101 Pandas Exercises for Data Analysis

101 python pandas exercises are designed to challenge your logical muscle and to help internalize data manipulation with python's favorite package for data analysis. The questions are of 3 levels of difficulties with L1 being the easiest to L3 being the hardest.

101 Pandas Exercises. Photo by Chester Ho.

You might also like to practice the 101 NumPy exercises (https://www.machinelearningplus.com/python/101-numpy-exercises-python/), they are often used together.

1. How to import pandas and check the version?

Show Solution

```
import numpy as np  # optional
import pandas as pd
print(pd.__version__)
print(pd.show_versions(as_json=True))
```

```
0.20.3
{'system': {'commit': None}, 'dependencies': {'pandas': '0.20.3', 'pytest': '3.2.1',
'pip': '9.0.1', 'setuptools': '36.5.0.post20170921', 'Cython': '0.26.1', 'numpy':
'1.13.3', 'scipy': '0.19.1', 'xarray': None, 'IPython': '6.1.0', 'sphinx': '1.6.3',
'patsy': '0.4.1', 'dateutil': '2.6.1', 'pytz': '2017.2', 'blosc': None, 'bottleneck'
: '1.2.1', 'tables': '3.4.2', 'numexpr': '2.6.2', 'feather': None, 'matplotlib': '2.
1.0', 'openpyxl': '2.4.8', 'xlrd': '1.1.0', 'xlwt': '1.2.0', 'xlsxwriter': '1.0.2',
'lxml': '4.1.0', 'bs4': '4.6.0', 'html5lib': '0.999999999', 'sqlalchemy': '1.1.13',
'pymysql': None, 'psycopg2': None, 'jinja2': '2.9.6', 's3fs': None, 'pandas_gbq': No
ne, 'pandas_datareader': None}}
None
```

2. How to create a series from a list, numpy array and dict?

Create a pandas series from each of the items below: a list, numpy and a dictionary

Input

```
import numpy as np
mylist = list('abcedfghijklmnopqrstuvwxyz')
myarr = np.arange(26)
mydict = dict(zip(mylist, myarr))
```

Show Solution

3. How to convert the index of a series into a column of a dataframe?

Difficulty Level: L1

Convert the series `ser` into a dataframe with its index as another column on the dataframe.

Input

```
mylist = list('abcedfghijklmnopqrstuvwxyz')
myarr = np.arange(26)
mydict = dict(zip(mylist, myarr))
ser = pd.Series(mydict)
```

Show Solution

```
# Input
mylist = list('abcedfghijklmnopqrstuvwxyz')
myarr = np.arange(26)
mydict = dict(zip(mylist, myarr))
ser = pd.Series(mydict)

# Solution
df = ser.to_frame().reset_index()
print(df.head())
```

```
  index  0
0     a  0
1     b  1
2     c  2
3     d  4
4     e  3
```

4. How to combine many series to form a dataframe?

Difficulty Level: L1

Combine ser1 and ser2 to form a dataframe.

Input

```
import numpy as np
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

Show Solution

```
# Input
import numpy as np
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))

# Solution 1
df = pd.concat([ser1, ser2], axis=1)

# Solution 2
df = pd.DataFrame({'col1': ser1, 'col2': ser2})
print(df.head())
```

```
  col1  col2
0    a     0
1    b     1
2    c     2
3    e     3
4    d     4
```

5. How to assign name to the series' index?

Difficulty Level: L1

Give a name to the series `ser` calling it 'alphabets'.

Input

```
ser = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
```

Show Solution

```
# Input
ser = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))

# Solution
ser.name = 'alphabets'
ser.head()
```

```
0    a
1    b
2    c
3    e
4    d
Name: alphabets, dtype: object
```

6. How to get the items of series A not present in series B?

Difficulty Level: L2

From `ser1` remove items present in `ser2`.

```
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
```

Show Solution

```
# Input
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])

# Solution
ser1[~ser1.isin(ser2)]
```

```
0    1
1    2
2    3
dtype: int64
```

7. How to get the items not common to both series A and series B?

Difficulty Level: L2

Get all items of `ser1` and `ser2` not common to both.

Input

```
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
```

Show Solution

```
# Input
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])

# Solution
ser_u = pd.Series(np.union1d(ser1, ser2))  # union
ser_i = pd.Series(np.intersect1d(ser1, ser2))  # intersect
ser_u[~ser_u.isin(ser_i)]
```

```
0    1
1    2
2    3
5    6
6    7
7    8
dtype: int64
```

8. How to get the minimum, 25th percentile, median, 75th, and max of a numeric series?

Difficuty Level: L2

Compute the minimum, 25th percentile, median, 75th, and maximum of `ser`.

Input

```
ser = pd.Series(np.random.normal(10, 5, 25))
```

Show Solution

```
# Input
state = np.random.RandomState(100)
ser = pd.Series(state.normal(10, 5, 25))

# Solution
np.percentile(ser, q=[0, 25, 50, 75, 100])
```

```
array([  1.39267584,   6.49135133,  10.2578186 ,  13.06985067,  25.80920994])
```

9. How to get frequency counts of unique items of a series?

Difficulty Level: L1

Calculte the frequency counts of each unique value `ser`.

Input

```
ser = pd.Series(np.take(list('abcdefgh'), np.random.randint(8, size=30)))
```

Show Solution

```
# Input
ser = pd.Series(np.take(list('abcdefgh'), np.random.randint(8, size=30)))

# Solution
ser.value_counts()
```

```
f    8
g    7
b    6
c    4
a    2
e    2
h    1
dtype: int64
```

10. How to keep only top 2 most frequent values as it is and replace everything else as 'Other'?

Difficulty Level: L2

From `ser`, keep the top 2 most frequent items as it is and replace everything else as 'Other'.

Input

```
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))
```

Show Solution

```
# Input
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))

# Solution
print("Top 2 Freq:", ser.value_counts())
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
ser
```

```
Top 2 Freq: 4    5
3    3
2    2
1    2
dtype: int64

0     Other
1     Other
2         3
3         4
4     Other
5         4
6         4
7         3
8         3
9         4
10        4
11    Other
dtype: object
```

11. How to bin a numeric series to 10 groups of equal size?

Difficulty Level: L2

Bin the series `ser` into 10 equal deciles and replace the values with the bin name.

Input

```
ser = pd.Series(np.random.random(20))
```

Desired Output

```
# First 5 items
0    7th
1    9th
2    7th
3    3rd
4    8th
dtype: category
Categories (10, object): [1st < 2nd < 3rd < 4th ... 7th < 8th < 9th < 10th]
```

Show Solution

```
# Input
ser = pd.Series(np.random.random(20))
print(ser.head())

# Solution
pd.qcut(ser, q=[0, .10, .20, .3, .4, .5, .6, .7, .8, .9, 1],
        labels=['1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th', '10t
h']).head()
```

```
0    0.556912
1    0.892955
2    0.566632
3    0.146656
4    0.881579
dtype: float64

0    7th
1    9th
2    7th
3    3rd
4    8th
dtype: category
Categories (10, object): [1st < 2nd < 3rd < 4th ... 7th < 8th < 9th < 10th]
```

Feedback

12. How to convert a numpy array to a dataframe of given shape? (L1)

Difficulty Level: L1

Reshape the series `ser` into a dataframe with 7 rows and 5 columns

Input

```
ser = pd.Series(np.random.randint(1, 10, 35))
```

Show Solution

```
# Input
ser = pd.Series(np.random.randint(1, 10, 35))

# Solution
df = pd.DataFrame(ser.values.reshape(7,5))
print(df)
```

```
   0  1  2  3  4
0  1  2  1  2  5
1  1  2  4  5  2
2  1  3  3  2  8
3  8  6  4  9  6
4  2  1  1  8  5
5  3  2  8  5  6
6  1  5  5  4  6
```

13. How to find the positions of numbers that are multiples of 3 from a series?

Difficulty Level: L2

Find the positions of numbers that are multiples of 3 from `ser`.

Input

```
ser = pd.Series(np.random.randint(1, 10, 7))
```

Show Solution

```
# Input
ser = pd.Series(np.random.randint(1, 10, 7))
ser

# Solution
print(ser)
np.argwhere(ser % 3==0)
```

```
0    6
1    8
2    6
3    7
4    6
5    2
6    4
dtype: int64

array([[0],
       [2],
       [4]])
```

14. How to extract items at given positions from a series

Difficulty Level: L1

From `ser`, extract the items at positions in list `pos`.

Input

```
ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))
pos = [0, 4, 8, 14, 20]
```

Show Solution

```
# Input
ser = pd.Series(list('abcdefghijklmnopqrstuvwxyz'))
pos = [0, 4, 8, 14, 20]

# Solution
ser.take(pos)
```

```
0     a
4     e
8     i
14    o
20    u
dtype: object
```

15. How to stack two series vertically and horizontally ?

Difficulty Level: L1

Stack ser1 and ser2 vertically and horizontally (to form a dataframe).

Input

```
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))
```

Show Solution

```
# Input
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))

# Output
# Vertical
ser1.append(ser2)

# Horizontal
df = pd.concat([ser1, ser2], axis=1)
print(df)
```

```
   0  1
0  0  a
1  1  b
2  2  c
3  3  d
4  4  e
```

16. How to get the positions of items of series A in another series B?

Difficulty Level: L2

Get the positions of items of `ser2` in `ser1` as a list.

Input

```
ser1 = pd.Series([10, 9, 6, 5, 3, 1, 12, 8, 13])
ser2 = pd.Series([1, 3, 10, 13])
```

Show Solution

```
# Input
ser1 = pd.Series([10, 9, 6, 5, 3, 1, 12, 8, 13])
ser2 = pd.Series([1, 3, 10, 13])

# Solution 1
[np.where(i == ser1)[0].tolist()[0] for i in ser2]

# Solution 2
[pd.Index(ser1).get_loc(i) for i in ser2]
```

```
[5, 4, 0, 8]
```

17. How to compute the mean squared error on a truth and predicted series?

Difficulty Level: L2

Compute the mean squared error of `truth` and `pred` series.

Input

```
truth = pd.Series(range(10))
pred = pd.Series(range(10)) + np.random.random(10)
```

Show Solution

```
# Input
truth = pd.Series(range(10))
pred = pd.Series(range(10)) + np.random.random(10)

# Solution
np.mean((truth-pred)**2)
```

```
0.28448128110629545
```

18. How to convert the first character of each element in a series to uppercase?

Difficulty Level: L2

Change the first character of each word to upper case in each word of `ser`.

```
ser = pd.Series(['how', 'to', 'kick', 'ass?'])
```

Show Solution

```
# Input
ser = pd.Series(['how', 'to', 'kick', 'ass?'])

# Solution 1
ser.map(lambda x: x.title())

# Solution 2
ser.map(lambda x: x[0].upper() + x[1:])

# Solution 3
pd.Series([i.title() for i in ser])
```

```
0     How
1      To
2    Kick
3    Ass?
dtype: object
```

19. How to calculate the number of characters in each word in a series?

Difficulty Level: L2

Input

```
ser = pd.Series(['how', 'to', 'kick', 'ass?'])
```

Show Solution

```
# Input
ser = pd.Series(['how', 'to', 'kick', 'ass?'])

# Solution
ser.map(lambda x: len(x))
```

```
0    3
1    2
2    4
3    4
dtype: int64
```

20. How to compute difference of differences between consequtive numbers of a series?

Difficulty Level: L1

Difference of differences between the consequtive numbers of `ser`.

Input

```
ser = pd.Series([1, 3, 6, 10, 15, 21, 27, 35])
```

Desired Output

```
[nan, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 8.0]
[nan, nan, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0]
```

Show Solution

```
# Input
ser = pd.Series([1, 3, 6, 10, 15, 21, 27, 35])

# Solution
print(ser.diff().tolist())
print(ser.diff().diff().tolist())
```

```
[nan, 2.0, 3.0, 4.0, 5.0, 6.0, 6.0, 8.0]
[nan, nan, 1.0, 1.0, 1.0, 1.0, 0.0, 2.0]
```

21. How to convert a series of date-strings to a timeseries?

Difficiulty Level: L2

Input

```
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04', '2014-05-05',
'2015-06-06T12:20'])
```

Desired Output

```
0   2010-01-01 00:00:00
1   2011-02-02 00:00:00
2   2012-03-03 00:00:00
3   2013-04-04 00:00:00
4   2014-05-05 00:00:00
5   2015-06-06 12:20:00
dtype: datetime64[ns]
```

Show Solution

```
# Input
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04', '2014-05-05'
, '2015-06-06T12:20'])

# Solution 1
from dateutil.parser import parse
ser.map(lambda x: parse(x))

# Solution 2
pd.to_datetime(ser)
```

```
0   2010-01-01 00:00:00
1   2011-02-02 00:00:00
2   2012-03-03 00:00:00
3   2013-04-04 00:00:00
4   2014-05-05 00:00:00
5   2015-06-06 12:20:00
dtype: datetime64[ns]
```

22. How to get the day of month, week number, day of year and day of week from a series of date strings?

Difficiulty Level: L2

Get the day of month, week number, day of year and day of week from `ser`.

Input

```
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04', '2014-05-05',
'2015-06-06T12:20'])
```

Desired output

```
Date:  [1, 2, 3, 4, 5, 6]
Week number:  [53, 5, 9, 14, 19, 23]
Day num of year:  [1, 33, 63, 94, 125, 157]
Day of week:  ['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Monday', 'Saturday']
```

Show Solution

```
# Input
ser = pd.Series(['01 Jan 2010', '02-02-2011', '20120303', '2013/04/04', '2014-05-05'
, '2015-06-06T12:20'])

# Solution
from dateutil.parser import parse
ser_ts = ser.map(lambda x: parse(x))

# day of month
print("Date: ", ser_ts.dt.day.tolist())

# week number
print("Week number: ", ser_ts.dt.weekofyear.tolist())

# day of year
print("Day number of year: ", ser_ts.dt.dayofyear.tolist())

# day of week
print("Day of week: ", ser_ts.dt.weekday_name.tolist())
```

```
Date:  [1, 2, 3, 4, 5, 6]
Week number:  [53, 5, 9, 14, 19, 23]
Day num of year:  [1, 33, 63, 94, 125, 157]
Day of week:  ['Friday', 'Wednesday', 'Saturday', 'Thursday', 'Monday', 'Saturday']
```

23. How to convert year-month string to dates corresponding to the 4th day of the month?

Difficiulty Level: L2

Change `ser` to dates that start with 4th of the respective months.

Input

```
ser = pd.Series(['Jan 2010', 'Feb 2011', 'Mar 2012'])
```

Desired Output

```
0   2010-01-04
1   2011-02-04
2   2012-03-04
dtype: datetime64[ns]
```

Show Solution

```
import pandas as pd
# Input
ser = pd.Series(['Jan 2010', 'Feb 2011', 'Mar 2012'])

# Solution 1
from dateutil.parser import parse
# Parse the date
ser_ts = ser.map(lambda x: parse(x))

# Construct date string with date as 4
ser_datestr = ser_ts.dt.year.astype('str') + '-' + ser_ts.dt.month.astype('str') +
'-' + '04'

# Format it.
[parse(i).strftime('%Y-%m-%d') for i in ser_datestr]

# Solution 2
ser.map(lambda x: parse('04 ' + x))
```

```
0   2010-01-04
1   2011-02-04
2   2012-03-04
dtype: datetime64[ns]
```

24. How to filter words that contain atleast 2 vowels from a series?

Difficiulty Level: L3

From `ser`, extract words that contain atleast 2 vowels.

Input

```
ser = pd.Series(['Apple', 'Orange', 'Plan', 'Python', 'Money'])
```

Desired Output

```
0     Apple
1    Orange
4     Money
dtype: object
```

Show Solution

```
# Input
ser = pd.Series(['Apple', 'Orange', 'Plan', 'Python', 'Money'])

# Solution
from collections import Counter
mask = ser.map(lambda x: sum([Counter(x.lower()).get(i, 0) for i in list('aeiou')])
>= 2)
ser[mask]
```

```
0     Apple
1    Orange
4     Money
dtype: object
```

25. How to filter valid emails from a series?

Difficiulty Level: L3

Extract the valid emails from the series `emails`. The regex pattern for valid emails is provided as reference.

Input

```
emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com', 'matt@t.co', 'n
arendra@modi.com'])
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
```

Desired Output

```
1    rameses@egypt.com
2            matt@t.co
3    narendra@modi.com
dtype: object
```

Show Solution

```
# Input
emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com', 'matt@t.co',
'narendra@modi.com'])

# Solution 1 (as series of strings)
import re
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
mask = emails.map(lambda x: bool(re.match(pattern, x)))
emails[mask]

# Solution 2 (as series of list)
emails.str.findall(pattern, flags=re.IGNORECASE)

# Solution 3 (as list)
[x[0] for x in [re.findall(pattern, email) for email in emails] if len(x) > 0]
```

```
['rameses@egypt.com', 'matt@t.co', 'narendra@modi.com']
```

26. How to get the mean of a series grouped by another series?

Difficiulty Level: L2

Compute the mean of `weights` of each `fruit`.

Input

```
fruit = pd.Series(np.random.choice(['apple', 'banana', 'carrot'], 10))
weights = pd.Series(np.linspace(1, 10, 10))
print(weight.tolist())
print(fruit.tolist())
#> [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
#> ['banana', 'carrot', 'apple', 'carrot', 'carrot', 'apple', 'banana', 'carrot', 'app
le', 'carrot']
```

Desired output

```
# values can change due to randomness
apple     6.0
banana    4.0
carrot    5.8
dtype: float64
```

Show Solution

```
# Input
fruit = pd.Series(np.random.choice(['apple', 'banana', 'carrot'], 10))
weights = pd.Series(np.linspace(1, 10, 10))

# Solution
weights.groupby(fruit).mean()
```

```
apple     7.4
banana    2.0
carrot    6.0
dtype: float64
```

27. How to compute the euclidean distance between two series?

Difficiulty Level: L2

Compute the euclidean distance [https://en.wikipedia.org/wiki/Euclidean_distance] between series (points) p and q, without using a packaged formula.

Input

```
p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])
```

Desired Output

```
18.165
```

Show Solution

```
# Input
p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])

# Solution
sum((p - q)**2)**.5

# Solution (using func)
np.linalg.norm(p-q)
```

```
18.165902124584949
```

28. How to find all the local maxima (or peaks) in a numeric series?

Difficiulty Level: L3

Get the positions of peaks (values surrounded by smaller values on both sides) in `ser`.

Input

```
ser = pd.Series([2, 10, 3, 4, 9, 10, 2, 7, 3])
```

Desired output

```
array([1, 5, 7])
```

Show Solution

```
# Input
ser = pd.Series([2, 10, 3, 4, 9, 10, 2, 7, 3])

# Solution
dd = np.diff(np.sign(np.diff(ser)))
peak_locs = np.where(dd == -2)[0] + 1
peak_locs
```

```
array([1, 5, 7])
```

29. How to replace missing spaces in a string with the least frequent character?

Replace the spaces in `my_str` with the least frequent character.

Difficiulty Level: L2

Input

```
my_str = 'dbc deb abed gade'
```

Desired Output

```
'dbccdebcabedcgade'  # least frequent is 'c'
```

Show Solution

```
# Input
my_str = 'dbc deb abed gade'

# Solution
ser = pd.Series(list('dbc deb abed gade'))
freq = ser.value_counts()
print(freq)
least_freq = freq.dropna().index[-1]
"".join(ser.replace(' ', least_freq))
```

```
d    4
b    3
e    3
     3
a    2
g    1
c    1
dtype: int64

'dbccdebcabedcgade'
```

30. How to create a TimeSeries starting '2000-01-01' and 10 weekends (saturdays) after that having random numbers as values?

Difficiulty Level: L2

Desired output

```
# values can be random
2000-01-01    4
2000-01-08    1
2000-01-15    8
2000-01-22    4
2000-01-29    4
2000-02-05    2
2000-02-12    4
2000-02-19    9
2000-02-26    6
2000-03-04    6
```

Show Solution

```
# Solution
ser = pd.Series(np.random.randint(1,10,10), pd.date_range('2000-01-01', periods=10,
 freq='W-SAT'))
ser
```

```
2000-01-01    6
2000-01-08    7
2000-01-15    4
2000-01-22    6
2000-01-29    8
2000-02-05    6
2000-02-12    5
2000-02-19    8
2000-02-26    1
2000-03-04    7
Freq: W-SAT, dtype: int64
```

31. How to fill an intermittent time series so all missing dates show up with values of previous non-missing date?

Difficiulty Level: L2

`ser` has missing dates and values. Make all missing dates appear and fill up with value from previous date.

Input

