Importing pandas

Getting started and checking your pandas setup

1. Import pandas under the alias `pd`.

In [101…
```
import pandas as pd
import numpy as np
```

2. Print the version of pandas that has been imported.

In [102…
```
print(pd. __version__)
```

```
1.5.3
```

3. Try checking for the help of any of the function in pandas.

In [ ]:
```
```

DataFrame basics

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

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', 'no', 'yes',
'no', 'no']}

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

4. Create a DataFrame `df` from this dictionary `data` which has the index `labels`.

In [103…
```
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', 'do
        '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', 'no', 'no
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

In [104…
```
df=pd.DataFrame(data,index=labels)
```

In [105…
```
df.head()
```

Out[105]:

	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

5. Display a summary of the basic information about this DataFrame and its data (*hint: there is a single method that can be called on the DataFrame*).

In [106…
```
df.describe(include='all')
```

Out[106]:

	animal	age	visits	priority
count	10	8.000000	10.000000	10
unique	3	NaN	NaN	2
top	cat	NaN	NaN	no
freq	4	NaN	NaN	6
mean	NaN	3.437500	1.900000	NaN
std	NaN	2.007797	0.875595	NaN
min	NaN	0.500000	1.000000	NaN
25%	NaN	2.375000	1.000000	NaN
50%	NaN	3.000000	2.000000	NaN
75%	NaN	4.625000	2.750000	NaN
max	NaN	7.000000	3.000000	NaN

6. Return the first 3 rows of the DataFrame `df`.

In [107…
```
df.head(3)
```

Out[107]:

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

7. Select just the 'animal' and 'age' columns from the DataFrame `df`.

In [108…
```
df[['animal','age']]
```

Out[108]:

	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

8. Select the data in rows `[3, 4, 8]` *and* in columns `['animal', 'age']`.

In [109…
```
df.loc[['c', 'd', 'h'],['animal', 'age']]
```

Out[109]:

	animal	age
c	snake	0.5
d	dog	NaN
h	cat	NaN

9. Select only the rows where the number of visits is greater than 3.

In [110…
```
df[df.visits>3]
```

Out[110]:

	animal	age	visits	priority

10. Check for missing values in the data.

In [111…
```
df.isnull().sum()
```

Out[111]:
```
animal      0
age         2
visits      0
priority    0
dtype: int64
```

11. Select the rows where the animal is a cat *and* the age is less than 3.

In [112…
```
df[(df.animal == 'cat') & (df.age<3)]
```

Out[112]:

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

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

In [113…
```
df[df['age'].between(2, 4)]
```

Out[113]:

	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

13. Change the age in row 'f' to 1.5.

In [114…
```
df.age.loc['f']=1.5
```

```
/tmp/ipykernel_32/371777984.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.age.loc['f']=1.5
```

In [115…
```
df
```

Out[115]:

	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

14. Calculate the sum of all visits in `df` (i.e. find the total number of visits).

In [116…
```
df.visits.sum()
```

Out[116]:
```
19
```

15. Calculate the mean age for each different animal in `df`. Explore the groupby function.

In [117…
```
df.groupby(['animal']).mean()
```

```
/tmp/ipykernel_32/229052335.py:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  df.groupby(['animal']).mean()
```

Out[117]:

animal	age	visits
cat	2.333333	2.0
dog	5.000000	2.0
snake	2.500000	1.5

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

In [ ]:

```
```

17. Count the number of each type of animal in `df`.

In [118…

```
df.groupby('animal').animal.count()
```

Out[118]:

```
animal
cat      4
dog      4
snake    2
Name: animal, dtype: int64
```

18. Sort `df` first by the values in the 'age' in *decending* order, then by the value in the 'visit' column in *ascending* order (so row `i` should be first, and row `d` should be last).

In [119…

```
df.sort_values(by='age')
```

Out[119]:

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

19. 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`.

In [120…

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

In [121…

```
df['priority']
```

Out[121]:
```
a     True
b     True
c    False
d     True
e    False
f    False
g    False
h     True
i    False
j    False
Name: priority, dtype: bool
```

20. In the 'animal' column, change the 'snake' entries to 'python'.

In [122…
```
df
```

Out[122]:

	animal	age	visits	priority
a	cat	2.5	1	True
b	cat	3.0	3	True
c	snake	0.5	2	False
d	dog	NaN	3	True
e	dog	5.0	2	False
f	cat	1.5	3	False
g	snake	4.5	1	False
h	cat	NaN	1	True
i	dog	7.0	2	False
j	dog	3.0	1	False

In [123…
```
df['animal'] = df['animal'].replace('snake', 'python')
```

In [124…
```
df['animal']
```

Out[124]:
```
a       cat
b       cat
c    python
d       dog
e       dog
f       cat
g    python
h       cat
i       dog
j       dog
Name: animal, dtype: object
```

In [ ]:
```

```