9  0.28284   0.814612   0.542779   0.325938   0.387805
```


Correct Answer = C:)
```
print prices.tail(1).values
```
.tail = display the last 5 rows in array including column headers
.tail(1) = display only the last row in the array including column headers
.tail(1).values = display only the values of the last row in the array without column headers

Question 9 + Answer & Validation

Which of the following is a valid output of the following code?

Code:

```
import numpy as np
a = np.random.randint(10, 30, size=(2, 4))
print a
print a.size
print a.shape[0]
print a.shape[1]
```

Select one answer:
```
a) [[5 14 10 10]
    [29 26 40 13]]
    8
    2
    4
b) [[21 14 10 10]
    [29 26 19 13]]
    8
    4
    2
c) [[21 14 10 10]
    [29 26 19 13]]
    8
    2
    4
d) [[21 14]
```

```
        [0 26]
        [29 15]
        [12 4]]
        4
        2
        4
```

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.random.randint(10, 30, size=(2, 4))
>>> print a
[[21 14 10 10]
        [29 26 19 13]]
>>> print a.size
8
>>> print a.shape[0]
2
>>> print a.shape[1]
4
```

Question 11 + Answer & Validation

A donut shop owner sells 3 items (coffee, donuts, and bagels) and wants to calculate
the following for a week of sales
(1) total sales, (2) daily sales, and (3) item sales

Code:

```
import numpy as np
import pandas as pd

weekSales =  pd.DataFrame(np.random.random([7,3])*100,
            index=['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'],
            columns=['Coffee', 'Donuts', 'Bagels'])

print weekSales

print _A_
print _B_
print _C_
```

Output:

```
          Coffee      Donuts      Bagels
Sun     33.530433    9.862755   53.781308
Mon     29.831218   84.473341   34.325790
Tues    78.021005   84.196225   27.752726
Wed     85.667444    2.104389    0.696575
Thurs   74.328857   56.910230   24.484673
Fri     76.807376   67.794258   17.389399
Sat     84.438981    3.740902   58.141248

988.27913306218318

Sun       97.174496
Mon      148.630350
Tues     189.969956
Wed       88.468408
```

```
Thurs      155.723760
Fri        161.991033
Sat        146.321131


Coffee     462.625316
Donuts     309.082100
Bagels     216.571718
```

What code A, B, and C are used to produce that output?
Select one answer:
a) weekSales.sum(), weekSales.sum(axis=1), weekSales.sum(axis=0)
b) weekSales.sum().sum(), weekSales.sum(axis=0), weekSales.sum(axis=1)
c) neither a nor b
d) either a or b

correct answer: c

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> weekSales = pd.DataFrame(np.random.random([7,3])*100, index=['Sun', 'Mon', 'Tues',
'Wed', 'Thurs', 'Fri', 'Sat'], columns=['Coffee', 'Donuts', 'Bagels'])
>>> print weekSales
           Coffee      Donuts      Bagels
Sun     33.530433    9.862755   53.781308
Mon     29.831218   84.473341   34.325790
Tues    78.021005   84.196225   27.752726
Wed     85.667444    2.104389    0.696575
Thurs   74.328857   56.910230   24.484673
Fri     76.807376   67.794258   17.389399
Sat     84.438981    3.740902   58.141248

>>> print weekSales.sum().sum()
988.27913306218318

>>> print weekSales.sum(axis=1)
Sun        97.174496
Mon       148.630350
Tues      189.969956
Wed        88.468408
Thurs     155.723760
Fri       161.991033
Sat       146.321131
dtype: float64

>>> print weekSales.sum(axis=0)
Coffee     462.625316
Donuts     309.082100
Bagels     216.571718
dtype: float64
```

Question 12 + Answer + Validation

"How should section A be filled in to complete code that will cause the following
output:"

Code

```
import numpy as np

x = np.array([[1,2,3,4],
              [4,5,6,7],
              [8,9,0,1]])

print _A_


"""

Output:
[[1, 3],
[4, 6],
[8, 0]])


Select one answer:
a) x[0,2]
b) x[:,0,2]
c) x[::2]
d) x[:,[0,2]]

Correct answer: d)

In[1]: import numpy as np
In[2]: x = np.array([[1,2,3,4],
              [4,5,6,7],
              [8,9,0,1]])
In[3]: print x[:,[0,2]]
[[1 3]
[4 6]
[8 0]]

"""


Question 13  + Answer + Validation
```

How should section A be filled in to complete code that will cause the following output:

```
import numpy as np
import pandas as pd

dates = pd.date_range('2015-01-01', '2015-01-04', name='Date')
df = pd.DataFrame(index=dates)
df['count'] = [i for i in range(len(df))]

print df

__A__

df.set_index('count', inplace=True)

print df


Output:
            count
Date
```

```
2015-01-01        0
2015-01-02        1
2015-01-03        2
2015-01-04        3
            Date
count
0      2015-01-01
1      2015-01-02
2      2015-01-03
3      2015-01-04
```

Select one answer:

a) df.reset_index(['Date'])
b) df.drop_index(['Date'])
c) df = df.reset_index(['Date'])
d) df.drop_index(['Date'], inplace=True)

Correct answer: c)

Python Transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> dates = pd.date_range('2015-01-01', '2015-01-04', name='Date')
>>> df = pd.DataFrame(index=dates)
>>> df['count'] = [i for i in range(len(df))]
>>> print df
            count
Date
2015-01-01        0
2015-01-02        1
2015-01-03        2
2015-01-04        3
>>> df = df.reset_index(['Date'])
>>> df.set_index('count', inplace=True)
>>> print df
            Date
count
0      2015-01-01
1      2015-01-02
2      2015-01-03
3      2015-01-04
```


Question 14   + Answer + Validation

What is the output of the following code?

```
import pandas
import numpy
df = numpy.array([2, 3, 4, 5])
print pandas.rolling_mean(df, window=2)
```

Select one:
a) [nan 2 3 4]
b) [nan 2.5 3.5 4.5]
c) [2 2.5 3.25 4.125]
d) [1 2 3 4]

```
Correct answer: b
```

```
Python transcript:
>>> import pandas
>>> import numpy
>>> df = numpy.array([2, 3, 4, 5])
>>> print pandas.rolling_mean(df, window=2)
[ nan 2.5 3.5 4.5]
```

Question 15   + Answer + Validation

What is the output of this python code?

Code:

```
import numpy as np
a = np.arange(5, 0, -1)
print a[a < 3]
```

Select one answer:

```
a) [0, 1, 2]
b) [5, 4, 3]
c) [False, False, False, True, True]
d) [2, 1]
```

```
Correct answer: d)
```

Python transcript:

```
>>> import numpy as np
>>> a = np.arange(5, 0, -1)
>>> print a[a < 3]
[2 1]
```

Question 16   + Answer + Validation

What line of code should be added to "Section A", using pandas DataFrames, so that it
produces the following output:

Code:

```
import pandas as pd
df1 = pd.DataFrame([[10, 'w'],
                    [20, 'x'],
                    [30, 'y'],
                    [40, 'z']],
                   columns=['numbers', 'letters'],
                   index=['a', 'b', 'c', 'd'])


df2 = pd.DataFrame([[1.0, '#'],
                    [2.0, '@'],
                    [3.0, '%'],
                    [4.0, '$']],
                   columns=['floats', 'symbols'],
```

```
                       index=['a', 'b', 'c', 'e'])
```

Section A

Output:

```
  numbers letters  floats symbols
a      10        w       1       #
b      20        x       2       @
c      30        y       3       %
d      40        z     NaN     NaN
e     NaN      NaN       4       $
```

Select one answer:
a) print df1.join(df2, how='right')
b) print df1.join(df2, how='inner')
c) print df1.join(df2, how='outer')
d) print df1.join(df2, how='left')

Correct Answer: c)

Python transcript:

```
import pandas as pd
df1 = pd.DataFrame([[10, 'w'],
                    [20, 'x'],
                    [30, 'y'],
                    [40, 'z']],
                 columns=['numbers', 'letters'],
                 index=['a', 'b', 'c', 'd'])


df2 = pd.DataFrame([[1.0, '#'],
                    [2.0, '@'],
                    [3.0, '%'],
                    [4.0, '$']],
                 columns=['floats', 'symbols'],
                 index=['a', 'b', 'c', 'e'])

print df1.join(df2, how='outer')
```

Question 17   + Answer + Validation

How should section A be filled so that the code causes the following output?

Code:

```
import numpy as np
j=np.random.random([4,4])
print j
print A
```

Output:

```
[[ 0.77193745  0.65987068  0.07110931  0.34828411]
 [ 0.57139421  0.58080777  0.45935194  0.05061515]
 [ 0.01467635  0.84673314  0.78251514  0.96852681]
```

```
[ 0.71871822  0.57120611  0.30561734  0.71769405]]
[[ 0.45935194  0.05061515]
[ 0.78251514  0.96852681]]
```

Select one answer:

a) j[1:2,2:]
b) j[2:3,3:4]
c) j[1:3,2:]
d) j[2:,1:2]

Correct answer: c

Python Script:

```
>>> import numpy as np
>>> j=np.random.random([4,4])
>>> print j
[[ 0.77193745  0.65987068  0.07110931  0.34828411]
[ 0.57139421  0.58080777  0.45935194  0.05061515]
[ 0.01467635  0.84673314  0.78251514  0.96852681]
[ 0.71871822  0.57120611  0.30561734  0.71769405]]
>>> print j[1:3,2:]
[[ 0.45935194  0.05061515]
[ 0.78251514  0.96852681]]
```

Question 18   + Answer + Validation

What is the output of the variable "value"?

```
import numpy as np
a = np.random.randint(0,10,size=(3,3))
print "Matrix a is ",a
value = np.mean(a.min(axis = 0))
print "Variable value is", value
```

```
OUTPUT
======
Matrix a is [[7 5 2]
            [8 5 1]
            [1 3 0]]
```

Select one answer:

a) 1.0
b) [5.33333333 4.33333333 1.0]
c) 1.3333333
d) [4.66666667 4.66666667 1.33333333]

Correct answer: c

```
Python transcript
=================
>>> import numpy as np
>>> a = np.random.randint(0,10,size=(3,3))
>>> print "Matrix a is ",a
Matrix a is [[7 5 2]
            [8 5 1]
            [1 3 0]]
>>> value = np.mean(a.min(axis = 0))
```

```
>>> print "Variable value is", value
Variable value is 1.33333333333
```

Question 19 + Answer + Validation

What is the output of the following python code?

Code:

```
import numpy as np
ary = np.array([[[1, 2], [3, 4]], [[5, 6], [7,8]]])
print ary[:, :, 0]
```

Output of "print ary" is:
```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

Select one answer:
a)
```
[[1 2]
[3 4]]
```
b)
```
[[1 2]
[5 6]]
```
c)
```
[[1 3]
[5 7]]
```
d)
```
[[1 5]
[3 7]]
```

Correct answer: c)

Python transcript:
```
>>> import numpy as np
>>> ary = np.array([[[1, 2], [3, 4]], [[5, 6], [7,8]]])
>>> print ary[:, :, 0]
[[1 3]
[5 7]]
>>> print ary
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

Question 20 + Answer + Validation

What is the output of the following code?

Code:

```
import numpy as np

a = np.array([[1,2,3,4,5],
             [6,7,8,9,10],
             [11,12,13,14,15],
```

```
                 [16,17,18,19,20],
                 [21,22,23,24,25]])

print a[:,1:5:2]

Select one answer:
a) [[ 1  3  5]
    [ 6  8 10]
    [11 13 15]
    [16 18 20]
    [21 23 25]]

b) [[ 2  4]
    [ 7  9]
    [12 14]
    [17 19]
    [22 24]]

c) [[ 1  2  3  4  5]
    [11 12 13 14 15]
    [21 22 23 24 25]]

c) [[ 6  7  8  9 10]
    [16 17 18 19 20]]

Correct answer: b)

Python transcript:

>>> import numpy as np
>>> a = np.array([[1,2,3,4,5],
                  [6,7,8,9,10],
                  [11,12,13,14,15],
                  [16,17,18,19,20],
                  [21,22,23,24,25]])
>>> print a
[[ 2  4]
[ 7  9]
[12 14]
[17 19]
[22 24]]


Question 21  + Answer + Validation

What is the output of the following code?

import numpy as np
a=np.ones((3,3))*2
b=a.dot(a)
print(a)
print(b)

A)
[[ 2.  2.  2.]
 [ 2.  2.  2.]
 [ 2.  2.  2.]]
[[ 4.  4.  4.]
 [ 4.  4.  4.]
 [ 4.  4.  4.]]
```

```
B)
[[ 2.  2.  2.]
 [ 2.  2.  2.]
 [ 2.  2.  2.]]
[[ 12.  12.  12.]
 [ 12.  12.  12.]
 [ 12.  12.  12.]]


C)
[[ 4.  4.  4.]
 [ 4.  4.  4.]
 [ 4.  4.  4.]]
[[ 12.  12.  12.]
 [ 12.  12.  12.]
 [ 12.  12.  12.]]


D)
[[ 2.  2.  2.]
 [ 2.  2.  2.]
 [ 2.  2.  2.]]
[[ 8.  8.  8.]
 [ 8.  8.  8.]
 [ 8.  8.  8.]]
```

Answer: B

Python Transcript:

```
>>> import numpy as np
>>> a=np.ones((3,3))*2
>>> b=a.dot(a)
>>> print(a)
[[ 2.  2.  2.]
 [ 2.  2.  2.]
 [ 2.  2.  2.]]
>>> print(b)
[[ 12.  12.  12.]
 [ 12.  12.  12.]
 [ 12.  12.  12.]]
```


Question 22 + Answer + Validation

You are given two data frames, df1 and df2, in the code below. They are joined
together to form a third data frame. What is the output of the below code?

Code:

```
import pandas as pd

df1 = pd.DataFrame( {'var1': [1,2,3,4,5], 'var2': [11,12,13,14,15]}, index =
['cat','dog', 'bird', 'fish', 'turtle'] )
df2 = pd.DataFrame( {'var3': [101,102,103,104,105]}, index = ['dog', 'bird', 'cat',
'turtle', 'fish'] )
df3 = df1.join(df2)
print df3.ix['bird','var2']
```


Select one answer:
a) [11, 12, 13, 14, 15]
b) [3, 13, 102]

```
c) 13
d) 102

Correct answer: c)



Python transcript:

>>> import pandas as pd

>>> df1 = pd.DataFrame( {'var1': [1,2,3,4,5], 'var2': [11,12,13,14,15]}, index =
['cat','dog', 'bird', 'fish', 'turtle'] )
>>> df2 = pd.DataFrame( {'var3': [101,102,103,104,105]}, index = ['dog', 'bird',
'cat', 'turtle', 'fish'] )
>>> df3 = df1.join(df2)
>>> print df3.ix['bird','var2']
13
```

Question 23 + Answer + Validation

How would section _A_ be filled to complete code that will address any gaps in the
data (missing cells) after reindexing a data frame and produce the following output:

Code:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index=['2013-01-02', '2013-01-03', '2013-01-
04', '2013-01-05', '2013-01-08'], columns=['a', 'b', 'c'])
df = df.reindex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04', '2013-01-
05','2013-01-07', '2013-01-08', '2013-01-09'])
print df
_A_
print df
```

Output:

```
                  a          b          c
2013-01-01       NaN        NaN        NaN
2013-01-02  0.490073   0.132083  -0.404633
2013-01-03  0.972129   0.596112  -0.744198
2013-01-04 -0.650304   0.299980   0.093164
2013-01-05 -1.049114  -0.212860   0.698289
2013-01-07       NaN        NaN        NaN
2013-01-08  0.671206  -0.611449  -0.215637
2013-01-09       NaN        NaN        NaN


                  a          b          c
2013-01-01  0.490073   0.132083  -0.404633
2013-01-02  0.490073   0.132083  -0.404633
2013-01-03  0.972129   0.596112  -0.744198
2013-01-04 -0.650304   0.299980   0.093164
2013-01-05 -1.049114  -0.212860   0.698289
2013-01-07 -1.049114  -0.212860   0.698289
2013-01-08  0.671206  -0.611449  -0.215637
2013-01-09  0.671206  -0.611449  -0.215637
```

Select one answer:
a)   df.fillna(method='ffill', inplace=True)
```

b)          df.fillna(method='bfill', inplace=True)

c)          a followed by b

d)          b followed by a

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> import numpy as np
>>> df = pd.DataFrame(np.random.randn(5, 3), index=['2013-01-02', '2013-01-03', '2013-
01-04', '2013-01-05', '2013-01-08'], columns=['a', 'b', 'c'])
>>> df=df.reindex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04', '2013-01-
05','2013-01-07', '2013-01-08', '2013-01-09'])
>>> print df
                    a          b          c
2013-01-01        NaN        NaN        NaN
2013-01-02   0.490073   0.132083  -0.404633
2013-01-03   0.972129   0.596112  -0.744198
2013-01-04  -0.650304   0.299980   0.093164
2013-01-05  -1.049114  -0.212860   0.698289
2013-01-07        NaN        NaN        NaN
2013-01-08   0.671206  -0.611449  -0.215637
2013-01-09        NaN        NaN        NaN
>>> df.fillna(method='ffill', inplace=True)
>>> df.fillna(method='bfill', inplace=True)
>>> print df
                    a          b          c
2013-01-01   0.490073   0.132083  -0.404633
2013-01-02   0.490073   0.132083  -0.404633
2013-01-03   0.972129   0.596112  -0.744198
2013-01-04  -0.650304   0.299980   0.093164
2013-01-05  -1.049114  -0.212860   0.698289
2013-01-07  -1.049114  -0.212860   0.698289
2013-01-08   0.671206  -0.611449  -0.215637
2013-01-09   0.671206  -0.611449  -0.215637
```

Question 24 + Answer + Validation

What is the output of this code?

Code:

```
import numpy as np
k= np.array([[2, 4, 8, 16], [16.0, 8.0, 4.0, 2.0]])
print k[1,2:-1]/k[-2,-2]
```

Select one answer:
a) [1.0]
b) [2.0]
c) [0.25]
d) [0.5]

Correct answer: d)

Python transcript:

```
>>> import numpy as np
```

```
>>> k= np.array([[2, 4, 8, 16], [16.0, 8.0, 4.0, 2.0]])
>>> print k[1,3] / k[-2,-3]
[ 0.5]
```

Question 25 + Answer + Validation

What is the output of this Python code?

Code:

```
import numpy as np
j = np.array([(1,2,3),(4,5,6),(7,8,9)])
k = j[:,-1:]
print k.shape
```

Select one answer:
a) 3
b) (3, 1)
c) (1, 3)
d) [3 6 9]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> j = np.array([(1,2,3),(4,5,6),(7,8,9)])
>>> k = j[:,-1:]
>>> print k.shape
(3, 1)
```

Question 26 + Answer + Validation

What does the following code output?

Code:

```
import numpy as np
syms=['GOOG','AAPL','GLD','XOM']
num_syms = len(syms)
allocs = [1.0 / num_syms] * num_syms
print allocs
```

Select the answer which the above code will output:
a) [0.25, 0.25, 0.25, 0.25]
b) [1.0]
c) 1.0
d) TypeError: unsupported operand type(s) for /: 'list' and 'float'

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> syms=['GOOG','AAPL','GLD','XOM']
>>> num_syms = len(syms)
>>> allocs = [1.0 / num_syms] * num_syms
>>> print allocs
[0.25, 0.25, 0.25, 0.25]
```

Question 26 + Answer + Validation

What is the output of this python code?

Code:

```
    import pandas as pd
    df1 = pd.DataFrame({'Type': ["Stock", None]})
    df2 = df1
    df1.fillna("Mutual Fund", inplace=True)
    df2.fillna("ETF")
    print df1
    print df2
```

Select one answer:
a)

```
          Type
0        Stock
1         None
          Type
0        Stock
1          ETF
```

b)
```
          Type
0        Stock
1  Mutual Fund
          Type
0        Stock
1          ETF
```

c)
```
          Type
0        Stock
1  Mutual Fund
          Type
0        Stock
1         None
```

d)
```
          Type
0        Stock
1  Mutual Fund
          Type
0        Stock
1  Mutual Fund
```

Correct answer: d)

Python transcript:

```
Python 2.7.11 |Anaconda 2.4.1 (64-bit)| (default, Jan 29 2016, 14:26:21) [MSC v.1500
64 bit (AMD64)] on win32
In[3]:     import pandas as pd
    df1 = pd.DataFrame({'Type': ["Stock", None]})
    df2 = df1
    df1.fillna("Mutual Fund", inplace=True)
    df2.fillna("ETF")
    print df1
    print df2
Backend Qt4Agg is interactive backend. Turning interactive mode on.
          Type
0        Stock
1  Mutual Fund
          Type
0        Stock
1  Mutual Fund
```

Question 27 + Answer + Validation

What is the output of the following code?

```
import pandas as pd
import numpy as np

array0 = np.array([3,2,5,2,1])
df = pd.DataFrame(array0, columns=['numbers'], index=['a','b','c','d','e'])
df = df.ix[df.index[2:5]].sum()
dr = df**2
print  df
```

Select one answer:
a) 64
b) numbers 8
c) 8
d) 9

Correct answer: b)

Python transcript:

```
>>> import pandas as pd
>>> import numpy as np
>>> array0 = np.array([3,2,5,2,1])
>>> df = pd.DataFrame(array0, columns=['numbers'], index=['a','b','c','d','e'])
>>> df = df.ix[df.index[2:5]].sum()
>>> dr = df**2
>>> print df
numbers 8
```

Question 28 + Answer + Validation

What is the output of the following code?

Given a CSV file that contains the following data:

```
Name,Value,Company,Founded
George Soros,4000000000,Soros Fund Management,1969
Tucker Balch,1000000000000,Lucena Research,
```

```
import pandas as pd
df = pd.read_csv('top_hedgefund_managers.csv',
index_col="Company").sort(columns='Value', ascending=False).fillna('TBD')
print(df)
```

Select one answer:

a)

	Name	Value	Founded
Company			
Lucena Research	Tucker Balch	1000000000000	TBD
Soros Fund Management	George Soros	4000000000	1969

b)

	Name	Value	Founded

```
Company
Soros Fund Management   George Soros      4000000000      1969
Lucena Research         Tucker Balch  1000000000000       TBD
```

c)

```
                             Name          Value   Founded
Company
Lucena Research         Tucker Balch  1000000000000       NaN
Soros Fund Management   George Soros      4000000000      1969
```

d)

```
                   Value                Company Founded
Name
Tucker Balch   1000000000000      Lucena Research      TBD
George Soros      4000000000  Soros Fund Management    1969
```

Correct answer: a)

Python transcript:

```
>>> import pandas as pd
>>> data = '''
... Name,Value,Company,Founded
... George Soros,4000000000,Soros Fund Management,1969
... Tucker Balch,1000000000000,Lucena Research,
... '''
>>> fp = open("top_hedgefund_managers.csv", 'w')
>>> fp.write(data)
>>> fp.close()
>>> df = pd.read_csv('top_hedgefund_managers.csv',
index_col="Company").sort(columns='Value', ascending=False).fillna('TBD')
>>> print(df)
                             Name          Value Founded
Company
Lucena Research         Tucker Balch  1000000000000     TBD
Soros Fund Management   George Soros      4000000000    1969
```

Question 29 + Answer + Validation

What is the output of the following code snippet?

Code:

```
import numpy as np
m = np.array([[1, 3, 5], [2, 4, 6]])
n = np.array([[1, 2, 3], [1, 2, 3]])
print m.sum(axis=0)+n.mean(axis=0)
```

```
Select one answer:
a) [ 5, 11, 17]
b) 33
c) [ 11.,  14.]
d) [  4.,   9.,  14.]
```

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> m = np.array([[1, 3, 5], [2, 4, 6]])
>>> n = np.array([[1, 2, 3], [1, 2, 3]])
>>> print m.sum(axis=0)+n.mean(axis=0)
[  4.    9.   14.]
```

Question 30 + Answer + Validation

What is the output of the following python codes?

Codes:

```
import numpy as np
a = np.array([1,2,3,4,5])
print a * a[::-1]
```

```
Select one answer:
a) [5 8 9 8 8]
b) [5 8 8 8 5]
c) [5 8 9 8 5]
d) [5 9 9 8 5]
```

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([1,2,3,4,5])
>>> print a * a[::-1]
[5 8 9 8 5]
```

```
Question 31  + Answer + Validation

What is the output of this Python code?

Code:

        import pandas as pd

        data = {'A': [100, 60, 30, 130],
                'B': [5, 15, 15, 5],
                'C': [100, 300, 100, 200],
                'D': [20, 20, 30, 30]}

        df = pd.DataFrame(data)

        # print df
        #      A    B    C    D
        #0   100    5  100   20
        #1    60   15  300   20
        #2    30   15  100   30
        #3   130    5  200   30

        df = df.shift(2)

        print df.mean()

Select one answer:

a)
        A     40
        B      5
        C    100
        D     10

b)
        A     80
        B     10
        C    200
        D     20

c)
        A     80
        B     10
        C    175
        D     25

d)
        A    NaN
        B    NaN
        C    NaN
```

```
        D    NaN
```

Correct answer: b)

Python transcript:

```
>>> import pandas as pd
>>> data = {'A': [100, 60, 30, 130],
>>>         'B': [5, 15, 15, 5],
>>>         'C': [100, 300, 100, 200],
>>>         'D': [20, 20, 30, 30]}
>>> df = pd.DataFrame(data)

>>> # print df
>>> #      A    B    C    D
>>> #0  100    5  100   20
>>> #1   60   15  300   20
>>> #2   30   15  100   30
>>> #3  130    5  200   30
>>> df = df.shift(2)
>>> print df.mean()
A     80
B     10
C    200
D     20
dtype: float64
```

Question 32 + Answer + Validation

What will be the output of the following code:

```
a = [[]]*4
a[0].append(15)
print a
```

Select one answer:

```
a) [[[15]], [[]], [[]], [[]]]
b) [[15], [15], [15], [15]]
c) [[15], [], [], []]
d) [[[15]], [[15]], [[15]], [[15]]]
```

Correct answer b)

Python transcript:

```
>>> a = [[]]*4
>>> a[0].append(15)
>>> print a
[[15], [15], [15], [15]]
```

Question 33 + Answer + Validation

How should section A be filled in to complete code that will cause the following
output:

Code:

```
import pandas as pd
df = pd.DataFrame({
    "A": [1, 2],
    "B": [3, 4]
})
print df
print __A__
print df
```

Output:

```
   A  B
0  1  3
1  2  4
   A
0  1
1  2
   A  B
0  1  3
1  2  4
```

Select one answer:
a) df.drop("B", inplace=True)
b) df.drop("B", axis=0)
c) df.drop("B", axis=1)
d) df.drop("B", axis=1, inplace=True)

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> df = pd.DataFrame({
        "A": [1, 2],
        "B": [3, 4]
    })
>>> print df
   A  B
0  1  3
1  2  4
>>> print df.drop("B", axis=1)
   A
0  1
```

```
 1  2
>>> print df
    A  B
0  1  3
1  2  4
```

Question 34 + Answer + Validation

Fill the blank __A__ below to cause this Python code to give the following output:

Code:
```
import pandas as pd

i= [1,2,3,4,5,6]
d = [2,2.5,2.99,3.5,3.8,10]
df = pd.DataFrame(index=i, data=d)

print __A__
```

Output:
```
           0
1        NaN
2   0.250000
3   0.196000
4   0.170569
5   0.085714
6   1.631579
```

Select one answer:

a) df/df.shift(1) - 1
b) df/(df-1)
c) df/df.shift(1)
d) df*(df-1)

```
Correct answer: a)


Python transcript:

>>> import pandas as pd

>>> i= [1,2,3,4,5,6]

>>> d = [2,2.5,2.99,3.5,3.8,10]

>>> df = pd.DataFrame(index=i, data=d)

>>> print df/df.shift(1) - 1
            0
1       NaN
2  0.250000
3  0.196000
4  0.170569
5  0.085714
6  1.631579
```

```
Question 35  + Answer + Validation

What is the output of the following python code?

import numpy as np
x = np.array([[0,1,2],[3,4,5],[4,3,2]])
x = x**2
print x[-1] * x[1]
```

```
A)[144 144 100]
B)[0 4 10]
C)[0 16 100]
D)[48 48 40]
```

Answer: A

Transcript:

```
>>> import numpy as np
>>> x = np.array([[0,1,2],[3,4,5],[4,3,2]])
>>> x = x**2
>>> print x[-1] * x[1]
[144 144 100]
```

Question 36 + Answer + Validation

What is the output of the following code:

Code:

```
import numpy as np
import pandas as pd
s = pd.DataFrame([0, 1, 2, np.nan])
print s.size, s.count()
```

Select one answer:
```
a) 3 0    3
b) 3 0    4
c) 4 0    3
d) 4 0    4
```

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> s = pd.DataFrame([0, 1, 2, np.nan])
>>> print s.size, s.count()
4 0    3
dtype: int64
```

```
Question 37  + Answer + Validation

What is the output of the following python code:

Code:

import numpy as np
import pandas as pd
df = pd.DataFrame(np.array([(1,2),(3,4),(5,6)]),index=['a','c','e'],columns=
['data1','data2'])
df = df.reindex(['a','b','c','d','e'])
df = df.fillna(method='ffill')
df = df.sum(axis=1)
print df

Select one answer:

a)
data1    13
data2    18
dtype: float64

b)
a      3
b      3
c      7
d      7
e     11
dtype: float64

c)
data1    17
data2    22
dtype: float64

d)
a      3
b      7
c      7
d     11
e     11
dtype: float64

Correct answer: b)

Python transcript:

>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.array([(1,2),(3,4),(5,6)]),index=['a','c','e'],columns=
['data1','data2'])
>>> df = df.reindex(['a','b','c','d','e'])
>>> df = df.fillna(method='ffill')
>>> df = df.sum(axis=1)
>>> print df
a      3
```

```
b     3
c     7
d     7
e    11
dtype: float64
```

Question 38 + Answer + Validation

What is the output of the following Python code?

Code:

```
import numpy as np
import pandas as pd

arr = np.ones([1,5])
df = pd.DataFrame(arr)
df.ix[:,2:3] = 0
print df.values
```

Select one answer:
a) [1, 1, 0, 1, 1]
b) [1, 1, 0, 0, 1]
c) [1, 1, 1, 1, 1]
d) [0, 0, 0, 0, 0]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> arr = np.ones([1,5])
>>> df = pd.DataFrame(arr)
>>> df.ix[:,2:3] = 0
>>> print df.values
[[ 1.  1.  0.  0.  1.]]
```

```
Question 39  + Answer + Validation

How should section A be filled in to complete code that will cause the following
output:

Code:

import numpy as np
m = np.arange(20).reshape(4,5)
print m
print m.__A__

Output:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]

[[ 0  1  3  6 10]
 [ 5 11 18 26 35]
 [10 21 33 46 60]
 [15 31 48 66 85]]


Select one answer:

a) m.sum(axis=0)
b) m.sum(axis=1)
c) m.cumsum(axis=0)
d) m.cumsum(axis=1)

Correct answer: d)


Python transcript:
>>> import numpy as np
>>> m = np.arange(20).reshape(4,5)
>>> print m
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
>>> print m.cumsum(axis=1)
[[ 0  1  3  6 10]
 [ 5 11 18 26 35]
 [10 21 33 46 60]
 [15 31 48 66 85]]
```

Question 40 + Answer + Validation

What is the output of the following code:

Code:

import pandas

import numpy

df = numpy.array([25,4,78,54,6,21,45,2])

print pandas.rolling_max(df,window=2,min_periods=2)

Output:

Select one answer:

```
a)[ nan  78.  78.  78.  54.  21.  45.  45.]
b)[ nan  nan  25.  78.  78.  54.  21.  45.]
c)[ nan  6.   25.  78.  54.  21.  45.  45.]
d)[ nan  25.  78.  78.  54.  21.  45.  45.]
```

Correct answer:d)

Python transcript:

```
>>> import pandas

>>> import numpy

>>> df = numpy.array([25,4,78,54,6,21,45,2])

>>> print pandas.rolling_max(df,window=2,min_periods=2)

[ nan  25.  78.  78.  54.  21.  45.  45.]
```

Question 41 + Answer + Validation

What is the output of the following code?

Code:

```
import numpy as np
A = np.zeros((4,4), dtype=int)
A[1::2, ::2] = 1
A[::2, 1::2] = 1
print(A)
```

Select one answer:

a) [[1 0 1 0]
    [0 1 0 1]
    [1 0 1 0]
    [0 1 0 1]]

b) [[0 1 0 1]
    [1 0 1 0]
    [0 1 0 1]
    [1 0 1 0]]

c) [[0 1 0 0]
    [1 1 0 0]
    [0 0 0 0]
    [0 0 0 0]]

d) None of the above

Correct answer: b)

Python transcript:
```
>>> import numpy as np
>>> A = np.zeros((4,4),dtype=int)
>>> A[1::2, ::2] = 1
>>> A[::2,1::2] = 1
>>> print(A)
[[0 1 0 1]
 [1 0 1 0]
 [0 1 0 1]
 [1 0 1 0]]
```

Question 42 + Answer + Validation

What statement in place of _A_ would cause the following output:

Code:

```
import numpy as np

x = np.random.randint(5, size=(4, 5))
print x
_A_
print x
```

Output:

```
[[4 2 3 4 1]
 [2 2 1 0 1]
 [2 1 0 3 4]
 [3 2 1 1 3]]
```

```
[[  4    2    3    4    1]
 [777    2 777    0 777]
 [  2    1    0    3    4]
 [  3    2    1    1    3]]
```

```
Select one answer:
a) x[1][2::] = 777
b) x[1:3:5] = 777
c) x[1::3] = 777
d) x[1][::2] = 777

Correct answer: d)

Python Transcript:

ml4t@ml4t-VirtualBox:~/ml4t/mc1_p2$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x = np.random.randint(5, size=(4, 5))
>>> print x
[[0 1 2 0 1]
 [3 4 3 2 0]
 [1 0 1 1 1]
 [0 0 2 0 1]]
>>> x[1][::2] = 777
>>> print x
[[  0    1    2    0    1]
 [777    4 777    2 777]
 [  1    0    1    1    1]
 [  0    0    2    0    1]]
```

```
Question 43  + Answer + Validation

red = np.array([[3,4],[2,4]])
x = ([2,5],[2,3])
x = np.asarray(x)
blue = red * x[-2,:]

print blue

Select one answer:

a)[[3 4]
   [2 4]]

b)[[ 6 20]
   [ 4 20]]

c)[[2 5]
   [2 3]]

d)[[6 8]
   [4 8]]

Answer: b

Python transcript:
>>> import numpy as np
>>> red = np.array([[3,4],[2,4]])
>>> x = ([2,5],[2,3])
>>> x = np.asarray(x)
>>> blue = red * x[-2,:]
>>> print blue
[[ 6 20]
 [ 4 20]]
```

```
Question 44  + Answer + Validation

What is the output of this python code?

Code:
import numpy as np

arr = np.array([[1,2,5],[2,4,2],[3,3,2]])
x = arr[:,1].sum(),arr[0:].sum().sum(axis=0)
print x

Select one answer:
a) (9,24)
b) (9,8)
c) (8,24)
d) 33

correct answer: a

Python transcript:
```

```
>>> import numpy as np
>>> arr = np.array([[1,2,5],[2,4,2],[3,3,2]])
>>> x = arr[:,1].sum(),arr[0:].sum().sum(axis=0)
>>> print x
(9, 24)
```

```
Question 45  + Answer + Validation
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import numpy as np

x = np.array([[0, 1, 2, 3, 4],\
                        [5, 6, 7, 8, 9],\
                        [10, 11, 12, 13, 14]])

print x
print _A_
```

```
Output:
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
 [[3]
 [8]]
```

```
Select one answer:
a) x[:2,4]
b) x[3:4,0:2]
c) x[2,3]
d) x[0:2,3:4]
```

Correct answer: d)

```
Python transcript:
>>> import numpy as np
>>> x = np.array([[0, 1, 2, 3, 4],\
... [5, 6, 7, 8, 9],\
... [10, 11, 12, 13, 14]])
>>> print x
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
>>> print x[0:2,3:4]
[[3]
```

3/5/2016

```
  [8]]
Question 46  + Answer + Validation
```

What is the output of this Python code?

Code:

```
import numpy as np
j = np.array([(1,2,3),(4,5,6),(7,8,9)])
k = j[:,-1:]
print k.shape
```

Select one answer:
a) 3
b) (3, 1)
c) (1, 3)
d) [3 6 9]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> j = np.array([(1,2,3),(4,5,6),(7,8,9)])
>>> k = j[:,-1:]
>>> print k.shape
(3, 1)
```

```
Question 47  + Answer + Validation
```

What is the output of the following code?

```
>>>import numpy as np

>>>a=np.random.randint(0,10,size=(5,4))
>>>print 'a=\n',a

 a=
 [[4 1 6 9]
  [4 9 2 5]
  [7 0 9 7]
  [6 1 9 5]
  [6 6 3 9]]
```

what is the output of the following code?
```
print a[1:3,1:2].sum()
```
a) 25
b) 26

```
c) 11
d) 9

answer d) 9

Python transcript:
>>>import numpy as np
>>>a=np.random.randint(0,10,size=(5,4))
>>>print 'a=\n',a
  a=
   [[4 1 6 9]
    [4 9 2 5]
    [7 0 9 7]
    [6 1 9 5]
    [6 6 3 9]]
>>>print a[1:3,1:2]
   [[9]
    [0]]
```

Question 48

What is the output of the following python code?

Code:

```
import pandas as pd
data = { 'key': [ 'B','D','C','A' ], 'value': [ 1,2,3,4 ]}
sub = pd.DataFrame.from_dict( data ).sort( columns=[ 'key' ]).ix[ 0:1 ]
print sub.value.mean()
```

Select one answer:
a) 1.5
b) 2.0
c) 2.5
d) 3.0

Answer: b

Python transcript:

```
>>> import pandas as pd
>>> data = { 'key': [ 'B','D','C','A' ], 'value': [ 1,2,3,4 ]}
>>> sub = pd.DataFrame.from_dict( data ).sort( columns=[ 'key' ]).ix[ 0:1 ]
>>> print sub.value.mean()
2.0
```

Question 49

How should section A be filled in to complete code that will cause the following
output:
(Select all the values bigger than 10 from the second column)

Code:

```
import numpy as np
r = np.random.randint(20, size=(4, 6))
print r
print _A_
```

Output:
```
[[18 15 12 12 17  7]
 [18  6  9 16  5  2]
 [ 9 18 11 10  6 18]
 [12  8 19 19  4 16]]
[[15 18]]
```

Select one answer:
a) r[np.where(r[:,0]>10),0]
b) r[np.where(r[:,1]>10),1]
c) r[1,np.where(r[:,1]>10)]
d) r[0,np.where(r[:,0]>10)]

Correct answer: b)

Python Transcript:

```
>>> import numpy as np
>>> r = np.random.randint(20, size=(4, 6))
>>> print r
[[18 15 12 12 17  7]
 [18  6  9 16  5  2]
 [ 9 18 11 10  6 18]
 [12  8 19 19  4 16]]
>>> print r[np.where(r[:,1]>10),1]
[[15 18]]
>>>
```


Question 50

What is the output of this Python code?

```
import numpy as np
x = np.array([8,1,3,4,5,2,1,9,6,4,7])
print(np.mean(x[2:7][::-1][:-1]))
```

Select one answer:

a) 5
b) 3
c) 5.0
d) 3.0

Answer: d

```
>>> import numpy as np
>>> x = np.array([8,1,3,4,5,2,1,9,6,4,7])
>>> print(np.mean(x[2:7][::-1][:-1]))
3.0
```

```
Question 51

What is the output of the following python code?

Code:

import numpy as np
a = np.array([2, 7, 11, 9, 3])
a[a < 5] = 5
a[2] = 10
print a


Select one answer:
a) [ 2   5 10   5   3]
b) [ 5   7 10   5   3]
c) [ 5   7 10   9   5]
d) [ 5 10 11   9   5]

Correct answer: c)

Python transcript:

>>> import numpy as np
>>> a = np.array([2, 7, 11, 9, 3])
>>> a[a < 5] = 5
>>> a[2] = 10
>>> print a
[ 5   7 10   9   5]
```

```
Question 52
```

Given a pandas data frame df, which block of statements will label the x and y axes in the resultant plot?

Choose the best answer from those provided:

a)

```
        df.plot(title='P Values', fontsize=12, x-axis='Date', y-axis='P Value')
        plt.show()
```

b)

```
        df.plot(title='P Values', fontsize=12)
        df.xaxis = 'Date'
        df.yaxis = 'P Value'
        plt.show()
```

c)

```
        ax = df.plot(title='P Values', fontsize=12)
        ax.set_xlabel('Date')
        ax.set_ylabel('P Value')
        plt.show()
```

d)

```
        df.plot(title='P Values', fontsize=12, xlabel='Date', ylabel='P Value')
        plt.show()
```

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> dates = pd.date_range('2010-01-01', periods=8)
>>> df = pd.DataFrame(np.random.randn(8, 1), index=dates, columns=list('P'))
>>> ax = df.plot(title='P Values', fontsize=12)
>>> ax.set_xlabel('Date')
<matplotlib.text.Text object at 0x7f710cb12cd0>
>>> ax.set_ylabel('P Value')
<matplotlib.text.Text object at 0x7f70f52dec50>
>>> plt.show()
```
Question 53

Which is NOT an example of code to replace section _A_ that can calculate the difference between neighboring rows in an ndarray (as illustrated by the output)?

```
import pandas as pd
a = pd.Series([1, 2, 4, 7, 11, 16])
_A_
print a
print b.values[-1]*1.0
```

Output:
```
0     1
1     2
2     4
3     7
4    11
5    16
dtype: int64
5.0
```


a)
```
b = a-a.shift(1)
b=b[1:]
```
b)

```
b=pd.Series([])
for i in range(len(a)-1):
    b[i] = a[i+1] - a[i]
c)
b = a-a.shift(1)
d)
b = a-a.shift(1)
b=b[:-1]
```

Answer: (d)

Question 54

Given a dataframe df depicting the daily returns of a portfolio,
describe the output of the print statement below:

Code:
```
import pandas as pd
data = [1.0, 1.05, 1.2, 0.99, 1.5]
# create dates starting Jan 1st and ending Jan 5th
ind = pd.date_range(start='2016-01-01', end='2016-01-05')
name = ['daily_returns']
df = pd.DataFrame(data=data, index=ind, columns=name)
a = df['daily_returns'] - df['daily_returns'].mean()
b = (a**2).sum()
c = df.shape[0]-1
# what is the output of the print statement below?
print (b/c)**0.5
```

a) The sample standard deviation of daily_returns
b) The population standard deviation of daily_returns
c) The sum of squared error of daily_returns
d) The root mean squared error of daily_returns

Correct answer: a

Python transcript:
```
>>> import pandas as pd
>>> data = [1.0, 1.05, 1.2, 0.99, 1.5]
>>> # create dates starting Jan 1st and ending Jan 5th
>>> ind = pd.date_range(start='2016-01-01', end='2016-01-05')
>>> name = ['daily_returns']
>>> df = pd.DataFrame(data=data, index=ind, columns=name)
>>> a = df['daily_returns'] - df['daily_returns'].mean()
>>> b = (a**2).sum()
>>> c = df.shape[0]-1
>>> # what is the output of the print statement below?
>>> print (b/c)**0.5
0.213939
>>> # pandas function for sample std
>>> df.std()
daily_returns    0.213939
```
Question 55

How should section A be filled to print out only the first two values in the array as
seen in the sample output below:

Which of the following returns the last five values of the array

Code:

```
import numpy as np

a = np.random.random([5])
print a
print _A_
```

Output:

```
[ 0.70059652  0.98449675  0.75068418  0.31808623  0.14485409]
[ 0.70059652  0.98449675]
```

Select one answer:
a) a[-2]
b) a[:-2]
c) a[-5:-3]
d) a[2:-1]

Correct answer: c)

Python transcript:

```
>>> import numpy as np

>>> a = np.random.random([5])
>>> print a
[ 0.70059652  0.98449675  0.75068418  0.31808623  0.14485409]
>>> print print a[-5:-3]
[ 0.70059652  0.98449675]
```

Question 56

Given 100 days of data (sampled daily), which is the proper python formula to
calculate sharpe ratio? Assume sr is Sharpe Ratio, dr is daily return, rfr is risk-
free-rate and np is a numpy library.
    a. sr = np.sqrt(252) * (dr - rfr).mean()/dr.std()
    b. sr = np.sqrt(252) * np.std(dr - rfr)/np.mean(dr -rfr)
    c. sr = np.sqrt(252) * np.mean(dr - rfr)/np.std(dr)
    d. sr = np.sqrt(100) * (dr - rfr).mean()/dr.std()

correct answer: a

proof of difference:
```
import pandas as pd
import numpy as np

def assess_port():
    data = [["2010-12-08", .0028],["2010-12-09", .0015],["2010-12-10", -.0054],["2010-
12-11", -.0058],["2010-12-12", .0061],["2010-12-13", .0011]]
    dr = pd.DataFrame(data)

    pd_sr = np.sqrt(252) * (dr).mean()/dr.std()
    np_sr = np.sqrt(252) * np.mean(dr).mean()/np.std(dr)

    return pd_sr.ix[1], np_sr.ix[1]
```

```
if __name__ == "__main__":
    pd_sr, np_sr = assess_port()
    print "Pandas Sharpe Ratio: ", pd_sr
    print "Numpy Sharpe Ratio: ", np_sr
```

Question 57

Which is the correct code for Section _A_ to set the start date values in a Pandas
Datafame to 0 assuming there are two or more securities?

```
dates = pd.date_range(dt.datetime(2005,01,01), dt.datetime(2005,01,05))
# data_frame is a Pandas Data Frame retrieved using the util.py helper function
data_frame = get_data(['GOOG','AAPL','GLD'], dates)

#set first trading day in series to 0
_A_

print data_frame
```

```
Output:
              SPY     GOOG    AAPL     GLD
2005-01-03    0.00    0.00    0.00    0.00
2005-01-04  102.65  194.50   31.83   42.74
2005-01-05  101.94  193.51   32.11   42.67
```

Select one answer:
a) data_frame.ix[:,0] = 0
b) data_frame.ix[0,:] = 0
c) data_frame[0,:] = 0
d) data_frame[:,0] = 0

Correct Answer: b)

```
Transcript:
>>> dates = pd.date_range(dt.datetime(2005,01,01), dt.datetime(2005,01,05))
>>> data_frame = get_data(['GOOG','AAPL','GLD'], dates)
>>>
>>> data_frame.ix[0,:] = 0
>>> print data_frame
              SPY     GOOG    AAPL     GLD
2005-01-03    0.00    0.00    0.00    0.00
2005-01-04  102.65  194.50   31.83   42.74
2005-01-05  101.94  193.51   32.11   42.67
```

Question 58

What is the output of the following code:

```
import numpy as np

a=np.array([(10.0,20.0),(1.0,2.0)])
```

```
b=np.array([(100,200),(1,2)])

print "\nDivide a by b:\n", a/b
```

Select one answer:

a)   Divide a by b:
```
        [[0 0]
         [1 1]]
```

b)   Divide a by b:
```
        [[ 0.1  0.1]
         [ 1.   1. ]]
```

c)   Divide a by b:
```
        [[0.0 0.0]
         [1.0 1.0]]
```

d)   Divide a by b:
```
        [[10 10]
         [.10 .10]]
```

Correct answer: b)

Python transcript:

```
>>>import numpy as np
>>>a=np.array([(10.0,20.0),(1.0,2.0)])
>>>b=np.array([(100,200),(1,2)])
>>>print "\nDivide a by b:\n", a/b

Divide a by b:
[[ 0.1  0.1]
 [ 1.   1. ]]
```

Question 59

How should 'section A' be filled in to complete code that will cause the following
output:

```
Code:
import pandas as pd
d = {'one' : [1., 2., 3., 4.],
     'two' : [2., 3., 4., 5.],
     'chaz': [3., 10., -2., 0]}
df = pd.DataFrame(d)
df_dic = {}
for i in range(-2,3):
        df_dic["{0}".format(i)] = df * i
print _A_
```

Output:
```
   chaz  one  two
0    12    4    8
1    40    8   12
2    -8   12   16
3     0   16   20
```

Select one answer:
a) df_dic[1] * 4
b) df_dic[2]
c) df_dic['2']
d) df_dic['2'] - df_dic['-2']

```
Correct answer: d)

Python transcript:

>>> import pandas as pd
>>> d = {'one' : [1., 2., 3., 4.],
     'two' : [2., 3., 4., 5.],
     'chaz': [3., 10., -2., 0]}
>>> df = pd.DataFrame(d)
>>> df_dic = {}
>>> for i in range(-2,3):
               df_dic["{0}".format(i)] = df * i

>>> print df_dic['2'] - df_dic['-2']
```

Question 60

What is the output of the following code?

