CIS-2266 Pandas dropbox

- Complete the exercise below
- Output completed notebook to HTML or PDF
- Upload to the Dropbox for grading

Importing pandas

Getting started and checking your pandas setup

Difficulty: *easy*

1. Import pandas under the name `pd`. (1 point)

```
In [1]: import pandas as pd
```

2. Print the version of pandas that has been imported. (1 point)

```
In [2]: print('Pandas version ' , pd.__version__)
```

```
Pandas version  1.1.3
```

DataFrame basics

A few of the fundamental routines for selecting, sorting, adding and aggregating data in DataFrames

Difficulty: *easy*

Consider the following Python dictionary `data` and Python list `labels` :

```
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat',
'snake', 'cat', 'dog', 'dog'],
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'n
o', 'yes', 'no', 'no']}

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

3. Create a DataFrame `df` from this dictionary `data` which has the index `labels` . (1 point)

In [3]:

```
#Use:
import numpy as np
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake'
        'age': [2.5, 3, 0.5, np.nan, 5, 2, 4.5, np.nan, 7, 3],
        'visits': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'priority': ['yes', 'yes', 'no', 'yes', 'no', 'no', 'no', 'yes

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

# Code Goes Below:
df = pd.DataFrame(data, index =labels)
print(df)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  2.0       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

4. Display a summary of the basic information about this DataFrame and its data. (1 point)

In [4]:

```
print("Summary of the basic info about DataFrame and its data: ", df.i
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   animal    10 non-null     object 
 1   age       8 non-null      float64
 2   visits    10 non-null     int64  
 3   priority  10 non-null     object 
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
Summary of the basic info about DataFrame and its data:  None
```

5. Return the first 3 rows of the DataFrame `df`. (1 point)

In [5]:
```
print(df.iloc[:3])
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
```

6. Select just the 'animal' and 'age' columns from the DataFrame `df`. (1 point)

In [6]:
```
print(df[['animal','age']])
```

```
  animal  age
a    cat  2.5
b    cat  3.0
c  snake  0.5
d    dog  NaN
e    dog  5.0
f    cat  2.0
g  snake  4.5
h    cat  NaN
i    dog  7.0
j    dog  3.0
```

7. Select the data in rows `[3, 4, 8]` *and* in columns `['animal', 'age']`. (1 point)

In [7]:
```
print(df.iloc[[3,4,8], [0,1]])
```

```
  animal  age
d    dog  NaN
e    dog  5.0
i    dog  7.0
```

8. Select only the rows where the number of visits is greater than 3. (1 point)

In [8]:
```
print("Rows where number of visits >3: ")
print(df[df['visits']>3])
```

```
Rows where number of visits >3: 
Empty DataFrame
Columns: [animal, age, visits, priority]
Index: []
```

9. Select the rows where the age is missing, i.e. is `NaN`. (1 point)

```
In [9]: print(df[df['age'].isnull()])
```

```
  animal  age  visits priority
d    dog  NaN       3      yes
h    cat  NaN       1      yes
```

10. Select the rows where the animal is a cat *and* the age is less than 3. (1 point)

```
In [10]: print(df[(df['animal'] == 'cat') & (df['age'] < 3)])
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
f    cat  2.0       3       no
```

11. Select the rows the age is between 2 and 4 (inclusive). (1 point)

```
In [11]: print(df[df['age'].between(2,4)])
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
f    cat  2.0       3       no
j    dog  3.0       1       no
```

12. Change the age in row 'f' to 1.5. (1 point)

```
In [12]: df.loc['f', 'age'] = 1.5
         print(df)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

13. Calculate the sum of all visits (the total number of visits). (1 point)

In [13]:

```
print(df['visits'].sum())
```

```
19
```

14. Calculate the mean age for each different animal in `df` . (1 point)

In [14]:

```
print("mean :" , df['age'].mean())
```

```
mean : 3.375
```

15. Append a new row 'k' to `df` with your choice of values for each column. Then delete that row to return the original DataFrame. (1 point)

In [15]:

```
df.loc['k'] = ['horse', 5.5, 2, 'yes']
print(df)

df = df.drop('k')
print("\nOriginal DataFrame:")
print(df)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
k  horse  5.5       2      yes

Original DataFrame:
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

16. Count the number of each type of animal in `df` . (1 point)

In [ ]:

```
```

17. Sort `df` first by the values in the 'age' in *decending* order, then by the value in the 'visit' column in *ascending* order. (1 point)

In [16]:

```
df.sort_values(by=['age', 'visits'], ascending=[False, True])
print(df)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

18. The 'priority' column contains the values 'yes' and 'no'. Replace this column with a column of boolean values: 'yes' should be `True` and 'no' should be `False` . (1 point)

In [17]:

```
print(df['priority'].map({'yes': True, 'no':False}))
```

```
a     True
b     True
c    False
d     True
e    False
f    False
g    False
h     True
i    False
j    False
Name: priority, dtype: bool
```

19. In the 'animal' column, change the 'snake' entries to 'python'. (1 point)

```
In [18]: df['animal'].replace('snake', 'python')
         print(df)
```

```
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
d    dog  NaN       3      yes
e    dog  5.0       2       no
f    cat  1.5       3       no
g  snake  4.5       1       no
h    cat  NaN       1      yes
i    dog  7.0       2       no
j    dog  3.0       1       no
```

20. Produce a sum of each 'priority' for each animal type . (1 point)

```
In [19]: print(df['priority'].sum())
```

```
yesyesnoyesnononoyesnono
```

DataFrames: beyond the basics

Slightly trickier: you may need to combine two or more methods to get the right answer

Difficulty: *medium*

The previous section was a tour through some basic but essential DataFrame operations. Below are some ways that you might need to cut your data, but for which there is no single "out of the box" method.

21. You have a DataFrame `df` with a column 'A' of integers. For example:

```
df = pd.DataFrame({'A': [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 7]})
```

How do you filter out rows which contain the same integer as the row immediately above? (2 points)

In [20]:
```
#Use:
df = pd.DataFrame({'A': [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 7]})

#Code goes below:

```

22. Given a DataFrame of numeric values, say

```
df = pd.DataFrame(np.random.random(size=(5, 3))) # a 5x3 frame of float values
```

How do you subtract the row mean from each element in the row? (2 points)

In [21]:
```
#Use:
df = pd.DataFrame(np.random.random(size=(5, 3)))

#Code goes below:

```

23. Suppose you have DataFrame with 10 columns of real numbers, for example:

```
df = pd.DataFrame(np.random.random(size=(5, 10)), columns=list('abcdefghij'))
```

Which column of numbers has the smallest sum? (Find that column's label.) (2 points)

In [22]:
```
#Use:
df = pd.DataFrame(np.random.random(size=(5, 10)), columns=list('abcdef

#Code goes below:

```

24. How do you count how many unique rows in DataFrame `df` (i.e. ignore all rows that are duplicates)? (2 points)

In [ ]:
```

```

25. Using the DataFrame `df` from # 24...

Create a new DataFrame `df2` of just column 'a' where all values are rounded to the second decimal and output `df2`. (2 points)

In [ ]:

End

In [ ]: