pandas-75-exercises-with-solutions

December 9, 2020

1 Welcome to this Kernel

This kernel is a compilation of 75 exercises with solutions from this webpage:

https://www.machinelearningplus.com/python/101-pandas-exercises-python/

```
[ ]: # Allow several prints in one cell
     from IPython.core.interactiveshell import InteractiveShell
     InteractiveShell.ast_node_interactivity = "all"
```

2 Pandas exercise

1. How to import pandas and check the version?

```
[1]: import pandas as pd
     print(pd.__version__)

     # Print all pandas dependencies
     print(pd.show_versions(as_json=True))
```

```
1.0.1
{'system': {'commit': None, 'python': '3.7.6.final.0', 'python-bits': 64, 'OS':
'Linux', 'OS-release': '5.4.0-29-generic', 'machine': 'x86_64', 'processor':
'x86_64', 'byteorder': 'little', 'LC_ALL': 'None', 'LANG': 'zh_CN.UTF-8',
'LOCALE': 'zh_CN.UTF-8'}, 'dependencies': {'pandas': '1.0.1', 'numpy': '1.18.1',
'pytz': '2019.3', 'dateutil': '2.8.1', 'pip': '20.0.2', 'setuptools':
'45.2.0.post20200210', 'Cython': '0.29.15', 'pytest': '5.3.5', 'hypothesis':
'5.5.4', 'sphinx': '2.4.0', 'blosc': None, 'feather': None, 'xlsxwriter':
'1.2.7', 'lxml.etree': '4.6.2', 'html5lib': '1.0.1', 'pymysql': None,
'psycopg2': None, 'jinja2': '2.11.1', 'IPython': '7.12.0', 'pandas_datareader':
None, 'bs4': '4.8.2', 'bottleneck': '1.3.2', 'fastparquet': None, 'gcsfs': None,
'matplotlib': '3.1.3', 'numexpr': '2.7.1', 'odfpy': None, 'openpyxl': '3.0.3',
'pandas_gbq': None, 'pyarrow': None, 'pytables': None, 'pyxlsb': None, 's3fs':
None, 'scipy': '1.4.1', 'sqlalchemy': '1.3.13', 'tables': '3.6.1', 'tabulate':
None, 'xarray': None, 'xlrd': '1.2.0', 'xlwt': '1.3.0', 'numba': '0.48.0'}}
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
a_list = list("abcdefg")
numpy_array = np.arange(1, 10)
dictionary = {"A":  0, "B":1, "C":2, "D":3, "E":5}
```

[ ]:
```
series1 = pd.Series(a_list)
print(series1)
series2 = pd.Series(numpy_array)
print(series2)
series3 = pd.Series(dictionary)
print(series3)
```

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

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)
print(ser[:5])
```

[ ]:
```
# solution 1 using DataFrame
ser_df = pd.DataFrame(ser)
ser_df.reset_index()

# using pandas to_frame()
ser_df = ser.to_frame().reset_index()
ser_df
```

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

Combine ser1 and ser2 to form a dataframe.

[ ]:
```
# input
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

[ ]:
```
# using pandas DataFrame
ser_df = pd.DataFrame(ser1, ser2).reset_index()
ser_df.head()
# using pandas DataFrame with a dictionary, gives a specific name to the column
ser_df = pd.DataFrame({"col1":ser1, "col2":ser2})
ser_df.head(5)
# using pandas concat
```

```
ser_df = pd.concat([ser1, ser2], axis = 1)
ser_df.head()
```

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

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

[ ]:
```
# input
ser = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
```

[ ]:
```
# using series rename method
ser.rename("alphabets")
# using series attribute
ser.name = "other_name"
ser
```

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

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])
```

[ ]:
```
ser1[~ser1.isin(ser2)]
```

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

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])
```

[ ]:
```
# using pandas
a_not_b = ser1[~ser1.isin(ser2)]
b_not_a = ser2[~ser2.isin(ser1)]

a_not_b.append(b_not_a, ignore_index = True)

# using numpy union and intersection
ser_u = pd.Series(np.union1d(ser1, ser2))
ser_i = pd.Series(np.intersect1d(ser1, ser2))
ser_u[~ser_u.isin(ser_i)]
```

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

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

```
[ ]: # input
     state = np.random.RandomState(100)
     ser = pd.Series(state.normal(10, 5, 25))
```

```
[ ]: # using pandas
     ser.describe()

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

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

Calculate the frequency counts of each unique value ser.

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

```
[ ]: ser.value_counts()
```

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

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]))
     ser
```

```
[ ]: ser.value_counts()
     ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
     ser
     # we do value_counts to see the repetitions for each value, then we do ~ser.
      ↪isin value_counts, filter by index the first 2 and = "Other renames the␣
      ↪values"
```

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

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

```
[ ]: # input
     ser = pd.Series(np.random.random(20))
     ser
```

```
[ ]: pd.qcut(ser, q = 10)
     # we can also pass labels
     pd.qcut(ser, q = [0, .10, .20, .30, .40, .50, .60, .70, .80, .90, 1],␣
      ↪labels=['1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th',␣
      ↪'10th']).head()
```