Code:

```
import numpy as np
data = np.array([
        [2.0, 4.0, 8.0],
        [1.0, 2.0, 4.0],
        [4.0, 8.0, 16.0]])
output = data.sum(axis=1)
print(output)
```

Choices:

A) 49.
B) [7., 14., 28.]
C) 14.
D) [14., 7., 28.]

Correct Answer: D

Python transcript:

```
>>> import numpy as np
>>> data = np.array([[2.0, 4.0, 8.0],[1.0, 2.0, 4.0],[4.0, 8.0, 16.0]])
>>> output = data.sum(axis=1)
>>> print(output)
>>> [14., 7., 28.]
```

Question 61

What is the output of this python code?

Code:
```
array_1 = [[0, 1, 2, 3, 4],
           [5, 6, 7, 8, 9],
           [10, 11, 12, 13, 14],
```

```
              [15, 16, 17, 18, 19],
              [20, 21, 22, 23, 24],
              [25, 26, 27, 28, 29]]
array_2 = [[0, 1, 2, 3, 4],
              [5, 6, 7, 8, 9]]
df = pd.DataFrame(array_1, columns=list('abcde'))
x = np.array(array_2)
df = df.ix[2:2, ['a', 'b']] * x[-1, 1:5:2]
print df.get_values()
```

```
Select one answer:
a) [[25 42]]
b) [[60 88]]
c) [[50 77]]
d) Error message
```

Correct answer: b)

```
Python transcript:
>>> import pandas as pd
>>> import numpy as np
>>> array_1 = [[0, 1, 2, 3, 4],
...            [5, 6, 7, 8, 9],
...            [10, 11, 12, 13, 14],
...            [15, 16, 17, 18, 19],
...            [20, 21, 22, 23, 24],
...            [25, 26, 27, 28, 29]]
>>> array_2 = [[0, 1, 2, 3, 4],
...            [5, 6, 7, 8, 9]]
>>> df = pd.DataFrame(array_1, columns=list('abcde'))
>>> x = np.array(array_2)
>>>
>>> df = df.ix[2:2, ['a', 'b']] * x[-1, 1:5:2]
>>> print df.get_values()
[[60 88]]
Question 62
```

What is the output of the following code?

```
import numpy as np
a = np.array([[ 1,  2,  3,  4],
              [ 5,  6,  7,  8],
              [ 9, 10, 11, 12],
              [13, 14, 15, 16]])
print a[-1:,1:3]
```

```
Select one answer from the following options:
(a) [10 11]
(b) [14 15]
(c) [14 15 16]
(d) [13 14 15 16]
```

Correct answer: (b)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[ 1,  2,  3,  4],
...               [ 5,  6,  7,  8],
...               [ 9, 10, 11, 12],
```

```
...                            [13, 14, 15, 16]])
>>> print a[-1:,1:3]
[14 15]
```

Question 63

What would be the output of the following code:

Code:

```
import numpy as np
a = np.array([[ 1, 2, 3],
              [ 4, 5, 6],
              [ 7, 8, 9]])

print a[0,:] * sum(a[0:2,-2])
```

Select one answer:
a) [5 10 15]
b) [2 4 6]
c) [7 14 21]
d) [7 28 49]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[ 1, 2, 3],
...               [ 4, 5, 6],
...               [ 7, 8, 9]])
...
>>> print a[0,:]
[1 2 3]
>>> print sum(a[0:2,-2])
7
>>> print a[0,:] * sum(a[0:2,-2])
[ 7 14 21]
```

Question 64

By default, numpy's std() function calculates the uncorrected sample standard
deviation of a given numpy array. If x is a numpy array of real numbers, then,
which of the following lines of code will return the same value as numpy.std(x)?

```
a) sum((x - x.mean())**2) / (len(x))
b) sum((x - x.mean())**2) / (len(x) - 1)
c) math.sqrt(sum((x - x.mean())**2) / (len(x)))
d) math.sqrt(sum((x - x.mean())**2) / (len(x) - 1))
```

------ ANSWER ------

The correct answer is C.

A calculates the uncorrected sample variance, B calculates the corrected sample
variance, and D calculates the corrected sample standard deviation.

---- TRANSCRIPT ----

```
import math
import numpy

x = numpy.random.rand(1000)

a = sum((x - x.mean())**2) / (len(x))
b = sum((x - x.mean())**2) / (len(x) - 1)
c = math.sqrt(sum((x - x.mean())**2) / (len(x)))
d = math.sqrt(sum((x - x.mean())**2) / (len(x) - 1))

assert(a != numpy.std(x) and a == numpy.var(x))
assert(b != numpy.std(x) and b == numpy.var(x, ddof=1))
assert(c == numpy.std(x))
assert(d != numpy.std(x) and d == numpy.std(x, ddof=1))
```

Question 65

Choose the correct output of the print statement of the following code:

```
import numpy as np
C = np.ndarray([2,2], buffer=np.matrix([[1, 2], [3, 4]]), dtype=int)
print C[:,0:1]
```

Select one answer:
a) [1 3]
b) [[1]
    [3]]
c) [[1 2]
    [3 4]]
d) [[1 2]]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> C = np.ndarray([2,2], buffer=np.matrix([[1, 2], [3, 4]]), dtype=int)
>>> print C
[[1]
 [3]]
```

```
Question 66

These pieces of python code gives out outcomes as:

>>> x = np.arange(1,3).reshape(-1,1)
[[1],
 [2]]

>>> a = np.array([1.0, 2.0, 3.0])
>>> b = [2.0]
>>> print a * b
[ 2.  4.  6.]

Then what is the output of this python code?

>>> import numpy as np
>>> a = np.arange(1,10).reshape(-1,1)
>>> b = np.arange(1,10)
>>> print a*b

Choices:
a)
[ 1   4   9 16 25 36 49 64 81]


b)
[[ 1   2   3   4   5   6   7   8   9]
 [ 2   4   6   8 10 12 14 16 18]
 [ 3   6   9 12 15 18 21 24 27]
 [ 4   8 12 16 20 24 28 32 36]
 [ 5 10 15 20 25 30 35 40 45]
 [ 6 12 18 24 30 36 42 48 54]
 [ 7 14 21 28 35 42 49 56 63]
 [ 8 16 24 32 40 48 56 64 72]
 [ 9 18 27 36 45 54 63 72 81]]


c)
[ 9 16 21 24 25 24 21 16  9]


d)
[285]    # np.sum([1,  4,  9, 16, 25, 36, 49, 64, 81]) = 285


Correct answer : b)


Python transcript:

#For answer b)
>>> import numpy as np
>>> a = np.arange(1,10).reshape(-1,1)
>>> b = np.arange(1,10)
```

```
>>> print a
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
>>> print b
[1 2 3 4 5 6 7 8 9]
>>> print a*b
[[ 1  2  3  4  5  6  7  8  9]
 [ 2  4  6  8 10 12 14 16 18]
 [ 3  6  9 12 15 18 21 24 27]
 [ 4  8 12 16 20 24 28 32 36]
 [ 5 10 15 20 25 30 35 40 45]
 [ 6 12 18 24 30 36 42 48 54]
 [ 7 14 21 28 35 42 49 56 63]
 [ 8 16 24 32 40 48 56 64 72]
 [ 9 18 27 36 45 54 63 72 81]]
>>> print b*a   ## The same as a*b
[[ 1  2  3  4  5  6  7  8  9]
 [ 2  4  6  8 10 12 14 16 18]
 [ 3  6  9 12 15 18 21 24 27]
 [ 4  8 12 16 20 24 28 32 36]
 [ 5 10 15 20 25 30 35 40 45]
 [ 6 12 18 24 30 36 42 48 54]
 [ 7 14 21 28 35 42 49 56 63]
 [ 8 16 24 32 40 48 56 64 72]
 [ 9 18 27 36 45 54 63 72 81]]
```

Question 67

How should section A be filled in to complete code that will cause the following output:

Code:

```
import pandas as pd
left_frame = pd.DataFrame({'key': range(5),
                           'left_value': ['a', 'b', 'c', 'd', 'e']})
right_frame = pd.DataFrame({'key': range(2, 7),
                            'right_value': ['f', 'g', 'h', 'i', 'j']})
print left_frame
print right_frame

print _A_
```

Output:

```
   key left_value
0    0          a
1    1          b
2    2          c
3    3          d
4    4          e
   key right_value
0    2          f
1    3          g
2    4          h
3    5          i
```

```
4     6           j
   key left_value right_value
0     2            c            f
1     3            d            g
2     4            e            h
```

Select one answer:
a) pd.merge(left_frame, right_frame, on='key', how='outer')
b) pd.concat([left_frame, right_frame])
c) pd.merge(left_frame, right_frame, on='key', how='inner')
d) pd.concat([left_frame, right_frame], axis=1)

Question 68

What is the output of the following python code?

Code:
```python
import numpy as np
array = np.random.randint(4, 5, size = (6, 7))
print array.shape[0]
```

Select one answer:
a) 4
b) 5
c) 6
d) 7

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> array = np.random.randint(4, 5, size = (6, 7))
>>> print array.shape[0]
6
>>>
```


Question 69

What is the output of the following code?

Code:

```python
import numpy as np
x = np.array([5,4,3,2,1])
print x[-1:] * x[:-1]
```

Select one answer:
a) [20 12 6 2]
b) [5 8 9 8]
c) [10 12 12 10]
d) [5 4 3 2]

Correct answer: d)

Transcript:

```
>>> import numpy as np
>>> x = np.array([5,4,3,2,1])
>>> print x[-1:] * x[:-1]
[5 4 3 2]
```

Question 70

How do we normalize a 2d numpy matrix such as the one below, so that sum of all
elements is 1.0?

```
[[ 0.,   1.,   2.],
 [ 3.,   4.,   5.],
 [ 6.,   7.,   8.]]
```

How should section A be filled in to complete code that will cause b to have the value
shown in output below.

```
import numpy as np
a = np.arange(9, dtype=float)
a = a.reshape(3,3)
b = _A_
```

Output:

```
[[ 0.        ,  0.02777778,  0.05555556],
 [ 0.08333333,  0.11111111,  0.13888889],
 [ 0.16666667,  0.19444444,  0.22222222]]
```

Select one answer:

a) sum(a)/a
b) a/sum(a)
c) a/sum(sum(a))
d) none of the above

```
>>> import numpy as np
>>> a = np.arange(9, dtype=float)
>>> a = a.reshape(3,3)
>>> b = a/sum(sum(a))
>>> b
array([[ 0.        ,  0.02777778,  0.05555556],
       [ 0.08333333,  0.11111111,  0.13888889],
       [ 0.16666667,  0.19444444,  0.22222222]])
```

What is the output of this python code?

Code:

```
import pandas as pd

w = [0, 1, 2]
df = pd.DataFrame([[1, 1, 1], [2, 2, 2]], columns=["A", "B", "C"], index=["X", "Y"])
df = df.multiply(w)

print df.ix["Y", "C"]
```

Select one answer:
a) 4
b) 2
c) 1
d) 0

Correct answer: a)

Python transcript:
```
>>> import pandas as pd
>>>
>>> w = [0, 1, 2]
>>> df = pd.DataFrame([[1, 1, 1], [2, 2, 2]], columns=["A", "B", "C"], index=["X",
"Y"])
>>> df = df.multiply(w)
>>>
>>> print df.ix["Y", "C"]
4
```

How should section A be filled in to complete code that will cause the following
output:

```
===============CODE================

import numpy as np
import pandas as pd

j = pd.DataFrame(np.random.randn(3, 3),columns=['a', 'b', 'c'])
j.ix[1,:] = np.nan

print "BEFORE:"
print j

j.fillna(method='__A__', inplace="TRUE")

print "\n\nAFTER:"
print j
```

```
==============OUTPUT==============
BEFORE:
          a          b          c
0  1.413829   1.752905  -0.597698
1       NaN        NaN        NaN
2 -0.396044   1.682260   2.131227


AFTER:
          a          b          c
0  1.413829   1.752905  -0.597698
1 -0.396044   1.682260   2.131227
2 -0.396044   1.682260   2.131227
```

a)forward
b)ffill
c)backward
d)bfill

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> j = pd.DataFrame(np.random.randn(3, 3),columns=['a', 'b', 'c'])
>>> j.ix[1,:] = np.nan
>>> print j
          a          b          c
0  0.591730 -0.063186   0.439702
1       NaN        NaN        NaN
2 -0.625829   0.827364  -0.139017
>>> j.fillna(method='bfill', inplace="TRUE")
>>> print j
          a          b          c
0  0.591730 -0.063186   0.439702
1 -0.625829   0.827364  -0.139017
2 -0.625829   0.827364  -0.139017
```

What is the output of this python code?

Code:

```
import numpy as np
x = np.ones([2,3])
y = [[1,0.5,-0.5],[1,0.5,1]]
print 2*(x-y)
```

Select one answer:
a) [[0. 1. 3.]
    [0. 1. 0.]]

b) [[0. 1. 4.]
    [0. 1. 0.]]

c) [[0. 2. 3.]
    [0. 2. 0.]]

d) None of the above. The print statement will produce an error since 'x' is of type 'numpy.ndarray' and 'y' is of type 'list'

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> x = np.ones([2,3])
>>> y = [[1,0.5,-0.5],[1,0.5,1]]
>>> print 2*(x-y)
[[ 0.  1.  3.]
 [ 0.  1.  0.]]
```


How should section _A_ filled in to get the below outputs? with extra column (dv) has the sum of the equities each day.

Code:
```
import pandas as pd
import numpy as np

df = pd.DataFrame(  data=[[1.0, 2.0, 3.0],[1.0, 2.0, 3.0],[1.0, 2.0, 3.0]]
                  , columns=['GOOG','APPL','XOM'],index=
['1/1/2016','1/2/2016','1/3/2016'])

print df
_A_
print df
```

Output:
```
          GOOG  APPL  XOM  dv
1/1/2016     1     2    3   6
1/2/2016     1     2    3   6
1/3/2016     1     2    3   6
```


Select one answer:
 A) df['dv'] = np.sum(df,axis=1)
 B) df += df['dv'].sum(axis=1)
 C) df['dv'] = df[:,1:3].sum()
 D) df.columns.add('dv').sum(df,axis=1)

 Correct answer: a)

Python transsscript:
```
>>> import pandas as pd
>>> import numpy as np
>>> df = pd.DataFrame(  data=[[1.0, 2.0, 3.0],[1.0, 2.0, 3.0],[1.0, 2.0, 3.0]]
...                   , columns=['GOOG','APPL','XOM'],index=
['1/1/2016','1/2/2016','1/3/2016'])
>>> print df
          GOOG  APPL  XOM
1/1/2016     1     2    3
1/2/2016     1     2    3
1/3/2016     1     2    3
>>> df['dv'] = np.sum(df,axis=1)
>>> print df
          GOOG  APPL  XOM  dv
1/1/2016     1     2    3   6
```

```
1/2/2016      1      2      3      6
1/3/2016      1      2      3      6
```

In the following program what should be the output marked as _OUTPUT_ below?

Code:

```
>>> import pandas as pd, numpy as np
>>> cols = ["AGE","HEIGHT", "WEIGHT"]
>>> data = [[1,1,None],[2,None,2]]
>>> df1 = pd.DataFrame(data, columns=cols)
>>> print df1
   AGE  HEIGHT  WEIGHT
0    1       1     NaN
1    2     NaN       2

>>> df1.fillna(20)
   AGE  HEIGHT  WEIGHT
0    1       1      20
1    2      20       2

>>> print df1['HEIGHT'].mean() / df1['WEIGHT'].mean()
_OUTPUT_
```

Options:
a) NaN
b) 0.5
c) 1.0
d) 0.95454545454545459

Correct Answer: b)

Python transcript:

```
>>> print df1['HEIGHT'].mean() / df1['WEIGHT'].mean()
0.5
```

The fillna() method takes a parameter to replace the values "inplace", which is by default False. Therefore, df1.fillna(20) has no effect on original dataframe df1.

What is the expected output of the following code?

Code:

```
import pandas as pd
my_df1 = pd.DataFrame({'X' : ['x1','x2','x3'], 'Y' : ['y1','y2','y3']})
my_df2 = pd.DataFrame({'X' : ['x1','x2','x3'], 'Z' : ['b1','b2','b3']})

my_object = [my_df1, my_df2]

my_result = pd.concat(my_object)

print(my_result)
```

Select one answer:
a)
```
    X    Y    Z
0  x1   y1  NaN
1  x2   y2  NaN
2  x3   y3  NaN
```

```
0   x1    y1    b1
1   x2    y2    b2
2   x3    y3    b3
```

b)
ERROR

c)
```
    X     Y     Z
0   x1    y1    b1
1   x2    y2    b2
2   x3    y3    b3
```

d)
```
    X     Y     Z
0   x1    y1    NaN
1   x2    y2    NaN
2   x3    y3    NaN
0   x1    NaN   b1
1   x2    NaN   b2
2   x3    NaN   b3
```

Correct answer: d)

Python transcript:

```
>>> import pandas as pd
>>> my_df1 = pd.DataFrame({'X' : ['x1','x2','x3'], 'Y' : ['y1','y2','y3']})
>>> my_df2 = pd.DataFrame({'X' : ['x1','x2','x3'], 'Z' : ['b1','b2','b3']})
>>>
>>> my_object = [my_df1, my_df2]
>>>
>>> my_result = pd.concat(my_object)
>>>
>>> print(my_result)
    X     Y     Z
0   x1    y1    NaN
1   x2    y2    NaN
2   x3    y3    NaN
0   x1    NaN   b1
1   x2    NaN   b2
2   x3    NaN   b3
```

```
import pandas as pd

data =[ [ 126.29,   665.41,   409.47,   155.92],
        [ 126.49,   668.28,   411.67,   156.71],
        [ 126.82,   659.01,   416.24,   157.78],
        [ 126.50,   650.02,   420.59,   157.20],
        [ 126.80,   622.46,   419.93,   156.50],
        [ 127.90,   623.14,   421.43,   158.64],
        [ 127.97,   625.96,   420.74,   159.67],
        [ 128.28,   629.64,   419.59,   160.38] ]

cols = ['SPY', 'AAPL', 'GOOG', 'GLD']
dates = pd.date_range('2010-01-01', '2010-01-08')
df = pd.DataFrame(data, columns=cols, index=dates)

print _A_
```

Fill the blank (_A_) to get the following output.

```
              GLD
2010-01-01  155.92
2010-01-03  157.78
2010-01-05  156.50
2010-01-07  159.67
```

Select one answer:

```
a) df.ix[:4, 'GLD']
b) df.ix[0:8:2, -1:]
c) df.ix[1:8, -1].head(4)
d) df['GLD'].head(4)
```

Correct answer: b)

Python Transcript:

```
>>>
>>>
>>> import pandas as pd
>>> data =[ [ 126.29,   665.41,   409.47,   155.92],
...         [ 126.49,   668.28,   411.67,   156.71],
...         [ 126.82,   659.01,   416.24,   157.78],
...         [ 126.50,   650.02,   420.59,   157.20],
...         [ 126.80,   622.46,   419.93,   156.50],
...         [ 127.90,   623.14,   421.43,   158.64],
...         [ 127.97,   625.96,   420.74,   159.67],
...         [ 128.28,   629.64,   419.59,   160.38] ]
>>> cols = ['SPY', 'AAPL', 'GOOG', 'GLD']
>>> dates = pd.date_range('2010-01-01', '2010-01-08')
>>> df = pd.DataFrame(data, columns=cols, index=dates)
>>> print df
              SPY     AAPL     GOOG      GLD
2010-01-01  126.29  665.41  409.47  155.92
2010-01-02  126.49  668.28  411.67  156.71
2010-01-03  126.82  659.01  416.24  157.78
2010-01-04  126.50  650.02  420.59  157.20
2010-01-05  126.80  622.46  419.93  156.50
2010-01-06  127.90  623.14  421.43  158.64
2010-01-07  127.97  625.96  420.74  159.67
2010-01-08  128.28  629.64  419.59  160.38

>>> print df.ix[0:8:2, -1:]
              GLD
2010-01-01  155.92
2010-01-03  157.78
2010-01-05  156.50
2010-01-07  159.67
>>>
```


Complete the code below to select first three rows of the dataframe df and divide by
the minimum of each column, then print the result.

Code:

```
import pandas as pd
arr=[[12,4],[6,20],[18,40],[3,16]]
```

```
df=pd.DataFrame(arr,columns=['A','B'])

print _____
```

Select one answer:
a)df[:3,]/df.min(axis=0)
b)df[0:2,]/df.min(axis=1)
c)df.iloc[:-1,]/df.min(axis=0)
d)df.iloc[0:3,]/df.min(axis=1)

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> arr=[[12,4],[6,20],[18,40],[3,16]]
>>> df=pd.DataFrame(arr,columns=['A','B'])
>>> print df.iloc[:-1,]/df.min(axis=0)
   A   B
0  4   1
1  2   5
2  6  10
```


What is the output of this code:

```
import numpy as np
import pandas as pd
df = pd.DataFrame([1, 2, 3, 5, 8])
df.apply(lambda x: x ** x)
```

Select one answer:
a)

```
    0
0   1
1   4
2   9
3  25
4  64
```

b)

```
          0
0         1
1         4
2        27
3      3125
4  16777216
```

c)

```
    0
0   2
1   4
2   6
3  10
4  16
```

d)

```
    0
0   1
1   2
```

```
2   3
3   5
4   8
```

Correct answer: b)

Python transcript:
```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame([1, 2, 3, 5, 8])
>>> df.apply(lambda x: x ** x)
            0
0           1
1           4
2          27
3        3125
4  16777216
>>>
```

What is the output of the following code?

```
# Credit: borrowed and adapted from the lectures
(https://github.com/syednasar/ml4t/blob/master/lessons/1.8%20Optimizers.ipynb)

import scipy.optimize as spo

def f(x):
    '''Given a scalar x, return some value'''
    y = (x-7)**2 + 13
    return y

x_guess = 2.0

min_result = spo.minimize(f, x_guess,
                          method = 'SLSQP')

print "Minimum found at: X = {}, Y = {}".format(min_result.x, min_result.fun)
```

Select one answer:
a) Minimum found at: X = [-7.], Y = [13.]
b) Minimum found at: X = [-7.], Y = [-13.]
c) Minimum found at: X = [2.], Y = [38.]
d) Minimum found at: X = [7.], Y = [13.]

Correct answer: d)

Python transcript:

```
>>> import scipy.optimize as spo
>>> def f(x):
...     '''Given a scalar x, return some value'''
...     y = (x-7)**2 + 13
...     return y
>>> x_guess = 2.0
>>> min_result = spo.minimize(f, x_guess,
...                           method = 'SLSQP')
>>> print "Minimum found at: X = {}, Y = {}".format(min_result.x, min_result.fun)
Minimum found at: X = [ 7.], Y = [ 13.]
```

How should the blank be filled in to satisfy the mentioned constraint that will cause
the following output:

Code:
```python
import scipy.optimize as spo

def f(x):
    '''Arbitrary function'''
    n = sum(x**2)
    return n

guess = [0.5, 0.1, 1.2, 0.2]
const = _BLANK_  'Fill in constraint that makes sure the sum of the minimized result
adds up to two'

result = spo.minimize(f, guess, method = 'SLSQP', constraints=const)

print result.x
```

Output:
```
[ 0.5  0.5  0.5  0.5]
```

Select one answer:
a) const = ({ 'type': 'eq', 'fun': lambda x: sum(x) == 2})
b) const = ({ 'type': 'eq', 'fun': lambda x: 2 - sum(x)})
c) const = ({ 'type': 'ineq', 'fun': lambda x: sum(x) == 2})
d) const = ({ 'type': 'ineq', 'fun': lambda x: 2 - sum(x)})

Correct answer: b)

Python transcript:
```python
>>> import scipy.optimize as spo
>>> def f(x):
...     n = sum(x**2)
...     return n
...
>>> guess = [0.5, 0.1, 1.2, 0.2]
>>> const = ({ 'type': 'eq', 'fun': lambda x: 2 - sum(x)})
>>> result = spo.minimize(f, guess, method = 'SLSQP', constraints=const)
>>> print result.x
[ 0.5  0.5  0.5  0.5]
```

What is the output of this python code?

Code:
```python
import numpy as np
m = np.matrix([[1,1,0,1],
               [0,0,1,0],
               [1,1,0,1],
               [0,1,1,0]])

print m[1:3,-1] + m[-3:-1,1]
```

Select one answer:
a) [[1 2]]

b) [[1 2 0]]

c) [[0]
    [2]]

d) [[0]
    [2]

```
    [1]]

Correct Answer: c)

Python transcript:

>>> import numpy as np
>>> m = np.matrix([[1,1,0,1],
                   [0,0,1,0],
                   [1,1,0,1],
                   [0,1,1,0]])
>>> print m[1:3,-1] + m[-3:-1,1]
[[0]
 [2]]
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import pandas as pd
a = pd.DataFrame([1,2,3], columns=['First'])
print a
b = pd.DataFrame([1,2,3,4], columns=['Second'])
print b
print __A__
```

Output:

```
    First
0       1
1       2
2       3

    Second
0        1
1        2
2        3
3        4

    First  Second
0       1       1
1       2       2
2       3       3
3     NaN       4
```

Select one answer:
a) a.join(b)
b) a.join(b, how='inner')
c) a.join(b, how='outer')
d) pd.concat([a,b])

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> a = pd.DataFrame([1,2,3], columns=['First'])
>>> print a
    First
```

```
0      1
1      2
2      3
>>> b = pd.DataFrame([1,2,3,4], columns=['Second'])
>>> print b
   Second
0      1
1      2
2      3
3      4
>>> print a.join(b, how='outer')
   First  Second
0      1       1
1      2       2
2      3       3
3    NaN       4
```

```
Question
--------
```

What is the output of the following script:

Code:

```
import pandas as pd

df = pd.DataFrame(data={'price':[1,2,3,4,5]}, index=pd.date_range('2010-01-01','2010-
01-05'))
pd.rolling_mean(df, window=2)
```

Select one answer:
a)
```
              price
    2010-01-01   1.0
    2010-01-02   1.5
    2010-01-03   3.0
    2010-01-04   4.5
    2010-01-05   6.0
```
a)
```
       mean
    1    1.5
    2    2.5
    3    3.5
    4    4.5
```
c)
```
    ValueError: min_periods must be >= 0
```
d)
```
              price
    2010-01-01   NaN
    2010-01-02   1.5
    2010-01-03   2.5
    2010-01-04   3.5
    2010-01-05   4.5
```

Correct Answer: d)

Python transcript:

```
>>> import pandas as pd
>>> df = pd.DataFrame(data={'price':[1,2,3,4,5]}, index=pd.date_range('2010-01-
01','2010-01-05'))
>>> pd.rolling_mean(df, window=2)
                price
    2010-01-01    NaN
    2010-01-02    1.5
    2010-01-03    2.5
    2010-01-04    3.5
    2010-01-05    4.5
```

How should _A_ be filled to find the index of the column with the largest value

Code:

```
import numpy as np

x = np.array([[1,9,1],[1,1,1],[1,1,1]])
print x
print _A_
```

Output:

```
[[1 9 1]
 [1 1 1]
 [1 1 1]]
1
```

Select one answer:

```
a) x.argmax(axis=1).max()
b) x.argmax(axis=1)
c) x[x.max(axis=1)].ix
d) np.maximum(x[0],x[1]).max()
```

Correct answer: a)

Python transcript:

```
>>>import numpy as np
>>>x = np.array([[1,9,1],[1,1,1],[1,1,1]])
>>>print x
[[1 9 1]
 [1 1 1]
 [1 1 1]]
>>>print x.argmax(axis=1).max()
1
```

What is the output of the following code?

Code:

```
import numpy as np
arr = np.arange(10)
arr_slice = arr[3:5]
arr_slice[:] = 4
temp_number = 1
arr_slice = temp_number
```

```
print arr
```

Select one answer:

a) [1 2 3 4 4 6 7 8 9 10]
b) [0 1 2 4 4 5 6 7 8 9]
c) [1 2 3 1 6 7 8 9 10]
d) [0 1 2 3 4 5 6 7 8 9]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> arr = np.arange(10)
>>> arr_slice = arr[3:5]
>>> arr_slice[:] = 4
>>> temp_number = 1
>>> arr_slice = temp_number
>>> print arr
[0 1 2 4 4 5 6 7 8 9]
```

What is the output of the following code?

```
import numpy as np
import pandas as pd

a = np.array([[ 1, 2, 3],
              [ 4, 5, 6],
              [ 7, 8, 9]])
df = pd.DataFrame(a)
df.columns = [['N1', 'N2', 'N3']]
print(df['N2'][1:3].sum())
```

Select one answer:
a) 11
b) 13
c) 15
d) 9

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> a = np.array([[ 1, 2, 3],
...               [ 4, 5, 6],
...               [ 7, 8, 9]])
>>> df = pd.DataFrame(a)
>>> df.columns = [['N1', 'N2', 'N3']]
>>> print(df['N2'][1:3].sum())
13
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
pts = [{'x': 0, 'y': 0, 'z': 1},
       {'x': 2, 'y': 0, 'z': 1},
```

```
                 {'x': 3, 'y': 3, 'z': 1}]
df = pd.DataFrame(pts)
print df
 _A_
print df

Output:

    x  y  z
0   0  0  1
1   2  0  1
2   3  3  1
    x  y  z
0   0  0  1
1   4  0  1
2   6  6  1
```

```
Select one answer:
a) df = df*3
b) df.ix[:, 0:1] = df.ix[:, 0:1]*2
c) df.ix[0:2, 0:3] = df.ix[:, 0:2]*2
d) df.ix[:, 0:2] = df.ix[:, 0:2]*2
```

```
Correct answer: d)

Python transcript:

>>> import pandas as pd

>>> pts = [{'x': 0, 'y': 0, 'z': 1},
           {'x': 2, 'y': 0, 'z': 1},
           {'x': 3, 'y': 3, 'z': 1}]
>>> df = pd.DataFrame(pts)
>>> print df
    x  y  z
0   0  0  1
1   2  0  1
2   3  3  1
>>> df.ix[:, 0:2] = df.ix[:, 0:2]*2
>>> print df
    x  y  z
0   0  0  1
1   4  0  1
2   6  6  1
```

```
Fill in the blank for "y" to cause the following output:

Code:
import numpy as np

x = np.random.random([2,5])
print x
y = ???
print y


Output:
```

```
[[ 0.05736545  0.66388265  0.90395058  0.9522113   0.92785198]
 [ 0.77764742  0.25293629  0.27912528  0.98815477  0.10810053]]
[[ 0.66388265  0.9522113 ]
 [ 0.25293629  0.98815477]]
```

 Select one answer:

 a) y = x[:, 1:4:2]
 b) y = x[1:4:2, :]
 c) y = x[0:1, 1:2:4]
 d) y = x[1:2:4, 0:1]

 Correct answer: A

 Python transcript:
```
 >>> import numpy as np
>>> x = np.random.random([2,5])
>>> print x
[[ 0.05736545  0.66388265  0.90395058  0.9522113   0.92785198]
 [ 0.77764742  0.25293629  0.27912528  0.98815477  0.10810053]]
>>> y = x[:,1:4:2]
>>> print y
[[ 0.66388265  0.9522113 ]
 [ 0.25293629  0.98815477]]
>>>
```


What output does the following code snippet produce?

```
def my_func(x, y=-1):
    return x ** 2 + y

print my_func(3) + my_func(6, 5)
```

A. 25
B. 45
C. 50
D. 49

Correct answer: D

Python transcript
Running the follow line proves the answer is D, 49

```
# validation
print "my_func(3) + my_func(6,5) = {} + {} = {}".format(my_func(3), my_func(6,5),
my_func(3) + my_func(6, 5))

output: my_func(3) + my_func(6,5) = 8 + 41 = 49
```


Code:

```
import numpy as np
a = np.arange(0,9).reshape((3,3))
print a
print __A__
```

```
Output:

[[0 1 2]
 [3 4 5]
 [6 7 8]]
[0]
```

Select one answer:
a) a[0]
b) a[a>1]
c) a[-1]
d) a[a==0]

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> a = np.arange(0,9).reshape((3,3))
>>> print a
[[0 1 2]
 [3 4 5]
 [6 7 8]]
>>> print a[a==0]
[0]
```

What is the output of this python code?

```
code:
import numpy as np

xxarray = np.array([[3,4,5],[6,7,8]])
print xxarray[::-1,::-1]
```

Select one answer:
a)
8

b)
[[5 4 3]
 [8 7 6]]

c)
[[8 7 6]
 [5 4 3]]

d)
[8 7 6]

correct answer: c)

python transcript:

```
>>> import numpy as np

>>> xxarray = np.array([[3,4,5],[6,7,8]])
>>> print xxarray
>>> print xxarray[::-1,::-1]

[[8 7 6]
 [5 4 3]]
```

```
Slicing question

What is the output of this python code?

array = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])
array = array[:3, :3]
array = array[1:, 1:]
print(array)

Options
a)

[[ 6,  7,  8]
 [10, 11, 12]
 [14, 15, 16]]

b)

[[ 11 ]]

c)

[[ 6,  7]
 [10, 11]]

d)

[[ 0,  0,  0,  4]
 [ 0,  0,  0,  0]
 [ 0,  0,  0,  0]
 [13,  0,  0,  0]]


Correct Answer:
answer: c

Python Transcript

-> np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]])

[[ 1,  2,  3,  4]
 [ 5,  6,  7,  8]
 [ 9, 10, 11, 12]
 [13, 14, 15, 16]]

-> array[:3, :3]

[[ 1,  2,  3]
 [ 5,  6,  7]
 [ 9, 10, 11]]
```

```
-> array[1:, 1:]

[[ 6,  7]
 [10, 11]]
```

What is the output of this python code?

Code:

```
import numpy as np

A = np.arange(0, 100, 10)
B = A[[2, 4, 6]]
C = - A[[2]]
res = B - C
print res
```

Select one answer:
a) [0 2 4]
b) [4 6 8]
c) [40 60 80]
d) [20 40 60]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> A = np.arange(0, 100, 10)
>>> B = A[[2, 4, 6]]
>>> C = - A[[2]]
>>> res = B - C
>>> print A
[ 0 10 20 30 40 50 60 70 80 90]
>>> print B
[20 40 60]
```

```
>>> print C
[-20]
>>> print res
[40 60 80]
```

What is the output of following code?

Code:

```
import numpy as np
alphabets = np.array(['A', 'B', 'C'])
numbers = np.array([[1, 2, 3],[4,5,6],[7,8,9]])
print numbers[alphabets == 'B', 2:]
```

Select one answer:

```
a) [['B']]
b) [4,5,6]
c) [7,8,9]
d) [[6]]
```

Correct Answer: d)

Python transcript:

```
>>> import numpy as np
>>> alphabets = np.array(['A', 'B', 'C'])
>>> numbers = np.array([[1, 2, 3],[4,5,6],[7,8,9]])
>>> print numbers[alphabets == 'B', 2:]
[[6]]
>>>
```

Was this output created by A, B, C, or D?

```
    0
1  2
2  4
```

```
# A
print [[' ', 0], [1, 2], [2, 4]]
```

```
# B
import pandas as pd
print pd.DataFrame([2, 4], columns=[0])
```

```
# C
import pandas as pd
print pd.DataFrame([[1, 2], [2, 4]], columns=[' ', 0])

# D
import pandas as pd
print pd.DataFrame([2, 4], index=(1, 2))

Correct answer: D

Python transcript:

>>> # D is the correct answer
... import pandas as pd
>>> print pd.DataFrame([2, 4], index=(1, 2))
   0
1  2
2  4
>>>
>>> # other answers give incorrect results
... #
... # A
... print [[' ', 0], [1, 2], [2, 4]]
[[' ', 0], [1, 2], [2, 4]]
>>> #
... # B
... print pd.DataFrame([2, 4], columns=[0])
   0
0  2
1  4
>>> #
... # C
... print pd.DataFrame([[1, 2], [2, 4]], columns=[' ', 0])
     0
0  1  2
1  2  4
>>>
```

If you want to generate a list that is the length of another list where each entry
will have the same value and each value will sum up to one, which of the following
will accomplish that.


```
old_l_length = 5
print new_list
```

Output:

```
[0.2,0.2,0.2,0.2,0.2]
```

Select one answer:
a) new_list = old_l_length*[1]
b) new_list = [(1/h) for h in range(0,1)]
c) new_list = old_l_length*[old_l_length]
d) new_list = old_l_length*[1.0/old_l_length]

Correct answer: d)

Python transcript:

```
>>> old_l_length=5
>>> new_list=old_l_length*[1.0/old_l_length]
>>> print new_list
[0.2, 0.2, 0.2, 0.2, 0.2]
```

Given following allocations:

```
allocs = [0.1,0.2,0.3,0.4]
```

Which of the following code will reverse the allocations?

a) allocs = allocs[::-1]
b) allocs = allocs * -1
c) allocs = allocs[-1::]
d) allocs = allocs[:-1:]

correct answer: a)

python transcript:

```
allocs = [0.1,0.2,0.3,0.4]
```

```
print allocs

allocs = allocs[::-1]

print allocs

/Users/Himanshu/anaconda/bin/python
/Users/Himanshu/Documents/GT/ml4t/share/mc1_p1/question.py
[0.1, 0.2, 0.3, 0.4]
[0.4, 0.3, 0.2, 0.1]
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
print _A_
```

Output:

```
((0.0, 1.0), (0.0, 1.0), (0.0, 1.0))
```

Select one answer:

a) ((0., 1.),)*3
b) [(0., 1.) for i in [0,0,0]]
c) (0., 1.)*3
d) [(0., 1.) for i in 3]

Correct answer: a)

Python transcript:

```
Python 2.7.11 (default, Dec 26 2015, 17:47:53)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print ((0., 1.),)*3
((0.0, 1.0), (0.0, 1.0), (0.0, 1.0))
>>> print [(0., 1.) for i in [0,0,0]]
[(0.0, 1.0), (0.0, 1.0), (0.0, 1.0)]
>>> print (0., 1.)*3
(0.0, 1.0, 0.0, 1.0, 0.0, 1.0)
>>> print [(0., 1.) for i in 3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

Submit the following code in Python 2.7

```
series1 = [1,2,3,4,5,6,7,8,9]
print (series1[1:2],series1[5:7]*2)
```

What is the output of this python code?

```
a).  ([2], [12, 14])
b).  ([2], [6, 7, 6, 7])
c).  ([2, 3], [6, 7, 8, 6, 7, 8])
d).  ([2, 3], [12, 14, 16])
```

Correct answer: b)

```
Python transcript:
>>> series1 = [1,2,3,4,5,6,7,8,9]
>>> print (series1[1:2],series1[5:7]*2)
([2], [6, 7, 6, 7])
```

What is the output of this python code?

Code:

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
a[0:3,2]
```

Select one answer:

```
a) 24
b) array([7, 8, 9])
c) IndexError: index 3 is out of bounds for axis 0 with size 3
d) array([3, 6, 9])
```

Answer:
d)

Python Transcript

```
>>>import numpy as np
>>>a = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>>a
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
>>>a[0:3,2]
array([3, 6, 9])
```

Question:

What is the output of the following python code?

```
import numpy as np
arr = np.ones([5,5])
arr[:] = 2
arr[2:] = 3
arr[:, -1] = 4
print arr
```

Choices:

A)

```
[[ 2.  2.  2.  2.  4.]
 [ 2.  2.  2.  2.  4.]
 [ 3.  3.  3.  3.  4.]
 [ 3.  3.  3.  3.  4.]
 [ 3.  3.  3.  3.  4.]]
```

B)

```
[[ 3.  3.  3.  3.  4.]
 [ 3.  3.  3.  3.  4.]
 [ 2.  2.  2.  2.  4.]
 [ 2.  2.  2.  2.  4.]
 [ 2.  2.  2.  2.  4.]]
```

C)
```
[[ 2.  2.  2.  2.  2.]
 [ 2.  2.  2.  2.  2.]
 [ 3.  3.  3.  3.  3.]
 [ 3.  3.  3.  3.  3.]
 [ 4.  4.  4.  4.  4.]]
```

D)
```
[[ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
 [ 1.  1.  1.  1.  1.]
```

```
   [ 1.  1.  1.  1.  1.]]
```

Correct answer: A

Python transcript:

```
>>> import numpy as np
>>> arr = np.ones([5,5])
>>> arr[:] = 2
>>> arr[2:] = 3
>>> arr[-1, :] = 4
>>> print arr
[[ 2.  2.  2.  2.  2.]
 [ 2.  2.  2.  2.  2.]
 [ 3.  3.  3.  3.  3.]
 [ 3.  3.  3.  3.  3.]
 [ 4.  4.  4.  4.  4.]]
```

What is the output of this python code?

```
import numpy as np

alana_array = np.array([[22, 7, 8], [45, 1, 1], [123, 7, 45], [20, 0, 14]])
print alana_array.shape[1]
```

A) 3
B) 4x3
C) 3x4
D) 4

Correct answer: A)

Python transcript:

```
>>> import numpy as np
>>> alana_array = np.array([[22, 7, 8], [45, 1, 1], [123, 7, 45], [20, 0, 14]])
>>> print alana_array.shape[1]
3
```

```
Which of the following answers is the correct output?

import pandas as pd
import numpy as np
df = pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10],[2,3,4,5,6],[7,8,9,10,11],[0,1,0,1,0]])
a = np.arange(4,0,-2)
df1 = df.ix[-3:,a]
print df1

Options
a)     4   2
   2   6   4
   3  11   9
   4   0   0

b) Empty DataFrame
Columns: []
Index: [0, 1]

c)     4   2
   0   5   3
   1  10   8
   2   6   4
   3  11   9
   4   0   0

d)    2   3   4
   4   0   1   0
   2   4   5   6

correct answer: c)

Python transcript:

>>import pandas as pd
>>import numpy as np
>>df = pd.DataFrame([[1,2,3,4,5],[6,7,8,9,10],[2,3,4,5,6],[7,8,9,10,11],[0,1,0,1,0]])
>>a = np.arange(4,0,-2)
>>df1 = df.ix[-3:,a]
>>print df1

output:

       4   2
0      5   3
1     10   8
```

```
2    6    4
3   11    9
4    0    0
```

What is the output of the following code?

Code:

```python
import numpy as np
a=np.array([[0, 1, 0],[2, 0, 3],[0, 4, 0]])
print a.max(axis=0).min()
```

Select one answer:
a) 1
b) 2
c) 3
d) 4

Correct answer: b)

Python transcript:
```
>>> import numpy as np
>>> a=np.array([[0,1,0],[2,0,3],[0,4,0]])
>>> print a.max(axis=0).min()
2
```

What is the output of the following code:

```
import numpy as np
a = np.array([(12,20,8,15,6,14),(18,22,0,8,19,15)])
a[a>15] = 15
print a
```

Select one answer:
a) [[12 20 8 15 6 14]
    [18 22 0 8 19 15]]
b) [[15 15 15 15 15 15]
    [15 15 15 15 15 15]]
c) [[12 15 8 15 6 14]
    [15 15 0 8 15 15]]
d) [[12 20 8 15 6 14]
    [18 22 0 8 19 15]]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([(12,20,8,15,6,14),(18,22,0,8,19,15)])
>>> a[a>15] = 15
>>> print a
[[12 15  8 15  6 14]
 [15 15  0  8 15 15]]
```

If you needed to compute the daily returns of a given dataframe, and subsequently set the first rows for each column/stock to 0.0, which of the following code snippets should you use?

NOTE: "df" is a single-column dataframe containing the properly calculated portfolio values for each day. The format for df looks like:

```
        2010-01-04        1000000.000000
        2010-01-05         994880.982851
        2010-01-06         995136.933709
        2010-01-07         993601.228654
        Freq: D, dtype: float 64
```

NOTE: there is extra emphasis on the term "should" - this indicates using the proper
code to ensure the initial dataframe does not change (i.e. there is only one answer
that is the optimal choice that ensures the original state of the passed in dataframe
is not changed)

```
import pandas as pd
import numpy as np

def get_daily_returns(df):
        <CODE SNIPPET HERE>


A)        dr = df.copy()
        dr[:1] = (df[1:] / df[-1:].values) - 1
        dr[0] = 0.0
        return dr

B)         dr[1:] = (df[1:] / df[:-1].values) - 1
        dr[0] = 0.0
        return dr

C)        dr = df.copy()
        dr[1:] = (df[1:] / df[:-1].values) - 1
        dr[0] = 0.0
        return dr

D)        dr = df.copy()
        dr[1:] = (df[1:] / df[-1:]) -1
        dr[1:] = 0.0
        return dr
```

Correct Answer: C

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> import dt as datetime
>>> dates = pd.date_range(dt.datetime(2010,1,4), dt.datetime(2010,1,7))
>>> port_val = pd.DataFrame(index=dates)
>>> daily_ret = compute_daily_returns(port_val)
>>> print daily_ret
```

```
Output:
2010-01-04          0.000000
2010-01-05         -0.005119
2010-01-06          0.000257
2010-01-07         -0.001543
```

THE QUESTION:

How should section A be filled in to complete code that will cause the following
output:

THE CODE:

```
import pandas as pd
numbers= pd.DataFrame([[8, 4, 1], [2, 5, 7], [6, 2, 3]])
numbers= numbers/__A__
print numbers
```


THE OUTPUT:
```
        0           1           2
0      1.00        1.00          1
1      0.25        1.25          7
2      0.75        0.50          3
```


SELECT ONE ANSWER:

a) numbers[0,:]
b) numbers[:,0]
c) numbers.ix[0,:]
d) numbers.ix[:,0]

CORRECT ANSWER: c)

THE TRANSCRIPT:

```
>>> import numpy as np
>>> numbers= pd.DataFrame([[8, 4, 1], [2, 5, 7], [6, 2, 3]])
>>> numbers= numbers/numbers.ix[0,:]
>>> print(numbers)
        0           1           2
0      1.00        1.00          1
1      0.25        1.25          7
2      0.75        0.50          3
```

The answers 'a' and 'b' give an error, and 'd' gives the incorrect output.

```
Given the following code:

import numpy as np

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


which line of code produces the following output?:

[ 4.  4.  4.]


a)  print (a*2).min(axis=0)
b)  print (a*2).max(axis=1)
c)  print (a*2).mean(axis=0)
d)  print (a*2).mean(axis=1)


Correct answer: d)

Python transcript:

import numpy as np

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

#print (a*2).min(axis=0)
#print (a*2).max(axis=1)
#print (a*2).mean(axis=0)
print (a*2).mean(axis=1)
```

A numpy 2d array contains this value [[5,3],[10,2]]. Please fill in section X to get the following output:

```
import numpy as np
a = np.array([[5,3],[10,2]])
print "Input:\n", a
print "\nOutput:\n", X
```

```
Input:
[[ 5  3]
 [10  2]]
```

```
Output:
[[15 13]
 [20 12]]
```

```
a) X = a + np.max(a[0,:])
b) X = a + np.max(a[:,1])
c) X = a + np.max(a[:,:])
d) X = a + np.min(a[:,:])
```

Correct answer: c

```
Python transcript:
>>> import numpy as np
>>> a = np.array([[5,3],[10,2]])
>>> print "Input:\n", a
Input:
[[ 5  3]
 [10  2]]
>>> print "\nOutput:\n", a + np.max(a[:,:])
```

```
Output:
[[15 13]
 [20 12]]
```

How should the TODO section be filled in to cause the following output:

Code:

```
import numpy as np
myArray = np.random.randint(0,10,size=(4,4))
print "Before..."
print myArray

print "After..."
#TODO - add code here
print myArray
```

Output:

```
Before...
[[3 6 6 2]
 [0 3 6 3]
 [7 0 8 3]
 [7 9 7 5]]
```

```
After...
[[3 6 6 2]
 [0 3 6 3]
 [7 0 8 3]
 [3 6 6 2]]
```

Select one answer:
a) myArray[0:1] = myArray[3:4]
b) myArray[0:1,] = myArray[-1:,]
c) myArray[-1:,] = myArray[0:1,]
d) myArray[-4:,] = myArray[0:4,]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> myArray = np.random.randint(0,10,size=(4,4))
>>> print "Before..."
Before...
>>> print myArray
[[3 6 6 2]
 [0 3 6 3]
 [7 0 8 3]
 [7 9 7 5]]
>>> print "After..."
After...
>>> myArray[-1:,] = myArray[0:1,]
>>> print myArray
[[3 6 6 2]
 [0 3 6 3]
 [7 0 8 3]
 [3 6 6 2]]
```

You want to compare the evolution of the value of each column. How should you normalize a dataframe by column so that the first row is all ones ?