```
ser = pd.Series([1,10,3,np.nan], index=pd.to_datetime(['2000-01-01', '2000-01-03', '20
00-01-06', '2000-01-08']))
print(ser)
#> 2000-01-01     1.0
#> 2000-01-03    10.0
#> 2000-01-06     3.0
#> 2000-01-08     NaN
#> dtype: float64
```

Desired Output

```
2000-01-01     1.0
2000-01-02     1.0
2000-01-03    10.0
2000-01-04    10.0
2000-01-05    10.0
2000-01-06     3.0
2000-01-07     3.0
2000-01-08     NaN
```

Show Solution

```
# Input
ser = pd.Series([1,10,3, np.nan], index=pd.to_datetime(['2000-01-01', '2000-01-03',
'2000-01-06', '2000-01-08']))

# Solution
ser.resample('D').ffill()  # fill with previous value

# Alternatives
ser.resample('D').bfill()  # fill with next value
ser.resample('D').bfill().ffill()  # fill next else prev value
```

```
2000-01-01     1.0
2000-01-02    10.0
2000-01-03    10.0
2000-01-04     3.0
2000-01-05     3.0
2000-01-06     3.0
2000-01-07     3.0
2000-01-08     3.0
Freq: D, dtype: float64
```

Feedback

32. How to compute the autocorrelations of a numeric series?

Difficiulty Level: L3

Compute autocorrelations for the first 10 lags of `ser`. Find out which lag has the largest correlation.

Input

```
ser = pd.Series(np.arange(20) + np.random.normal(1, 10, 20))
```

Desired output

```
# values will change due to randomness
[0.29999999999999999, -0.11, -0.17000000000000001, 0.46000000000000002, 0.280000000000
00003, -0.040000000000000001, -0.37, 0.41999999999999998, 0.47999999999999998, 0.17999
999999999999]
Lag having highest correlation:  9
```

Show Solution

```
# Input
ser = pd.Series(np.arange(20) + np.random.normal(1, 10, 20))

# Solution
autocorrelations = [ser.autocorr(i).round(2) for i in range(11)]
print(autocorrelations[1:])
print('Lag having highest correlation: ', np.argmax(np.abs(autocorrelations[1:]))+1)
```

```
[0.29999999999999999, -0.11, -0.17000000000000001, 0.46000000000000002, 0.2800000000
0000003, -0.040000000000000001, -0.37, 0.41999999999999998, 0.47999999999999998, 0.1
7999999999999999]
Lag having highest correlation:  9
```

33. How to import only every nth row from a csv file to create a dataframe?

Difficiulty Level: L2

Import every 50th row of BostonHousing dataset [https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv] as a dataframe.

Show Solution

```
# Solution 1: Use chunks and for-loop
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHo
using.csv', chunksize=50)
df2 = pd.DataFrame()
for chunk in df:
    df2 = df2.append(chunk.iloc[0,:])


# Solution 2: Use chunks and list comprehension
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHo
using.csv', chunksize=50)
df2 = pd.concat([chunk.iloc[0] for chunk in df], axis=1)
df2 = df2.transpose()

# Solution 3: Use csv reader
import csv
with open('BostonHousing.csv', 'r') as f:
    reader = csv.reader(f)
    out = []
    for i, row in enumerate(reader):
        if i%50 == 0:
            out.append(row)

df2 = pd.DataFrame(out[1:], columns=out[0])
print(df2.head())
```

```
                  crim    zn  indus chas                   nox     rm   age  \
0              0.21977   0.0   6.91    0  0.44799999999999995  5.602  62.0
1               0.0686   0.0   2.89    0                0.445  7.416  62.5
2   2.7339700000000002   0.0  19.58    0                0.871  5.597  94.9
3               0.0315  95.0   1.47    0  0.40299999999999997  6.975  15.3
4  0.19072999999999998  22.0   5.86    0                0.431  6.718  17.5

      dis rad  tax ptratio       b  lstat  medv
0  6.0877   3  233    17.9   396.9   16.2  19.4
1  3.4952   2  276    18.0   396.9   6.19  33.2
2  1.5257   5  403    14.7  351.85  21.45  15.4
3  7.6534   3  402    17.0   396.9   4.56  34.9
4  7.8265   7  330    19.1  393.74   6.56  26.2
```

34. How to change column values when importing csv to a dataframe?

Difficulty Level: L2

Import the boston housing dataset (https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv), but while importing change the `'medv'` (median house value) column so that values < 25 becomes 'Low' and > 25 becomes 'High'.

Show Solution

```
# Solution 1: Using converter parameter
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHo
using.csv',
                 converters={'medv': lambda x: 'High' if float(x) > 25 else 'Low'})


# Solution 2: Using csv reader
import csv
with open('BostonHousing.csv', 'r') as f:
    reader = csv.reader(f)
    out = []
    for i, row in enumerate(reader):
        if i > 0:
            row[13] = 'High' if float(row[13]) > 25 else 'Low'
        out.append(row)

df = pd.DataFrame(out[1:], columns=out[0])
print(df.head())
```

```
                   crim    zn indus chas                   nox  \
0               0.00632  18.0  2.31    0   0.5379999999999999
1               0.02731   0.0  7.07    0                0.469
2               0.02729   0.0  7.07    0                0.469
3  0.032369999999999996   0.0  2.18    0  0.45799999999999996
4               0.06905   0.0  2.18    0  0.45799999999999996

                  rm   age     dis rad  tax ptratio       b lstat  medv
0              6.575  65.2    4.09   1  296    15.3   396.9  4.98   Low
1              6.421  78.9  4.9671   2  242    17.8   396.9  9.14   Low
2              7.185  61.1  4.9671   2  242    17.8  392.83  4.03  High
3  6.997999999999999  45.8  6.0622   3  222    18.7  394.63  2.94  High
4              7.147  54.2  6.0622   3  222    18.7   396.9  5.33  High
```

35. How to create a dataframe with rows as strides from a given series?

Difficiulty Level: L3

Input

```
L = pd.Series(range(15))
```

Desired Output

```
array([[ 0,  1,  2,  3],
       [ 2,  3,  4,  5],
       [ 4,  5,  6,  7],
       [ 6,  7,  8,  9],
       [ 8,  9, 10, 11],
       [10, 11, 12, 13]])
```

Show Solution

```
L = pd.Series(range(15))

def gen_strides(a, stride_len=5, window_len=5):
    n_strides = ((a.size-window_len)//stride_len) + 1
    return np.array([a[s:(s+window_len)] for s in np.arange(0, a.size, stride_len)[:
n_strides]])

gen_strides(L, stride_len=2, window_len=4)
```

```
array([[ 0,  1,  2,  3],
       [ 2,  3,  4,  5],
       [ 4,  5,  6,  7],
       [ 6,  7,  8,  9],
       [ 8,  9, 10, 11],
       [10, 11, 12, 13]])
```

36. How to import only specified columns from a csv file?

Difficulty Level: L1

Import 'crim' and 'medv' columns of the BostonHousing dataset (https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv) as a dataframe.

Show Solution

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv', usecols=['crim', 'medv'])
print(df.head())
```

```
      crim  medv
0  0.00632  24.0
1  0.02731  21.6
2  0.02729  34.7
3  0.03237  33.4
4  0.06905  36.2
```

37. How to get the n*rows,* ncolumns, datatype, summary stats of each column of a dataframe? Also get the array and list equivalent.

Difficulty Level: L2

Get the number of rows, columns, datatype and summary statistics of each column of the Cars93 (https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv) dataset. Also get the numpy array and list equivalent of the dataframe.

Show Solution

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

#  number of rows and columns
print(df.shape)

# datatypes
print(df.dtypes)

# how many columns under each dtype
print(df.get_dtype_counts())
print(df.dtypes.value_counts())

# summary statistics
df_stats = df.describe()

# numpy array
df_arr = df.values

# list
df_list = df.values.tolist()
```

```
(93, 27)
Manufacturer           object
Model                  object
Type                   object
Min.Price             float64
Price                 float64
Max.Price             float64
MPG.city              float64
MPG.highway           float64
AirBags                object
DriveTrain             object
Cylinders              object
EngineSize            float64
Horsepower            float64
RPM                   float64
Rev.per.mile          float64
Man.trans.avail        object
Fuel.tank.capacity    float64
Passengers            float64
Length                float64
Wheelbase             float64
Width                 float64
Turn.circle           float64
Rear.seat.room        float64
Luggage.room          float64
Weight                float64
Origin                 object
Make                   object
dtype: object
float64    18
object      9
dtype: int64
float64    18
object      9
dtype: int64
```

38. How to extract the row and column number of a particular cell with given criterion?

Difficulty Level: L1

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Which manufacturer, model and type has the highest `Price`? What is the row and column number of the cell with the highest `Price` value?

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
# Get Manufacturer with highest price
df.loc[df.Price == np.max(df.Price), ['Manufacturer', 'Model', 'Type']]

# Get Row and Column number
row, col = np.where(df.values == np.max(df.Price))

# Get the value
df.iat[row[0], col[0]]
df.iloc[row[0], col[0]]

# Alternates
df.at[row[0], 'Price']
df.get_value(row[0], 'Price')

# The difference between `iat` - `iloc` vs `at` - `loc` is:
# `iat` snd `iloc` accepts row and column numbers.
# Whereas `at` and `loc` accepts index and column names.
```

```
61.899999999999999
```

39. How to rename a specific columns in a dataframe?

Difficulty Level: L2

Rename the column `Type` as `CarType` in `df` and replace the '.' in column names with '_'.

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
print(df.columns)
#> Index(['Manufacturer', 'Model', 'Type', 'Min.Price', 'Price', 'Max.Price',
#>        'MPG.city', 'MPG.highway', 'AirBags', 'DriveTrain', 'Cylinders',
#>        'EngineSize', 'Horsepower', 'RPM', 'Rev.per.mile', 'Man.trans.avail',
#>        'Fuel.tank.capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
#>        'Turn.circle', 'Rear.seat.room', 'Luggage.room', 'Weight', 'Origin',
#>        'Make'],
#>       dtype='object')
```

Desired Solution

```
print(df.columns)
#> Index(['Manufacturer', 'Model', 'CarType', 'Min_Price', 'Price', 'Max_Price',
#>        'MPG_city', 'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders',
#>        'EngineSize', 'Horsepower', 'RPM', 'Rev_per_mile', 'Man_trans_avail',
#>        'Fuel_tank_capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
#>        'Turn_circle', 'Rear_seat_room', 'Luggage_room', 'Weight', 'Origin',
#>        'Make'],
#>       dtype='object')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
# Step 1:
df=df.rename(columns = {'Type':'CarType'})
# or
df.columns.values[2] = "CarType"

# Step 2:
df.columns = df.columns.map(lambda x: x.replace('.', '_'))
print(df.columns)
```

```
Index(['Manufacturer', 'Model', 'CarType', 'Min_Price', 'Price', 'Max_Price',
       'MPG_city', 'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders',
       'EngineSize', 'Horsepower', 'RPM', 'Rev_per_mile', 'Man_trans_avail',
       'Fuel_tank_capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
       'Turn_circle', 'Rear_seat_room', 'Luggage_room', 'Weight', 'Origin',
       'Make'],
      dtype='object')
```

40. How to check if a dataframe has any missing values?

Difficulty Level: L1

Check if `df` has any missing values.

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
df.isnull().values.any()
```

41. How to count the number of missing values in each column?

Difficulty Level: L2

Count the number of missing values in each column of `df`. Which column has the maximum number of missing values?

