Install and import pandas, checking setup

Getting started and checking your pandas setup Difficulty: easy

In [2]:

```
# 0. Install pandas.
%pip install pandas

# 1. Import pandas under the alias pd.
import pandas as pd

# 2. Print the version of pandas that has been imported.
print(pd.__version__)

# 3. Print out all the version information of the libraries that are required by the pand
as library.
print(pd.show_versions())
```

```
Requirement already satisfied: pandas in d:\projects\mlprojects\tutor-kak-arief\.venv\lib
\site-packages (2.2.2)
Requirement already satisfied: numpy>=1.22.4 in d:\projects\mlprojects\tutor-kak-arief\.v
env\lib\site-packages (from pandas) (2.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\projects\mlprojects\tutor-kak
-arief\.venv\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in d:\projects\mlprojects\tutor-kak-arief\.ve
nv\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in d:\projects\mlprojects\tutor-kak-arief\.
venv\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in d:\projects\mlprojects\tutor-kak-arief\.venv\l
ib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
2.2.2

INSTALLED VERSIONS
------------------
commit                : d9cdd2ee5a58015ef6f4d15c7226110c9aab8140
python                : 3.10.0.final.0
python-bits           : 64
OS                    : Windows
OS-release            : 10
Version               : 10.0.19045
machine               : AMD64
processor             : AMD64 Family 23 Model 96 Stepping 1, AuthenticAMD
byteorder             : little
LC_ALL                : None
LANG                  : None
LOCALE                : English_United States.1252

pandas                : 2.2.2
numpy                 : 2.1.0
pytz                  : 2024.1
dateutil              : 2.9.0.post0
setuptools            : 57.4.0
pip                   : 24.2
Cython                : None
pytest                : None
hypothesis            : None
sphinx                : None
blosc                 : None
feather               : None
xlsxwriter            : None
lxml.etree            : None
html5lib              : None
pymysql               : None
psycopg2              : None
jinja2                : 3.1.4
IPython               : 8.26.0
IPython               : 8.26.0
pandas_datareader     : None
adbc-driver-postgresql: None
adbc-driver-sqlite    : None
bs4                   : 4.12.3
bottleneck            : None
dataframe-api-compat  : None
fastparquet           : None
fsspec                : None
gcsfs                 : None
matplotlib            : None
numba                 : None
numexpr               : None
odfpy                 : None
openpyxl              : None
pandas_gbq            : None
pyarrow               : None
pyreadstat            : None
python-calamine       : None
pyxlsb                : None
s3fs                  : None
scipy                 : None
sqlalchemy            : None
tables                : None
tabulate              : None
xarray                : None
xlrd                  : None
zstandard             : None
tzdata                : 2024.1
qtpy                  : None
pyqt5                 : None
None
```

DataFrame Basics

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

In [3]:

```
import numpy as np

# 4. Create a DataFrame df from this dictionary data which has the index labels
data = {'animal': ['cat', 'cat', 'snake', 'dog', 'dog', 'cat', 'snake', 'cat', 'dog', 'd
og'],
        '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']

df = pd.DataFrame(data, index=labels)

# 5. Display a summary of the basic information about this DataFrame and its data
print(df.info())

# 6. Return the first 3 rows of the DataFrame df
print(df.head(3))

# 7. Select just the 'animal' and 'age' columns from the DataFrame df
print(df[['animal', 'age']])

# 8. Select the data in rows [3, 4, 8] and in columns ['animal', 'age']
print(df.loc[df.index[[3, 4, 8]], ['animal', 'age']])

# 9. Select only the rows where the number of visits is greater than 3
print(df[df['visits'] > 3])

# 10. Select the rows where the age is missing, i.e. it is NaN
print(df[df['age'].isna()])
```

```python
# 11. Select the rows where the animal is a cat and the age is less than 3
print(df[(df['animal'] == 'cat') & (df['age'] < 3)])

# 12. Select the rows where the age is between 2 and 4 (inclusive)
print(df[df['age'].between(2, 4)])

# 13. Change the age in row ‘f’ to 1.5
df.loc['f', 'age'] = 1.5
print(df)

# 14. Calculate the sum of all visits in df (i.e. find the total number of visits)
print(df['visits'].sum())

# 15. Calculate the mean age for each different animal in df
print(df.groupby('animal')['age'].mean())

# 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
df.loc['k'] = ['dog', 5.5, 2, 'yes']
print(df)
df = df.drop('k')
print(df)

# 17. Count the number of each type of animal in df
print(df['animal'].value_counts())

# 18. Sort df first by the values in the ‘age’ in descending order, then by the value in
the ‘visits’ column in ascending order
print(df.sort_values(by=['age', 'visits'], ascending=[False, True]))

# 19. Replace the 'priority' column with boolean values: ‘yes’ should be True and ‘no’ sh
ould be False
df['priority'] = df['priority'].map({'yes': True, 'no': False})
print(df)

# 20. In the ‘animal’ column, change the ‘snake’ entries to ‘python’
df['animal'] = df['animal'].replace('snake', 'python')
print(df)

# 21. For each animal type and each number of visits, find the mean age (use pivot table)
print(df.pivot_table(index='animal', columns='visits', values='age', aggfunc='mean'))
```

```
<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
None
  animal  age  visits priority
a    cat  2.5       1      yes
b    cat  3.0       3      yes
c  snake  0.5       2       no
  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
  animal  age
d    dog  NaN
e    dog  5.0
```

```
i    dog  7.0
Empty DataFrame
Columns: [animal, age, visits, priority]
Index: []
  animal  age  visits priority
d    dog  NaN       3      yes
h    cat  NaN       1      yes
  animal  age  visits priority
a    cat  2.5       1      yes
f    cat  2.0       3       no
  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
  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
19
animal
cat      2.333333
dog      5.000000
snake    2.500000
Name: age, dtype: float64
  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    dog  5.5       2      yes
  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
animal
cat      4
dog      4
snake    2
Name: count, dtype: int64
  animal  age  visits priority
i    dog  7.0       2       no
e    dog  5.0       2       no
g  snake  4.5       1       no
j    dog  3.0       1       no
b    cat  3.0       3      yes
a    cat  2.5       1      yes
f    cat  1.5       3       no
c  snake  0.5       2       no
h    cat  NaN       1      yes
d    dog  NaN       3      yes
  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
   animal  age  visits  priority
a      cat  2.5       1      True
b      cat  3.0       3      True
c   python  0.5       2     False
d      dog  NaN       3      True
e      dog  5.0       2     False
f      cat  1.5       3     False
g   python  4.5       1     False
h      cat  NaN       1      True
i      dog  7.0       2     False
j      dog  3.0       1     False
visits    1    2     3
animal
cat     2.5  NaN  2.25
dog     3.0  6.0   NaN
python  4.5  0.5   NaN
```

DataFrames: Beyond the Basics

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

In [6]:

```python
# 22. Filter out rows which contain the same integer as the row immediately above
df = pd.DataFrame({'A': [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 7]})
df_filtered = df.loc[df['A'].shift() != df['A']]
print(df_filtered)

# 23. Subtract the row mean from each element in the row
df = pd.DataFrame(np.random.random(size=(5, 3)))
df_subtracted = df.sub(df.mean(axis=1), axis=0)
print(df_subtracted)

# 24. Return the column label with the smallest sum
df = pd.DataFrame(np.random.random(size=(5, 10)), columns=list('abcdefghij'))
smallest_sum_col = df.sum().idxmin()
print(smallest_sum_col)

# 25. Count unique rows ignoring duplicates
df = pd.DataFrame(np.random.randint(0, 2, size=(10, 3)))
unique_rows = df.drop_duplicates().shape[0]
print(unique_rows)

# 26. Find the column which contains the third NaN value for each row
nan = np.nan
data = [[0.04,  nan,  nan, 0.25,  nan, 0.43, 0.71, 0.51,  nan,  nan],
        [ nan,  nan,  nan, 0.04, 0.76,  nan,  nan, 0.67, 0.76, 0.16],
        [ nan,  nan, 0.5 ,  nan, 0.31, 0.4 ,  nan,  nan, 0.24, 0.01],
        [0.49,  nan,  nan, 0.62, 0.73, 0.26, 0.85,  nan,  nan,  nan],
        [ nan,  nan, 0.41,  nan, 0.05,  nan, 0.61,  nan, 0.48, 0.68]]
columns = list('abcdefghij')
df = pd.DataFrame(data, columns=columns)
third_nan = df.apply(lambda x: x.isna().cumsum().eq(3).idxmax(), axis=1)
print(third_nan)

# 27. Find the sum of the three greatest values for each group
df = pd.DataFrame({'grps': list('aaabbcaabcccbbc'),
                   'vals': [12,345,3,1,45,14,4,52,54,23,235,21,57,3,87]})

# Group by 'grps', then use `nlargest` and aggregate the sum
```

```python
result = df.groupby('grps')['vals'].apply(lambda x: x.nlargest(3).sum())
print(result)


# 28. For each group of 10 consecutive integers in 'A', calculate the sum of the corresponding values in column 'B'
df = pd.DataFrame(np.random.RandomState(8765).randint(1, 101, size=(100, 2)), columns = ["A", "B"])

# Setting observed=True to adopt the future default behavior
result = df.groupby(pd.cut(df['A'], bins=range(0, 101, 10)), observed=True)['B'].sum()
print(result)
```

```
   A
0  1
1  2
3  3
4  4
5  5
8  6
9  7
          0         1         2
0 -0.260180 -0.322444  0.582624
1  0.490632 -0.288122 -0.202510
2 -0.429190  0.166004  0.263186
3 -0.242244  0.017533  0.224711
4 -0.003536 -0.260068  0.263604
a
5
0    e
1    c
2    d
3    h
4    d
dtype: object
grps
a    409
b    156
c    345
Name: vals, dtype: int64
A
(0, 10]      635
(10, 20]     360
(20, 30]     315
(30, 40]     306
(40, 50]     750
(50, 60]     284
(60, 70]     424
(70, 80]     526
(80, 90]     835
(90, 100]    852
Name: B, dtype: int32
```