```
Code :

import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.random([3,3]))
print _A_

Output :

          0         1         2
0  1.000000  1.000000   1.000000
1  31.028261  0.520715  11.288351
2   0.313378  0.938353  12.367858
```

Select one answer:

```
a) df/df[:].max()
b) df/df.ix[0]
c) df/df.ix[:,0]
d) df.ix[:,0]=[1,1,1]
```

Correct answer : b)

Python Transcript

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.random([3,3]))
>>> print df/df.ix[0]
          0         1         2
0  1.000000  1.000000  1.000000
1  0.285340  0.849128  1.299684
2  1.009725  0.327130  0.243831
```

What does the last line of the following code output:

Code:

```
import pandas as pd
from util import get_data

dates = pd.date_range('2010-08-23', '2010-08-26')
symbols = ['IBM', 'GOOG', 'GLD', 'XOM']
df = get_data(symbols, dates)

print df
print df[['GOOG', 'GLD', 'XOM']][1:3] # What does this line output?

Output:

              IBM     GOOG      GLD     XOM
2010-08-23  122.30  464.07  119.78  56.68
2010-08-24  120.78  451.39  120.36  56.15
2010-08-25  121.14  454.62  121.36  56.12
2010-08-26  118.73  450.98  120.96  55.71
#
#   WHAT ELSE HERE?
#
```

Select one answer:

```
a)                  GOOG      GLD     XOM
      2010-08-24    451.39   120.36   56.15
      2010-08-25    454.62   121.36   56.12


b)                    IBM     GOOG      GLD     XOM
      2010-08-24    120.78   451.39   120.36   56.15
      2010-08-25    121.14   454.62   121.36   56.12

c)                  GOOG      GLD     XOM
      2010-08-24    451.39   120.36   56.15
      2010-08-25    454.62   121.36   56.12
      2010-08-26    450.98   120.96   55.71

d)                  GOOG      GLD     XOM
      2010-08-24    451.39   120.36      NaN
      2010-08-25    454.62   121.36   56.12
```

Correct answer: a)

Python transcript:

```
>>> import pandas as pd
>>> from util import get_data
>>>
>>> dates = pd.date_range('2010-08-23', '2010-08-26')
>>> symbols = ['IBM', 'GOOG', 'GLD', 'XOM']
>>> df = get_data(symbols, dates, addSPY=False)
>>>
>>> print df
                 IBM     GOOG      GLD     XOM
2010-08-23    122.30   464.07   119.78   56.68
2010-08-24    120.78   451.39   120.36   56.15
2010-08-25    121.14   454.62   121.36   56.12
2010-08-26    118.73   450.98   120.96   55.71
>>> print df[['GOOG', 'GLD', 'XOM']][1:3]
                GOOG      GLD     XOM
      2010-08-24   451.39   120.36   56.15
      2010-08-25   454.62   121.36   56.12
```

What is the output of the following code executed in a Python console:

Code:

```
import numpy as np

def foo(x):
    return 1 + x ** 2

a = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
b = [0.2, 0.2, 0.2, 0.2, 0.2]
print(foo(a))
print(foo(b))
```

Select one answer:
a)
```
[ 1.04  1.04  1.04  1.04  1.04]
[ 1.04  1.04  1.04  1.04  1.04]
```

b)
```
[ 1.04  1.04  1.04  1.04  1.04]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in foo
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

c)
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in foo
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in foo
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

d)
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in foo
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
[ 1.04  1.04  1.04  1.04  1.04]
```


Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> def foo(x):
...     return 1 + x ** 2
...
>>> a = np.array([0.2, 0.2, 0.2, 0.2, 0.2])
>>> b = [0.2, 0.2, 0.2, 0.2, 0.2]
>>> print(foo(a))
[ 1.04  1.04  1.04  1.04  1.04]
>>> print(foo(b))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in foo
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
>>>
```

What code would you put in for _blank_ to produce the console output produced by this method?

```python
def normalize():
    prices = np.array([(5.19,8.5),(5.1,8.6),(5.15,8.7),(5.3,8.8),(5.25,8.9)])
    df = pd.DataFrame(prices, columns=['SYM1','SYM2'])
    print df / _blank_
```

```
Console Output:
        SYM1      SYM2
0  1.000000  1.000000
1  0.982659  1.011765
2  0.992293  1.023529
3  1.021195  1.035294
4  1.011561  1.047059
```

a) df.ix[0,:]
b) df.ix[head]
c) df.head()
d) df[0,:]

--Correct answer
a)

Proof:

```python
import pandas as pd
import numpy as np

def normalize():
    prices = np.array([(5.19,8.5),(5.1,8.6),(5.15,8.7),(5.3,8.8),(5.25,8.9)])
    df = pd.DataFrame(prices, columns=['SYM1','SYM2'])
    print df / df.ix[0,:]

if __name__ == "__main__":
    normalize()
```

```
Output
        SYM1      SYM2
0  1.000000  1.000000
1  0.982659  1.011765
2  0.992293  1.023529
3  1.021195  1.035294
4  1.011561  1.047059
```

What is the output of the following code?

Code:

```
import numpy as np
a = np.matrix([[0,1,2,3],
               [4,5,6,7],
               [8,9,10,11],
               [12,13,14,15]])
a = a[::-1,::-1]
print a[1:3,1:]
```

Select one answer:
a)
```
[[15 14 13]
 [11 10  9]]
```
b)
```
[[10  9  8]
 [ 6  5  4]]
```
c)
```
[[10  9]
 [ 6  5]
 [ 2  1]]
```
d)
```
[[ 5  6]
 [ 9 10]
 [13 14]]
```

Correct answer b)

Python transcript:

```
>>> import numpy as np
>>> a = np.matrix([[0,1,2,3],
```

```
...                          [4,5,6,7],
...                          [8,9,10,11],
...                          [12,13,14,15]])
>>> a = a[::-1,::-1]
>>> print a[1:3, 1:]
[[10  9  8]
 [ 6  5  4]]
```

What is the output of this python code?

Code:

```
import numpy as np
a = np.arange(6)
a = a.reshape(3,2)
print a.min(axis=1)
```

Select one answer:
a) 0
b) [0 1]
c) [0 2 4]
d) [1 2]

Correct answer: c)

Python transcript:
```
>>> import numpy as np
>>> a = np.arange(6)
>>> a = a.reshape(3,2)
>>> print a.min(axis=1)
[0 2 4]
```

How should section A be filled in to complete the code that will cause the following output:

Code:

```
import pandas as pd
if __name__ == "__main__":
  data = [
    [None,  0.247489, -0.965233],
```

```
      [-0.623457, -1.046121, -2.218067],
      [None,  1.359514,  None],
      [1.401580, -1.736888,  1.295827],
      [-0.145410,  2.473498,  1.214342]
    ]
  df = pd.DataFrame(data)
  df = df._A_
  print df
```

Output:

```
          0         1         2
0       NaN  0.247489 -0.965233
1 -0.623457 -1.046121 -2.218067
2 -0.623457  1.359514 -2.218067
3  1.401580 -1.736888  1.295827
4 -0.145410  2.473498  1.214342
```

Select one answer:

a) ffill()
b) dropna()
c) fillfwd()
d) nafill()

Python transcript:

```
import pandas as pd
if __name__ == "__main__":
  data = [
    [None,  0.247489, -0.965233],
    [-0.623457, -1.046121, -2.218067],
    [None,  1.359514,  None],
    [1.401580, -1.736888,  1.295827],
    [-0.145410,  2.473498,  1.214342]
  ]
  df = pd.DataFrame(data)
  df = df._A_
  print df
```

```
          0         1         2
0       NaN  0.247489 -0.965233
1 -0.623457 -1.046121 -2.218067
2 -0.623457  1.359514 -2.218067
3  1.401580 -1.736888  1.295827
4 -0.145410  2.473498  1.214342
```

```
Fill the blank in above to cause this Python code to give the following output.
You are given a Pandas Dataframe 'df', containing 10 stocks from 1st Jan 2009 to 31st
Dec 2009, and you are required to select
data of 'AAPL' and 'GLD' for last five days.
Output:
"
           AAPL  GLD
2009-12-27  904  884
2009-12-28   75  172
2009-12-29  381    4
2009-12-30  892   92
2009-12-31  417  169
"
Select one answer:

a) df.ix[-1:-5,['AAPL','GLD']]
b) df.ix[-5:-1, ['AAPL', 'GLD']]
c) df[-5:, ['AAPL', 'GLD']]
d) df.ix[-5:, ['AAPL', 'GLD']]

Correct Answer: d

Python Script:
-----------------------------------------------------------------------------------------
-------
"""MC1-Homework-3: Design a Midterm Question.
Question tests students knowledge of Pandas dataframe. It tests row-slicing and column
selection.

Data is selected for 10 stocks: ['GOOG', 'AAPL', 'XON', 'SPY', 'GLD', 'IBM', 'BUD',
'CBG', 'KIJ', 'LMN'] in a specific range[2009-2010]. And, question asks
the student to select APPL and GLD stocks for the last 5 records.

"""

import pandas as pd
import numpy as np

a = np.random.random_integers(1000, size=(365., 10.))
dates = pd.date_range('2009-1-1', periods=365, freq='D')
df = pd.DataFrame(a)
df.index = dates
df.columns = [['GOOG', 'AAPL', 'XON', 'SPY', 'GLD', 'IBM', 'BUD', 'CBG', 'KIJ',
'LMN']]

#First Choice
#print df.ix[:-5,['AAPL','GLD']]

#Second Choice
#print df.ix[-5:-1, ['AAPL', 'GLD']]

#Third Choice
#print df[-5:, ['AAPL', 'GLD']]

#Fourth Choice (correct choice)
#print df.ix[-5:, ['AAPL', 'GLD']]
```

```
What is the output of the following code?

import numpy as np
x= np.array([[1,2,3],[4,5,6], [7,8,9]])
y = np.amax(x, axis=1)
z = x/y
print z

(a) [[0, 0, 0],
     [1, 0, 0],
     [2, 1, 1]]
(b) [[ 0.33333333,  0.33333333,  0.33333333],
     [ 1.33333333,  0.83333333,  0.66666667],
     [ 2.33333333,  1.33333333,  1.          ]]
(c) [[0, 0, 0],
     [0, 0, 0],
     [1, 1, 1]]
(d) [[ 0.14285714,  0.25       ,  0.33333333],
     [ 0.57142857,  0.625      ,  0.66666667],
     [ 1.         ,  1.         ,  1.          ]]

Correct answer: (a)

Python transcript:

>>> import numpy as np
>>> x= np.array([[1,2,3],[4,5,6], [7,8,9]])
>>> y = np.amax(x, axis=1)
>>> z = x/y
>>> print z
[[0 0 0]
 [1 0 0]
 [2 1 1]]
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import pandas as pd
prices = pd.DataFrame([[185.35, 4.71, 113.83],[186.27, 4.83, 112.97],[184.22, 4.97,
112.33]],columns=['SPY','OIL', 'GLD'])
print prices
print _A_
```

Output:

```
      SPY    OIL      GLD
0  185.35  4.71   113.83
1  186.27  4.83   112.97
2  184.22  4.97   112.33
      SPY    OIL      GLD
1  185.35  4.71   113.83
2  186.27  4.83   112.97
```

Select one answer:
a) prices.shift(1)[0:]
b) prices.shift(1)[:]
c) prices.shift(1)[1:]
d) prices.shift(-1)[1:]

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> prices = pd.DataFrame([[185.35, 4.71, 113.83],[186.27, 4.83, 112.97],[184.22,
4.97, 112.33]],columns=['SPY','OIL', 'GLD'])
>>> print prices
      SPY    OIL      GLD
0  185.35  4.71   113.83
1  186.27  4.83   112.97
2  184.22  4.97   112.33
>>> print prices.shift(1)[1:]
      SPY    OIL      GLD
1  185.35  4.71   113.83
2  186.27  4.83   112.97
```

What is the output of the following code?

Code:

```
import numpy as np

a = np.array([[1,2,3],
              [3,4,5],
              [4,5,6]])
print a[:,1:]
```

Select one answer:
a) [[1,2,3]
    [3,4,5]]
b) [[3,4,5]

```
    [4,5,6]]
c) [[2 3]
    [4 5]
    [5 6]]
d) [[1,2]
    [3,4]
    [4,5]]
```

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[1,2,3],
                  [3,4,5],
                  [4,5,6]])
>>> print a[:,1:]
[[2 3]
[4 5]
[5 6]]
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import numpy as np
j = np.array([[2,3],[1,1]])
k = np.array([[3,3],[3,3]])
print j
print k
print _A_
```

Output:
```
[[2 3]
 [1 1]]
[[3 3]
 [3 3]]
[[15 15]
 [ 6  6]]
```

Select one answer:
a) np.dot(j, k)
b) np.multiply(j, k)
c) j * k
d) j / k

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> j = np.array([[2,3],[1,1]])
>>> k = np.array([[3,3],[3,3]])
>>> print j
[[2 3]
 [1 1]]
```

```
>>> print k
[[3 3]
 [3 3]]
>>> print np.dot(j, k)
[[15 15]
 [ 6  6]]
```

How should section A be filled in to complete code that will cause the following
output:

Code:

```
import numpy as np
a = np.array([[1,2,3,4,5],
              [11,12,13,14,15],
              [21,22,23,24,25],
              [31,32,33,34,35],
              [41,42,43,44,45]])
b = np.array([[51,52,53,54,55]])
print _A_
```

Output:

```
[[ 1  2  3  4  5 51]
 [11 12 13 14 15 52]
 [21 22 23 24 25 53]
 [31 32 33 34 35 54]
 [41 42 43 44 45 55]]
```

Select one answer:
a) np.concatenate((a, b), axis=1)
b) np.concatenate((a, b.T), axis=0)
c) np.concatenate((a, b.T), axis=1)
d) np.concatenate((a, b), axis=0)

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[1,2,3,4,5],
...               [11,12,13,14,15],
...               [21,22,23,24,25],
...               [31,32,33,34,35],
...               [41,42,43,44,45]])
>>> b = np.array([[51,52,53,54,55]])
>>> print np.concatenate((a, b.T), axis=1)
[[ 1  2  3  4  5 51]
 [11 12 13 14 15 52]
 [21 22 23 24 25 53]
 [31 32 33 34 35 54]
 [41 42 43 44 45 55]]
```

Given the following python code, what will the values of x and y be after execution:

```
import numpy as np
tmparr = np.array( [[ 1, 2, 3],
                    [ 2, 1, 4]])

x = np.sum(tmparr , axis=1)[0]
y = x / tmparr[-1,-1]
```

Select one answer:
a) x: 3   y: 4
b) x: 3   y: 0.75
c) x: 6   y: 1
d) x: 6   y: 1.5

Correct answer: c)

Python Transcript:

```
>>> import numpy as np
>>> tmparr = np.array( [[ 1, 2, 3], [ 2, 1, 4]])
>>> x =  np.sum(tmparr , axis=1)[0]
>>> y = x / tmparr[-1,-1]
>>> print "x=" ,  x , " y=" , y
x= 6   y= 1
```

```
import pandas as pd
import numpy as np

test_frame = pd.DataFrame(np.random.rand(8,4),
                          index=list('abdcefgh'),
                          columns=['st1','st2','st3','st4'])

print 'Test Frame'
print test_frame

print 'Mean for first 4 Rows of st2'
print __ANSWER__.mean()

print 'Standard Deviation for first 4 Rows of st2'
print __ANSWER__.std()

Output:

Test Frame
        st1        st2        st3        st4
a  0.338524  0.126643  0.776153  0.610379
b  0.262568  0.103882  0.995729  0.207025
c  0.067094  0.394764  0.970601  0.282487
d  0.690560  0.573392  0.288221  0.061491
e  0.762193  0.679864  0.241871  0.223460
f  0.318419  0.167459  0.411278  0.862037
g  0.821099  0.390488  0.772339  0.890881
h  0.905461  0.891842  0.181957  0.471498

Mean for first 4 rows of
0.20842977928

Standard Deviation for first 4 Rows of st2
0.161771239476

#Select one asnwer:
#a) df.ix[:3, 'st2']
#b) df['st1']
#c) df.iloc[:3, 'st2']
#d) df.ix[0]

#correct answer: a)

python transcript:

>>> import pandas as pd
>>> import numpy as np
>>> test_frame = pd.DataFrame(np.random.rand(8,4), index=list('abcdefgh'), columns=
['st1','st2','st3','st4'])
>>> print 'Test Frame'
Test Frame
>>> print test_frame
        st1        st2        st3        st4
a  0.338524  0.126643  0.776153  0.610379
b  0.262568  0.103882  0.995729  0.207025
c  0.067094  0.394764  0.970601  0.282487
d  0.690560  0.573392  0.288221  0.061491
e  0.762193  0.679864  0.241871  0.223460
f  0.318419  0.167459  0.411278  0.862037
g  0.821099  0.390488  0.772339  0.890881
```

```
h  0.905461  0.891842  0.181957  0.471498

>>> print test_frame.ix[:3, 'st2'].mean()
0.20842977928
>>> print test_frame.ix[:3, 'st2'].std()
0.161771239476
```

What is the output from the following python code?

Code:

```
import numpy as np
m = np.array([[4,9,16],[25,36,49],[64,81,100]])
m = m / np.sqrt(m)
print(m)
```

Select one answer:

a)
```
[[ 0.20412415  0.45927933  0.81649658]
 [ 1.27577591  1.83711731  2.50052078]
 [ 3.26598632  4.13351394  5.10310363]]
```

b)
```
[[  2.   3.   4.]
 [  5.   6.   7.]
 [  8.   9.  10.]]
```

c)
```
2.   3.   4.  5.   6.   7.  8.   9.  10.
```

d)
```
[[   16    81   256]
 [  625  1296  2401]
 [ 4096  6561 10000]]
```

Correct answer: b

Python transcript:

```
>>> import numpy as np
>>> m = np.array([[4,9,16],[25,36,49],[64,81,100]])
>>> m = m / np.sqrt(m)
>>> print(m)
```

```
[[  2.   3.   4.]
 [  5.   6.   7.]
 [  8.   9.  10.]]
```
How should section A be filled in to complete code that will cause the following output:

Code:

```
import numpy as np
j = np.array([[0., 1., 2.],[3., 4., 5.],[6., 7., 8.]])
print j
print j/j[1,:]
```

Output:

```
[[ 0.  1.  2.]
 [ 3.  4.  5.]
 [ 6.  7.  8.]]
[[ 0.    0.25  0.4 ]
 [ 1.    1.    1.  ]
 [ 2.    1.75  1.6 ]]
```

Select one answer:
a) j / j[0,0]
b) j / j[:,1]
c) j / j.sum(axis = 1)
d) j / j[1,:]

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> j = np.array([[0., 1., 2.],[3., 4., 5.],[6., 7., 8.]])
>>> print j
[[ 0.  1.  2.]
 [ 3.  4.  5.]
 [ 6.  7.  8.]]
>>> print j / j[1,:]
[[ 0.    0.25  0.4 ]
 [ 1.    1.    1.  ]
 [ 2.    1.75  1.6 ]]
```

What is the output of the following code?

```
import numpy as np
j = np.array([[4,1],[5,5],[1,4]], np.float64)
nm = j/j.sum(axis=0)
pt = nm*100
print pt
```

Select one answer:
a) [[80. 20.]
    [50. 50.]
    [20. 80.]]
b) [[40. 10.]
    [50. 50.]
    [10. 40.]]
c) [[20. 5.]
    [25. 25.]

```
    [ 5.   20.]]
d) [[ 4.   1.]
    [ 5.   5.]
    [ 1.   4.]]
```

Correct answer: b)

Python transcript:
```
>>> import numpy as np
>>> j = np.array([[4,1],[5,5],[1,4]], np.float64)
>>> nm = j/j.sum(axis=0)
>>> pt = nm*100
>>> print pt
[[ 40.  10.]
 [ 50.  50.]
 [ 10.  40.]]
```

We desire to achieve Naive Scalar multiplication by two (two times the input object
vector range)
Input range: v = [1, 2, 3, 4, 5]
Desired Output (twice the input): [2, 4, 6, 8, 10]
Which program results in correct answer:
A:
```
v = range(1, 6)
print 2 * v
```

B:
```
import numpy as np
v = np.arange(1, 6)
print 2 * v
```

Answers:
1: A
2: B
3: Both A & B
4: Neither A & B

Answer: B

Scripts:
A:
```
v = range(1, 6)
print 2*v
Out: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

B:
```
import numpy as np
v = np.arange(1, 6)
print 2 * v
Out: [2, 4, 6, 8, 10]
```

B is the correct answer

How should section A be filled in to complete the code so that the last "print df"
will generate the following output:

Code:

```
import numpy as np
import pandas as pd

rand = np.random.randint(5, size=(3,3))
df = pd.DataFrame(rand)

print "=== original dataframe ==="
print df

print "=== modified dataframe ==="
_____A_____
print df
```

Output:
```
===original dataframe===
   0  1  2
0  4  4  4
1  4  1  2
2  1  2  0
===modified dataframe===
   0   1  2
0  4  16  4
1  4   1  2
2  1   4  0
```

Select one answer:
a) df.ix[:,1]^2
b) df.ix[:,1]**2
c) df.ix[:,1] = df.ix[:,1]^2
d) df.ix[:,1] = df.ix[:,1]**2

Correct answer: d)

Python transcript:

```
>>> import pandas as pd
>>> import numpy as np
>>> rand = np.random.randint(5,size=(3,3))
>>> df = pd.DataFrame(rand)
>>> print df
   0  1  2
0  4  4  4
1  4  1  2
2  1  2  0
>>> df.ix[:,1] = df.ix[:,1]**2
>>> print df
   0   1  2
0  4  16  4
1  4   1  2
2  1   4  0
```

What is the output of this python code?

Code:

```
import numpy as np
x = np.array([[1,2,3],[3,4,5],[5,6,7]])
print x[1]+x[-1]
```

Output:

?

Select one answer:

a) [4 8 12]
b) [5 9 13]
c) [8 10 12]
d) [6 8 10]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> x = np.array([[1,2,3],[3,4,5],[5,6,7]])
>>> print x[1]+x[-1]
[ 8 10 12]
```

What is the output produced by
 the following code segment?

```
import pandas as pd

s1 = pd.Series((2, 5, 7))
s2 = pd.Series((3, 8, 2))
s3 = pd.Series((5, 9, 4))
s4 = pd.Series((4, 7, 3))
df = pd.DataFrame([s1 + s2 + s3 + s4])
```

```
print df
```

a)
```
     0
0   14
1   13
2   18
3   14
```

b)
```
     0   1   2
0   14  29  16
```

c)
```
     0
0   59
```

d)
```
 [[2, 5, 7],
  [3, 8, 2],
  [5, 9, 4],
  [4, 7, 3]]
```


correct answer b)

Python transcript:

```
>>> import pandas as pd
>>> s1 = pd.Series((2, 5, 7))
>>> s2 = pd.Series((3, 8, 2))
>>> s3 = pd.Series((5, 9, 4))
>>> s4 = pd.Series((4, 7, 3))
>>> df = pd.DataFrame([s1 + s2 + s3 + s4])
>>> print df
    0   1   2
0  14  29  16
```

Choose the correct line of code to print the output shown below:

Code:

```
import numpy as np
import pandas as pd

df1 = pd.DataFrame(np.random.randn(10,3))
print df1
[ chosen code goes here ]
```

Output:

```
          0         1         2
0  1.629342 -1.081380 -0.790401
1  2.292899 -0.628032 -0.101210
2  0.562970 -1.014486  0.734165
3  0.912935  1.483613 -0.035802
4  1.456115  1.320073  1.736249
5 -1.250448  0.423665  1.689530
6  1.015360  1.011011  0.181923
7  0.396016 -1.921691  0.207542
8 -1.221396 -0.433596  0.806824
9  1.114499  0.457012  0.433003
7     0.396016
```

```
8   -1.221396
9    1.114499
Name: 0, dtype: float64
```

Select one answer:
a) print df1[3].tail(0)
b) print df1[0].tail(3)
c) print df1[:3]
d) print df1[3:]

Correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>>
>>> df1 = pd.DataFrame(np.random.randn(10,3))
>>> print df1
           0         1         2
0 -1.250064 -0.867570  0.318090
1  0.023197  1.131202  1.225183
2  0.827925 -0.036734  0.735396
3 -0.830061  0.753152  0.441009
4 -0.425413 -1.072667  0.140984
5  0.419763 -0.671618 -1.149441
6 -0.995779  0.875700 -1.010246
7  2.486033  1.364540  0.650869
8 -1.796426 -1.077509 -0.746713
9 -0.584872 -0.083137 -1.001605
>>> print df1[0].tail(3)
7    2.486033
8   -1.796426
9   -0.584872
Name: 0, dtype: float64
```

The numpy array below contains closing prices for six securities over a ten day period.
What is the output of this python code?

Code:
```
import numpy as np
                   # SPY      IBM     AAPL     HNZ      XOM      GLD
prices = np.array([[ 86.8 ,  81.64,  90.36,  33.95,  74.48,  86.23],
                   [ 86.7 ,  81.13,  94.18,  33.82,  74.47,  84.48],
                   [ 87.28,  83.38,  92.62,  33.38,  73.26,  85.13],
                   [ 84.67,  82.03,  90.62,  32.59,  71.39,  82.75],
                   [ 85.01,  81.46,  92.3 ,  31.99,  72.15,  84.46],
                   [ 83.19,  79.15,  90.19,  31.69,  70.77,  83.92],
                   [ 81.19,  80.09,  88.28,  31.49,  69.83,  80.76],
                   [ 81.34,  79.74,  87.34,  31.75,  71.09,  80.88],
                   [ 78.78,  77.74,  84.97,  30.65,  68.51,  79.79],
                   [ 78.81,  78.6 ,  83.02,  30.67,  69.94,  80.39]])


print prices[:,-1]  ### what is the output of this???
```

Select one answer:

a) A list of closing prices of all six securities on the 10th day
b) 80.39 (The closing price of GLD on the 10th day)
c) A list of the closing price of GLD for all ten days
d) It does not actually output any prices; it actually generates an IndexError
exception

Correct answer: c)

Python transcript:
```
>>> import numpy as np
>>>                      # SPY      IBM      AAPL     HNZ      XOM      GLD
... prices = np.array([[ 86.8 ,   81.64,   90.36,   33.95,   74.48,   86.23],
...                    [ 86.7 ,   81.13,   94.18,   33.82,   74.47,   84.48],
...                    [ 87.28,   83.38,   92.62,   33.38,   73.26,   85.13],
...                    [ 84.67,   82.03,   90.62,   32.59,   71.39,   82.75],
...                    [ 85.01,   81.46,   92.3 ,   31.99,   72.15,   84.46],
...                    [ 83.19,   79.15,   90.19,   31.69,   70.77,   83.92],
...                    [ 81.19,   80.09,   88.28,   31.49,   69.83,   80.76],
...                    [ 81.34,   79.74,   87.34,   31.75,   71.09,   80.88],
...                    [ 78.78,   77.74,   84.97,   30.65,   68.51,   79.79],
...                    [ 78.81,   78.6 ,   83.02,   30.67,   69.94,   80.39]])
>>>
>>>
>>> print prices[:,-1]   ### what is the output of this???
[ 86.23   84.48   85.13   82.75   84.46   83.92   80.76   80.88   79.79   80.39]
```

How should section A be filled in to complete code that will cause the following
output:

Code:

```
import numpy as np
a = np.arange(16).reshape(4,4)
print a
print _A_
```

Output:

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
[24 28 32 36]
```

Select one answer:
a) a.sum(axis=0)
b) a[3,:]
c) a.sum(axis=1)
d) a[:,3]

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> a = np.arange(16).reshape(4,4)
>>> print a
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
>>> print a.sum(axis=0)
```

```
[24 28 32 36]
```

WORKING CODE:

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.random_integers(1,50,size=(10, 3)), columns=['col1',
'col2', 'col3'])
print df
x = df[6:10]
y = x/x
dframes = [df[0:6], y]
df1 = pd.concat(dframes)
print df1
```

QUESTION CODE:

```
import numpy as np
import pandas as pd

df = pd.DataFrame(np.random.random_integers(1,50,size=(10, 3)), columns=['col1',
'col2', 'col3'])
print df
x = df[6:10]
y = x/x
__A__
df1 = pd.concat(dframes)
print df1
```

```
   col1  col2  col3
0    19    12    37
1    11    21     8
2    41    36    10
3    27    37    29
4    32    34     6
5    31    48    45
6    39    48    28
7    30    14    27
8     9    24    44
9    32    14     1
```

What code in __A__ will produce the output

```
   col1  col2  col3
0    19    12    37
1    11    21     8
2    41    36    10
3    27    37    29
4    32    34     6
5    31    48    45
6     1     1     1
7     1     1     1
8     1     1     1
9     1     1     1
```

```
A. dframes = [df[1:6], y]
B. dframes = [df[1:7], y]
C. dframes = [df[0:7], y]
D. dframes = [df[0:6], y]
```

Python Transcript:

```
ml4t@ml4t-VirtualBox:~$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.random_integers(1,50,size=(10, 3)), columns=['col1',
'col2', 'col3'])
>>> print df
   col1  col2  col3
0    19    12    37
1    11    21     8
2    41    36    10
3    27    37    29
4    32    34     6
5    31    48    45
6    39    48    28
7    30    14    27
8     9    24    44
9    32    14     1
>>> x = df[6:10]
>>> y = x/x
>>> dframes = [df[0:6], y]
>>> df1 = pd.concat(dframes)
>>> print df1
   col1  col2  col3
0    19    12    37
1    11    21     8
2    41    36    10
3    27    37    29
4    32    34     6
5    31    48    45
6     1     1     1
7     1     1     1
8     1     1     1
9     1     1     1
>>>


# This question accomplished two goals
# 1) If the student has completed the projects, he/she should know how to
#    properly access the indexed pandas dataframe using the .ix syntax
# 2) If the student has watched the lecture, he/she should be able to figure
#    out the difference between inner and outer joins
#
# The question is unambiguous, has only one correct answer, and have 3 attractive
alternatives
#




Given

>>> import pandas as pd

>>> project1_scores = pd.DataFrame([92.0, 87.0, 95.0],
                                   index=['Janet', 'Ariel', 'Laurel'],
                                   columns=['Project 1'])
```

```
>>> project2_scores = pd.DataFrame([89.0, 98.0, 90.0],
                                   index=['Ariel', 'Laurel', 'Boyd'],
                                   columns=['Project 2'])
```

How should section X and Y be filled in to complete code that will cause the following
output:

```
>>> combined_scores = project1_scores.join(project2_scores, how= X )

>>> print combined_scores

        Project 1  Project 2
Ariel          87         89
Boyd          NaN         90
Janet          92        NaN
Laurel         95         98

>>> ariels_score = Y

>>> print ariels_score

Project 1    87
Project 2    89
Name: Ariel, dtype: float64
```

Select one answer:
a) X 'inner', Y combined_scores.ix['Ariel']
b) X 'inner', Y combined_scores[0]
c) X 'outer', Y combined_scores.ix['Ariel']
d) X 'outer', Y combined_scores[0]

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> project1_scores = pd.DataFrame([92.0, 87.0, 95.0], index=['Janet', 'Ariel',
'Laurel'], columns=['Project 1'])
>>> project2_scores = pd.DataFrame([89.0, 98.0, 90.0], index=['Ariel', 'Laurel',
'Boyd'], columns=['Project 2'])
>>> combined_scores = project1_scores.join(project2_scores, how='outer')
>>> print combined_scores
        Project 1  Project 2
Ariel          87         89
Boyd          NaN         90
Janet          92        NaN
Laurel         95         98
>>> ariels_score = combined_scores.ix['Ariel']
>>> print ariels_score
Project 1    87
Project 2    89
Name: Ariel, dtype: float64
```

Python source code:

```
import pandas as pd
project1_scores = pd.DataFrame([92.0, 87.0, 95.0], index=['Janet', 'Ariel', 'Laurel'],
columns=['Project 1'])
```

```
project2_scores = pd.DataFrame([89.0, 98.0, 90.0], index=['Ariel', 'Laurel', 'Boyd'],
columns=['Project 2'])
combined_scores = project1_scores.join(project2_scores, how='outer')
print combined_scores
ariels_score = combined_scores.ix['Ariel']
print ariels_score
```

Type 1:

What is the output of the following python code?

Code:

```
import numpy as np
import pandas as pd
array = np.array([[1,2,3,4,5],[6,7,8,9,0],[0,9,8,7,6],[5,4,3,2,1]])
df=pd.DataFrame(array,columns=['C1', 'C2', 'C3', 'C4', 'C5'])
print df.ix[1:3,['C2','C4']]
```

a) 7   9
   9   2

b)   C2 C4
    1   7   9
    2   9   7
    3   4   2

c)   C2 C4
        9   7
        7   9
        2   4

d)   C2 C4
    1   7   9
    2   9   7

Correct Answer: b)

Python Transcript:

```
>>>import numpy as np
>>>import pandas as pd
>>>array = np.array([[1,2,3,4,5],[6,7,8,9,0],[0,9,8,7,6],[5,4,3,2,1]])
>>>df=pd.DataFrame(array,columns=['C1', 'C2', 'C3', 'C4', 'C5'])
>>>print df.ix[1:3,['C2','C4']]
```

How should you transform numpy array A into numpy array B, which basically subtract
the mean of each row of a matrix. Please use the numpy built-in operation.

```
A = [[  1.   2.   3.]
     [  0.  10.  20.]
     [  3.   4.   5.]]


B = [[ -1.,   0.,   1.],
     [-10.,   0.,  10.],
     [ -1.,   0.,   1.]]
```

```
So B = ?
Select one answer:

a)A - A.mean(axis = 1)
b)A - A.mean(axis = 1, keepdims = True)
c)A - A.mean(axis = 0)
d)A - A.mean(axis = 0, keepdims = True)

Correct answer: b)

Python transcript:

>>> import numpy as np
>>> list_A = [[1.0,2.0,3.0],[0.0,10.0,20.0],[3.0,4.0,5.0]]
>>> A = np.array(list_A)
>>> print A
[[  1.   2.   3.]
 [  0.  10.  20.]
 [  3.   4.   5.]]
>>> B = A - A.mean(axis = 1, keepdims = True)
>>> print B
[[ -1.   0.   1.]
 [-10.   0.  10.]
 [ -1.   0.   1.]]
```

```python
def pandas_only_daterange(df, bd, ed):
    '''
    INPUT: DataFrame, High, Low, Close prices
    OUTPUT: DataFrame

    Return a new pandas DataFrame which contains the entries for the provided date
range
    '''
                (a) df.ix([bd:ed])
                    (b) df[bd:ed]
                    (c) df.iloc[bd:ed]
                    (d) df.index([bd:ed])
```

Correct Answer (b)

```python
def only_positive(arr):
    '''
    INPUT: 2 DIMENSIONAL NUMPY ARRAY
    OUTPUT: 2 DIMENSIONAL NUMPY ARRAY

    Return a numpy array containing only the rows from arr where all the values
    are positive.

    E.g.  [[1, -1, 2], [3, 4, 2], [-8, 4, -4]]  ->  [[3, 4, 2]]
    '''
                (a) [i for i in np.nditer(arr) if i > 0]
                    (b) arr[np.min(arr, 1) > 0]
                    (c) [i for i in arr>0]
                    (d) np.argmin(arr)
```

Correct Answer (b)What is the possible output of this python code?

```
In [1]: import numpy as np
```

```
In [2]: r = np.random.rand(4)

In [3]: print r
```

Select one answer:
a) [0.00000001, 0.98765443, 0.00084734, 0.23423342]
b) [0.00000001, 0.98765443, 0.00084734, -0.23423342]
c) [0.00000001, 0.98765443, 0.00084734, 1.23423342]
d) [0.00000001, 0.98765443, 0.00084734, 1.00000000]

Answer a is the correct answer
However, in the np.random.rand(), the range is output actually is

0.0 <= output < 1.0

Answer b or c or d are not possible

What is the output of this python code? In this example, we provide Python code, and
then several potential example answers.

Code:
```
import numpy as np

test = np.array([1, 2, 3, 4, 5, 6])
test.shape = (2,3)
print test.cumsum(axis = 0)
```


Potential output below, select one answer:

a) [5 , 7 , 9]

b) [[1 , 3 ,   6]
    [4 , 9 , 15]]

c) [[1 , 2]
        [4 , 6]
        [9 , 12]]

d) [[1 , 2 , 3]
    [5 , 7 , 9]]

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>>
>>> test = np.array([1, 2, 3, 4, 5, 6])
>>> test.shape = (2,3)
>>> print test.cumsum(axis = 0)
[[1 2 3]
 [5 7 9]]
>>>
```


Question 1:

Given the following code, choose which is the correct answer.

```
Code:
import numpy as np
a = np.array([[1, 2],[4, 5]])
indices = np.array([1, 0])
print a[indices]

Select one answer:
a)[[4 5]
   [1 2]]
b) [[1 2]
   [4 5]
c) [[2 1]
   [5 4]]
d) [[1 4]
   [2 5]]

  Python Transcript:

>>> import numpy as np
>>> a = np.array([[1, 2],[4, 5]])
>>> indices = np.array([1, 0])
>>> print a[indices]
[[4 5]
 [1 2]]
>>>
```

```
"""What is the output of the following code?"""

import numpy as np
a = np.array([[ 1,  2,  3,  4],
              [ 5,  6,  7,  8],
              [ 9, 10, 11, 12],
              [13, 14, 15, 16]])
print a[2,:]

"""
Select one answer:
a) [ 2  6  10 14]
b) [ 5  6  7  8 ]
c) [ 9 10 11 12 ]
d) [ 12 11 10 9 ]

Correct answer: c)

Python transcript:
>>> print a[2,:]
[ 9 10 11 12]
>>>
```

```
What would be the output of the following Python code:

import numpy as np
A = np.array([[1.,3.,5.,7.],[2.,4.,6.,8.]])
B = A[:,::2]
C = B.mean(axis=1)
print C
```

Select one answer:

a) [3., 4.]
b) [1.5, 5.5]
c) [3.5]
d) [4., 5.]

Correct Answer: a)

Python transcript:
```
>>> import numpy as np
>>> A = np.array([[1.,3.,5.,7.],[2.,4.,6.,8.]])
>>> B = A[:,::2]
>>> C = B.mean(axis=1)
>>> print C
[3., 4.]
```

What would be the output from the following code:

```
import numpy as np
arr1 = np.array(([1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]))
arr2 = arr1[1:3,1:3] / 2
print arr2
```

Select one answer:
a) [[3 3.5][5 5.5]]
b) [3 3.5 5 5.5]
c) [[1 3][5 7]]
d) [[3 3][5 5]]

Correct answer: d

Python transcript:

```
>>> import numpy as np
>>> arr1 = np.array(([1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]))
>>> arr2 = arr1[1:3,1:3] / 2
>>> print arr2
[[3 3]
 [5 5]]
```


You are working on a project that involves time series data (time on one axis, value
on the other) for a biological experiement involving bacterial growth. Unfortunately,
the lab technician on your team is somewhat of a Python beginner. They have written a
function, getData(), that takes the name of a given bacterial species as a string
input and returns a one-dimenstional numpy ndarray of that bacterial species'
population per time point, but without any time labels.

Your teammate does, however, inform you that all measurements were taken once per hour
for 12 hours starting at t=0 hrs, such that there are 13 entries in each array. How
would you complete the following code to construct a pandas dataframe with columns
corresponding to the bacterial species, and rows corresponding to the measurement
times?

Code:
===
```
import numpy as np
import pandas as pd
import random
```

teammate's code. generates synthetic data. see note at bottom of this document.

```
bacterial_species = ['eColi', 'cDiff', 'staphA', 'gNeg', 'strep']
bacterialData = {}

for bs in bacterial_species:
    seed = random.randint(0, 1000)
    time = np.arange(0, 13, 1)
    bacterialData[bs] = np.asarray([seed*2**t for t in time]).astype('float')

def getData(s, data=bacterialData):
    return data[s]

# your code
bacterial_species = ['eColi', 'cDiff', 'staphA', 'gNeg', 'strep']
time_points = np.arange(0, 13, 1)

df = pd.DataFrame(index=time_points, columns=bacterial_species)

for bs in bacterial_species:
        population = getData(bs)
        ## what line should go here? ##

print df
```

Output:
```
================================================================================
       eColi    cDiff   staphA      gNeg    strep
0        763      920      175       598       58
1       1526     1840      350      1196      116
2       3052     3680      700      2392      232
3       6104     7360     1400      4784      464
4      12208    14720     2800      9568      928
5      24416    29440     5600     19136     1856
6      48832    58880    11200     38272     3712
7      97664   117760    22400     76544     7424
8     195328   235520    44800    153088    14848
9     390656   471040    89600    306176    29696
10    781312   942080   179200    612352    59392
11   1562624  1884160   358400   1224704   118784
12   3125248  3768320   716800   2449408   237568
```

Select one answer:
```
================================================================================
a) df.ix[bacterial_species.index(bs), :] = population
b) df[:, bacterial_species.index(bs)] = population
c) df[[bs]] = population
d) df[bs] = population
```

Correct answer: d

Python transcript:
```
================================================================================
>>> import numpy as np
>>> import pandas as pd
>>> import random
>>> bacterial_species = ['eColi', 'cDiff', 'staphA', 'gNeg', 'strep']
>>> bacterialData = {}
>>> for bs in bacterial_species:
...     seed = random.randint(0, 1000)
...     time = np.arange(0, 13, 1)
...     bacterialData[bs] = np.asarray([seed*2**t for t in time]).astype('float')
```

```
...
>>> def getData(s, data=bacterialData):
...      return data[s]
...
>>> bacterial_species = ['eColi', 'cDiff', 'staphA', 'gNeg', 'strep']
>>> time_points = np.arange(0, 13, 1)
>>> df = pd.DataFrame(index=time_points, columns=bacterial_species)
>>> for bs in bacterial_species:
...      population = getData(bs)
...      df[bs] = population
...
>>> print df
      eColi    cDiff   staphA     gNeg    strep
0       941      987      793      310      861
1      1882     1974     1586      620     1722
2      3764     3948     3172     1240     3444
3      7528     7896     6344     2480     6888
4     15056    15792    12688     4960    13776
5     30112    31584    25376     9920    27552
6     60224    63168    50752    19840    55104
7    120448   126336   101504    39680   110208
8    240896   252672   203008    79360   220416
9    481792   505344   406016   158720   440832
10   963584  1010688   812032   317440   881664
11  1927168  2021376  1624064   634880  1763328
12  3854336  4042752  3248128  1269760  3526656


Note:
================================================================================


When I first wrote this question, I kept the portion of code labeled "teammate's code"
above separate from "your code." The teammate's code could be stored in
MyTeammatesCode.py:

    import numpy as np
    import random

    bacterial_species = ['eColi', 'cDiff', 'staphA', 'gNeg', 'strep']
    bacterialData = {}
    for bs in bacterial_species:
        seed = random.randint(0, 1000)
        time = np.arange(0, 13, 1)
        bacterialData[bs] = np.asarray([seed*2**t for t in time]).astype('float')


    def getData(s, data=bacterialData):
        return data[s]
```

The student would then only be shown the portion labeled "your code," which would be
modified to import the function getData from MyTeammatesCode:

```
    from MyTeammatesCode import getData
```

This hides the synthetic generation of the data and, in my opinion, makes the question
more relevant. The reason I did not implement it this way was because the assignment
specification stated that all code must be self contained...

Given the following dataframe, df:

```
df =
                SPY
2016-01-01      10
2016-01-02      11
2016-01-03      NaN
2016-01-04      NaN
2016-01-05      14
2016-01-06      15
```

Which line of code would produce the following output:

```
df =
                SPY
2016-01-01      10
2016-01-02      11
2016-01-03      14
2016-01-04      14
2016-01-05      14
2016-01-06      15
```

Options:

```
a) df = df.fillna(method='ffill')
b) df = df.fillna(method='bfill')
c) df = df.dropna()
d) df = df.interpolate()
```

Correct answer: b

Proof:

```
import pandas as pd

df = pd.DataFrame([10, 11,  pd.np.NaN, pd.np.NaN , 14, 15],
                  columns=['SPY'],
                  index=[pd.date_range(start='1/1/2016', end='1/6/2016')])


df = df.fillna(method='bfill')

print df
```
What is the output of the python code below?

Code:

```
import numpy as np
a = np.array([10,30,50,70])
b = a / a[0]
print b[-1]
```

Select one answer:
a) IndexError
b) 10
c) 5
d) 7

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([10,30,50,70])
```

```
>>> b = a / a[0]
>>> print b[-1]
7
```

Given the following python code fragment,

```
import pandas as pd
from util import get_data
dates = pd.date_range("2010-01-06", "2010-01-07")
symbols = [ 'IBM', 'GOOG', 'AAPL', 'XOM', 'HP' ]
symbols = [ w for w in sorted(symbols[1:]) ]
prices = get_data(symbols,dates)
sym = symbols[-1]
print "sym = {}\n{}".format(sym,prices[sym])
```

```
#-----------------------------------------------------------------------------
```

What would be the output?

```
(a)
sym = AAPL
2010-01-06     210.07
2010-01-07     209.68
Freq: D, Name: AAPL, dtype: float64
```

```
(b)
sym = XOM
2010-01-06     65.36
2010-01-07     65.15
Freq: D, Name: XOM, dtype: float64
```

```
(c)
sym = IBM
2010-01-06     123.90
2010-01-07     123.47
Freq: D, Name: IBM, dtype: float64
```

```
(d)
sym = HP
2010-01-06     45.40
2010-01-07     45.69
Freq: D, Name: HP, dtype: float64
```

```
#-----------------------------------------------------------------------------
```

```
Answer:
(b)
sym = XOM
2010-01-06     65.36
2010-01-07     65.15
Freq: D, Name: XOM, dtype: float64
```

Proof (Transcript):

```
>>> import pandas as pd
>>> from util import get_data
>>> dates = pd.date_range("2010-01-06", "2010-01-07")
>>> symbols = [ 'IBM', 'GOOG', 'AAPL', 'XOM', 'HP' ]
```

```
>>> symbols = [ w for w in sorted(symbols[1:]) ]
>>> prices = get_data(symbols,dates)
>>> sym = symbols[-1]
>>> print "sym = {}\n{}".format(sym,prices[sym])
sym = XOM
2010-01-06    65.36
2010-01-07    65.15
Freq: D, Name: XOM, dtype: float64


#---------------------------------------------------------------------------

The other answers:

for s in symbols:
    print "sym = {}\n{}".format(s,prices[s])

>>> for s in symbols:
...     print "sym = {}\n{}".format(s,prices[s])
...
sym = AAPL
2010-01-06    210.07
2010-01-07    209.68
Freq: D, Name: AAPL, dtype: float64
sym = GOOG
2010-01-06    608.26
2010-01-07    594.10
Freq: D, Name: GOOG, dtype: float64
sym = HP
2010-01-06    45.40
2010-01-07    45.69
Freq: D, Name: HP, dtype: float64
sym = XOM
2010-01-06    65.36
2010-01-07    65.15
Freq: D, Name: XOM, dtype: float64

Since IBM was dropped [1:]

sym = IBM
2010-01-06    123.90
2010-01-07    123.47
Freq: D, Name: IBM, dtype: float64
```

If adr = Average Daily Returns and sddr = Volatility, what is the correct formula to calculate the sr (Sharpe Ratio) is section A:

Code:

```
def test_run():
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
    daily_returns = compute_daily_returns(df)

    adr=daily_returns['SPY'].mean()
    sddr=daily_returns['SPY'].std()
    sqrt_frf = 15.87450786638754
    sr = _A_

    print "Sharpe Ratio:", sr
```

```
Output:
Sharpe Ratio: 0.754609034965

Select one answer:
a) sr = adr / (sqrt_frf * sddr)
b) sr = sddr / (sqrt_frf * adr)
c) sr = sqrt_frf * (adr / sddr)
d) sr = sqrt_frf * (sddr / adr)


Correct answer: c)

Python transcript:
import pandas as pd
import matplotlib.pyplot as plt
from util import get_data, plot_data

def compute_daily_returns(df):
    daily_returns = df.copy()
    daily_returns[1:]=(df[1:]/df[:-1].values)-1
    daily_returns.ix[0,:] = 0
    return daily_returns


def test_run():
    dates = pd.date_range('2009-01-01', '2012-12-31')
    symbols = ['SPY']
    df = get_data(symbols, dates)
    daily_returns = compute_daily_returns(df)

    adr=daily_returns['SPY'].mean()
    sddr=daily_returns['SPY'].std()
    sqrt_frf = 15.87450786638754
    sr = sqrt_frf * (adr / sddr)

    print "Sharpe Ratio:", sr

    #calc incorrect answers
    print "answer A: ", adr / (sqrt_frf * sddr)
    print "answer B: ", sddr / (sqrt_frf * adr)
    print "answer D: ", sqrt_frf * (sddr / adr)

if __name__ == "__main__":
    test_run()




Code:

import numpy as np
x = np.array([[1,3,5],[7,9,11],[13,15,17]])
print x
print __A__

output:

[[ 1  3  5]
 [ 7  9 11]
 [13 15 17]]

[4, 10, 16]
```

```
Select one answer:
a) x[1,:]
b) x[1:]+1
c) x[:,1]+1
d) x[:1]+1

Correct answer: c)