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
n_missings_each_col = df.apply(lambda x: x.isnull().sum())
n_missings_each_col.argmax()
```

```
'Luggage.room'
```

42. How to replace missing values of multiple numeric columns with the mean?

Difficulty Level: L2

Replace missing values in `Min.Price` and `Max.Price` columns with their respective mean.

Input

```
df = pd.read_csv('https://raz.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
df_out = df[['Min.Price', 'Max.Price']] = df[['Min.Price', 'Max.Price']].apply(lambda x: x.fillna(x.mean()))
print(df_out.head())
```

```
   Min.Price  Max.Price
0  12.900000  18.800000
1  29.200000  38.700000
2  25.900000  32.300000
3  17.118605  44.600000
4  17.118605  21.459091
```

Feedback

43. How to use apply function on existing columns with global variables as additional arguments?

Difficulty Level: L3

In `df`, use `apply` method to replace the missing values in `Min.Price` with the column's mean and those in `Max.Price` with the column's median.

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Use Hint from StackOverflow [https://stackoverflow.com/questions/32437435/passing-additional-arguments-to-python-pandas-dataframe-apply]

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
d = {'Min.Price': np.nanmean, 'Max.Price': np.nanmedian}
df[['Min.Price', 'Max.Price']] = df[['Min.Price', 'Max.Price']].apply(lambda x, d: x.fillna(d[x.name](x)), args=(d, ))
```

44. How to select a specific column from a dataframe as a dataframe instead of a series?

Difficulty Level: L2

Get the first column (`a`) in `df` as a dataframe (rather than as a Series).

Input

```
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))
```

Show Solution

```
# Input
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))

# Solution
type(df[['a']])
type(df.loc[:, ['a']])
type(df.iloc[:, [0]])

# Alternately the following returns a Series
type(df.a)
type(df['a'])
type(df.loc[:, 'a'])
type(df.iloc[:, 1])
```

```
pandas.core.series.Series
```

45. How to change the order of columns of a dataframe?

Difficulty Level: L3

Actually 3 questions.

1. In `df`, interchange columns `'a'` and `'c'`.
2. Create a generic function to interchange two columns, without hardcoding column names.
3. Sort the columns in reverse alphabetical order, that is colume `'e'` first through column `'a'` last.

Input

```
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))
```

Show Solution

```
# Input
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))

# Solution Q1
df[list('cbade')]

# Solution Q2 - No hard coding
def switch_columns(df, col1=None, col2=None):
    colnames = df.columns.tolist()
    i1, i2 = colnames.index(col1), colnames.index(col2)
    colnames[i2], colnames[i1] = colnames[i1], colnames[i2]
    return df[colnames]

df1 = switch_columns(df, 'a', 'c')

# Solution Q3
df[sorted(df.columns)]
# or
df.sort_index(axis=1, ascending=False, inplace=True)
```

46. How to set the number of rows and columns displayed in the output?

Difficulty Level: L2

Change the pamdas display settings on printing the dataframe `df` it shows a maximum of 10 rows and 10 columns.

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
pd.set_option('display.max_columns', 10)
pd.set_option('display.max_rows', 10)
# df

# Show all available options
# pd.describe_option()
```

47. How to format or suppress scientific notations in a pandas dataframe?

Difficulty Level: L2

Suppress scientific notations like 'e-03' in `df` and print upto 4 numbers after decimal.

Input

```
df = pd.DataFrame(np.random.random(4)**10, columns=['random'])
df
#>          random
#> 0  3.474280e-03
#> 1  3.951517e-05
#> 2  7.469702e-02
#> 3  5.541282e-28
```

Desired Output

```
#>    random
#> 0  0.0035
#> 1  0.0000
#> 2  0.0747
#> 3  0.0000
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.random(4)**10, columns=['random'])

# Solution 1: Rounding
df.round(4)

# Solution 2: Use apply to change format
df.apply(lambda x: '%.4f' % x, axis=1)
# or
df.applymap(lambda x: '%.4f' % x)

# Solution 3: Use set_option
pd.set_option('display.float_format', lambda x: '%.4f' % x)

# Solution 4: Assign display.float_format
pd.options.display.float_format = '{:.4f}'.format
print(df)

# Reset/undo float formatting
pd.options.display.float_format = None
```

```
   random
0  0.0002
1  0.5942
2  0.0000
3  0.0030
```

48. How to format all the values in a dataframe as percentages?

Difficulty Level: L2

Format the values in column `'random'` of `df` as percentages.

Input

```
df = pd.DataFrame(np.random.random(4), columns=['random'])
df
#>       random
#> 0    .689723
#> 1    .957224
#> 2    .159157
#> 3    .21082
```

Desired Output

```
#>      random
#> 0    68.97%
#> 1    95.72%
#> 2    15.91%
#> 3    2.10%
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.random(4), columns=['random'])

# Solution
out = df.style.format({
    'random': '{0:.2%}'.format,
})

out
```

	random
0	21.66%
1	44.90%
2	85.69%
3	92.12%

49. How to filter every nth row in a dataframe?

Difficulty Level: L1

From `df`, filter the `'Manufacturer'`, `'Model'` and `'Type'` for every 20th row starting from 1st (row 0).

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv')

# Solution
print(df.iloc[::20, :][['Manufacturer', 'Model', 'Type']])
```

```
   Manufacturer    Model     Type
0         Acura  Integra    Small
20     Chrysler  LeBaron  Compact
40        Honda  Prelude   Sporty
60      Mercury   Cougar  Midsize
80       Subaru   Loyale    Small
```

50. How to create a primary key index by combining relevant columns?

Difficulty Level: L2

In `df`, Replace `NaN`s with 'missing' in columns `'Manufacturer'`, `'Model'` and `'Type'` and create a index as a combination of these three columns and check if the index is a primary key.

Input

```
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv', usecols=[0,1,2,3,5])
```

Desired Output

```
                       Manufacturer    Model     Type  Min.Price  Max.Price
Acura_Integra_Small           Acura  Integra    Small       12.9       18.8
missing_Legend_Midsize      missing   Legend  Midsize       29.2       38.7
Audi_90_Compact               Audi       90  Compact       25.9       32.3
Audi_100_Midsize              Audi      100  Midsize        NaN       44.6
BMW_535i_Midsize               BMW     535i  Midsize        NaN        NaN
```

Show Solution

```
# Input
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv', usecols=[0,1,2,3,5])

# Solution
df[['Manufacturer', 'Model', 'Type']] = df[['Manufacturer', 'Model', 'Type']].fillna('missing')
df.index = df.Manufacturer + '_' + df.Model + '_' + df.Type
print(df.index.is_unique)
```

```
True
```

51. How to get the row number of the nth largest value in a column?

Difficulty Level: L2

Find the row position of the 5th largest value of column `'a'` in `df`.

Input

```
df = pd.DataFrame(np.random.randint(1, 30, 30).reshape(10,-1), columns=list('abc'))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1, 30, 30).reshape(10,-1), columns=list('abc'))

# Solution
n = 5
df['a'].argsort()[::-1][n]
```

```
    a   b   c
0  27   7  25
1   8   4  20
2   1   7  17
3  24   9  17
4  21  15   9
5  21  16  20
6  19  27  25
7  12   8  20
8  11  16  28
9  24  13   4


4
```

52. How to find the position of the nth largest value greater than a given value?

Difficulty Level: L2

In `ser`, find the position of the 2nd largest value greater than the mean.

Input

```
ser = pd.Series(np.random.randint(1, 100, 15))
```

Show Solution

```
# Input
ser = pd.Series(np.random.randint(1, 100, 15))

# Solution
print('ser: ', ser.tolist(), 'mean: ', round(ser.mean()))
np.argwhere(ser > ser.mean())[1]
```

```
ser:  [7, 77, 16, 86, 60, 38, 34, 36, 83, 27, 16, 52, 50, 52, 54] mean:  46


array([3])
```

53. How to get the last n rows of a dataframe with row sum > 100?

Difficulty Level: L2

Get the last two rows of `df` whose row sum is greater than 100.

```
df = pd.DataFrame(np.random.randint(10, 40, 60).reshape(-1, 4))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(10, 40, 60).reshape(-1, 4))

# Solution
# print row sums
rowsums = df.apply(np.sum, axis=1)

# last two rows with row sum greater than 100
last_two_rows = df.iloc[np.where(rowsums > 100)[0][-2:], :]
```

54. How to find and cap outliers from a series or dataframe column?

Difficulty Level: L2

Replace all values of `ser` in the lower 5%ile and greater than 95%ile with respective 5th and 95th %ile value.

Input

```
ser = pd.Series(np.logspace(-2, 2, 30))
```

Show Solution

```
# Input
ser = pd.Series(np.logspace(-2, 2, 30))

# Solution
def cap_outliers(ser, low_perc, high_perc):
    low, high = ser.quantile([low_perc, high_perc])
    print(low_perc, '%ile: ', low, '|', high_perc, '%ile: ', high)
    ser[ser < low] = low
    ser[ser > high] = high
    return(ser)

capped_ser = cap_outliers(ser, .05, .95)
```

```
0.05 %ile:  0.016049294077 | 0.95 %ile:  63.8766722202
```

55. How to reshape a dataframe to the largest possible square after removing the negative values?

Difficulty Level: L3

Reshape `df` to the largest possible square with negative values removed. Drop the smallest values if need be. The order of the positive numbers in the result should remain the same as the original.