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

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

```
[ ]: # input
     ser = pd.Series(np.random.randint(1, 10, 35))
     ser
```

```
[ ]: # using numpy
     pd.DataFrame(np.array(ser).reshape(7, 5))

     # using only pandas
     pd.DataFrame(ser.values.reshape(7, 5))
```

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

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

```
[ ]: # input

     np.random.RandomState(100)
     ser = pd.Series(np.random.randint(1, 5, 10))
     ser
```

```
[ ]: # using the where clause
     ser.where(lambda x: x%3 == 0).dropna()

     # using numpy and reshape to get a pandas series
     #pd.Series(np.argwhere(ser%3 == 0).reshape(4))
     np.argwhere(ser%3 == 0)
```

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

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

```
[ ]: # input

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

```
[ ]: # using loc
     ser.loc[pos]

     # using series take
     ser.take(pos)
```

15. How to stack two series vertically and horizontally ?

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

[ ]:
```
# input
ser1 = pd.Series(range(5))
ser2 = pd.Series(list('abcde'))
```

[ ]:
```
# vertical
ser1.append(ser2)
# or using pandas concat and axis = 0
pd.concat([ser1, ser2], axis = 0)

# horizontal
pd.concat([ser1, ser2], axis = 1)
```

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

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])
```

[ ]:
```
# get's the index, but it's sorts the index
list(ser1[ser1.isin(ser2)].index)

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

# using pandas Index and get location
[pd.Index(ser1).get_loc(i) for i in ser2]
```

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

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)
```

[ ]:
```
# BAD, don't use it
(np.mean([(truth_i - pred_i)**2 for truth_i, pred_i in zip(truth, pred)]))

# using numpy
np.mean((truth-pred)**2)

# using sklear metrics
from sklearn.metrics import mean_squared_error
mean_squared_error(truth, pred)
```

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

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

```
[ ]: # input
     ser = pd.Series(['just', 'a', 'random', 'list'])
     ser
```

```
[ ]: # using python string method title() Assumes we only encounter string in the␣
      ↪list
     [i.title() for i in ser]

     # using lambda
     ser.map(lambda x: x.title())

     # other solution
     ser.map(lambda x: x[0].upper() + x[1:])
```

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

```
[ ]: # input
     ser = pd.Series(['just', 'a', 'random', 'list'])
```

```
[ ]: # using list comprehension
     [len(i) for i in ser]

     # using series map
     ser.map(len)

     # using series apply
     ser.apply(len)
```

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

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]
```

```
[ ]: # using pandas diff()
     ser.diff(periods = 1).tolist()
     ser.diff(periods = 1).diff(periods = 1).tolist()
```

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

```
[ ]: # 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
'''
```

[ ]:
```
# using pands to_datetime
pd.to_datetime(ser)

# using dateutil parse
from dateutil.parser import parse
ser.map(lambda x: parse(x))
```

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

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']
'''
```

[ ]:
```
# day
pd.to_datetime(ser).dt.day.to_list()
# week
pd.to_datetime(ser).dt.week.to_list()
# another method
pd.to_datetime(ser).dt.weekofyear.to_list()
```

```
# day of year
pd.to_datetime(ser).dt.dayofyear.to_list()
# day of week in words
week_dict = {0:"Monday", 1:"Tuesday", 2:"Wednesday", 3:"Thursday", 4:"Friday",␣
↪5:"Saturday", 6:"Sunday"}
pd.to_datetime(ser).dt.dayofweek.map(week_dict).to_list()
# another method
pd.to_datetime(ser).dt.weekday_name.to_list()
```

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

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]

'''
```

[ ]:
```
# solution using parser
from dateutil.parser import parse
ser.map(lambda x: parse('04 ' + x))

# another solution

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]
```

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

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
     '''
```

```
[ ]: # using nested loops
     vowels = list("aeiou")
     list_ = []
     for w in ser:
         c = 0
         for l in list(w.lower()):
             if l in vowels:
                 c += 1
         if c >= 2:
             print(w)
             list_.append(w)

     ser[ser.isin(list_)]

     # another solution using counter

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

25. How to filter valid emails from a series?

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', 'narendra@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
'''
```

[ ]:
```
# using powerful regex
import re
re_ = re.compile(pattern)
emails[emails.str.contains(pat = re_, regex = True)]

# other solutions
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]

# using str.findall
emails.str.findall(pattern, flags=re.IGNORECASE)

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

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

Compute the mean of weights of each fruit.

[ ]:
```
# doesn't incliude the upper limit
fruit = pd.Series(np.random.choice(['apple', 'banana', 'carrot'], 10))
fruit
weights = pd.Series(np.linspace(1, 10, 10))
weights
#print(weights.tolist())
#print(fruit.tolist())

'''
Desired output

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

[ ]:
```
# using pandas groupby
df = pd.concat([fruit, weights], axis = 1)
df
df.groupby(0).mean()
```

```
# use one list to calculate a kpi from another
weights.groupby(fruit).mean()
```

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

Compute the 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
'''
```

```
[ ]: # using list comprehension
suma = np.sqrt(np.sum([(p - q)**2 for p, q in zip(p, q)]))
suma

# using series one to one operation
sum((p - q)**2)**.5

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

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

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

'''
Desired output

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

```
[ ]: # using pandas shift
local_max = ser[(ser.shift(1) < ser) & (ser.shift(-1) < ser)]
local_max.index

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

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.