Python transcript:

>>> import numpy as np
>>> x = np.array([[1,3,5],[7,9,11],[13,15,17]])
>>> print x
[[ 1   3  5]
 [ 7   9 11]
 [13  15 17]]
>>> print x[:,1]+1
[ 4 10 16]
>>>
```

You are a gambler and don't care about risk in your stocks. You only care about which
stocks have produced the highest cumulative returns over a given period. But you
aren't a crazy person and you want to spread your assets so that no stock has more
than 30% of your starting investment. What bounds and constraints should you use for
the following code to accomplish your goal?

Code:

```
import numpy as np
import scipy.optimize as spo

def calc_cum_ret(allocs, prices, sv=100):
    normed = prices/prices[0]
    alloced = normed * allocs
    pos_vals = alloced * sv
    port_val = pos_vals.sum(axis=1)
    cum_ret = (port_val[-1]/port_val[0]) - 1
    return cum_ret

def f(allocs, prices, sv = 100):
    cum_ret = calc_cum_ret(allocs, prices, sv)
    return -1 * cum_ret

prices = np.random.random([5,4])
#norms the stock prices

prices = prices/prices[0]
#outputs stock prices

print 'Normed values:\n', prices
#outputs the delta from last row to first row

print 'Delta last row to first row:\n',prices[-1] - prices[0]
guess = [0.25, 0.25, 0.25, 0.25]
const = ({ 'type': 'eq', 'fun': lambda x: })
bounds = __bounds__
result = spo.minimize(f, guess, args=(prices), method='SLSQP', constraints=const,
bounds=bounds)
print result.x

Output:
```

```
Normed values:
[[ 1.           1.           1.           1.         ]
 [ 4.83057394   0.87928325   4.30013505   3.67002576]
 [ 0.22853987   0.50879323   3.92298928   1.60302687]
 [ 9.55912976   0.7723291    6.04157082   4.58556235]
 [ 8.22897886   1.03077582   4.05262634   2.02358999]]
Delta last row to first row:
[ 7.22897886   0.03077582   3.05262634   1.02358999]
[ 0.3   0.1   0.3   0.3]
```

```
Select one answer:
a) __const__ = sum(x), __bounds__ = [(0,0.3) for x in prices]
b) __const__ = sum(x), __bounds__ = [(0,0.3) for x in guess]
c) __const__ = sum(x) - 1, __bounds__ = [(0,0.3) for x in prices]
d) __const__ = sum(x) - 1, __bounds__ = [(0,0.3) for x in guess]
```

answer: d)

Python transcript:

```
>>> import numpy as np
>>> import scipy.optimize as spo
>>>
>>> def calc_cum_ret(allocs, prices, sv=100):
...     normed = prices/prices[0]
...     alloced = normed * allocs
...     pos_vals = alloced * sv
...     port_val = pos_vals.sum(axis=1)
...     cum_ret = (port_val[-1]/port_val[0]) - 1
...     return cum_ret
...
>>> def f(allocs, prices, sv = 100):
...     cum_ret = calc_cum_ret(allocs, prices, sv)
...     return -1 * cum_ret
...
>>> prices = np.random.random([5,4])
>>> #norms the stock prices

... prices = prices/prices[0]
>>> #outputs stock prices

... print 'Normed values:\n', prices
Normed values:
[[ 1.           1.           1.           1.         ]
 [ 4.83057394   0.87928325   4.30013505   3.67002576]
 [ 0.22853987   0.50879323   3.92298928   1.60302687]
 [ 9.55912976   0.7723291    6.04157082   4.58556235]
 [ 8.22897886   1.03077582   4.05262634   2.02358999]]
>>> #outputs the delta from last row to first row

... print 'Delta last row to first row:\n',prices[-1] - prices[0]
Delta last row to first row:
[ 7.22897886   0.03077582   3.05262634   1.02358999]
>>> guess = [0.25, 0.25, 0.25, 0.25]
>>> const = ({ 'type': 'eq', 'fun': lambda x: sum(x) - 1})
>>> bounds = [(0,0.3) for x in guess]
>>> result = spo.minimize(f, guess, args=(prices), method='SLSQP', constraints=const,
bounds=bounds)
>>> print result.x
[ 0.3   0.1   0.3   0.3]
```

How should section A be filled in to complete code that will cause the following
output:

Code:


```
import pandas

dict1 = {'a':[0,1,3],"b":[0,1,2]}
dict2 = {'a':[0,1,2],"d":["x","y","z"]}

data_frame1 = pandas.DataFrame(dict1)
data_frame2 = pandas.DataFrame(dict2)

data_frame = data_frame1.merge(data_frame2, how=_A_)
print data_frame
```

Output:

```
    a   b   d
0   0   0   x
1   1   1   y
2   2 NaN   z
```

Select one answer:
a) left
b) right
c) inner
d) outer

Correct answer: b)

Python transcript:

```
>> import pandas

>> dict1 = {'a':[0,1,3],"b":[0,1,2]}
>> dict2 = {'a':[0,1,2],"d":["x","y","z"]}

>> data_frame1 = pandas.DataFrame(dict1)
>> data_frame2 = pandas.DataFrame(dict2)

>> data_frame = data_frame1.merge(data_frame2, how="right")
>> print data_frame
    a   b   d
0   0   0   x
1   1   1   y
2   2 NaN   z
```


```
import numpy as np
j = np.random.random([3,3])
print j
print _A_
```

Output:

```
[[ 0.99560912  0.2936611   0.66510217]
 [ 0.52336501  0.58238854  0.30215874]
 [ 0.63356296  0.76165895  0.8700516 ]]

[[ 1.  1.  1.]
 [ 1.  1.  1.]
```

```
 [ 1.  1.  1.]]
```

Select one answer:
a) j / j[:,:]
b) j / j[:,0]
c) j / j[1,1]
d) j / j[:,1]

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> j = np.random.random([3,3])
>>> print j
[[ 0.99560912  0.2936611   0.66510217]
 [ 0.52336501  0.58238854  0.30215874]
 [ 0.63356296  0.76165895  0.8700516 ]]
>>> print j/j[:,:]
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]]
```

Example 2

```
import numpy as np
j = np.random.random([3,3])
print j
print _A_
```

Output:

```
[[ 0.65242743  0.40133462  0.08575517]
 [ 0.52589383  0.49806351  0.71443455]
 [ 0.25223838  0.89513972  0.73972009]]

[[ 1.3099282   0.80579006  0.17217718]
 [ 1.05587707  1.          1.43442461]
 [ 0.50643818  1.79724012  1.48519231]]
```


Select one answer:
a) j / j[2,1]
b) j / j[0,0]
c) j / j[1,1]
d) j / j[0,1]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> j = np.random.random([3,3])
>>> print j
[[ 0.65242743  0.40133462  0.08575517]
 [ 0.52589383  0.49806351  0.71443455]
 [ 0.25223838  0.89513972  0.73972009]]

>>> print j/j[1,1]
[[ 1.3099282   0.80579006  0.17217718]
 [ 1.05587707  1.          1.43442461]
 [ 0.50643818  1.79724012  1.48519231]]
```

Example 3

```
import numpy as np
j = np.random.random([3,3])
print j
print _A_
```

Output:

```
[[ 0.60390091  0.66816072  0.12569322]
 [ 0.73579602  0.06948963  0.27994467]
 [ 0.31339756  0.7180695   0.87592347]]

[[ 1.          1.          1.         ]
 [ 1.21840522  0.10400136  2.22720577]
 [ 0.51895528  1.07469577  6.96874064]]
```

Select one answer:
a) j / j[:]
b) j / j[::]
c) j / j[0:]
d) j / j[:1]

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> j = np.random.random([3,3])
>>> print j
[[ 0.27172487  0.22566461  0.7666046 ]
 [ 0.12110203  0.526352    0.57373312]
 [ 0.32745802  0.89627414  0.66634067]]

>>> print j/j[:1]
[[ 1.          1.          1.         ]
 [ 0.44567887  2.33245256  0.74840813]
 [ 1.20510875  3.97170889  0.86921037]]
```

```
import numpy as np
j = np.random.random([4,3])
print j
print _A_
```

Output:

```
[[ 0.21331125  0.04612688  0.36021628]
 [ 0.80857742  0.73366879  0.42214925]
 [ 0.0038377   0.37583067  0.31555916]
 [ 0.69313871  0.08621987  0.98246983]]

[[ 0.26381055  0.06287154  0.85329129]
 [ 1.          1.          1.         ]
 [ 0.00474623  0.51226204  0.74750613]
 [ 0.85723234  0.11751879  2.32730443]]
```

Select one answer:

```
a) j / j[2:3:1]
b) j / j[0:2:2]
c) j / j[3:4:1]
d) j / j[1:2:1]
```

Correct answer: d)

Python transcript:

```
>>> import numpy as np
>>> j = np.random.random([4,3])
>>> print j
[[ 0.4437299   0.96579418  0.48318863]
 [ 0.16248441  0.23711833  0.27095995]
 [ 0.35959424  0.23281634  0.46297012]
 [ 0.40742835  0.19568303  0.73013975]]

>>> print j/j[1:2:1]
[[ 2.7309075   4.0730473   1.78324741]
 [ 1.          1.          1.         ]
 [ 2.21309989  0.9818572   1.70862933]
 [ 2.5074919   0.82525475  2.69464085]]
```

How should section A be filled in to complete code that will cause the following
output:

Code:

```
import numpy as np
a = np.random.rand(3,3)
print a
_A_
print a
```

Output:

```
[[ 0.13291185  0.8271588   0.79832582]
 [ 0.7124346   0.21054717  0.80106292]
 [ 0.08765459  0.43224489  0.12674358]]
[[ 0.13291185  0.          0.79832582]
 [ 0.7124346   1.          0.80106292]
 [ 0.08765459  2.          0.12674358]]
```

Select one answer:
```
a) a[:, 1] = [0, 1, 2]
b) a[:, 2] = [0, 1, 2]
c) a[1, :] = [0, 1, 2]
d) [0, 1, 2] = a[:, 2]
```

Correct answer: a)

Python transcript:
```
>>> import numpy as np
>>> a = np.random.rand(3,3)
>>> print a
[[ 0.13291185  0.8271588   0.79832582]
 [ 0.7124346   0.21054717  0.80106292]
 [ 0.08765459  0.43224489  0.12674358]]
>>> a[:, 1] = [0, 1, 2]
>>> print a
[[ 0.13291185  0.          0.79832582]
 [ 0.7124346   1.          0.80106292]
 [ 0.08765459  2.          0.12674358]]
```

```
What will be printed out:

Code:
a = ((1, 2, 3), )
b = a * 2
print b

Select one answer:
a) ((1, 2, 3, 1, 2, 3))
b) ((2, 4, 6), )
c) ((1, 2, 3), (1, 2, 3))
d) None of the above

Correct answer: c)


Python transcript:

>>> a = ((1, 2, 3),)
>>> b = a * 2
>>> print b
((1, 2, 3), (1, 2, 3))
```

```
How should section A be filled in to complete code that will cause the following
output:

Code:
import numpy as np
a = np.array(((2,3), (1,0)))
b = np.array(((1,2), (3,4)))
print A

Output:
[[11 16]
 [ 1  2]]

Select one answer:
a) a % b
b) a * b
c) np.dot(a,b)
d) np.sum(a)

Correct answer: c

Python transcript:

>>>import  numpy as np

>>>a = np.array(((2,3),(1, 0)))
>>>b = np.array(((1,2), (3, 4)))

>>>print np.dot(a,b)

[[11 16]
 [ 1  2]]
```

```
Code:
```

```
>>> import numpy as np
>>> testArray = np.arange(0,16).reshape((4,4))
>>> newArray = testArray[:3,:2,]
>>> newArray [:1,] = -1
>>> testAverage = np.average(newArray)
```

What is the final value of testAverage

i) 4.0

ii) 2.0

iii) 2.667

iv) -0.5

Correct answer is i)

Transcript:

```
>>> testArray = np.arange(0,16).reshape((4,4))
>>> testArray
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> newArray = testArray[:3,:2,]
>>> newArray
array([[0, 1],
       [4, 5],
       [8, 9]])
>>> newArray [:1,] = -1
>>> newArray
array([[-1, -1],
       [ 4,  5],
       [ 8,  9]])
>>> testAverage = np.average(newArray)
>>> testAverage
4.0
```


Explanations for other choices:

If the student confuses [:1,] with the first row, then he can get the second answer

If newArray [1,] = -1, then the third answer would have been correct

If newArray [1:,] = -1, then the fourth answer would have been correct

What is the output of this python code?

Code:

```
import numpy as np
import pandas as pd
ascending_sequence = pd.DataFrame(np.array([0,1,2,3,4,5]))
rolling_mean = pd.rolling_mean(ascending_sequence, window =3)
print(rolling_mean.values)
```

Select one answer:

```
a) [2.5]

b) [3.]

c) [[  0.0]
    [  0.5]
    [  1.0]
    [  2.0]
    [  3.0]
    [  4.0]]

 d) [[ nan]
    [ nan]
    [  1.]
    [  2.]
    [  3.]
    [  4.]]
```

Correct answer: d)

Python Transcript:

```
import numpy as np
import pandas as pd
ascending_sequence = pd.DataFrame(np.array([0,1,2,3,4,5]))
rolling_mean = pd.rolling_mean(ascending_sequence, window =3)
print(rolling_mean.values)

[[ nan]
 [ nan]
 [  1.]
 [  2.]
 [  3.]
 [  4.]]
```

What is the output of the following code?

```
import numpy as np
a = np.array([[ 1,  2,  3,  4,  5],
              [ 6,  7,  8,  9, 10],
              [11, 12, 13, 14, 15],
              [16, 17, 18, 19, 20]])
print a[:,3]
```

Select one answer:
a) [1 2 3 4]
b) [16 17 18 19 20]
c) [4 9 14 19]
d) [3 8 13 18]

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[ 1,  2,  3,  4,  5],
...               [ 6,  7,  8,  9, 10],
...               [11, 12, 13, 14, 15],
...               [16, 17, 18, 19, 20]])
>>> print a[:,3]
```

```
[ 4   9 14 19]
```
What would the code print as an output (shown as _A_) when executed?

Code:

```python
import pandas as pd
df = pd.DataFrame({'c1': [2, 3, 2, 4, 6, 3, 2, 6],'c2': np.random.randn(8),'c3':['a',
'd', 'c', 'a', 'd', 'c', 'a', 'c']})
print df.duplicated('c1')[2]
```


Output:

_A_

Select one answer:

a) 3
b) True
c) [2,3,6]
d) None of the above

Correct answer: b)

Python Transcript:

```python
>>> import pandas as pd
>>> df = pd.DataFrame({'c1': [2, 3, 2, 4, 6, 3, 2, 6],'c2': np.random.randn(8),'
c3':['a', 'd', 'c', 'a', 'd', 'c', 'a', 'c']})
>>> print df.duplicated('c1')[2]
True
```


What is the output of the below program ?

```python
import pandas as pd
import numpy as np
import math as m


if __name__ == "__main__":
    arr=np.array([[1,-2,3],[-3,-4,5],[-5,4,-3],[-3,2,-1]])
    i=1
    while arr[i:-i,:].size:
        val= np.sum(np.multiply(arr[i:,i:-i],np.transpose(arr[i:-i,-i:])),0)
        i=i+1
    print val
```

a)[-10 -6]
b)Error thrown at multiply statement: Could not be broadcast together with shapes
c)[10 -6]
d)-16

Correct answer:c)

```
ml4t@ml4t-VirtualBox:~$ python test.py
[10 -6]
```

What is the output of this python code:

```python
import numpy as np

i = np.array([41, 51, 57, 50, 31])
j = np.array([[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14], [15, 16, 17, 18,
19], [20, 21, 22, 23, 24]])

print j[i <= 50]
```

Select one answer:

a)
```
[[ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

b)
```
[[ 0  3  4]
 [ 5  8  9]
 [10 13 14]
 [15 18 19]
 [20 23 24]]
```

c)
```
[[ 0  1  2  3  4]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

d)
```
[[ 1  2  3]
 [ 6  7  8]
 [11 12 13]
 [16 17 18]
 [21 22 23]]
```

Correct answer: c)

Python transcript:

```
>>> import numpy as np
>>> i = np.array([41, 51, 57, 50, 31])
>>> j = np.array([[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14], [15, 16, 17,
18, 19], [20, 21, 22, 23, 24]])
>>> print i <= 50
[ True False False  True  True]
>>> print j[i <= 50]
[[ 0  1  2  3  4]
 [15 16 17 18 19]
 [20 21 22 23 24]]What is the output of the code below:
```

Code:

```python
import pandas as pd
if __name__ == "__main__":
    df = pd.DataFrame([[0,1],[2,3],[4,5],[6,7]],columns=None,index=None)
```

```
       print df.tail(2).values
```

Select one answer:
a) [[2 3]]
b) [[4 5]]
c)[[0 1
         [2 3]]
d)[[4 5]
         [6 7]]

For this question, assume every stock has a stock_id , say id for GILD is 1,
GOOG is 2 and id for APPL is 9. This program counts number of occurences of
each stock. Which answer choice correctly displays the following output?

code:

```
import numpy as np
import pandas as pd


arr = np.array([(1, 10),(1, 12),(1, 14),(1, 16),(2, 100),(2, 102),(2, 104),(9, 105) ])
arr = pd.DataFrame(arr, columns = ['stock_id', 'price'])
print arr

arr = //add code here
print arr
```

Output:

```
stock_id
1     4
2     3
9     1
```

Select one answer:

```
a) arr.groupby('price').count()
b) arr.groupby('stock_id').size()
c) arr.groupby('stock_id').shape()
d) arr.countby('stock_id').shape()
```

correct answer: b)

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>>
>>>
>>>
>>> arr = np.array([(1, 10),(1, 12),(1, 14),(1, 16),(2, 100),(2, 102),(2, 104),(9,
105) ])
>>> arr = pd.DataFrame(arr, columns = ['stock_id', 'price'])
>>>
>>> print arr
   stock_id  price
0          1     10
1          1     12
2          1     14
3          1     16
4          2    100
```

```
5          2      102
6          2      104
7          9      105
>>>
>>> arr = arr.groupby('stock_id').size()
>>>
>>> print arr
stock_id
1    4
2    3
9    1
dtype: int64
>>>
```

What is the output of the following code?

```
import numpy as np
arr = np.array([[1, 2, 3],
                [3, 4, 5],
                [5, 6, 7]])

target = np.array([0, 2])
master_target = arr[target][0] + arr[target][1]

print arr * master_target
```

Select one answer:
a) [[6 16 30],
    [18 32 50],
    [30 48 70]]

b) [[1 4 9],
    [3 8 15],
    [5 12 21]]

c) [[5 12 21],
    [15 24 25],
    [25 26 49]]

d) [[6 14 24],
    [16 26 38],
    [26 38 52]]

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> arr = np.array([[1 2 3],
...                 [3 4 5],
...                 [5 6 7]])
>>> target = np.array([0, 2])
>>> master_target = arr[target][0] + arr[target][1]
>>> print arr * master_target

[[ 6 16 30]
 [18 32 50]
 [30 48 70]]
```

What is the output of this python code assuming

weekly returns and 52 weeks in a trading year?

```
Code:
import numpy as np

def compute_sharpe(awr, stdr, rfr, sf):
    sr = (awr - rfr) / stdr * np.sqrt(sf)
    print(sr)

if __name__ == "__main__":
    compute_sharpe(0.02, 0.05, 0.01, 52)
```

Select one answer:
a) 0.2
b) 1.0
c) 1.4
d) 2.8


```
(python27)bash-3.2$ python
Python 2.7.11 |Continuum Analytics, Inc.| (default, Dec  6 2015, 18:57:58)
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import numpy as np
>>> def compute_sharpe(awr, stdr, rfr, sf):
...     sr = (awr - rfr) / stdr * np.sqrt(sf)
...     print(sr)
...
>>> compute_sharpe(0.02, 0.05, 0.01, 52)
1.44222051019
```


Complete the following code to reproduce the output:

```
import numpy as np
import pandas as pd
data = pd.DataFrame(np.arange(16).reshape(4,4),
                    index=list('abcd'),
                    columns=['one', 'two', 'three', 'four'])
```

Output:

```
a     1
b     5
c     9
d    13
Name: two, dtype: int64
```

Select one answer:
a) data['two']
b) data.ix[:, 1]
c) data.ix[:, 'two']
d) All of the above

Correct answer: d

Python transcript:

```
>>> import numpy as np
>>> import pandas as pd
>>> data = pd.DataFrame(np.arange(16).reshape(4,4),
```

```
...                                   index=list('abcd'),
...                                   columns=['one', 'two', 'three', 'four'])
>>> data['two']
a      1
b      5
c      9
d     13
Name: two, dtype: int64
>>> data.ix[:, 1]
a      1
b      5
c      9
d     13
Name: two, dtype: int64
>>> data.ix[:, 'two']
a      1
b      5
c      9
d     13
Name: two, dtype: int64
```

How should section <A> be filled in to complete code that will create a multi-line
plot?

Code:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randn(10, 4).cumsum(0), columns=['A', 'B', 'C', 'D'],
index=np.arange(0, 100, 10))
<A>
plt.show()
```

Select one answer:
a) df.plot()
b) df.plot(kind="barh", stacked=True)
c) df['A'].hist()
d) plt.scatter(df['A'], df['B'])

Correct Answer: a

Python transcript:
```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.random.randn(10, 4).cumsum(0), columns=['A', 'B', 'C', 'D'],
index=np.arange(0, 100, 10))
>>> df.plot()
<matplotlib.axes._subplots.AxesSubplot object at 0x7f86363e6290>
>>> plt.show()
```

Which statment gives the expected output?

Code:

```
import pandas as pd
```

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
        'B': ['B0', 'B1', 'B2', 'B3'],
        'C': ['C0', 'C1', 'C2', 'C3'],
        'D': ['D0', 'D1', 'D2', 'D3']},
        index=[0, 1, 2, 3])
df2 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],
        'D': ['D2', 'D3', 'D6', 'D7'],
        'F': ['F2', 'F3', 'F6', 'F7']},
        index=[2, 3, 6, 7])
```

```
Expected output:
     A    B    C    D    B    D    F
2   A2   B2   C2   D2   B2   D2   F2
3   A3   B3   C3   D3   B3   D3   F3
```

```
Select one answer:
a) pd.concat([df1, df2], axis=1, join="inner")
b) pd.concat([df1, df2], axis=1, join="outer")
c) pd.concat([df1, df2], axis=0, join="inner")
d) pd.concat([df1, df2], axis=0, join="outer")
```

```
Correct answer: a)
```

```
Python transcript:
>>> import pandas as pd
>>> df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
...       'B': ['B0', 'B1', 'B2', 'B3'],
...       'C': ['C0', 'C1', 'C2', 'C3'],
...       'D': ['D0', 'D1', 'D2', 'D3']},
...       index=[0, 1, 2, 3])
>>> df2 = pd.DataFrame({'B': ['B2', 'B3', 'B6', 'B7'],
...       'D': ['D2', 'D3', 'D6', 'D7'],
...       'F': ['F2', 'F3', 'F6', 'F7']},
...       index=[2, 3, 6, 7])
>>> pd.concat([df1, df2], axis=1, join="inner")
     A    B    C    D    B    D    F
2   A2   B2   C2   D2   B2   D2   F2
3   A3   B3   C3   D3   B3   D3   F3

[2 rows x 7 columns]
```

```
What is the output?