Input

```
df = pd.DataFrame(np.random.randint(-20, 50, 100).reshape(10,-1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(-20, 50, 100).reshape(10,-1))
print(df)

# Solution
# Step 1: remove negative values from arr
arr = df[df > 0].values.flatten()
arr_qualified = arr[~np.isnan(arr)]

# Step 2: find side-length of largest possible square
n = int(np.floor(arr_qualified.shape[0]**.5))

# Step 3: Take top n^2 items without changing positions
top_indexes = np.argsort(arr_qualified)[::-1]
output = np.take(arr_qualified, sorted(top_indexes[:n**2])).reshape(n, -1)
print(output)
```

```
    0   1   2   3   4   5   6   7   8   9
0  25 -13  17  16   0   6  22  44  10 -19
1  47   4  -1  29 -13  12  41 -13  49  42
2  20 -20   9  16 -17  -1  37  39  41  37
3  27  44  -5   5   3 -12   0 -13  23  45
4   8  27  -8  -3  48 -16  -5  40  16  10
5  12  12  41 -12   3 -17  -3  27 -15  -1
6  -9  -3  41 -13   1   0  28  33  -2  18
7  18 -14  35   5   4  14   4  44  14  34
8   1  24  26  28 -10  17 -14  14  38  17
9  13  12   5   9 -16  -7  12 -18   1  24
[[ 25.  17.  16.   6.  22.  44.  10.  47.]
 [  4.  29.  12.  41.  49.  42.  20.   9.]
 [ 16.  37.  39.  41.  37.  27.  44.   5.]
 [  3.  23.  45.   8.  27.  48.  40.  16.]
 [ 10.  12.  12.  41.   3.  27.  41.  28.]
 [ 33.  18.  18.  35.   5.   4.  14.   4.]
 [ 44.  14.  34.  24.  26.  28.  17.  14.]
 [ 38.  17.  13.  12.   5.   9.  12.  24.]]
```

56. How to swap two rows of a dataframe?

Difficulty Level: L2

Swap rows 1 and 2 in `df`.

Input

```
df = pd.DataFrame(np.arange(25).reshape(5, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.arange(25).reshape(5, -1))

# Solution
def swap_rows(df, i1, i2):
    a, b = df.iloc[i1, :].copy(), df.iloc[i2, :].copy()
    df.iloc[i1, :], df.iloc[i2, :] = b, a
    return df

print(swap_rows(df, 1, 2))
```

```
    0   1   2   3   4
0   0   1   2   3   4
1  10  11  12  13  14
2   5   6   7   8   9
3  15  16  17  18  19
4  20  21  22  23  24
```

57. How to reverse the rows of a dataframe?

Difficulty Level: L2

Reverse all the rows of dataframe `df`.

Input

```
df = pd.DataFrame(np.arange(25).reshape(5, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.arange(25).reshape(5, -1))

# Solution 1
df.iloc[::-1, :]

# Solution 2
print(df.loc[df.index[::-1], :])
```

```
    0   1   2   3   4
4  20  21  22  23  24
3  15  16  17  18  19
2  10  11  12  13  14
1   5   6   7   8   9
0   0   1   2   3   4
```

58. How to create one-hot encodings of a categorical variable (dummy variables)?

Difficulty Level: L2

Get one-hot encodings for column `'a'` in the dataframe `df` and append it as columns.

Input

```
df = pd.DataFrame(np.arange(25).reshape(5,-1), columns=list('abcde'))
    a   b   c   d   e
0   0   1   2   3   4
1   5   6   7   8   9
2  10  11  12  13  14
3  15  16  17  18  19
4  20  21  22  23  24
```

Output

```
   0  5  10  15  20   b   c   d   e
0  1  0   0   0   0   1   2   3   4
1  0  1   0   0   0   6   7   8   9
2  0  0   1   0   0  11  12  13  14
3  0  0   0   1   0  16  17  18  19
4  0  0   0   0   1  21  22  23  24
```

Feedback

Show Solution

```
# Input
df = pd.DataFrame(np.arange(25).reshape(5,-1), columns=list('abcde'))

# Solution
df_onehot = pd.concat([pd.get_dummies(df['a']), df[list('bcde')]], axis=1)
print(df_onehot)
```

```
    a   b   c   d   e
0   0   1   2   3   4
1   5   6   7   8   9
2  10  11  12  13  14
3  15  16  17  18  19
4  20  21  22  23  24
   0  5  10  15  20   b   c   d   e
0  1  0   0   0   0   1   2   3   4
1  0  1   0   0   0   6   7   8   9
2  0  0   1   0   0  11  12  13  14
3  0  0   0   1   0  16  17  18  19
4  0  0   0   0   1  21  22  23  24
```

59. Which column contains the highest number of row-wise maximum values?

Difficulty Level: L2

Obtain the column name with the highest number of row-wise maximum's in `df`.

```
df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1))

# Solution
print('Column with highest row maxes: ', df.apply(np.argmax, axis=1).value_counts().
index[0])
```

```
Column with highest row maxes:  2
```

Feedback

60. How to create a new column that contains the row number of nearest column by euclidean distance?

Create a new column such that, each row contains the row number of nearest row-record by euclidean distance.

Difficulty Level: L3

Input

```
df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1), columns=list('pqrs'),
 index=list('abcdefghij'))
df
#     p   q   r   s
# a  57  77  13  62
# b  68   5  92  24
# c  74  40  18  37
# d  80  17  39  60
# e  93  48  85  33
# f  69  55   8  11
# g  39  23  88  53
# h  63  28  25  61
# i  18   4  73   7
# j  79  12  45  34
```

Desired Output

```
df
#    p   q   r   s nearest_row   dist
# a  57  77  13  62           i  116.0
# b  68   5  92  24           a  114.0
# c  74  40  18  37           i   91.0
# d  80  17  39  60           i   89.0
# e  93  48  85  33           i   92.0
# f  69  55   8  11           g  100.0
# g  39  23  88  53           f  100.0
# h  63  28  25  61           i   88.0
# i  18   4  73   7           a  116.0
# j  79  12  45  34           a   81.0
```

Show Solution

```
df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1), columns=list('pqrs'
), index=list('abcdefghij'))

# Solution
import numpy as np

# init outputs
nearest_rows = []
nearest_distance = []

# iterate rows.
for i, row in df.iterrows():
    curr = row
    rest = df.drop(i)
    e_dists = {}  # init dict to store euclidean dists for current row.
    # iterate rest of rows for current row
    for j, contestant in rest.iterrows():
        # compute euclidean dist and update e_dists
        e_dists.update({j: round(np.linalg.norm(curr.values - contestant.values))})
    # update nearest row to current row and the distance value
    nearest_rows.append(max(e_dists, key=e_dists.get))
    nearest_distance.append(max(e_dists.values()))

df['nearest_row'] = nearest_rows
df['dist'] = nearest_distance
```

61. How to know the maximum possible correlation value of each column against other columns?

Difficulty Level: L2

Compute maximum possible absolute correlation value of each column against other columns in `df`.

Input

```
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1), columns=list('pqrstuvwx
y'), index=list('abcdefgh'))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1), columns=list('pqrstuvwxy'), index=list('abcdefgh'))
df

# Solution
abs_corrmat = np.abs(df.corr())
max_corr = abs_corrmat.apply(lambda x: sorted(x)[-2])
print('Maximum Correlation possible for each column: ', np.round(max_corr.tolist(), 2))
```

```
Maximum Correlation possible for each column:  [ 0.91  0.57  0.55  0.71  0.53  0.26  0.91  0.71  0.69  0.71]
```

62. How to create a column containing the minimum by maximum of each row?

Difficulty Level: L2

Compute the minimum-by-maximum for every row of `df`.

```
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))

# Solution 1
min_by_max = df.apply(lambda x: np.min(x)/np.max(x), axis=1)

# Solution 2
min_by_max = np.min(df, axis=1)/np.max(df, axis=1)
```

63. How to create a column that contains the penultimate value in each row?

Difficulty Level: L2

Create a new column `'penultimate'` which has the second largest value of each row of `df`.

Input

```
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))

# Solution
out = df.apply(lambda x: x.sort_values().unique()[-2], axis=1)
df['penultimate'] = out
print(df)
```

```
    0   1   2   3   4   5   6   7   8   9  penultimate
0  52  69  62   7  20  69  38  10  57  17           62
1  52  94  49  63   1  90  14  76  20  84           90
2  78  37  58   7  27  41  27  26  48  51           58
3   6  39  99  36  62  90  47  25  60  84           90
4  37  36  91  93  76  69  86  95  69   6           93
5   5  54  73  61  22  29  99  27  46  24           73
6  71  65  45   9  63  46   4  93  36  18           71
7  85   7  76  46  65  97  64  52  28  80           85
```

64. How to normalize all columns in a dataframe?

Difficulty Level: L2

1. Normalize all columns of `df` by subtracting the column mean and divide by standard deviation.
2. Range all columns of `df` such that the minimum value in each column is 0 and max is 1.

Don't use external packages like sklearn.

Input

```
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))

# Solution Q1
out1 = df.apply(lambda x: ((x - x.mean())/x.std()).round(2))
print('Solution Q1\n',out1)

# Solution Q2
out2 = df.apply(lambda x: ((x.max() - x)/(x.max() - x.min())).round(2))
print('Solution Q2\n', out2)
```

```
Solution Q1
       0     1     2     3     4     5     6     7     8     9
0   1.09  0.64 -0.33 -0.96 -1.30  0.06  0.38  1.18 -1.60  1.66
1  -0.93 -2.36  0.87  1.47 -1.15  1.27  0.07 -0.87 -0.18  0.23
2   1.53  0.48 -0.90  0.18 -0.33  0.81 -1.29  0.34  0.06 -0.55
3   0.59 -0.24 -1.06  0.61  1.18 -1.23 -0.53 -0.45  0.34 -1.25
4   0.18  0.33  1.07  1.17  0.50 -0.26 -0.25 -1.45  1.11  1.11
5  -1.16  0.64 -0.93 -0.59 -0.15  0.63  1.02  1.13  1.20 -0.19
6  -0.58  0.07 -0.20 -0.87 -0.22 -1.62 -1.04  0.81 -1.23 -1.04
7  -0.73  0.45  1.47 -1.02  1.47  0.34  1.65 -0.71  0.31  0.02
Solution Q2
       0     1     2     3     4     5     6     7     8     9
0   0.16  0.00  0.71  0.98  1.00  0.42  0.43  0.00  1.00  0.00
1   0.91  1.00  0.24  0.00  0.95  0.00  0.54  0.78  0.49  0.49
2   0.00  0.05  0.93  0.52  0.65  0.16  1.00  0.32  0.41  0.76
3   0.35  0.29  1.00  0.35  0.10  0.86  0.74  0.62  0.31  1.00
4   0.50  0.10  0.16  0.12  0.35  0.53  0.65  1.00  0.03  0.19
5   1.00  0.00  0.95  0.83  0.58  0.22  0.22  0.02  0.00  0.64
6   0.78  0.19  0.66  0.94  0.61  1.00  0.91  0.14  0.87  0.93
7   0.84  0.06  0.00  1.00  0.00  0.32  0.00  0.72  0.32  0.56
```

65. How to compute the correlation of each row with the suceeding row?

Difficulty Level: L2

Compute the correlation of each row of `df` with its succeeding row.

