Learning_Pandas_Part_9_MoreInPandas-3

June 20, 2021

0.0.1 Prepared by Abhishek Kumar

0.0.2 https://www.linkedin.com/in/abhishekkumar-0311/

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

%matplotlib inline
```

[4]:
```
# Setup : DataFrame creation

salary = [['1','Abhishek Kumar','AIML', 'Machine Learning Engineer','M', 'Y',␣
 ↪'04051990', 1121000],
          ['2','Arjun Kumar','DM', 'Tech Lead','M', 'Y', '09031992', 109000],
          ['3','Vivek Raj','DM', 'Devops Engineer','M', 'N', np.NaN , 827000],
          ['4','Mika Singh','DM', 'Data Analyst','F', 'Y', '15101991',  np.NaN],
          ['5','Anusha Yenduri','AIML', 'Data Scientist','F', 'Y', '01011989', ␣
 ↪921000],
          ['6','Ritesh Srivastava','AIML', 'Data Engineer','M', 'Y', np.NaN,␣
 ↪785000]]

columns_name=['Emp_Id','Emp_Name','Department','Role','Gender', 'WFH Status',␣
 ↪'DOB', 'Salary']

emp_df = pd.DataFrame(salary,columns=columns_name)
emp_df
```

[4]:

	Emp_Id	Emp_Name	Department	Role	Gender	\
0	1	Abhishek Kumar	AIML	Machine Learning Engineer	M	
1	2	Arjun Kumar	DM	Tech Lead	M	
2	3	Vivek Raj	DM	Devops Engineer	M	
3	4	Mika Singh	DM	Data Analyst	F	

```
4      5     Anusha Yenduri       AIML             Data Scientist      F
5      6  Ritesh Srivastava       AIML              Data Engineer      M

  WFH Status       DOB     Salary
0          Y  04051990  1121000.0
1          Y  09031992   109000.0
2          N       NaN   827000.0
3          Y  15101991        NaN
4          Y  01011989   921000.0
5          Y       NaN   785000.0
```

[5]:
```
import numpy as np
import pandas as pd
sample = {
'col_a':['Houston,TX', 'Dallas,TX', 'Chicago,IL', 'Phoenix,AZ',      'San␣
↪Diego,CA'],
'col_b':['62K-70K', '62K-70K', '69K-76K', '62K-72K', '71K-78K' ],
'col_c':['A','B','A','a','c'],
'col_d':['  1x', ' 1y', '2x  ', '1x', '1y  ']
}
df_sample = pd.DataFrame(sample)
df_sample
```

[5]:
```
          col_a    col_b col_c col_d
0    Houston,TX  62K-70K     A    1x
1     Dallas,TX  62K-70K     B    1y
2    Chicago,IL  69K-76K     A  2x
3    Phoenix,AZ  62K-72K     a    1x
4  San Diego,CA  71K-78K     c  1y
```

1 Functions discussed in this Notebook - Part 3

Function	Description	Part
apply()	Apply a function along an axis of the DataFrame.	1
applymap()	Apply a function to a Dataframe elementwise.	1
map()	map() is used to substitute each value in a Series with another value.	1
transform()	Call func on self producing a DataFrame with transformed values.	1

Function	Description	Part
df.assign()	Assign new columns to a DataFrame.	2
pipe()	Apply func(self, *args, **kwargs).	2
df.update()	Modify in place using non-NA values from another DataFrame.	2
df.take	Return the elements in the given positional indices along an axis.	2
df.truncate	Truncate a Series or DataFrame before and after some index value.	2

Function	Description	Part
df.items	Iterates over the DataFrame columns, returning a tuple with the column name and the content as a Series.	3
df.iteritems	Iterates over the DataFrame columns, returning a tuple with the column name and the content as a Series.	3
df.iterrows	Iterate over DataFrame rows as (index, Series) pairs.	3
df.itertuples	Iterate over DataFrame rows as namedtuples.	3

1.1 Description

This Part is about Iteration over Dataframe, be it rows or columns.

The explicit looping is Not as Efficient as the Implicit techniques.

The following blogs give a complete idea about looping Dataframes.

1.1.1 1. https://www.dataindependent.com/pandas/pandas-iterate-over-rows/

1.1.2 2. https://realpython.com/fast-flexible-pandas/

1.1.3 3. https://stackoverflow.com/questions/24870953/does-pandas-iterrows-have-performance-issues/24871316#24871316

1.2 Summary

- **Use `vectorized operations`: Pandas methods and functions with no for-loops.**
- **Use the `.apply()` method with a callable.**

- **Use `.itertuples()`: iterate over DataFrame rows as namedtuples from Python's collections module.**
- **Use `.iterrows()`: iterate over DataFrame rows as (index, pd.Series) pairs. While a Pandas Series is a flexible data structure, it can be costly to construct each row into a Series and then access it.**
- **Use "element-by-element" for loops, updating each cell or row one at a time with `df.loc` or `df.iloc`. (Or, .at/.iat for fast scalar access.)**

[ ]:

[ ]:

[ ]:

2 What's More..?? Upcoming HDFStore

2.1 Prevent Reprocessing with HDFStore

Pandas has a built-in solution for this which uses HDF5 , a high-performance storage format designed specifically for storing tabular arrays of data. Pandas' HDFStore class allows you to store your DataFrame in an HDF5 file so that it can be accessed efficiently, while still retaining column types and other metadata. It is a dictionary-like class, so you can read and write just as you would for a Python dict object.

Here's how you would go about storing your pre-processed DataFrame, df, in an HDF5 file

[7]:
```
# Create storage object with filename `processed_data`
data_store = pd.HDFStore('processed_data.h5')

# Put DataFrame into the object setting the key as 'preprocessed_df'
data_store['preprocessed_df'] = emp_df
data_store.close()
```

```
C:\Users\abhi0\anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3418: PerformanceWarning:
your performance may suffer as PyTables will pickle object types that it cannot
map directly to c-types [inferred_type->mixed,key->block1_values]
[items->Index(['Emp_Id', 'Emp_Name', 'Department', 'Role', 'Gender', 'WFH
Status',
       'DOB'],
      dtype='object')]

  exec(code_obj, self.user_global_ns, self.user_ns)
```

[4]:
```
# Access data store
data_store = pd.HDFStore('processed_data.h5')

# Retrieve data using key
```

```
preprocessed_emp_df = data_store['preprocessed_df']
data_store.close()
```

[5]:
```
preprocessed_emp_df
```

[5]:

	Emp_Id	Emp_Name	Department	Role	Gender	WFH Status	DOB	Salary
0	1	Abhishek Kumar	AIML	Machine Learning Engineer	M	Y	04051990	1121000.0
1	2	Arjun Kumar	DM	Tech Lead	M	Y	09031992	109000.0
2	3	Vivek Raj	DM	Devops Engineer	M	N	NaN	827000.0
3	4	Mika Singh	DM	Data Analyst	F	Y	15101991	NaN
4	5	Anusha Yenduri	AIML	Data Scientist	F	Y	01011989	921000.0
5	6	Ritesh Srivastava	AIML	Data Engineer	M	Y	NaN	785000.0

[ ]:
```

```