Section **??**

[ ]:
```
# input
my_str = 'dbc deb abed ggade'

'''
Desired Output

'dbccdebcabedcggade'  # least frequent is 'c'
'''
```

[ ]:
```
# using Counter
from collections import Counter
my_str_ = my_str
Counter_ = Counter(list(my_str_.replace(" ", "")))
Counter_
minimum = min(Counter_, key = Counter_.get)

print(my_str.replace(" ", minimum))

# using pandas
ser = pd.Series(list(my_str.replace(" ", "")))
ser.value_counts()
minimum = list(ser.value_counts().index)[-1]
minimum
print(my_str.replace(" ", minimum))
```

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

[ ]:
```
'''
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
'''
```

```
[ ]: dti = pd.Series(pd.date_range('2000-01-01', periods=10, freq='W-SAT'))
     random_num = pd.Series([np.random.randint(1, 10) for i in range(10)])


     df = pd.concat({"Time":dti, "Numbers":random_num}, axis = 1)
     df

     # for more about time series functionality
     # https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.
      ↪html#timeseries-offset-aliases

     # another solution just using pandas Series
     ser = pd.Series(np.random.randint(1,10,10), pd.date_range('2000-01-01',␣
      ↪periods=10, freq='W-SAT'))
     ser
```

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

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', '2000-01-06', '2000-01-08']))

     '''
     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
     '''
```

```
[ ]: # Solution 1
     # first let's fill the missing dates
     indx = pd.date_range("2000-01-01", "2000-01-08")
     # now let's reindex the series ser with the new index
     # we have to reasing back to ser
     ser = ser.reindex(indx)
     # lastly let's populate the missing values
     ser.fillna(method = "ffill")
```

```
# Solution 2
ser = pd.Series([1,10,3,np.nan], index=pd.to_datetime(['2000-01-01',␣
 ↪'2000-01-03', '2000-01-06', '2000-01-08']))
ser.resample('D').ffill()  # fill with previous value
ser.resample('D').bfill()  # fill with next value
ser.resample('D').bfill().ffill()  # fill next else prev value
```

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

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.
 ↪28000000000000003, -0.040000000000000001, -0.37, 0.41999999999999998, 0.
 ↪47999999999999998, 0.17999999999999999]
Lag having highest correlation:  9
'''
```

[ ]:
```
# using pandas autocorr
# ser.autocorr(lag = 10)

# solution using list comprehension
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)
```

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

Import every 50th row of BostonHousing dataset as a dataframe.

[ ]:
```
# input
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[ ]:
```
# data comes without headers, but we searched for it
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',␣
 ↪'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# pure Python implementation
```

```
with open("/kaggle/input/boston-house-prices/housing.csv") as f:
    data = f.read()
    nth_rows = []
    for i, rows in enumerate(data.split("\n")):
        if i%50 == 0:
            nth_rows.append(rows)

# nth_rows is a list of strings separated by blank spaces " "
# the next list comprehension will do the trick

nth_rows[0]
data_ = [nth_rows[i].split() for i in range(len(nth_rows))]
df = pd.DataFrame(data_, columns=names)
df

# other solutions

# Solution 2: Use chunks and for-loop
# df = pd.read_csv("/kaggle/input/boston-house-prices/housing.csv",␣
 ↪chunksize=50)
# df2 = pd.DataFrame()
# for chunk in df:
#     df2 = df2.append(chunk.iloc[0,:])
# df2

# Solution 3: Use chunks and list comprehension
# df = pd.read_csv("/kaggle/input/boston-house-prices/housing.csv",␣
 ↪chunksize=50)
# df2 = pd.concat([chunk.iloc[0] for chunk in df], axis=1)
# df2 = df2.transpose()
# df2
```

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

Import the boston housing dataset, but while importing change the 'medv' (median house value) column so that values <25 becomes 'Low' and >25 becomes 'High'.

[ ]:
```
# input
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

[ ]:
```
# first let's import using the previuos code and save as a normal csv

names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',␣
 ↪'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
```

```
with open("/kaggle/input/boston-house-prices/housing.csv") as f:
    data = f.read()
    nth_rows = []
    for i, rows in enumerate(data.split("\n")):
        nth_rows.append(rows)

data_ = [nth_rows[i].split() for i in range(len(nth_rows))]

df = pd.DataFrame(data_, columns=names)
df.head()
df.to_csv("housing_preprocessed.csv")
del df
```

[ ]:
```
# now let's start importing as normal and use converters to convert the values
# skipfooter because we had the last rows with nan values and index_col to␣
 ↪specify that the first column is the index
df = pd.read_csv("housing_preprocessed.csv",  index_col = 0, skipfooter=1, ␣
 ↪converters = {"MEDV": lambda x: "HIGH" if float(x) >= 25 else "LOW"})
df
```

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

[ ]:
```
# 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]])
'''
```

[ ]:
```
# using slicing
# let's generate a list of indexes we need to use
# outputs array([ 0,  2,  4,  6,  8, 10, 12, 14])
index_ = np.arange(0, 15, 2)
index_
my_list = []
for i in range(6):
    my_list.append(list(L[index_[i]:index_[i+2]]))
np.array(my_list)

# above code as list comprehension
```

```
np.array([L[index_[i]:index_[i+2]] for i in range(6)])

# another solution
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)
```

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

```
[ ]: # input

     # code that generates the housing_preprocessed.csv file
     names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',␣
      ↪'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
     with open("/kaggle/input/boston-house-prices/housing.csv") as f:
         data = f.read()
         nth_rows = []
         for i, rows in enumerate(data.split("\n")):
             nth_rows.append(rows)

     data_ = [nth_rows[i].split() for i in range(len(nth_rows))]

     df = pd.DataFrame(data_, columns=names)
     df.to_csv("housing_preprocessed.csv")
     del df

     # use the /kaggle/input/boston-house-prices/housing_preprocessed.csv file
     import os
     for dirname, _, filenames in os.walk('/kaggle/input'):
         for filename in filenames:
             print(os.path.join(dirname, filename))
```

```
[ ]: file = "housing_preprocessed.csv"
     # using index
     df = pd.read_csv(file, usecols = [1, 2, 4], skipfooter=1)
     df.head()
     # using column names
     df = pd.read_csv(file, usecols = ["CRIM", "ZN", "CHAS"])
     df.head()
```

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

```
[ ]: # input
     # use the "housing_preprocessed.csv" file
```

```
[ ]: df = pd.read_csv("housing_preprocessed.csv", index_col=0 ,skipfooter=1)
     # number of rows and columns
     df.shape

     # each type of column
     df.dtypes

     # a more general view of the earlier code
     df.info()

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

     # all the statistics
     df.describe()
```

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

```
[ ]: # input
     # use the "housing_preprocessed.csv" file
```

```
[ ]: # solution 1
     df = pd.read_csv("housing_preprocessed.csv", skipfooter=1, index_col=0)
     # let's get the maximum value
     max_tax = df["TAX"].max()
     max_tax

     # now let's find the column and cell that has the maximum value
     df[df["TAX"] == max_tax]

     # solution 2
     df.loc[df["TAX"] == np.max(df["TAX"]), ["CRIM", "ZN", "TAX"]]

     # solution 3
     # get the row and column number
     row, col = np.where(df.values == np.max(df["TAX"]))
     for i, j in zip(row, col):
         print(i , j)

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

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

# 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.
```

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

```
[ ]: # input
     # Rename the column Type as CarType in df and replace the '.' in column names␣
      ↪with '_'.
     cars93 = pd.read_csv("../input/cars93/Cars93.csv", index_col=0)
     cars93.head()

     '''
     Desired Output

     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')
     '''
```

```
[ ]: # Solution 1: in 2 steps
     # Step1
     # first let's rename the Type to CarType
     cars93 = pd.read_csv("../input/cars93/Cars93.csv", index_col=0)
     cars93.rename(columns={"Type":"CarType"}, inplace = True)
     cols = cars93.columns
     # or
     df.columns.values[2] = "CarType"
     # Step2
     # replace the "." with "-"
     cols = list(map(lambda x: x.replace(".", "_"), cols))
     cars93.columns = cols
     cars93.head()

     # Solution 2: working only with lists
     cars93 = pd.read_csv("../input/cars93/Cars93.csv", index_col=0)
     cols = cars93.columns
     cols = list(map(lambda x: x.replace(".", "_"), cols))
```

```
cols[cols.index("Type")] = "CarType"
cars93.columns = cols
cars93.head()
```

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

```
[ ]: # input
     df = pd.read_csv("../input/cars93/Cars93.csv")
     df
```

```
[ ]: # Solution 1
     print("Our df has a total of {} null values".format(df.isnull().sum().sum()))
     print()

     # Solution 2
     df.isnull().values.any()
     print()

     # Solution 3
     # A more detailed one
     def report_nulls(df):
         '''
         Show a fast report of the DF.
         '''
         rows = df.shape[0]
         columns = df.shape[1]
         null_cols = 0
         list_of_nulls_cols = []
         for col in list(df.columns):
             null_values_rows = df[col].isnull().sum()
             null_rows_pcn = round(((null_values_rows)/rows)*100, 2)
             col_type = df[col].dtype
             if null_values_rows > 0:
                 print("The column {} has {} null values. It is {}% of total rows".
      ↪format(col, null_values_rows, null_rows_pcn))
                 print("The column {} is of type {}.\n".format(col, col_type))
                 null_cols += 1
                 list_of_nulls_cols.append(col)
         null_cols_pcn = round((null_cols/columns)*100, 2)
         print("The DataFrame has {} columns with null values. It is {}% of total␣
      ↪columns.".format(null_cols, null_cols_pcn))
         return list_of_nulls_cols

     report_nulls(df)
```

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

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("../input/cars93/Cars93.csv")
```

```
[ ]: # Solution 1
     df_null = pd.DataFrame(df.isnull().sum())
     df_null[df_null[0] > 0][0].argmax()
     df_null[df_null[0] > 0][0].idxmax()

     # Solution 2
     # find the total number of nulls per column
     n_missings_each_col = df.apply(lambda x: x.isnull().sum())

     # find the maximum nulls
     n_missings_each_col.argmax()
     n_missings_each_col.idxmax()
```

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

Replace missing values in Luggage.room columns with their respective mean.

```
[ ]: # input
     df = pd.read_csv("../input/cars93/Cars93.csv")
```

```
[ ]: # Solution 1
     beg_null = df.isnull().sum().sum()
     print(beg_null)
     # notice that we have filtering the columns  as a list.
     df[["Luggage.room"]] = df[["Luggage.room"]].apply(lambda x: x.fillna(x.mean()))
     end_null = df.isnull().sum().sum()
     print(end_null)

     print("We have got rid of {} null values, filling them with the mean.".
      ↪format(beg_null - end_null))
```

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

In df, use apply method to replace the missing values in Rear.seat.room with mean Luggage.room with median by passing an argument to the function.

```
[ ]: # input
     df = pd.read_csv("../input/cars93/Cars93.csv")
```

```
[ ]: # input
     df = pd.read_csv("../input/cars93/Cars93.csv")

     # Solution 1
```

```
print("We have a total of {} nulls".format(df.isnull().sum().sum()))

d = {'Rear.seat.room': np.nanmean, 'Luggage.room': np.nanmedian}
df[['Rear.seat.room', 'Luggage.room']] = df[['Rear.seat.room', 'Luggage.room']].
 ↪apply(lambda x, d: x.fillna(d[x.name](x)), args=(d, ))

print("We have a total of {} nulls".format(df.isnull().sum().sum()))

df["Rear.seat.room"].sum()
df["Luggage.room"].sum()


# Solution 2
# impor the df
df = pd.read_csv("../input/cars93/Cars93.csv")

# check nulls
print("We have a total of {} nulls".format(df.isnull().sum().sum()))

# define a custom function
def num_inputer(x, strategy):
    if strategy.lower() == "mean":
        x = x.fillna(value = np.nanmean(x))
    if strategy.lower() == "median":
        x = x.fillna(value = np.nanmedian(x))
    return x

# apply the custon function and using args whe can pass the strategy we want
df['Rear.seat.room'] = df[['Rear.seat.room']].apply(num_inputer, args =␣
 ↪["mean"])
df['Luggage.room'] = df[['Luggage.room']].apply(num_inputer, args = ["median"])

# check for nulls
print("We have a total of {} nulls".format(df.isnull().sum().sum()))

df["Rear.seat.room"].sum()
df["Luggage.room"].sum()
```

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

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'))
```

```
[ ]: # Solution
     # using to_frame()
     type(df["a"].to_frame())
     # using pandas DataFrame
     type(pd.DataFrame(df["a"]))

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

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

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

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'))
```

```
[ ]: # Solution to question 1
     # we pass a list with the custom names BUT THIS DOESN'T change in place
     df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))
     df[["c", "b", "a", "d", "e"]]
     df

     # if we reasing that this will work
     df = df[["c", "b", "a", "d", "e"]]
     df

     # Solution to question 2
     def change_cols(df, col1, col2):
         df_columns = df.columns.to_list()
         index1 = df_columns.index(col1)
         index2 = df_columns.index(col2)
         # swaping values
         df_columns[index1], df_columns[index2] = col1, col2
```

```
    return df[df_columns]


df = change_cols(df, "b", "e")
df


# Solution to question 3
df = pd.DataFrame(np.arange(20).reshape(-1, 5), columns=list('abcde'))
col_list = list(df.columns)
col_list_reversed = col_list[::-1]
col_list
col_list_reversed
# using the trick from solution 1
df = df[col_list_reversed]
df


print("Solution from the website")
print("-------------------------")
# Others solution from the website

# 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?

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

```
[ ]: # input
     df = pd.read_csv("../input/cars93/Cars93.csv")
```

```
[ ]: # we use set_option to set the maximun rows and columns to display
     pd.set_option("display.max_columns",10)
     pd.set_option("display.max_rows",10)
     df
```

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

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

```
[ ]: # input
     df = pd.DataFrame(np.random.random(5)**10, columns=['random'])

     '''
     Desired Output
     #>    random
     #> 0  0.0035
     #> 1  0.0000
     #> 2  0.0747
     #> 3  0.0000
     '''
```

```
[ ]: print("Initial DF")
     df
     print("Using solution 1")
     # Solution 1
     df.round(4)
     df
     pd.reset_option('^display.', silent=True)

     print("Using solution 2")
     # Solution 2
     df.apply(lambda x: '%.4f' %x, axis=1).to_frame()
     df
     pd.reset_option('^display.', silent=True)

     print("Using solution 3")
     # Solution 3
     pd.set_option('display.float_format', lambda x: '%.4f'%x)
     df
     pd.reset_option('^display.', silent=True)
     df
```

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

Format the values in column ‘random’ of df as percentages.

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

```
[ ]: # Solution 1
# Using style.format we can pass a dictionary to each column and display as we␣
 ↪want
out = df.style.format({
    'random': '{0:.2%}'.format,
})
out

