Energy saved from recycling

Did you know that recycling saves energy by reducing or eliminating the need to make materials from scratch? For example, aluminum can manufacturers can skip the energy-costly process of producing aluminum from ore by cleaning and melting recycled cans. Aluminum is classified as a non-ferrous metal.

Singapore has an ambitious goal of becoming a zero-waste nation. The amount of waste disposed of in Singapore has increased seven-fold over the last 40 years. At this rate, Semakau Landfill, Singapore's only landfill, will run out of space by 2035. Making matters worse, Singapore has limited land for building new incineration plants or landfills.

The government would like to motivate citizens by sharing the total energy that the combined recycling efforts have saved every year. They have asked you to help them.

You have been provided with three datasets. The data come from different teams, so the names of waste types may differ.

datasets/wastestats.csv - Recycling statistics per waste type for the period 2003 to 2017
Source: Singapore National Environment Agency

- **waste_type:** The type of waste recycled.
- **waste_disposed_of_tonne:** The amount of waste that could not be recycled (in metric tonnes).
- **total_waste_recycle_tonne:** The amount of waste that could be recycled (in metric tonnes).
- **total_waste_generated:** The total amount of waste collected before recycling (in metric tonnes).
- **recycling_rate:** The amount of waste recycled per tonne of waste generated.
- **year:** The recycling year.

datasets/2018_2019_waste.csv - Recycling statistics per waste type for the period 2018 to 2019
Source: Singapore National Environment Agency

- **Waste Type:** The type of waste recycled.
- **Total Generated:** The total amount of waste collected before recycling (in thousands of metric tonnes).
- **Total Recycled:** The amount of waste that could be recycled. (in thousands of metric tonnes).
- **Year:** The recycling year.

datasets/energy_saved.csv - Estimations of the amount of energy saved per waste type in kWh

- **material:** The type of waste recycled.
- **energy_saved:** An estimate of the energy saved (in kiloWatt hour) by recycling a metric tonne of waste.
- **crude_oil_saved:** An estimate of the number of barrels of oil saved by recycling a metric tonne of waste.

Instructions

The Singapore government has asked you to help them determine how much energy they have saved per year by recycling. You need to answer the following question:

How much energy in kiloWatt hour (kWh) has Singapore saved per year by recycling glass, plastic, ferrous, and non-ferrous metals between 2015 and 2019?

Save your answer as a DataFrame named annual_energy_savings with the an index labelled year. Your DataFrame should consist of one column, total_energy_saved, which contains the total amount of energy in kWh saved per year across the four materials described above. It should resemble the following table:

total_energy_saved year 2015 xxxx 2016 xxxx 2017 xxxx 2018 xxxx 2019 xxxx

Note: Unlike the Guided and Unguided Projects that exist on our platform, if you get stuck in this task, you will not have access to any hints, nor will you be able to request a solution. Similarly, our testing process is focused on your answers and will not provide feedback to help you towards your solution. All steps required, including importing, exploration, cleaning, and analysis, will be up to you!

Project: Energy Savings: Python Certification

```
#List files ONLY in the current directory
import os

files = [f for f in os.listdir('.') if os.path.isfile(f)]
print(files)
```

```
     ['.profile', '.bashrc', '.bash_logout', '.startup.py']
```

```
#Import packages
import pandas as pd
```

```
#Read dataframes
df_recycle_before_18 = pd.read_csv('datasets/wastestats.csv', index_col='year')
df_recycle_from_18 = pd.read_csv('datasets/2018_2019_waste.csv', index_col='Year')
df_energy_saved = pd.read_csv('datasets/energy_saved.csv')
```

Understand columns and data types in each dataframe

```
df_recycle_before_18.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 225 entries, 2016 to 2017
     Data columns (total 5 columns):
     waste_type                     225 non-null object
     waste_disposed_of_tonne        225 non-null int64
     total_waste_recycled_tonne     225 non-null float64
     total_waste_generated_tonne    225 non-null int64
     recycling_rate                 225 non-null float64
     dtypes: float64(2), int64(2), object(1)
     memory usage: 10.5+ KB
```

No NULL values - a good sign

waste_type - 225 non-null object - string

The other data types look alright

-waste_disposed_of_tonne - 225 non-null int64

-total_waste_recycled_tonne - 225 non-null float64

-total_waste_generated_tonne - 225 non-null int64

-recycling_rate - 225 non-null float64

-year - 225 non-null int64

```
df_recycle_from_18.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 30 entries, 2019 to 2018
     Data columns (total 3 columns):
     Waste Type                       30 non-null object
     Total Generated ('000 tonnes)    30 non-null int64
     Total Recycled ('000 tonnes)     30 non-null int64
     dtypes: int64(2), object(1)
     memory usage: 960.0+ bytes
```

NO NULL values - a good sign

Waste Type - 30 non-null object - string

The other data types look alright

-Total Generated ('000 tonnes) - 30 non-null int64

-Total Recycled ('000 tonnes) - 30 non-null int64

-Year - 30 non-null int64

```
df_energy_saved.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 5 entries, 0 to 4
     Data columns (total 6 columns):
     The table gives the amount of energy saved in kilowatt hour (kWh) and the amount of
     Unnamed: 1
     Unnamed: 2
```

```
Unnamed: 3
Unnamed: 4
Unnamed: 5
dtypes: object(6)
memory usage: 320.0+ bytes
```

The "Energy Saved" dataframe contains some NULL values in its columns.
Each field also seems to contain string values.

Explore value distribution and sample values in each dataframe

i. Recycling from 2003 to 2017

```
df_recycle_before_18.describe()
```

	waste_disposed_of_tonne	total_waste_recycled_tonne	total_waste_generated
count	2.250000e+02	2.250000e+02	2.2500
mean	3.697191e+05	4.896987e+05	8.5941
std	6.842470e+05	9.607678e+05	1.5791
min	1.300000e+03	0.000000e+00	1.4400
25%	2.460000e+04	1.830000e+04	1.1840
50%	1.062000e+05	9.110000e+04	3.3240
75%	5.000000e+05	5.200000e+05	8.0980
max	3.045200e+06	4.825900e+06	7.8515

```
df_recycle_before_18.head()
```

	waste_type	waste_disposed_of_tonne	total_waste_recycled_tonne	total_
year				
2016	Food	679900	111100.0	
2016	Paper/Cardboard	576000	607100.0	
2016	Plastics	762700	59500.0	
2016	C&D	9700	1585700.0	
2016	Horticultural waste	111500	209000.0	

ii. Recycling from 2018 to 2019

✓ [ ] ↳ *2 cells hidden*

iii. Energy Saved

✓ [ ] ↳ *5 cells hidden*

iv. Fix the 'Energy Saved' Dataframe

✓ [ ] ↳ *12 cells hidden*

Combine the 2 Recycling Statistics Dataframes

Next, let us combine the 2 Recycling Statistics Dataframes to simplify our analysis.

To do so, we need to extract the relevant columns from the "Recycling from 2003 to 2017" dataframe. This is because the "Recycling from 2003 to 2017" dataframe has more columns relative to the "Recycling from 2018 to 2019" dataframe irrelevant to our analysis.

Once done, let us append the "Recycling from 2018 to 2019" to the smaller "Recycling from 2003 to 2017" dataframe.

```
#extract the relevant columns
df_recycle_before_18 = df_recycle_before_18[['waste_type', 'total_waste_generated_tonne',

#convert the units for total waste generated and recycled from tonne to 1000 tonnes
df_recycle_before_18['total_waste_generated_tonne'] = df_recycle_before_18['total_waste_ge
df_recycle_before_18['total_waste_recycled_tonne'] = df_recycle_before_18['total_waste_rec

#then adjust the column labels to follow the more informative labels in df_recycle_from_18
df_recycle_before_18.rename(columns={"waste_type": "Waste Type",
                                     "total_waste_generated_tonne": "Total Generated ('000
                                     "total_waste_recycled_tonne": "Total Recycled ('000 t
                            inplace=True)


#Check that column labels have been replaced as expected
df_recycle_before_18.columns
```

```
     Index(['Waste Type', 'Total Generated ('000 tonnes)',
            'Total Recycled ('000 tonnes)'],
           dtype='object')
```

```
#below df_recycle_before_18, append records from df_recycle_from_18
df_recycle_all = df_recycle_before_18.append(df_recycle_from_18)
df_recycle_all.sort_index()
#df_recycle_all.sort_values(by=['Waste Type'])
```