Code:

import pandas as pd
import numpy as np
df=pd.DataFrame([10,20,30,40,50])
df_new=df.copy()
df_new.iloc[1]=0
df_new.loc[0]+=np.sum(df[:3])

print df_new


select one answer for the output:
a)    0
0   10
1   20
```

```
2   30
3   40
4   50


b)   0
0   70
1    0
2   30
3   40
4   50


c)   0
0   50
1    0
2   30
3   40
4   50


d)   0
0   40
1   0
2   30
3   40
4   50
```

Correct answer:
```
b)   0
0   70
1    0
2   30
3   40
4   50
```

Python transcript:
```
>>> import pandas as pd
>>> import numpy as np
>>> df=pd.DataFrame([10,20,30,40,50])
>>> df_new=df.copy()
>>> df_new.iloc[1]=0
>>> df_new.loc[0]+=np.sum(df[:3])
>>>
>>> print df_new
     0
0   70
1    0
2   30
3   40
4   50
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import numpy as np
d = np.array([[1,3,5,7], [2,4,6,8], [2,3,5,7], [1,4,9,16]])
print _A_
```

Output:

```
[[3 5 7]
 [4 6 8]]
```

```
Select one answer:
A. d[3:2,-1:4]
B. d[:2,:-1]
C. d[0:-2,1:]
D. d[1:3,2:4]
```

Correct answer: c)

```
>>> import numpy as np
>>> d = np.array([[1,3,5,7], [2,4,6,8], [2,3,5,7], [1,4,9,16]])
>>> print d[3:2,-1:4]
[]
>>> print d[:2,:-1]
[[1 3 5]
 [2 4 6]]
>>> print d[0:-2,1:]
[[3 5 7]
 [4 6 8]]
>>> print d[1:3,2:4]
[[6 8]
 [5 7]]
```

What gets printed?

Code:

```
names1 = ['Amir', 'Barry', 'Chales', 'Dao']
names2 = names1
names3 = names1[:]

names2[0] = 'Alice'
names3[1] = 'Bob'

sum = 0
for ls in (names1, names2, names3):
    if ls[0] == 'Alice':
        sum += 1
    if ls[1] == 'Bob':
        sum += 10

print sum
```

```
Select one answer:
a) 11
b) 12
c) 21
d) 22
```

Correct answer: b)

Python transcript:

```
>>> names1 = ['Amir', 'Barry', 'Chales', 'Dao']
>>> names2 = names1
>>> names3 = names1[:]
>>> names2[0] = 'Alice'
>>> names3[1] = 'Bob'
>>> sum=0
>>> for ls in (names1, names2, names3):
    if ls[0] == 'Alice':
        sum += 1
    if ls[1] == 'Bob':
```

```
        sum += 10


>>> print sum
12
```

What is the output of this python code?

Code:

```
import numpy as np
a = np.array([[ 0,  1,  2,  3],
              [ 4,  5,  6,  7],
              [ 8,  9, 10, 11]])
i = np.array([[0,1],
              [1,2]])
j = np.array([[2,3],
              [1,0]])
print  a[i,j]
```

Select one answer:
a) [[2 7]
    [5 8]]
b) [2 2]
c) [[1 6]
    [11 4]]
d) [[0 0]
    [0 0]
    [2 3]
    [1 0]]

Correct answer: a)

Python transcript:

```
>>> import numpy as np
>>> a = np.array([[ 0,  1,  2,  3],
                  [ 4,  5,  6,  7],
                  [ 8,  9, 10, 11]])
>>> i = np.array([[0,1],
                  [1,2]])
>>> j = np.array([[2,3],
                  [1,0]])
>>> print  a[i,j]

[[2 7]
 [5 8]]
```

In order to optimize sharpe ratio, we define an objective function and a function to calculate statistics (set rfr=0):

```
def statistics(allocs)
    ...
    k = np.sqrt(252)
    ...
    return _____

def min_func_sharpe_ratio(allocs):
    return -statistics(allocs)[3]
```

What will be the possible code in the blank line

A). return cummultive_return, average_daily_return, sharpe_ratio

B). return cummultive_return, average_daily_return, std_daily_return,
k*cummultive_return/std_daily_return

C). return cummultive_return, average_daily_return, std_daily_return,
k*average_daily_return/std_daily_return

D). return cummultive_return, average_daily_return, std_daily_return,
k*cummultive_return/average_daily_return

Answer: C

How should section A be filled in to complete code that will cause the following
output:

Code:

```
import pandas as pd
left = pd.DataFrame({'key': ['foo','bar'], 'lval':[1,2]})
right = pd.DataFrame({'key': ['foo','foo','bar'], 'rval':[3,4,5]})
print left
print right
print _A_
```

Output:

```
    key  lval
0   foo     1
1   bar     2
    key  rval
0   foo     3
1   foo     4
2   bar     5
    key  lval  rval
0   foo     1     3
1   foo     1     4
2   bar     2     5
```

Select one answer:
a) pd.concat([right], [left])
b) pd.concat([left], [right])
c) pd.merge(left, right, on='key')
d) pd.merge(left, right, on='lval')

Correct answer: c)

Python transcript:

```
>>> import pandas as pd
>>> left = pd.DataFrame({'key': ['foo','bar'], 'lval':[1,2]})
>>> right = pd.DataFrame({'key': ['foo','foo','bar'], 'rval':[3,4,5]})
>>> print left
    key  lval
0   foo     1
1   bar     2
>>> print right
    key  rval
0   foo     3
1   foo     4
2   bar     5
>>> print pd.merge(left, right, on='key')
    key  lval   rval
```

```
0  foo      1      3
1  foo      1      4
2  bar      2      5
```

Section A should be filled in to complete code that will generate a list of five 2-tuples (0,1):

Code:

```
_A_
print res
```

Output:
`[(0,1),(0,1),(0,1),(0,1),(0,1)]`

Select one answer that does NOT generate the output:
a) res = [(0,1),]*5
b) res = [(0,1),(0,1),(0,1),(0,1),(0,1)]
c)
```
res = []
for i in range(5):
        res += [(0,1)]
```
d)
```
res = []
for i in range(5):
        res.append(0,1)
```

Correct answer: d)

Python transcript:

```
>>> res = []
>>> res = [(0,1),]*5
>>> print res
[(0, 1), (0, 1), (0, 1), (0, 1), (0, 1)]
>>> res = []
>>> res = [(0,1),(0,1),(0,1),(0,1),(0,1)]
>>> print res
[(0, 1), (0, 1), (0, 1), (0, 1), (0, 1)]
>>> res = []
>>> for i in range(5):
        res += [(0,1)]


>>> print res
[(0, 1), (0, 1), (0, 1), (0, 1), (0, 1)]
>>> res = []
>>> for i in range(5):
        res.append(0,1)


Traceback (most recent call last):
  File "<pyshell#128>", line 2, in <module>
    res.append(0,1)
TypeError: append() takes exactly one argument (2 given)
>>> print res
[]
```

DataFrame scala initially contains the following columns: ['Gold', 'Silver', 'USD', 'CAD']. What are the columns after the following code is run?

```
Code:

import pandas as pd
scala = pd.DataFrame(columns = ['Gold', 'Silver', 'USD', 'CAD'])
scala = scala.dropna(subset=['USD'])
scala = scala.rename(columns = {'Gold': 'AU'})
scala = scala.join(pd.DataFrame(columns = ['Copper']), how = 'left')
print(scala.columns.tolist())

Select one answer:
a) ['AU', 'Silver', 'USD', 'CAD', 'Copper']
b) ['AU', 'Silver', 'CAD', 'Copper']
c) ['AU', 'Silver', 'CAD']
d) ['Gold', 'Silver', 'USD', 'CAD', 'Copper']

Correct answer: a)

Python transcript:

>>> import pandas as pd
>>> scala = pd.DataFrame(columns = ['Gold', 'Silver', 'USD', 'CAD'])
>>> scala = scala.dropna(subset=['USD'])
>>> scala = scala.rename(columns = {'Gold': 'AU'})
>>> scala = scala.join(pd.DataFrame(columns = ['Copper']), how = 'left')
>>> print(scala.columns.tolist())
['AU', 'Silver', 'USD', 'CAD', 'Copper']

What is the output of the following code?

import pandas as pd
import numpy as np

d = {'one':   np.array([1, 2, 3, 4]),
     'two':   np.array([5, 6, 7, 8]),
     'three': np.array([9, 10, 11, 12]),
     'four':  np.array([13, 14, 15, 16])}

df = pd.DataFrame(d)

print df[['one', 'three']][::2]


Select one answer:

a)      one    three
    0    1      9
    1    2      10

b)      one    three
    0    1      9
    2    3      11

c)      one    three
    0    1      9
    1    2      10
    2    3      11

d)      one    two    three
    0    1      5      9
    1           2      6      10
    2    3      7       11
```

```
Correct answer:  b)


Python transcript:

>>> import pandas as pd
>>> import numpy as np
>>> d = {'one':   np.array([1, 2, 3, 4]),
         'two':   np.array([5, 6, 7, 8]),
         'three': np.array([9, 10, 11, 12]),
         'four':  np.array([13, 14, 15, 16])}
>>> df = pd.DataFrame(d)
>>> print df[['one', 'three']][::2]
   one  three
0   1      9
2   3     11
```

```
How should section M be filled in to complete code that will cause the following
output:

import numpy as np
A = [[[1,2,3],[4,5,6]], [[7,8,9],[10,11,12]]]
B = np.array(A)
print _M_

Output:
array([1,  2,  3,  4],
      [5,  6,  7,  8],
      [9, 10, 11, 12]])
```

```
a)  B.reshape(3,-1).T
b)  B.reshape(4,-1).T
c)  B.reshape(-1,3)
d)  B.reshape(-1,4)
```

```
Correct answer: d)

Python transcript:

>>> import numpy as np
>>> A = [[[1,2,3],[4,5,6]], [[7,8,9],[10,11,12]]]
>>> B = np.array(A)
>>> print B.reshape(-1, 4)
array([1,  2,  3,  4],
      [5,  6,  7,  8],
      [9, 10, 11, 12]])
```

```
>>> import Numpy as np
>>> x = np.array([2, 4, 6, 8, 10, 12])
>>> x[1:5:2]
>>> print(x[1:5:2])
```

What is the output?

a) [4 8]
b) [4 6 8 10 12]
c) [4,6]
d) [2 6 10]

Correct answer: a)

Python Transcript:

```
>>> import Numpy as np
>>> x = np.array([2, 4, 6, 8, 10, 12])
>>> x[1:5:2]
>>> print(x[1:5:2])
[4 8]
```

What does the following code print as a result?

Code:

```
import pandas as pd
d = {'AAPL' : pd.Series([615.99,626.63,621.64], index=['2012-04-02','2012-04-
03','2012-04-04']),
     'GOOG' : pd.Series([646.92,642.62,635.15], index=['2012-04-02','2012-04-
03','2012-04-04']),
     'MSFT' : pd.Series([31.87,31.52,30.80], index=['2012-04-02','2012-04-03','2012-
04-04'])
     }
df1 = pd.DataFrame(d)
df2 = df1.apply(lambda x: x * 2)
result = df2 / df1.values
print result
```

Select one answer:

```
    a)              AAPL     GOOG     MSFT
       2012-04-02  1231.98  1293.84  63.74
       2012-04-03  1253.26  1285.24  63.04
       2012-04-04  1243.28  1270.30  61.60

    b)              AAPL     GOOG     MSFT
       2012-04-02  615.99   646.92   31.87
       2012-04-03  626.63   642.62   31.52
```

```
        2012-04-04  621.64  635.15  30.80

   c)                 AAPL    GOOG    MSFT
      2012-04-02        1       1       1
      2012-04-03        1       1       1
      2012-04-04        1       1       1

   d)                 AAPL    GOOG    MSFT
      2012-04-02        2       2       2
      2012-04-03        2       2       2
      2012-04-04        2       2       2
```

Correct answer: d)

Python transcript:

```
>>> import pandas as pd
>>> d = {'AAPL' : pd.Series([615.99,626.63,621.64], index=['2012-04-02','2012-04-
03','2012-04-04']),
...      'GOOG' : pd.Series([646.92,642.62,635.15], index=['2012-04-02','2012-04-
03','2012-04-04']),
...      'MSFT' : pd.Series([31.87,31.52,30.80], index=['2012-04-02','2012-04-
03','2012-04-04'])
...      }
>>> df1 = pd.DataFrame(d)
>>> df2 = df1.apply(lambda x: x * 2)
>>> result = df2 / df1.values
>>> print result
            AAPL  GOOG  MSFT
2012-04-02     2     2     2
2012-04-03     2     2     2
2012-04-04     2     2     2
```

What is the output of the following code?

Code:

```
import numpy as np

x = np.array([[1,2,3],[5,6,7]], dtype='int')
```

```
y = np.array(x/2)
x[1,0] = 0
print y.sum(axis=1)[-1]
```

Select one answer
a) 2
b) 4
c) 8
d) 9

Correct answer: c)

```
Python Transcript:
>>> import numpy as np
>>> x = np.array([[1,2,3],[5,6,7]], dtype='int')
>>> y = np.array(x/2)
>>> x[1,0] = 0
>>> print y.sum(axis=1)[-1]
8
```

What is the output of the following python code?

Code:

```
import numpy as np
a = np.arange(6).reshape(2,3)
b = a.sum(axis=1)
print b
```

Select one answer:
a) [3 5 7]
b) [[3]
    [12]]
c) [[3]
    [5]
    [7]]
d) [3 12]

Correct answer: d)

```
Python transcript:
>>> import numpy as np
>>> a = np.arange(6).reshape(2,3)
>>> b = a.sum(axis=1)
>>> print b
[ 3 12]
```

```
Which is of the following is the expected output of this python code:
   import pandas as pd
   df = pd.DataFrame([(1, 2, 3, 4, 5), (6, 7, 8, 9, 10), (11, 12, 13, 14, 15)])
   print df.ix[1:2,3:4]

a) 9 10
   14 15

b) 14 15

c)  3  4
  1 9  10
  2 14 15

d)  3  4
  1 14 15


Correct answer: c

Python Transcript:

Python 2.7.10 (default, Oct 23 2015, 18:05:06)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.DataFrame([(1, 2, 3, 4, 5), (6, 7, 8, 9, 10), (11, 12, 13, 14, 15)])
>>> print df.ix[1:2,3:4]
    3   4
1   9  10
2  14  15
>>>
```

```
What is the output of this python code?

import pandas as pd
from util import get_data
symbols = ['GOOG', 'AAPL', 'GLD', 'HNZ', 'SPY']
symbols = symbols[0:-1:2]
dates = pd.date_range('2011-10-12', '2011-10-14')
df = get_data(symbols, dates, addSPY=False)
print df
```

a)
```
              GOOG       GLD
2011-10-12  548.50    163.26
2011-10-13  558.99    162.30
2011-10-14  591.68    163.40
```

b)
```
              GOOG       GLD       SPY
2011-10-12  548.50    163.26   118.85
2011-10-13  558.99    162.30   118.61
2011-10-14  591.68    163.40   120.64
```

c)
```
              GOOG       GLD
2011-10-12  548.50    163.26
2011-10-13  558.99    162.30
```

d)
```
              GOOG       GLD       SPY
2011-10-12  548.50    163.26   118.85
2011-10-13  558.99    162.30   118.61
```

Correct answer: a)

```
>>> import pandas as pd
>>> from util import get_data
>>> symbols = ['GOOG', 'AAPL', 'GLD', 'HNZ', 'SPY']
>>> symbols = symbols[0:-1:2]
>>> dates = pd.date_range('2011-10-12', '2011-10-14')
>>> df = get_data(symbols, dates, addSPY=False)
>>> print df
              GOOG       GLD
2011-10-12  548.50    163.26
2011-10-13  558.99    162.30
2011-10-14  591.68    163.40
```

Given a dataframe df, where

```
df =
              GOOG    AAPL    GLD      XOM
2010-01-04  626.75  213.10  109.80  64.55
2010-01-05  623.99  213.46  109.70  64.80
2010-01-06  608.26  210.07  111.51  65.36
2010-01-07  594.10  209.68  110.82  65.15
2010-01-08  602.02  211.07  111.37  64.89
2010-01-11  601.11  209.21  112.85  65.62
2010-01-12  590.48  206.83  110.49  65.29
2010-01-13  587.09  209.75  111.54  65.03
```

What is the output of the following python code?

Code:

```
print df.ix['2010-01-11':'2010-01-06', ['AAPL', 'XOM']]
```

Select one answer:

a)
```
              GOOG    AAPL    GLD      XOM
2010-01-06  608.26  210.07  111.51  65.36
```

```
2010-01-07   594.10 209.68 110.82 65.15
2010-01-08   602.02 211.07 111.37 64.89
2010-01-11   601.11 209.21 112.85 65.62
```

b)
```
             AAPL    XOM
2010-01-06   210.07  65.36
2010-01-07   209.68  65.15
2010-01-08   211.07  64.89
2010-01-11   209.21  65.62
```

c)
```
Empty Dataframe
Columns: [AAPL, XOM]
Index: []
```

d)
```
             AAPL    XOM
2010-01-11   209.21  65.62
2010-01-08   211.07  64.89
2010-01-07   209.68  65.15
2010-01-06   210.07  65.36
```

Correct answer: c)

Python transcript (given df):

```
>>> print df.ix['2010-01-11':'2010-01-06',['AAPL','XOM']]
Empty DataFrame
Columns: [AAPL, XOM]
Index: []
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import pandas as pd
df = pd.DataFrame({'a':[3,3,3],
                   'b':[6,6,6],
                   'c':[3,3,3]})

print df
print _A_
```

Output:

```
   a  b   c
0  3  6   3
1  3  6   3
2  3  6   3

   a  b  c
1  2  1  2
2  2  1  2
```

Select one answer:
a) df[0,0]/df[:-1]
b) df.ix[0,0]/df[1:2]
c) df.ix[0,1]/df[1:]
d) df[1,0]/df[1:3]

```
Correct answer: c)

Python transcript:

>>> import pandas as pd
>>> df = pd.DataFrame({'a':[3,3,3],
...                    'b':[6,6,6],
...                    'c':[3,3,3]})
>>> print df
   a  b  c
0  3  6  3
1  3  6  3
2  3  6  3
>>> df.ix[0,1]/df[1:]
   a  b  c
1  2  1  2
2  2  1  2



import pandas as pd
from util import get_data

'''
Given unmodified df for SYM.csv where:

    print df

Output:

              SYM
2009-05-31    NaN
2009-06-01  43.78
2009-06-02    NaN
2009-06-03    NaN
2009-06-04  44.51


Which output would this code generate?
'''

def fill_dataframe(df):
    df.fillna(method='ffill', inplace=True)
    df.fillna(method='bfill', inplace=True)

dates = pd.date_range('2009-05-31', '2009-06-04')
df = get_data(['SYM'], dates, addSPY=False)
fill_dataframe(df)
print df

'''
a) Output:                  b) Output:

              SYM                        SYM
2009-05-31    NaN          2009-05-31  43.78
2009-06-01  43.78          2009-06-01  43.78
2009-06-02    NaN          2009-06-02  43.78
2009-06-03    NaN          2009-06-03  43.78
2009-06-04  44.51          2009-06-04  44.51

c) Output:                  d) Output:

              SYM                        SYM
```

```
2009-05-31      NaN         2009-05-31   43.78
2009-06-01      NaN         2009-06-01   43.78
2009-06-02      NaN         2009-06-02   44.51
2009-06-03      NaN         2009-06-03   44.51
2009-06-04      NaN         2009-06-04   44.51
```

Answer: Correct answer is b). DataFrame contents were derived from FAKE1, but modified to be
         condensed down to reduce problem length.

'''

what is the output of this code:

```python
import pandas as pd
import numpy as np
a = [[22,33,44,55],
     [1,2,3,np.nan],
     [111,222,np.nan,444],
     [11,22,33,44]]
df = pd.DataFrame(a,columns=['COL1','COL2','COL3','COL4'])
df=df.fillna(method='ffill')
df=df/df.ix[0,0:]
print df
```

a)
```
      COL1        COL2        COL3        COL4
0  22.000000   33.000000   44.000000   55.000000
1   0.045455    0.060606    0.068182    1.000000
2   5.045455    6.727273    0.068182    8.072727
3   0.500000    0.666667    0.750000    0.800000
```

b)
```
      COL1        COL2        COL3        COL4
0   1.000000    0.060606    0.068182    1.000000
1   0.045455    1.000000    0.068182    0.800000
2   5.045455    6.727273    1.000000    8.072727
3   0.500000    0.666667    0.750000    1.000000
```

c)
```
       COL1        COL2        COL3        COL4
0   1.000000    1.000000    1.000000    1.000000
1   0.045455    0.060606    0.068182         NaN
2   5.045455    6.727273         NaN    8.072727
3   0.500000    0.666667    0.750000    0.800000
```


d)
```
       COL1        COL2        COL3        COL4
0   1.000000    1.000000    1.000000    1.000000
1   0.045455    0.060606    0.068182    1.000000
2   5.045455    6.727273    0.068182    8.072727
3   0.500000    0.666667    0.750000    0.800000
```


ans) d

Python Transcript

```
>>> import pandas as pd
>>> import numpy as np
```

```
>>> a = [[22,33,44,55],
...      [1,2,3,np.nan],
...      [111,222,np.nan,444],
...      [11,22,33,44]]
>>> df = pd.DataFrame(a,columns=['COL1','COL2','COL3','COL4'])
>>> df=df.fillna(method='ffill')
>>> df=df/df.ix[0,0:]
>>> print df
        COL1      COL2      COL3      COL4
0   1.000000  1.000000  1.000000  1.000000
1   0.045455  0.060606  0.068182  1.000000
2   5.045455  6.727273  0.068182  8.072727
3   0.500000  0.666667  0.750000  0.800000
```

What is the output of the following program?

```
a = [1.1 ** i for i in range(0,5)]
dvs = pd.DataFrame(a)
drs = dvs[0] / dvs[0].shift(1) - 1
print [round(num, 1) for num in drs]
```

```
(a) [nan, 0.1, 0.1, 0.1, 0.1]
(b) [nan, 1.1, 1.1, 1.1, 1.1]
(c) [nan, 1.1, 1.21, 1.3, 1.5]
(d) [0.1, 0.1, 0.1, 0.1, 0.1]
```

correct answer (a)

```
>>> import pandas as pd
>>> a = [1.1 ** i for i in range(0,5)]
>>> dvs = pd.DataFrame(a)
>>> drs = dvs[0] / dvs[0].shift(1) - 1
>>> print [round(num, 1) for num in drs]
[nan, 0.1, 0.1, 0.1, 0.1]
```

What will be the output of the print statement for the code below:

Code:

```
import pandas as pd
import numpy as np

d = {'Portfolio_1': [0.00, 0.24, 0.32, 0.44],
     'Portfolio_2': [0.30, 0.12, 0.33, 0.25],
     'Portfolio_3': [0.13, 0.36, 0.19, 0.32],
     'Portfolio_4': [0.17, 0.16, 0.31, 0.36]}

df = pd.DataFrame(d)

print df.to_string(index=False)

threshold = df.iloc[3,2]
```

```
print threshold

allocations = np.array([0.41, 0.24, 0.56, 0.31, 0.32, 0.16, 0.33])

print [np.where( allocations > threshold )]
```

Select one answer:

```
a) [ 0.41  0.56  0.33]
b) [(array([0, 2, 6]),)]
c) [ 0.41  0.56  0.32  0.33]
d) [(array([0, 2, 4, 6]),)]
```

Correct answer: b)

Python transcript:

```
>>> import pandas as pd
>>> import numpy as np
>>> d = {'Portfolio_1': [0.00, 0.24, 0.32, 0.44],
     'Portfolio_2': [0.30, 0.22, 0.33, 0.25],
     'Portfolio_3': [0.13, 0.36, 0.19, 0.32],
     'Portfolio_4': [0.27, 0.16, 0.31, 0.36]}
>>> df = pd.DataFrame(d)
>>> print df.to_string(index=False)
Portfolio_1  Portfolio_2  Portfolio_3  Portfolio_4
       0.00         0.30         0.13         0.27
       0.24         0.22         0.36         0.16
       0.32         0.33         0.19         0.31
       0.44         0.25         0.32         0.36

>>> threshold = df.iloc[3,2]
>>> print threshold
0.32
>>> allocations = np.array([0.41, 0.24, 0.56, 0.31, 0.32, 0.16, 0.33])
```

How should section A be filled in to complete code that will cause the following output:

Code:

```
import numpy as np
a = np.array([(0,1,2),(2,3,4),(4,5,6),(6,7,8)])
b = np.array([(0,1,2,3),(4,5,6,7),(8,9,10,11),(12,13,14,15)])
_A_
print b
```

Output:

```
[[ 0  1  2  3]
 [ 4  2  3  4]
 [ 8  4  5  6]
 [12  6  7  8]]
```

Select one answer:
```
a) b[-1:,2:4] = a[-1:,0:2]
b) b[-1:,1:3] = a[-1:,0:2]
c) b[-3:,1:] = a[-3:,:]
d) b[-3:-1,1:] = a[-3:-1,:]
```

```
Correct answer: c)


Python transcript:

>>> import numpy as np
>>> a = np.array([(0,1,2), (2,3,4), (4,5,6), (6,7,8)])
>>> b = np.array([(0,1,2,3),(4,5,6,7),(8,9,10,11),(12,13,14,15)])
>>> b[-3:,1:] = a[-3:,:]
>>> print b
[[ 0  1  2  3]
 [ 4  2  3  4]
 [ 8  4  5  6]
 [12  6  7  8]]
```

Published by [Google Drive](#) – [Report Abuse](#) – Updated automatically every 5 minutes