# This applies to all the df
pd.options.display.float_format = '{:,.2f}%'.format
# to get the % multiply by 100
df*100
pd.reset_option('^display.', silent=True)
```

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

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

```
[ ]: # input
df = pd.read_csv("../input/cars93/Cars93.csv")
df
```

```
[ ]: # First let's import only the columns we need
df = pd.read_csv("../input/cars93/Cars93.csv", usecols=["Manufacturer",␣
 ↪"Model", "Type"])

# Solution 1
# Using normal python slicing
df[::20]

df = pd.read_csv("../input/cars93/Cars93.csv", usecols=["Manufacturer",␣
 ↪"Model", "Type"])

# Solution 2
# Using iloc
df.iloc[::20, :][['Manufacturer', 'Model', 'Type']]
```

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

In df, Replace NaNs 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("../input/cars93/Cars93.csv")
```

```
df
```

[ ]:
```
# Solution
df = pd.read_csv("../input/cars93/Cars93.csv", usecols=["Manufacturer",␣
↪"Model", "Type", "Min.Price", "Max.Price"])

# let's check if we have null
df.isnull().sum().sum()
df.fillna("missing")
# create new index
df["new_index"] = df["Manufacturer"] + df["Model"] + df["Type"]
# set new index
df.set_index("new_index", inplace = True)
df
```

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

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'))
df
```

[ ]:
```
# Solution 1

# argsort give the index of the smallest to largest number in an array
# arg_sort[0] is the index of the smallest number in df["a"]
arg_sort = df["a"].argsort()

#arg_sort.to_frame()
#arg_sort[0]

# now let's sort by arg_sort
#df
df = df.iloc[arg_sort]
df["arg_sort"] = arg_sort
df
n_largest = 5
print("The {} largest values in our DF is at row/index {} and the value is {}".
↪format(n_largest, (df[df["arg_sort"] == (n_largest-1)].index[0]),␣
↪df[df["arg_sort"] == (n_largest-1)]["a"].iloc[0]))

# Shorter solution
n = 5
# select column, argsort, inders (largest to smallest) and select the n largest
df['a'].argsort()[::-1][n]
```

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

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

[ ]:
```
# input
ser = pd.Series(np.random.randint(1, 100, 15))
```

[ ]:
```
# Solution using argsort and boolean filtering of pandas series
# I understood that I wanted the second largest of all values that is greter␣
↪than the mean
# so I sorted
#ser
sorted_ser = ser[ser.argsort()[::-1]]
#sorted_ser
sorted_ser[sorted_ser > sorted_ser.mean()].index[1]

# If you understood that the 2 value you encounter that is bigger than the mean
# This is the correct solution
print('ser: ', ser.tolist(), 'mean: ', round(ser.mean()))
np.argwhere(ser > ser.mean())[1]

# Another solution
ser[ser > ser.mean()].index[1]
```

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

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

[ ]:
```
# input
df = pd.DataFrame(np.random.randint(10, 40, 60).reshape(-1, 4))
df1 = df.copy(deep = True)
```

[ ]:
```
# Solution 1
df["sum"] = df.sum(axis = 1)
df

print("The index of the rows that are greater than 100 are {}".
↪format((df[df["sum"] > 100].index).to_list()[-2:]))

# Solution 2 using numpy
rowsums = df1.apply(np.sum, axis=1)

# last two rows with row sum greater than 100
last_two_rows = df1.iloc[np.where(rowsums > 100)[0][-2:], :]
last_two_rows
```

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

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))
ser1 = ser.copy(deep = True)
ser2 = ser.copy(deep = True)
```

```
[ ]: # Solution 1
# get the quantiles values
quantiles = np.quantile(ser, [0.05, 0.95])
ser

# filter ser using numpy to know where the values are below or greater than 5%␣
↪or 95% and replace the values
ser.iloc[np.where(ser < quantiles[0])] = quantiles[0]
ser.iloc[np.where(ser > quantiles[1])] = quantiles[1]

# or we can just do
ser1[ser1 < quantiles[0]] = quantiles[0]
ser1[ser1 > quantiles[1]] = quantiles[1]

ser1

# Solution from the webpage
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(ser2, .05, .95)
ser2
capped_ser
```

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

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))
```

```
[ ]: # This solution sorts the values.
# Not want we want
# my_array = np.array(df.values.reshape(-1, 1))
# my_array = my_array[my_array > 0]
# my_array.shape[0]
# lar_square = int(np.floor(my_array.shape[0]**0.5))
# arg_sort = np.argsort(my_array)[::-1]
# my_array[arg_sort][0:lar_square**2].reshape(lar_square, lar_square)


# Correct solution
my_array = np.array(df.values.reshape(-1, 1)) # convert to numpy
my_array = my_array[my_array > 0] # filter only positive values
lar_square = int(np.floor(my_array.shape[0]**0.5)) # find the largest square
arg_sort = np.argsort(my_array)[::-1][0:lar_square**2] # eliminate the smallest␣
↪values that will prevent from converting to a square
my_array = np.take(my_array, sorted(arg_sort)).reshape(lar_square, lar_square)␣
↪# filter the array and reshape back
my_array


# Solution from the webpage
# 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)
```

56. How to swap two rows of a dataframe?

Swap rows 1 and 2 in df.

[ ]:
```
# input
df = pd.DataFrame(np.arange(25).reshape(5, -1))
df
```

[ ]:
```
# THIS SWAPS the columns
print("Original DataFrame")
df
temp_col = df[1].copy(deep = True)
df[1], df[2] = df[2], temp_col
print("Swapped Columns DataFrame")
df

# # THIS SWAPS the rows
print("Original DataFrame")
```

```
df
temp_row = df.iloc[1].copy(deep = True)
df.iloc[1], df.iloc[2] = df.iloc[2], temp_row
print("Swapped Rows DataFrame")
df

# Solution from the webpage
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))
```

57. How to reverse the rows of a dataframe?

Reverse all the rows of dataframe df.

[ ]:
```
# input
df = pd.DataFrame(np.arange(25).reshape(5, -1))
```

[ ]:
```
# Solution 1
df
df.iloc[df.index.to_list()[::-1]]

# Solutions from the webpage
# Solution 2
df.iloc[::-1, :]

# Solution 3
print(df.loc[df.index[::-1], :])
```

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

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'))

'''
Desired 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
'''
```

```
[ ]: # Using pd.get_dummies
     dummies = pd.get_dummies(df["a"])
     df = pd.concat([dummies, df], axis = 1)
     df

     # Solution from the webpage
     # in one line
     df_onehot = pd.concat([pd.get_dummies(df['a']), df[list('bcde')]], axis=1)
     df_onehot
```

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

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

```
[ ]: # input
     df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1))
```

```
[ ]: # Solution 1
     def get_col(df):
         columns = list(df.columns)
         df["col_index_with_max"] = ""
         for i in range(len(df)):
             row_values = list(df.iloc[i, :-1].values)
             max_value = np.max(row_values)
             col_index = row_values.index(max_value)
             df["col_index_with_max"].iloc[i] = col_index

     get_col(df)

     df
     print("The col with maximum amont of maximun per row if {} with a total of {}␣
      ↪maximus".format(df.groupby("col_index_with_max").size()[::-1].index[0], \
                                                                                ␣
      ↪               df.groupby("col_index_with_max").size()[::-1].values[0]))

     # Solution 2
     # Another much more elegant solution from the webpage
     print('Column with highest row maxes: ', df.apply(np.argmax, axis=1).
      ↪value_counts().index[0])
```

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.

```
[ ]: # input
     df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1),␣
      ↪columns=list('pqrs'), index=list('abcdefghij'))
```

```
'''
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

'''
```

```
[ ]: ##############################################################################################
# Solution 1
# input
df = pd.DataFrame(np.random.randint(1,100, 40).reshape(10, -1),␣
 ↪columns=list('pqrs'), index=list('abcdefghij'))

# place holders
corr_list = []
index_list = []

# temporary var
max_corr = 0
current_index = ""

# nested loop to calculate
for i in range(len(df)):
    for j in range(len(df)):
        if i == j:
            pass
        else:
            # distance
            curr_corr = sum((df.iloc[i] - df.iloc[j])**2)**.5
            # correlation
            #curr_corr = df.iloc[i].corr(df.iloc[j])
            if curr_corr >= max_corr:
                max_corr = curr_corr
                current_index = list(df.index)[j]
    corr_list.append(max_corr)
    index_list.append(current_index)

    max_corr = 0
    current_index = ""

df["nearest_row"] = index_list
df["dist"] = corr_list
df
df.drop(["nearest_row", "dist"], axis = 1, inplace = True)

##############################################################################################

# Solution from the webpage
# 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
df
```

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

For each column get the maximum possible correlation with other columns (only 1 value)

```
[ ]: # input
     df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1),␣
      ↪columns=list('pqrstuvwxy'), index=list('abcdefgh'))
```

```
[ ]: # calculate the correlation, returns a matrix
     df_corr = np.abs(df.corr())
     # sorted -2 because it goes from min to max
     # max = 1 because it's correlation againts each other
     # so we pick -2
     max_corr = df_corr.apply(lambda x: sorted(x)[-2], axis = 0)
     max_corr
```

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

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

```
[ ]: # input
df = pd.DataFrame(np.random.randint(1,100, 80).reshape(8, -1))
df1 = df.copy(deep = True)
df2 = df.copy(deep = True)
```

```
[ ]: # Solution 1
df["min_by_max"] = (df.apply(min, axis = 1)/df.apply(max, axis = 1))
df

# Other solution from the webpage
# Solution 2
min_by_max = df1.apply(lambda x: np.min(x)/np.max(x), axis=1)
min_by_max
# Solution 3
min_by_max = np.min(df2, axis=1)/np.max(df2, axis=1)
min_by_max
```

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

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))
```

```
[ ]: # Using lambda and numpy partition
df["penultimate"] = df.apply(lambda x: np.partition(x, -2)[-2], axis = 1)
df
df.drop("penultimate", inplace = True, axis = 1)

# Using lambda and python lists
df["penultimate"] = df.apply(lambda x: sorted(list(x))[-2], axis = 1)
df
df.drop("penultimate", inplace = True, axis = 1)

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

64. How to normalize all columns in a dataframe?

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))
     df1 = df.copy(deep = True)
```

```
[ ]: # First normalization: mean and std
     df = df.apply(lambda x: ((x-np.mean(x))/np.std(x)), axis = 0)
     df

     # min max
     df1 = df1.apply(lambda x: ((x.max() - x)/(x.max() - x.min())).round(2))
     df1
```

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

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))
```

```
[ ]: df["corr"] = 0
     for i in range(len(df)-1):

         values1 = df.iloc[i, :-1].astype('float64')
         values2 = df.iloc[i+1, :-1].astype('float64')
         corr = values1.corr(values2)
         df["corr"].iloc[i] = corr
     df
     df.drop("corr", inplace = True, axis = 1)

     # Solution from the webpage
     # using list comprehension
     [df.iloc[i].corr(df.iloc[i+1]).round(2) for i in range(df.shape[0])[:-1]]
```

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

Replace both values in both diagonals of df with 0.

```
[ ]: # input
     df = pd.DataFrame(np.random.randint(1,100, 100).reshape(10, -1))
     df1 = df.copy(deep = True)

     '''
     Desired Output (might change because of randomness)

```

```
#     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
'''
```

[ ]:
```
# input
df = pd.DataFrame(np.random.randint(1,100, 100).reshape(10, -1))
df1 = df.copy(deep = True)

# Using nested loops
print("Original DF")
df
for i in range(len(df)):
    for j in range(len(df)):
        if i == j:
            df.iloc[i ,j] = 0
            # Inverse the matrix so that we can replace the other diagonal
            df[::-1].iloc[i, j] = 0

print("DF from the solution 1")
df

# Solution from the webpage
# Solution
for i in range(df1.shape[0]):
    df1.iat[i, i] = 0
    df1.iat[df1.shape[0]-i-1, i] = 0

print("DF from the solution 2")
df1
```

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

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'])
```

[ ]:
```
# Solution 1
pd.DataFrame(df_grouped)
df_grouped.groups["apple"]
df_grouped.get_group("apple")

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

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

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

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

[ ]:
```
# Solution 1
grouped_by = df["rating"].groupby(df["fruit"])
grouped_by.get_group("banana")
list(grouped_by.get_group("banana"))[1]

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

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

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)})
df
```

[ ]:
```
# Using pandas pivot table
df_grouped = pd.pivot_table(df[["fruit", "price"]], index = ["fruit"], aggfunc␣
 ↪= np.mean ).reset_index()
df_grouped
```

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

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

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)})
df1
df2
```

[ ]:
```
# Solution 1
# using pandas merge
merge_df = pd.merge(df1, df2, left_on=["fruit", "weight"], right_on=["pazham",␣
 ↪"kilo"])
merge_df



# Solution from the webpage
pd.merge(df1, df2, how='inner', left_on=['fruit', 'weight'],␣
 ↪right_on=['pazham', 'kilo'], suffixes=['_left', '_right'])
```

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

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, 10, 9)})

df2 = pd.DataFrame({'pazham': ['apple', 'orange', 'pine'] * 2,
                    'kilo': ['high', 'low'] * 3,
                    'price': np.random.randint(0, 10, 6)})


df1
df2
```

[ ]:
```
# We might use pandas merge
```

```
#df1.merge(df2, how = "inner", left_on = ["fruit", "weight", "price"], right_on␣
↪= ["pazham", "kilo", "price"])

df1["concat"] = df1["fruit"].astype(str) + df1["weight"].astype(str) +␣
↪df1["price"].astype(str)
#df1

df2["concat"] = df2["pazham"].astype(str) + df2["kilo"].astype(str) +␣
↪df2["price"].astype(str)
#df2

df1 = df1[~df1["concat"].isin(df2["concat"])]
df1.drop("concat", inplace = True, axis = 1)
df1

# Solution from the webpage, IMHO it's incorrect
#df1[~df1.isin(df2).all(1)]
```

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

Find the index where col fruit1 and fruit2 match

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

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

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

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'))
df

'''
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
'''
```

```
[ ]: df["lag1"] = df["a"].shift(1)
     df["lead1"] = df["b"].shift(-1)
     df
```

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

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'))
```

```
[ ]: # Solution
     pd.value_counts(df.values.ravel())
```

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

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'])

     df

     '''
     Desired Output

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

```
[ ]: # we do " ".join(x.split()) to replace multiple spaces to 1 space
     # we do split(None, 2, ) to split a string on the second space ()this way we␣
      ↪have West Bengal together
     df["re"] = df["row"].apply(lambda x: " ".join(x.split()).split(None, 2, ))
```

```
new_header = df["re"][0]
values = df["re"][1:]

# our values is a series of lists, we have to do some list comprehension no␣
↪extract the values
d = {new_header[0]:[int(values.iloc[i][0].replace(",", "")) for i in␣
↪range(len(values))], \
     new_header[1]:[values.iloc[i][1].replace(",", "") for i in␣
↪range(len(values))], \
     new_header[2]:[values.iloc[i][2].replace(",", "") for i in␣
↪range(len(values))]}

# create a pandas DF from a dict
new_df = pd.DataFrame(d)
new_df
```

3 That's all, thanks a lot. I hope you learned a lot of pandas.