	Waste Type	Total Generated ('000 tonnes)	Total Recycled ('000 tonnes)
2003	Horticultural Waste	304.6	119.3
2003	Paper/Cardboard	1084.7	466.2
2003	Plastics	579.9	39.1
2003	Construction Debris	422.9	398.3
2003	Wood/Timber	213.4	40.8
2003	Ferrous Metals	856.7	799.0
2003	Food waste	548.0	32.9
2003	Non-ferrous Metals	93.9	75.8
2003	Sludge	88.5	0.0
2003	Glass	65.5	6.2
2003	Textile/Leather	91.6	0.9
2003	Scrap Tyres	14.4	6.2
2003	Others (stones, ceramics & rubber etc)	103.8	0.0
2003	Total	4728.2	2223.2
2003	Used Slag	260.3	238.5
2004	Construction Debris	509.0	471.0
2004	Plastics	683.1	74.1
2004	Total	4789.7	2307.1
2004	Food waste	531.1	31.1
2004	Wood/Timber	222.3	73.7
2004	Paper/Cardboard	1132.1	519.9
2004	Horticultural Waste	227.0	127.9
2004	Others	114.1	11.0
2004	Non-ferrous Metals	86.9	71.8
2004	Used Slag	267.2	259.6
2004	Sludge	93.9	0.0
2004	Glass	73.6	4.9
2004	Textile/Leather	114.5	5.0
2004	Ferrous Metals	720.2	649.9
2004	Scrap Tyres	14.7	7.2
...	...	...	...

2018	Ferrous Metal	1269.0	126.0
2018	Paper/Cardboard	1054.0	586.0
2018	Plastics	949.0	41.0
2018	Food	763.0	126.0
2018	Wood	521.0	428.0
2018	Horticultural	320.0	227.0
2018	Non-Ferrous Metal	171.0	170.0
2018	Textile/Leather	220.0	14.0
2018	Used Slag	181.0	179.0
2018	Glass	64.0	12.0
2018	Scrap Tyres	32.0	29.0
2018	Ash & Sludge	240.0	25.0
2018	Construction& Demolition	1624.0	1618.0
2018	Overall	7695.0	4726.0
2018	Others (stones, ceramic, rubber, ect)	286.0	11.0
2019	Overall	7234.0	4247.0
2019	Scrap Tyres	33.0	31.0
2019	Glass	75.0	11.0
2019	Non-Ferrous Metal	126.0	124.0
2019	Used Slag	129.0	127.0
2019	Textile/Leather	168.0	6.0
2019	Ash & Sludge	252.0	25.0
2019	Horticultural	400.0	293.0
2019	Wood	438.0	289.0
2019	Food	7440.0	136.0
2019	Plastics	930.0	37.0
2019	Paper/Cardboard	1011.0	449.0

Combine the full Recycling Statistics Dataframe with the Energy Saved Dataframe

```
#do an INNER JOIN based on the Recycling Statistics Dataframe 'Waste Type' values found in
#with df_recycle_all on the left and df_energy_saved_transposed on the right
full_energy_saved = df_recycle_all.reset_index().merge(df_energy_saved_transposed,
                                                       how="inner",
                                                        left_on="Waste Type",
                                                        right_on="material",
                                                        sort=True,
                                                        copy=True).set_index('index')

#we also need to remove paper, which the "Instructions" did not want
full_energy_saved = full_energy_saved[full_energy_saved["Waste Type"] != "Paper"]

#left_index=False,
#right_index=True,


full_energy_saved.head()
```

	Waste Type	Total Generated ('000 tonnes)	Total Recycled ('000 tonnes)	material	energy_saved	crude_oil saved	energy_sav
index							
2013	Ferrous Metal	1416.0	1369.2	Ferrous Metal	642 Kwh	1.8 barrels	
2012	Ferrous Metal	1386.0	1331.2	Ferrous Metal	642 Kwh	1.8 barrels	
	Ferrous			Ferrous			

```
#this was used when I performed a LEFT JOIN
#to subset for only the materials found in both dataframes

#with the INNER JOIN approach above, I no longer need to perform this step
#materialList = []

#for material in df_energy_saved_transposed['material']:
#    materialList.append(material)


#materialList
#full_energy_saved[full_energy_saved['Waste Type'].isin(materialList)]


full_energy_saved.tail()
```

	Waste Type	Total Generated ('000 tonnes)	Total Recycled ('000 tonnes)	material	energy_saved	crude_oil saved	energy_sav
index							
2019	Glass	75.0	11.0	Glass	42 Kwh	NaN	
2018	Glass	64.0	12.0	Glass	42 Kwh	NaN	
2019	Non-Ferrous Metal	126.0	124.0	Non-Ferrous Metal	14000 Kwh	40 barrels	

