Author: Zoumana KEITA

BASICS FOR PANDAS

Consider the following Python dictionary data and Python list labels:

data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills', 'Cranes', 'plovers', 'Cranes', 'spoonbills', 'spoonbills'], 'age': [3.5, 4, 1.5, np.nan, 6, 3, 5.5, np.nan, 8, 4], 'visits': [2, 4, 3, 4, 3, 4, 2, 2, 3, 2], 'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

0. Import libraries

Entrée [34]:

```
import numpy as np
import pandas as pd
```

1. Create a DataFrame birds from this dictionary data which has the index labels.

Entrée [35]:

```
data = {'birds': ['Cranes', 'Cranes', 'plovers', 'spoonbills', 'spoonbills', 'Cranes', 'plo
        'age': [3.5, 4, 1.5, np.nan, 6, 3, 5.5, np.nan, 8, 4],
        'visits': [2, 4, 3, 4, 3, 4, 2, 2, 3, 2],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes', 'no', 'no']}

# Creating the DataFrame
panda_data_frame = pd.DataFrame.from_dict(data)
panda_data_frame
```

Out[35]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
1	Cranes	4.0	4	yes
2	plovers	1.5	3	no
3	spoonbills	NaN	4	yes
4	spoonbills	6.0	3	no
5	Cranes	3.0	4	no
6	plovers	5.5	2	no
7	Cranes	NaN	2	yes
8	spoonbills	8.0	3	no
9	spoonbills	4.0	2	no

2. Display a summary of the basic information about birds DataFrame and its data.

Entrée [36]:

```
'''
This code below will give me some statistical information about the data frame
but there are some missing data expressed by "NaN" that could include a bias in
my statistical informations.
'''
panda_data_frame.describe(include='all')
```

Out[36]:

	birds	age	visits	priority
count	10	8.000000	10.000000	10
unique	3	NaN	NaN	2
top	spoonbills	NaN	NaN	no
freq	4	NaN	NaN	6
mean	NaN	4.437500	2.900000	NaN
std	NaN	2.007797	0.875595	NaN
min	NaN	1.500000	2.000000	NaN
25%	NaN	3.375000	2.000000	NaN
50%	NaN	4.000000	3.000000	NaN
75%	NaN	5.625000	3.750000	NaN
max	NaN	8.000000	4.000000	NaN

**3. Print the first 2 rows of the birds dataframe **

Entrée [37]:

```
'''
The head(x) function allows me to print 'x' first rows of dataframe
'''
panda_data_frame.head(2)
```

Out[37]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
1	Cranes	4.0	4	yes

4. Print all the rows with only 'birds' and 'age' columns from the dataframe

Entrée [38]:

```
'''
I creat a variable called colums_to_get which the list containing all the
columns I want to get from my data frame.
'''
colums_to_get = ['birds', 'age']
print(panda_data_frame[colums_to_get])
```

```
        birds  age
0      Cranes  3.5
1      Cranes  4.0
2     plovers  1.5
3  spoonbills  NaN
4  spoonbills  6.0
5      Cranes  3.0
6     plovers  5.5
7      Cranes  NaN
8  spoonbills  8.0
9  spoonbills  4.0
```

5. select [2, 3, 7] rows and in columns ['birds', 'age', 'visits']

Entrée [39]:

```
# The columns I want to get.
colums_to_get = ['birds', 'age', 'visits']

# The indexes of the data I want to get.
indexes = [2,3,7]

#Create a new dataframe that will contain the final result.
reduced_data_frame = panda_data_frame[colums_to_get]

# Final result for the question
reduced_data_frame.iloc[indexes]
```

Out[39]:

	birds	age	visits
2	plovers	1.5	3
3	spoonbills	NaN	4
7	Cranes	NaN	2

6. select the rows where the number of visits is less than 4

Entrée [40]:

```
panda_data_frame[panda_data_frame['visits']<4]
```

Out[40]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
2	plovers	1.5	3	no
4	spoonbills	6.0	3	no
6	plovers	5.5	2	no
7	Cranes	NaN	2	yes
8	spoonbills	8.0	3	no
9	spoonbills	4.0	2	no

7. select the rows with columns ['birds', 'visits'] where the age is missing i.e NaN

Entrée [41]:

```
# The columns I want to get.
colums_to_get = ['birds','visits', 'age']

# Create a new dataframe that will contain the final result.
nan_data_frame = panda_data_frame[colums_to_get]

# I get all the rows (.any(axis=1)) where there is a NaN value (.isnull())
final_data_frame = nan_data_frame[nan_data_frame.isnull().any(axis=1)][['birds','visits']]

'''
I could get the result in one line like below or using an intermediate
variable called final_data_frame like previously
'''
#nan_data_frame[nan_data_frame.isnull().any(axis=1)][['birds','visits']]

# The final result
final_data_frame[['birds','visits']]
```

Out[41]:

	birds	visits
3	spoonbills	4
7	Cranes	2

8. Select the rows where the birds is a Cranes and the age is less than 4

Entrée [42]:

```
panda_data_frame[(panda_data_frame['birds']=='Cranes') & (panda_data_frame['age']<4)]
```

Out[42]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
5	Cranes	3.0	4	no

9. Select the rows the age is between 2 and 4(inclusive)

Entrée [43]:

```
panda_data_frame[(panda_data_frame['age']>2) & (panda_data_frame['age']<=4)]
```

Out[43]:

	birds	age	visits	priority
0	Cranes	3.5	2	yes
1	Cranes	4.0	4	yes
5	Cranes	3.0	4	no
9	spoonbills	4.0	2	no

