## **Intro To Pandas**

## 

## 

## **1 .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

# Inputs

import numpy as np

mylist = list('abcedfghijklmnopqrstuvwxyz')

myarr = np.arange(26)

mydict = dict(zip(mylist, myarr))

# Solution

ser1 = pd.Series(mylist)

ser2 = pd.Series(myarr)

ser3 = pd.Series(mydict)

print(ser3.head())

a 0

b 1

c 2

d 4

e 3

dtype: int64

## **2 .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)

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

## **3. 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))

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

## **4. 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])

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

## **5. 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)

Solution

# Input

truth = pd.Series(range(10))

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

# Solution

np.mean((truth-pred)\*\*2)

0.28448128110629545

## **6. 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']

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']

## **7. How to import only every nth row from a csv file to create a dataframe?**

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

Solution

# Solution 1: Use chunks and for-loop

df = pd.read\_csv('<https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.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/BostonHousing.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

## **8. How to change column values when importing csv to a dataframe?**

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’.

Solution

# Solution 1: Using converter parameter

df = pd.read\_csv('<https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.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())

## **9. How to check if a dataframe has any missing values?**

Check if df has any missing values.

Input

df = pd.read\_csv('<https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv>')

Solution

# Input

df = pd.read\_csv('<https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv>')

# Solution

df.isnull().values.any()

## **10. 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('<https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv>')

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'

## **11. How to replace missing values of multiple numeric columns with the mean?**

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

Input

df = pd.read\_csv('<https://raw.githubusercontent.com/selva86/datasets/master/Cars93_miss.csv>')

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

## **12. 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

# 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)

## **13. 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

#> random

#> 0 .689723

#> 1 .957224

#> 2 .159157

#> 3 .21082

Desired Output

#> random

#> 0 68.97%

#> 1 95.72%

#> 2 15.91%

#> 3 2.10%

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% |

## **14. 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))

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

## **15. How to create a column containing the minimum by maximum of each row?**

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

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

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)

## **16. 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))

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

## **17. 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)})

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', 'pounds'], 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 |

## **18. How to get the positions where values of two columns match?**

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]),)

## **References**

Machinelearningplus.com. 2021. [online] Available at: <https://www.machinelearningplus.com/python/101-pandas-exercises-python/> [Accessed 31 October 2021].