From the merged dataframe, we have:
-"**Waste Type**" (**and "material"**) values
-"**Total Recycled ('000 tonnes)**" values
-"**energy_saved_int**" integer values -> per ton

For each row, compute the total energy saved in kWh by multiplying "**Total Recycled ('000 tonnes)**" by a **factor of 1000** to unravel the "1000" tonne, and multiply with "**energy_saved_int**". Save the computed total energy saved in kWh for each row in a **new column "total_energy_saved**".

```
full_energy_saved['total_energy_saved'] = full_energy_saved["Total Recycled ('000 tonnes)"
```

```
full_energy_saved.head()
```

	Waste Type	Total Generated ('000 tonnes)	Total Recycled ('000 tonnes)	material	energy_saved	crude_oil saved	energy_sav
index							
2013	Ferrous Metal	1416.0	1369.2	Ferrous Metal	642 Kwh	1.8 barrels	
2012	Ferrous Metal	1386.0	1331.2	Ferrous Metal	642 Kwh	1.8 barrels	
	Ferrous			Ferrous			

```
full_energy_saved.reset_index().head()
```

	index	Waste Type	Total Generated ('000 tonnes)	Total Recycled ('000 tonnes)	material	energy_saved	crude_oil saved	energy_s
0	2013	Ferrous Metal	1416.0	1369.2	Ferrous Metal	642 Kwh	1.8 barrels	
1	2012	Ferrous Metal	1386.0	1331.2	Ferrous Metal	642 Kwh	1.8 barrels	
		Ferrous			Ferrous			

Aggregate total energy saved per year from 2015 to 2019

Perform a SUM of the **'total_energy_saved' values** across each year.

```
annual_energy_savings = full_energy_saved.groupby('index')['total_energy_saved'].sum()
```

Inspect the "annual_energy_savings" variable whether the contents look similar to the expected output requested in the Instructions panel.

```
print(annual_energy_savings)
```

```
     index
     2003    2.604000e+05
     2004    2.058000e+05
     2005    1.596000e+05
     2006    2.688000e+05
     2007    4.311036e+08
     2008    4.744950e+08
     2009    5.206542e+08
     2010    7.280439e+08
     2011    7.565808e+08
     2012    8.594430e+08
     2013    8.837472e+08
     2014    6.594000e+05
     2015    6.132000e+05
     2016    6.174000e+05
     2017    2.996140e+08
     2018    2.461774e+09
     2019    2.555612e+09
     Name: total_energy_saved, dtype: float64
```

Subset the dataframe to contain only the 'total_energy_saved' values from 2015 to 2019.

```
#annual_energy_savings[-5:]
annual_energy_savings = annual_energy_savings.tail(5)
```

```
print(annual_energy_savings)
```

```
     index
     2015    6.132000e+05
     2016    6.174000e+05
     2017    2.996140e+08
     2018    2.461774e+09
     2019    2.555612e+09
     Name: total_energy_saved, dtype: float64
```

Set the index column name to 'year'.

```
annual_energy_savings.index.name = 'year'
```

Set the value column name to 'total_energy_saved'.

```
annual_energy_savings.columns = ["total_energy_saved"]
```

Inspect the variable 'annual_energy_savings'.
It seems that the 'total_energy_saved' column name is not appearing after checking the last 5 rows.
After checking the data type of the variable 'annual_energy_savings', it seems that this is currently a pandas Series.

```
annual_energy_savings.tail()
```

```
year
2015    6.132000e+05
2016    6.174000e+05
2017    2.996140e+08
2018    2.461774e+09
2019    2.555612e+09
Name: total_energy_saved, dtype: float64
```

```
type(annual_energy_savings)
```

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

Therefore, let me change the variable 'annual_energy_savings' from a pandas Series to a dataframe.

```
annual_energy_savings = annual_energy_savings.to_frame()
```

```
type(annual_energy_savings)
```

```
pandas.core.frame.DataFrame
```

We now have a:

1. 'annual_energy_savings' dataframe
2. 'year' index column
3. 'annual_energy_savings' value in kWh column
4. show only values from 2015 to 2019

which follows what the Instructions panel wanted.

```
print(annual_energy_savings)
```

```
      total_energy_saved
year                    
2015        6.132000e+05
2016        6.174000e+05
2017        2.996140e+08
2018        2.461774e+09
2019        2.555612e+09
```

Yet I am encountering an issue after submitting this Jupyter notebook by clicking on "Check Project".

Some tests failed
TEST 1
There was an error while testing your code. Please double-check your submission.