Input

```
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))

# Solution
[df.iloc[i].corr(df.iloc[i+1]).round(2) for i in range(df.shape[0])[:-1]]
```

```
    0   1   2   3   4   5   6   7   8   9
0  93  49  26   2  96  56  11  73  90  65
1  54  17  47  52  65   9  21  87  94   4
2  51  11  44  77  37  57  17  25  95  26
3  84   8  61  43  63  63  59  65  69  29
4   8  27  53  95  10  35  16  61  39  83
5  30  70  91  26  12  44  37  71  21  48
6  66  44  47  44  29  99  86  78  31   1
7  17  40  28  12  89  95  79  54  81  47

[0.40999999999999998,
 0.47999999999999998,
 0.42999999999999999,
 -0.37,
 0.23000000000000001,
 0.14000000000000001,
 0.22]
```

66. How to replace both the diagonals of dataframe with 0?

Difficulty Level: L2

Replace both values in both diagonals of `df` with 0.

Input

```
df = pd.DataFrame(np.random.randint(1,100, 100).reshape(10, -1))
df
#     0   1   2   3   4   5   6   7   8   9
# 0  11  46  26  44  11  62  18  70  68  26
# 1  87  71  52  50  81  43  83  39   3  59
# 2  47  76  93  77  73   2   2  16  14  26
# 3  64  18  74  22  16  37  60   8  66  39
# 4  10  18  39  98  25   8  32   6   3  29
# 5  29  91  27  86  23  84  28  31  97  10
# 6  37  71  70  65   4  72  82  89  12  97
# 7  65  22  97  75  17  10  43  78  12  77
# 8  47  57  96  55  17  83  61  85  26  86
# 9  76  80  28  45  77  12  67  80   7  63
```

Desired output

```
#     0   1   2   3   4   5   6   7   8   9
# 0   0  46  26  44  11  62  18  70  68   0
# 1  87   0  52  50  81  43  83  39   0  59
# 2  47  76   0  77  73   2   2   0  14  26
# 3  64  18  74   0  16  37   0   8  66  39
# 4  10  18  39  98   0   0  32   6   3  29
# 5  29  91  27  86   0   0  28  31  97  10
# 6  37  71  70   0   4  72   0  89  12  97
# 7  65  22   0  75  17  10  43   0  12  77
# 8  47   0  96  55  17  83  61  85   0  86
# 9   0  80  28  45  77  12  67  80   7   0
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1,100, 100).reshape(10, -1))

# Solution
for i in range(df.shape[0]):
    df.iat[i, i] = 0
    df.iat[df.shape[0]-i-1, i] = 0
```

67. How to get the particular group of a groupby dataframe by key?

Difficulty Level: L2

This is a question related to understanding of grouped dataframe. From `df_grouped`, get the group belonging to `'apple'` as a dataframe.

Input

```
df = pd.DataFrame({'col1': ['apple', 'banana', 'orange'] * 3,
                   'col2': np.random.rand(9),
                   'col3': np.random.randint(0, 15, 9)})

df_grouped = df.groupby(['col1'])
```

```
# Input
df = pd.DataFrame({'col1': ['apple', 'banana', 'orange'] * 3,
                   'col2': np.random.rand(9),
                   'col3': np.random.randint(0, 15, 9)})

df_grouped = df.groupby(['col1'])

# Solution 1
df_grouped.get_group('apple')

# Solution 2
for i, dff in df_grouped:
    if i == 'apple':
        print(dff)
```

```
    col1      col2  col3
0  apple  0.673434     7
3  apple  0.182348    14
6  apple  0.050457     3
```

[/expand]

68. How to get the n'th largest value of a column when grouped by another column?

Difficulty Level: L2

In `df`, find the second largest value of `'taste'` for `'banana'`

Input

```
df = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                   'rating': np.random.rand(9),
                   'price': np.random.randint(0, 15, 9)})
```

```
```

Show Solution

```
# Input
df = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                   'taste': np.random.rand(9),
                   'price': np.random.randint(0, 15, 9)})

print(df)

# Solution
df_grpd = df['taste'].groupby(df.fruit)
df_grpd.get_group('banana').sort_values().iloc[-2]
```

```
    fruit  price     taste
0   apple      7  0.190229
1  banana      2  0.438063
2  orange      1  0.860182
3   apple      6  0.042149
4  banana      2  0.896021
5  orange      5  0.255107
6   apple      6  0.874533
7  banana      4  0.696274
8  orange      9  0.140713

0.69627423645996078
```

69. How to compute grouped mean on pandas dataframe and keep the grouped column as another column (not index)?

Difficulty Level: L1

In `df`, Compute the mean `price` of every `fruit`, while keeping the `fruit` as another column instead of an index.

Input

```
df = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                   'rating': np.random.rand(9),
                   'price': np.random.randint(0, 15, 9)})
```

```

```

Show Solution

```
# Input
df = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                   'rating': np.random.rand(9),
                   'price': np.random.randint(0, 15, 9)})

# Solution
out = df.groupby('fruit', as_index=False)['price'].mean()
print(out)
```

```
    fruit      price
0   apple  11.000000
1  banana   6.333333
2  orange   6.333333
```

70. How to join two dataframes by 2 columns so they have only the common rows?

Difficulty Level: L2

Join dataframes `df1` and `df2` by 'fruit-pazham' and 'weight-kilo'.

Input

```
df1 = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                    'weight': ['high', 'medium', 'low'] * 3,
                    'price': np.random.randint(0, 15, 9)})

df2 = pd.DataFrame({'pazham': ['apple', 'orange', 'pine'] * 2,
                    'kilo': ['high', 'low'] * 3,
                    'price': np.random.randint(0, 15, 6)})
```

Show Solution

```
# Input
df1 = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                    'weight': ['high', 'medium', 'low'] * 3,
                    'price': np.random.randint(0, 15, 9)})

df2 = pd.DataFrame({'pazham': ['apple', 'orange', 'pine'] * 2,
                    'kilo': ['high', 'low'] * 3,
                    'price': np.random.randint(0, 15, 6)})

# Solution
pd.merge(df1, df2, how='inner', left_on=['fruit', 'weight'], right_on=['pazham', 'po
unds'], suffixes=['_left', '_right'])
```

	fruit	price_left	weight	pazham	pounds	price_right
0	apple	5	high	apple	high	11
1	apple	10	high	apple	high	11
2	apple	8	high	apple	high	11
3	orange	6	low	orange	low	6
4	orange	7	low	orange	low	6
5	orange	0	low	orange	low	6

71. How to remove rows from a dataframe that are present in another dataframe?

Difficulty Level: L3

From `df1`, remove the rows that are present in `df2`. All three columns must be the same.

Input

```
df1 = pd.DataFrame({'fruit': ['apple', 'banana', 'orange'] * 3,
                    'weight': ['high', 'medium', 'low'] * 3,
                    'price': np.random.randint(0, 15, 9)})

df2 = pd.DataFrame({'pazham': ['apple', 'orange', 'pine'] * 2,
                    'kilo': ['high', 'low'] * 3,
                    'price': np.random.randint(0, 15, 6)})
```

Show Solution

```
# Input
df1 = pd.DataFrame({'fruit': ['apple', 'orange', 'banana'] * 3,
                    'weight': ['high', 'medium', 'low'] * 3,
                    'price': np.arange(9)})

df2 = pd.DataFrame({'fruit': ['apple', 'orange', 'pine'] * 2,
                    'weight': ['high', 'medium'] * 3,
                    'price': np.arange(6)})



# Solution
print(df1[~df1.isin(df2).all(1)])
```

```
    fruit  price  weight
2  banana      2     low
3   apple      3    high
4  orange      4  medium
5  banana      5     low
6   apple      6    high
7  orange      7  medium
8  banana      8     low
```

72. How to get the positions where values of two columns match?

Difficulty Level: L2

Show Solution

```
# Input
df = pd.DataFrame({'fruit1': np.random.choice(['apple', 'orange', 'banana'], 10),
                    'fruit2': np.random.choice(['apple', 'orange', 'banana'], 10)})

# Solution
np.where(df.fruit1 == df.fruit2)
```

```
(array([1, 5, 9]),)
```

73. How to create lags and leads of a column in a dataframe?

Difficulty Level: L2

Create two new columns in `df`, one of which is a lag1 (shift column `a` down by 1 row) of column 'a' and the other is a lead1 (shift column `b` up by 1 row).

Input

```
df = pd.DataFrame(np.random.randint(1, 100, 20).reshape(-1, 4), columns = list('abcd'
))
```

```
    a   b   c   d
0  66  34  76  47
1  20  86  10  81
2  75  73  51  28
3   1   1   9  83
4  30  47  67   4
```

Desired Output

```
    a   b   c   d  a_lag1  b_lead1
0  66  34  76  47     NaN     86.0
1  20  86  10  81    66.0     73.0
2  75  73  51  28    20.0      1.0
3   1   1   9  83    75.0     47.0
4  30  47  67   4     1.0      NaN
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1, 100, 20).reshape(-1, 4), columns = list('abc
d'))

# Solution
df['a_lag1'] = df['a'].shift(1)
df['b_lead1'] = df['b'].shift(-1)
print(df)
```

```
    a   b   c   d  a_lag1  b_lead1
0  29  90  43  24     NaN     36.0
1  94  36  67  66    29.0     76.0
2  81  76  44  49    94.0     97.0
3  55  97  10  74    81.0     43.0
4  32  43  62  62    55.0      NaN
```

74. How to get the frequency of unique values in the entire dataframe?

Difficulty Level: L2

Get the frequency of unique values in the entire dataframe `df`.

Input

```
df = pd.DataFrame(np.random.randint(1, 10, 20).reshape(-1, 4), columns = list('abcd'))
```

Show Solution

```
# Input
df = pd.DataFrame(np.random.randint(1, 10, 20).reshape(-1, 4), columns = list('abcd'
))

# Solution
pd.value_counts(df.values.ravel())
```

```
5    4
4    4
9    3
8    2
7    2
3    2
6    1
2    1
1    1
dtype: int64
```

75. How to split a text column into two separate columns?

Difficulty Level: L2

Split the string column in `df` to form a dataframe with 3 columns as shown.

Input

```
df = pd.DataFrame(["STD, City    State",
"33, Kolkata    West Bengal",
"44, Chennai    Tamil Nadu",
"40, Hyderabad    Telengana",
"80, Bangalore    Karnataka"], columns=['row'])

print(df)
#>                          row
#> 0          STD, City\tState
#> 1  33, Kolkata\tWest Bengal
#> 2   44, Chennai\tTamil Nadu
#> 3  40, Hyderabad\tTelengana
#> 4  80, Bangalore\tKarnataka
```

Desired Output

```
0 STD        City        State
1  33     Kolkata  West Bengal
2  44     Chennai   Tamil Nadu
3  40   Hyderabad    Telengana
4  80   Bangalore    Karnataka
```

Show Solution

```
# Input
df = pd.DataFrame(["STD, City    State",
"33, Kolkata    West Bengal",
"44, Chennai    Tamil Nadu",
"40, Hyderabad    Telengana",
"80, Bangalore    Karnataka"], columns=['row'])

# Solution
df_out = df.row.str.split(',|\t', expand=True)

# Make first row as header
new_header = df_out.iloc[0]
df_out = df_out[1:]
df_out.columns = new_header
print(df_out)
```

```
0 STD        City        State
1  33     Kolkata  West Bengal
2  44     Chennai   Tamil Nadu
3  40   Hyderabad    Telengana
4  80   Bangalore    Karnataka
```

To be continued . .

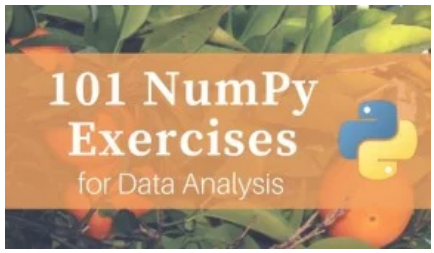
Related

[https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/]

Numpy Tutorial Part 2 - Vital Functions for Data Analysis [https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/]

February 14, 2018

In "Python"



[https://www.machinelearningplus.com/python/101-numpy-exercises-python/]

101 NumPy Exercises for Data Analysis (Python) [https://www.machinelearningplus.com/python/101-numpy-exercises-python/]

February 26, 2018

In "Python"

[https://www.machinelearningplus.com/python/101-numpy-tutorial-part1-array-python-examples/]

Numpy Tutorial Part 1 - Introduction to Arrays [https://www.machinelearningplus.com/python/101-numpy-tutorial-part1-array-python-examples/]

February 7, 2018

In "Python"

You may also like:

99_matplotlib_structure

NumPy Tutorial - A Complete Step-by-Step Guide with clear Examples

Portfolio Archive

02_distribution_of_doc min

Feedback

Vector Autoregression (VAR) - Comprehensive Guide with Examples in Python

22_Density_Plot_Matplmin

Statistical Significance Tests - Examples and How to find P Value?

Python debugging with pdb

Tags:Data_Manipulation [https://www.machinelearningplus.com/tag/data-manipulation/], Pandas [https://www.machinelearningplus.com/tag/pandas/], Python [https://www.machinelearningplus.com/tag/python/]

Andrzej • 5 months ago • edited

I just love how those problems challenge you to use python tools properly. Best way to get comfortable in pandas imo. Thank you <3

Alternate to #33

```
df = pd.read_csv('BostonHousing.csv', skiprows=lambda x: x%50!=0)
```

1 • Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #75 (A little slower, but in making it I learned about the n parameter for split, which I definitely will use again!)

```
(df["row"].iloc[1:]
.str.replace(",", "")
.str.split(expand=True, n=2)
.rename(columns= dict(zip(range(3),
df.iloc[0,0].replace(",", "").split())))
```

1 • Reply • Share ›

Arthur • a year ago

Hi! Thanks a lot for the exercises.
I created a jupyter notebook with all the questions without answers, if anybody is interested:

https://github.com/Mytakeon...

1 • Reply • Share ›

Richard Croft • 2 months ago

Question 71 reminds me of platform nine and three quarters...

• Reply • Share ›

Richard Croft • 2 months ago

Excellent set of question, thanks v much

• Reply • Share ›

Surya Teja Parnampedu • 4 months ago • edited

Alternate for #24:
ser[ser.str.count(pat=r'[aeiou]', flags=re.I) >= 2]

Alternate for #25:
emails[emails.str.match(pat=r"[A-z0-9._%+-]+@[A-z0-9.-]+\.[A-z]{2,4}")]

Alternate for #42:

```
df.fillna({
'Min_Price': df.Min_Price.mean(),
'Max_Price': df.Max_Price.mean()
})
```

• Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 25

```
emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com', 'matt@t.co',
'narendra@modi.com'])


pattern = '([A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4})'
print(emails.str.extract(pattern, flags=re.I))
```

```
0
0 NaN
1 rameses@egypt.com
2 matt@t.co
3 narendra@modi.com
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 24

probably slow, but easy.

```
ser[ser.apply(lambda x: sum(map(x.lower().count,'aeiou'))) >= 2]
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 23

```
pd.to_datetime("04 " + ser)
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 10

using value_counts (EASIER)

```
%%timeit

np.random.seed(100)
ser = pd.Series(np.random.randint(1, 5, [12]))

idx = ser.value_counts().head(2).index
ser[~ser.isin(idx)] = 'Other'
ser
1.5 ms ± 18.4 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

using counter (FASTER)

from collections import Counter

```
%%timeit
np.random.seed(100)
ser = pd.Series(np.random.randint(1, 5, [12]))

top2 = Counter(ser.values).most_common(2)
idx = [i[0] for i in top2]

ser[~ser.isin(idx)] = 'Other'
ser
1.11 ms ± 10.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 9

Easiest option is of course the given solution value_counts().

However, we can also do this using numpy.

```
np.random.seed(100)
```

```
np.random.seed(100)
ser = pd.Series([np.random.choice(list('abcdef')) for _ in range(30)])
ser.value_counts()
c 7
a 6
e 5
d 5
f 4
b 3
dtype: int64

# using numpy
u,c = np.unique(ser.values, return_counts= True)
np.array([u,c]).T
array([['a', 6],
['b', 3],
['c', 7],
['d', 5],
['e', 5],
['f', 4]], dtype=object)
```

Reply • Share ›

Bhishan Poudel • 4 months ago

**Qn **

```
ser.describe()
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn 7

```
s = pd.Series(np.setxor1d(ser1.values, ser2.values))
```

Reply • Share ›

Bhishan Poudel • 4 months ago • edited

Qn 6

```
%%timeit
ser1[~ser1.isin(ser2)]
449 µs ± 5.01 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

%%timeit
s = pd.Series(np.setdiff1d(ser1.values, ser2.values))
87.2 µs ± 309 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

Reply • Share ›

Bhishan Poudel • 4 months ago • edited

Qn 54

```
capped_ser = np.clip(ser, *np.percentile(ser,[5,95]) )
```

Reply • Share ›

Bhishan Poudel • 4 months ago

Qn:66 Using numpy rather than for-loop

```
v = df.values.copy()
```

Feedback

```
np.fill_diagonal(v,0)
v = np.rot90(v)
np.fill_diagonal(v,0)

df = pd.DataFrame(v)
df
```

Reply • Share ›

Bhishan Poudel • 4 months ago • edited

Question 75 Can also be done like this:

```
columns = df.iloc[0].str.replace('\s\s+',',').str.split(',').values
df.row.iloc[1:].str\
.replace('\s\s+',',')\
.str.split(',',expand=True)\
.rename(columns=lambda x: columns[0][x])
```

Reply • Share ›

Andrzej • 4 months ago • edited

I believe solution for 51 doesn't work.

it should be

```
df['a'].values.argsort()[::-1][n]
```

because if you reverse a Series you reverse its index as well, this way after reversing you index the same element.

Reply • Share ›

Anshuman Jadhav • 8 months ago

Alternate for 28

ser = pd.Series([2, 10, 3, 4, 9, 10, 2, 7, 3])
ser_n = ser.as_matrix()
import scipy.signal
indexes = scipy.signal.argrelmax(ser_n)
indexes

Reply • Share ›

rcodeprogramming ➚ Anshuman Jadhav • 8 months ago

Good one!

Reply • Share ›

Oleg Lokshyn • 9 months ago • edited

Alternate for #6:
np.setdiff1d(ser1, ser2)

Alternate for #7:
np.setxor1d(ser1, ser2)

Alternate for #8:
ser.describe()

BTW: expression np.random.RandomState(100) is useless: the object should be assigned to a variable and normal() should be called as it`s method:

state = np.random.RandomState(100)
state.normal(10, 5, 25)

Alternate for #16:
np.argwhere(ser1.isin(ser2))

Alternate for #18:
ser.apply(str.capitalize)

Alternate for #19:
ser.apply(len)

Alternate for #23:
from datetime import timedelta
pd.to_datetime(ser) + timedelta(days=3)

Alternate for #24:
ser[ser.apply(lambda x: len(set(x.lower()) & set('aeiou')) >= 2)]

• Reply • Share ›

rcodeprogramming ➚ Oleg Lokshyn • 8 months ago

Nice! Keep going Oleg.

• Reply • Share ›

Selva Prabhakaran Mod ➚ Oleg Lokshyn • 9 months ago

Nice work Oleg! Thanks for finding the RandomState(100) problem, I got it mixed up with how random seed works in R and Python.

• Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #66:

import itertools
from itertools import chain

for x, y in chain.from_iterable([
zip(range(10), range(10)),
zip(range(10), range(9, 0, -1))]):
df.iat[x, y] = 0

• Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #60:

from scipy.spatial.distance import pdist, squareform

dist = pdist(df, 'euclidean')
df_dist.apply(lambda x: x.idxmax(), axis=0)

(df
.assign(nearest_row = df_dist.apply(lambda x: x.idxmax(), axis=0))
.assign(dist = df_dist.apply(lambda x: x.max(), axis=0))
)

• Reply • Share ›

Feedback

Matthew Alhonte ➚ Matthew Alhonte • a year ago

Whoops. This one, rather:

```
from scipy.spatial.distance import pdist, squareform

dist = pdist(df, 'euclidean')

df_dist = pd.DataFrame(squareform(dist),
index=list('abcdefghij'),
columns=list('abcdefghij'))

(df
.assign(nearest_row = df_dist.apply(lambda x: x.idxmax(), axis=0))
.assign(dist = df_dist.apply(lambda x: x.max(), axis=0))
)
```

Reply • Share ›

Selva Prabhakaran • a year ago

Nice

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #55:

```
def getNextBestSquareArray(df):

arr = df.values.ravel()
onlyPos = np.extract(arr > 0, arr)

targetSideLen = int(np.floor(onlyPos.shape[0]**.5))
targetLen = targetSideLen**2

lenDiff = onlyPos.shape[0] - targetLen

#Using argpartition means we don't have to sort the entire array
clippedArray = np.delete(onlyPos,
np.argpartition(onlyPos, lenDiff-1)[:lenDiff])

return clippedArray.reshape(targetSideLen,
targetSideLen)
```

With %%timeit, got 91.8 µs ± 1.5 µs (the official solution had 2.96 ms ± 408 µs)

Reply • Share ›

Todd • a year ago

Loved the exercises thanks for sharing!!

Reply • Share ›

Selva Prabhakaran ➚ Todd • a year ago

You're welcome :)

Reply • Share ›

Feedback

Matthew Alhonte • a year ago

Alternate for #54:

fifth, ninetyfifth = ser.quantile([.05, .95])

ser.map(lambda x: (fifth if x ninetyfifth
else x))

• Reply • Share ›

Matthew Alhonte • a year ago

absolutely LOVE these exercise, by the way! :)

• Reply • Share ›

Selva Prabhakaran ➚ Matthew Alhonte • a year ago

Thanks Matt :)

• Reply • Share ›

Matthew Alhonte • a year ago

Alternate for 53 (very slight changes, mostly just using Pandas versions of NumPy functions and slightly terser indexing):

df[df.apply(lambda x: x.sum() > 100, axis=1)][-2:]

• Reply • Share ›

Matthew Alhonte • a year ago

I think #51 has a mistake. Shouldn't it be sort_values() instead of argsort()? (also, I think the index should be 4 instead of 5, since 0 would refer to the smallest):

df["a"].sort_values(ascending=False).index[4]

vs the official solution:
n = 5
df['a'].argsort()[::-1][n]

Argsort gives you the indices that would sort the series, but if you then select the 5th element of that series, it gives you the index you should give to the 5th element of the original series in order to sort it. But the 5th element of sort_values will give you the 5th largest value.

Similarly, I think #52 should be this:

ser[ser>ser.mean()].sort_values(ascending=False).index[1]

argwhere() gives you the indices where the predicate is true - and if you try to use index[5]. at least on mine, it throws an error because the the value at 5 was filtered out.

• Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #50 (I like method chaining :)):

(df
.fillna("missing")
.set_index(df.apply(lambda x:
"_".join(x[['Manufacturer', 'Model','Type']].astype(str)),
axis=1))

)

Reply • Share ›

hrishabh • a year ago

can anyone tell me how do we get most frequent value of a column and the coulmn name itself.
i did like this but didnt help
x=df['fruits'].value_counts.index.tolist()
return str(x[0])

Reply • Share ›

Jeff • a year ago

Thank you so much for the Alternates. Added to my growing fund of pandas...

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #33:

```
f = 'BostonHousing.csv'
n = 50
num_lines = sum(1 for _ in open(f))

skip_idx = [x for x in range(1, num_lines) if x % n !=0]

df = pd.read_csv(f, skiprows=skip_idx)
```

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #29:

```
ser = pd.Series(list(my_str))
lstUsed = ser[ser!=" "].value_counts().index[-1]
my_str.replace(" ", lstUsed)
```

Reply • Share ›

Jeff • a year ago

Excellent collection. Thank you so much. Impatiently Waiting for the remaining exercises...

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for 16:
ser1[ser1.isin(ser2)].index

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #13:
ser[ser % 3 == 0].index

Reply • Share ›

Matthew Alhonte • a year ago

Alternate for #10:

`ser.replace({x:"Other" for x in ser.value_counts()[2:].index})`

• Reply • Share ›

Selva Prabhakaran ➚ Matthew Alhonte • a year ago

Nice

• Reply • Share ›

Ankush Chandna • a year ago

This is awesome!

• Reply • Share ›

Paul • a year ago

heads up, error in the solution for #70. Solution should be:
pd.merge(df1, df2, how='inner', left_on=['fruit', 'weight'], right_on=['pazham', 'kilo'], suffixes=['_left', '_right'])

Search

Recent Posts

Vector Autoregression (VAR) – Comprehensive Guide with Examples in Python (https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/)

Mahalonobis Distance – Understanding the math with examples (python) (https://www.machinelearningplus.com/statistics/mahalanobis-distance/)

datetime in Python – Simplified Guide with Clear Examples (https://www.machinelearningplus.com/python/datetime-python-examples/)

Python Logging – Simplest Guide with Full Code and Examples (https://www.machinelearningplus.com/python/python-logging-guide/)

Matplotlib Histogram – How to Visualize Distributions in Python (https://www.machinelearningplus.com/plots/matplotlib-histogram-python-examples/)

ARIMA Model – Complete Guide to Time Series Forecasting in Python (https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)

Time Series Analysis in Python – A Comprehensive Guide with Examples (https://www.machinelearningplus.com/time-series/time-series-analysis-python/)

Matplotlib Tutorial – A Complete Guide to Python Plot w/ Examples (https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/)

Topic modeling visualization – How to present the results of LDA models? (https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/)

Top 50 matplotlib Visualizations – The Master Plots (with full python code) (https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

List Comprehensions in Python – My Simplified Guide (https://www.machinelearningplus.com/python/list-comprehensions-in-python/)

Python @Property Explained – How to Use and When? (Full Examples) (https://www.machinelearningplus.com/python/python-property/)

How Naive Bayes Algorithm Works? (with example and full code) (https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/)

Parallel Processing in Python – A Practical Guide with Examples (https://www.machinelearningplus.com/python/parallel-processing-python/)

Cosine Similarity – Understanding the math and how it works (with python codes) (https://www.machinelearningplus.com/nlp/cosine-similarity/)

Gensim Tutorial – A Complete Beginners Guide (https://www.machinelearningplus.com/nlp/gensim-tutorial/)

Lemmatization Approaches with Examples in Python (https://www.machinelearningplus.com/nlp/lemmatization-examples-python/)

Feature Selection – Ten Effective Techniques with Examples (https://www.machinelearningplus.com/machine-learning/feature-selection/)

101 Pandas Exercises for Data Analysis (https://www.machinelearningplus.com/python/101-pandas-exercises-python/)

LDA in Python – How to grid search best topic models? (https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/)

Topic Modeling with Gensim (Python) (https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/)

Python debugging with pdb (https://www.machinelearningplus.com/python/python-debugging/)

Caret Package – A Practical Guide to Machine Learning in R (https://www.machinelearningplus.com/machine-learning/caret-package/)

101 NumPy Exercises for Data Analysis (Python) (https://www.machinelearningplus.com/python/101-numpy-exercises-python/)

Numpy Tutorial Part 2 – Vital Functions for Data Analysis (https://www.machinelearningplus.com/python/numpy-tutorial-python-part2/)

Tags

@property (https://www.machinelearningplus.com/tag/property/) Bigrams (https://www.machinelearningplus.com/tag/bigrams/)

Classification (https://www.machinelearningplus.com/tag/classification/)

Corpus (https://www.machinelearningplus.com/tag/corpus/)

Cosine Similarity (https://www.machinelearningplus.com/tag/cosine-similarity/)

Data Manipulation (https://www.machinelearningplus.com/tag/data-manipulation/)

Debugging (https://www.machinelearningplus.com/tag/debugging/) Doc2Vec (https://www.machinelearningplus.com/tag/doc2vec/)

Evaluation Metrics (https://www.machinelearningplus.com/tag/evaluation-metrics/)

FastText (https://www.machinelearningplus.com/tag/fasttext/)

Feature Selection (https://www.machinelearningplus.com/tag/feature-selection/)

Gensim (https://www.machinelearningplus.com/tag/gensim/)

klaR (https://www.machinelearningplus.com/tag/klar/) LDA (https://www.machinelearningplus.com/tag/lda/)

Lemmatization (https://www.machinelearningplus.com/tag/lemmatization/)

Linear Regression (https://www.machinelearningplus.com/tag/linear-regression/)

Logistic (https://www.machinelearningplus.com/tag/logistic/) LSI (https://www.machinelearningplus.com/tag/lsi/)

Matplotlib (https://www.machinelearningplus.com/tag/matplotlib/)

Multiprocessing (https://www.machinelearningplus.com/tag/multiprocessing/)

Naive Bayes (https://www.machinelearningplus.com/tag/naive-bayes/)

NLP (https://www.machinelearningplus.com/tag/nlp/)

NLTK (https://www.machinelearningplus.com/tag/nltk/)

Numpy (https://www.machinelearningplus.com/tag/numpy/)

Pandas (https://www.machinelearningplus.com/tag/pandas/)

Parallel Processing (https://www.machinelearningplus.com/tag/parallel-processing/)

Phraser (https://www.machinelearningplus.com/tag/phraser/)

Practice Exercise (https://www.machinelearningplus.com/tag/practice-exercise/)

Python (https://www.machinelearningplus.com/tag/python/)

R (https://www.machinelearningplus.com/tag/r/)

Regex (https://www.machinelearningplus.com/tag/regex/)

Regression (https://www.machinelearningplus.com/tag/regression/)

Residual Analysis (https://www.machinelearningplus.com/tag/residual-analysis/)

Scikit Learn (https://www.machinelearningplus.com/tag/scikit-learn/)

Significance Tests (https://www.machinelearningplus.com/tag/significance-tests/)

Soft Cosine Similarity (https://www.machinelearningplus.com/tag/soft-cosine-similarity/)

spaCy (https://www.machinelearningplus.com/tag/spacy/) Summarization (https://www.machinelearningplus.com/tag/summarization/)

TaggedDocument (https://www.machinelearningplus.com/tag/taggeddocument/)

TextBlob (https://www.machinelearningplus.com/tag/textblob/) TFIDF (https://www.machinelearningplus.com/tag/tfidf/)

Time Series (https://www.machinelearningplus.com/tag/time-series/)

Topic Modeling (https://www.machinelearningplus.com/tag/topic-modeling/)

Visualization (https://www.machinelearningplus.com/tag/visualization/)

Word2Vec (https://www.machinelearningplus.com/tag/word2vec/)