10. Find the total number of visits of the bird Cranes

Entrée [44]:

```
# Create a new data frame with only 'Cranes'
cranes_dataframe = panda_data_frame[panda_data_frame['birds']=='Cranes']

# Calculate the total number of visits
np.sum(cranes_dataframe['visits'])
```

Out[44]:

```
12
```

11. Calculate the mean age for each different birds in dataframe.

Entrée [45]:

```
'''
To operate the mean operation, I think that it is important to handle NaN value
since some of my rows contain those missing values for the 'age' column.
'''
no_nan_data_frame = panda_data_frame[panda_data_frame['age'].notnull()]

# I can finally operate the mean
no_nan_data_frame.groupby(['birds'])['age'].agg(lambda x: x.unique().mean())
```

Out[45]:

```
birds
Cranes        3.5
plovers       3.5
spoonbills    6.0
Name: age, dtype: float64
```

12. Append a new row 'k' to dataframe with your choice of values for each column. Then delete that row to return the original DataFrame.

Entrée [46]:

```
# Adding the new row
panda_data_frame = panda_data_frame.append({'birds': 'Cranes', 'age': 4.5, 'visits': 5, 'pr
                        ignore_index=True)

print('Data frame with the new row\n', panda_data_frame)

# Removing the new row
panda_data_frame = panda_data_frame.drop(panda_data_frame.index[10])
print('\nOriginal data frame after removing the new row\n', panda_data_frame)
```

```
Data frame with the new row
          birds  age  visits priority
0       Cranes  3.5       2      yes
1       Cranes  4.0       4      yes
2      plovers  1.5       3       no
3   spoonbills  NaN       4      yes
4   spoonbills  6.0       3       no
5       Cranes  3.0       4       no
6      plovers  5.5       2       no
7       Cranes  NaN       2      yes
8   spoonbills  8.0       3       no
9   spoonbills  4.0       2       no
10      Cranes  4.5       5      yes

Original data frame after removing the new row
         birds  age  visits priority
0      Cranes  3.5       2      yes
1      Cranes  4.0       4      yes
2     plovers  1.5       3       no
3  spoonbills  NaN       4      yes
4  spoonbills  6.0       3       no
5      Cranes  3.0       4       no
6     plovers  5.5       2       no
7      Cranes  NaN       2      yes
8  spoonbills  8.0       3       no
9  spoonbills  4.0       2       no
```

13. Find the number of each type of birds in dataframe (Counts)

Entrée [47]:

```
panda_data_frame.groupby(['birds'])['birds'].count()
```

Out[47]:

```
birds
Cranes        4
plovers       2
spoonbills    4
Name: birds, dtype: int64
```

14. Sort dataframe (birds) first by the values in the 'age' in decending order, then by the value in the 'visits' column in ascending order.

Method N°1: Sort the two columns separatly

Entrée [48]:

```
# Sort the 'age' column in descending order
panda_data_frame.sort_values(['age'], ascending=False, inplace=True)
panda_data_frame
```

Out[48]:

	birds	age	visits	priority
8	spoonbills	8.0	3	no
4	spoonbills	6.0	3	no
6	plovers	5.5	2	no
1	Cranes	4.0	4	yes
9	spoonbills	4.0	2	no
0	Cranes	3.5	2	yes
5	Cranes	3.0	4	no
2	plovers	1.5	3	no
3	spoonbills	NaN	4	yes
7	Cranes	NaN	2	yes

Entrée [49]:

```
# Sort the 'visits' column in ascending order
panda_data_frame.sort_values(['visits'], ascending=True, inplace=True)
panda_data_frame
```

Out[49]:

	birds	age	visits	priority
6	plovers	5.5	2	no
9	spoonbills	4.0	2	no
0	Cranes	3.5	2	yes
7	Cranes	NaN	2	yes
8	spoonbills	8.0	3	no
4	spoonbills	6.0	3	no
2	plovers	1.5	3	no
1	Cranes	4.0	4	yes
5	Cranes	3.0	4	no
3	spoonbills	NaN	4	yes

Method N°2: Sort the two columns still remaining in the same data frame.

Entrée [50]:

```
panda_data_frame.sort_values(['age','visits'],
                             ascending=[False, True], inplace=True)
panda_data_frame
```

Out[50]:

	birds	age	visits	priority
8	spoonbills	8.0	3	no
4	spoonbills	6.0	3	no
6	plovers	5.5	2	no
9	spoonbills	4.0	2	no
1	Cranes	4.0	4	yes
0	Cranes	3.5	2	yes
5	Cranes	3.0	4	no
2	plovers	1.5	3	no
7	Cranes	NaN	2	yes
3	spoonbills	NaN	4	yes

15. Replace the priority column values with'yes' should be 1 and 'no' should be 0

Entrée [16]:

```
panda_data_frame.priority.replace(['yes', 'no'], [1, 0], inplace=True)
```

16. In the 'birds' column, change the 'Cranes' entries to 'trumpeters'.

Entrée [17]:

```
panda_data_frame.replace({'birds': {'Cranes': 'trumpeters'}})
```

Out[17]:

	birds	age	visits	priority
0	trumpeters	3.5	2	1
1	trumpeters	4.0	4	1
2	plovers	1.5	3	0
3	spoonbills	NaN	4	1
4	spoonbills	6.0	3	0
5	trumpeters	3.0	4	0
6	plovers	5.5	2	0
7	trumpeters	NaN	2	1
8	spoonbills	8.0	3	0
9	spoonbills	4.0	2	0

Entrée [ ]: