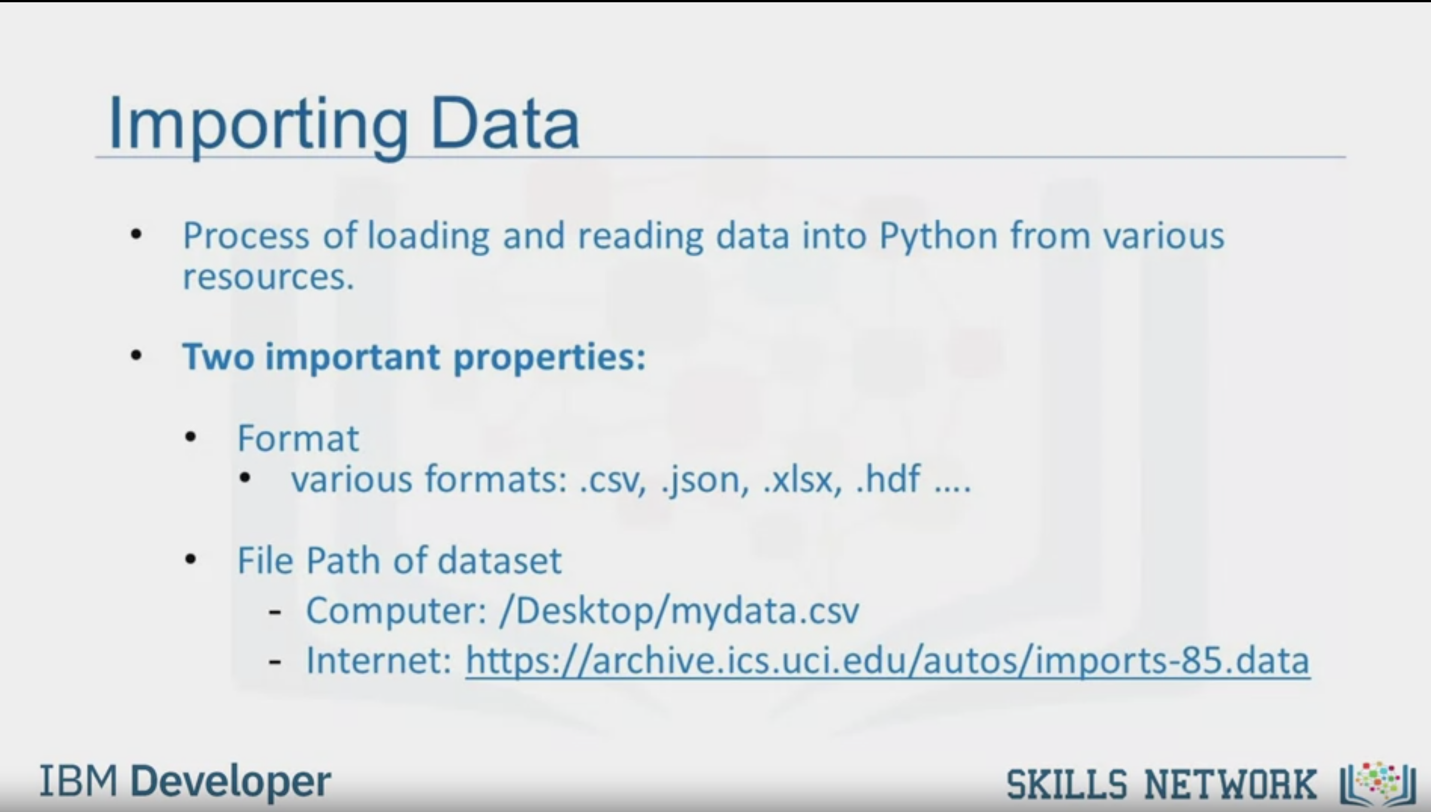


Importing and Exporting Data

in Python

IBM Developer

SKILLS NETWORK



Importing Data

Process of loading and reading data into Python from various

resources.

Two important properties:

Format

various formats: .csv, json, xlsx, .hdf ....

File Path of dataset

Computer: /Desktop/mydata.csv

Internet: https://archive.ics.uci.edu/autos/imports-85.data

IBM Developer

SKILLS NETWORK



Getting Data

data source : https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data

IBM Developer

SKILLS NETWORK



Importing a CSV into Python

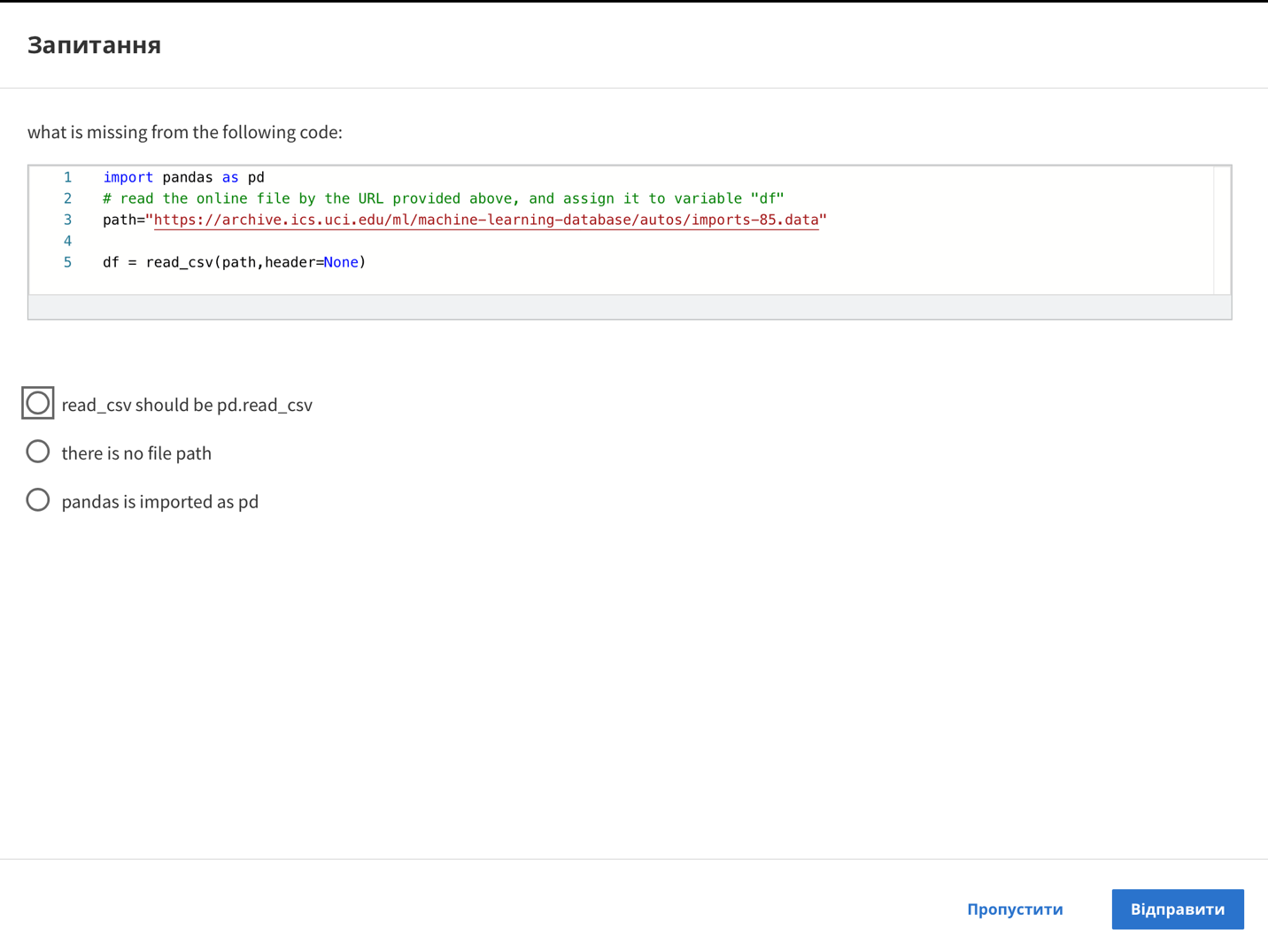
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learningdatabases/autos/imports-85.data"

df = pd.read\_csv (url)

IBM Developer

SKILLS NETWORK



Question

what is missing from the following code:

1 import pandas as pd

2 # read the online file by the URL provided above, and assign it to variable "df"

3 path="https://archive.ics.uci.edu/ml/machine-learning-database/autos/imports-85.data"

4

5 df = read\_csv (path, header=None)

read\_csv should be pd.read\_csv

there is no file path

pandas is imported as pd

Skip

Send



Importing a CSV without a header

import pandas as pd

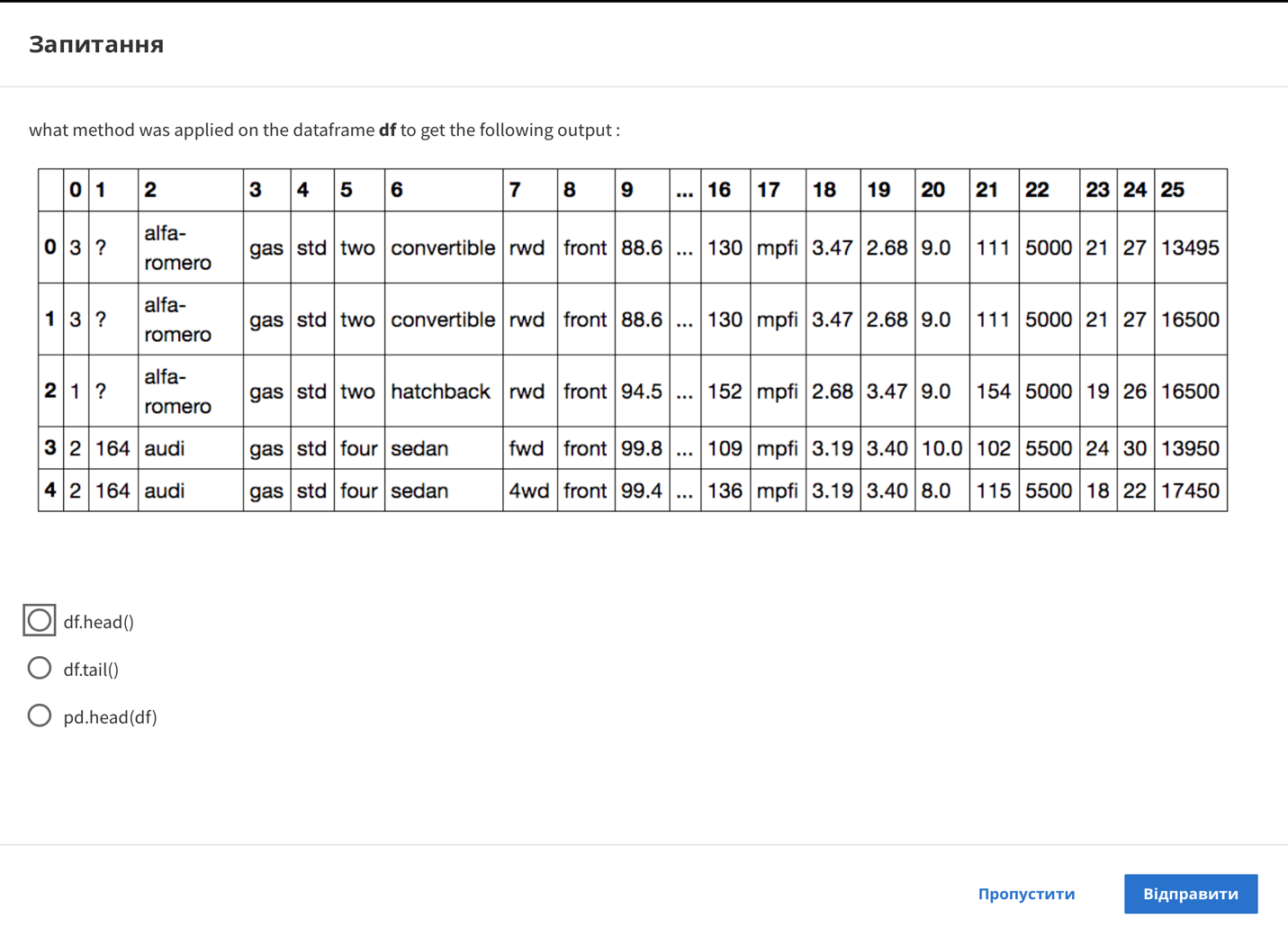
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-

85.data"

df = pd.read\_csv (url, header = None)

IBM Developer

SKILLS NETWORK



Question

what method was applied on the dataframe df to get the following output

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 0 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 | 111 | 5000 | 21 | 27 | 13495 |
| 1 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 | 111 | 5000 | 21 | 27 | 16500 |
| 2 | 1 | ? | alfa- romero | gas | std | two | hatchback | rwd | front | 94.5 | ... | 152 | mpfi | 2.68 | 3.47 | 9.0 | 154 | 5000 | 19 | 26 | 16500 |
| 3 | 2 | 164 | audi | gas | std | four | sedan | fwd | front | 99.8 | ... | 109 | mpfi | 3.19 | 3.40 | 10.0 | 102 | 5500 | 24 | 30 | 13950 |
| 4 | 2 | 164 | audi | gas | std | four | sedan | 4wd | front | 99.4 | ... | 136 | mpfi | 3.19 | 3.40 | 8.0 | 115 | 5500 | 18 | 22 | 17450 |

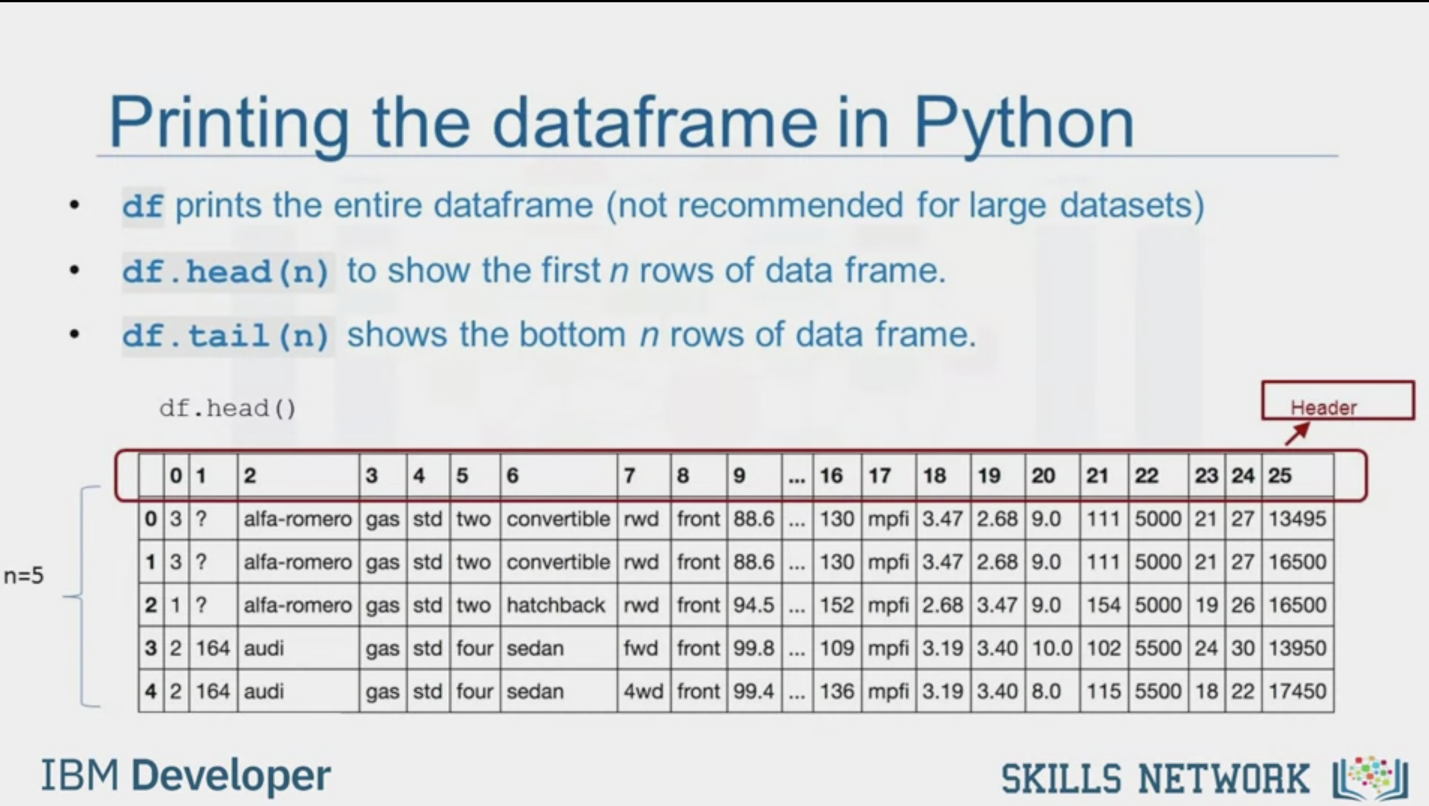
df.head()

df.tail()

pd.head(df)

Skip

Send



Printing the dataframe in Python

df prints the entire dataframe (not recommended for large datasets)

df. head (n) to show the first n rows of data frame.

df. tail (n) shows the bottom n rows of data frame.

df. head ()

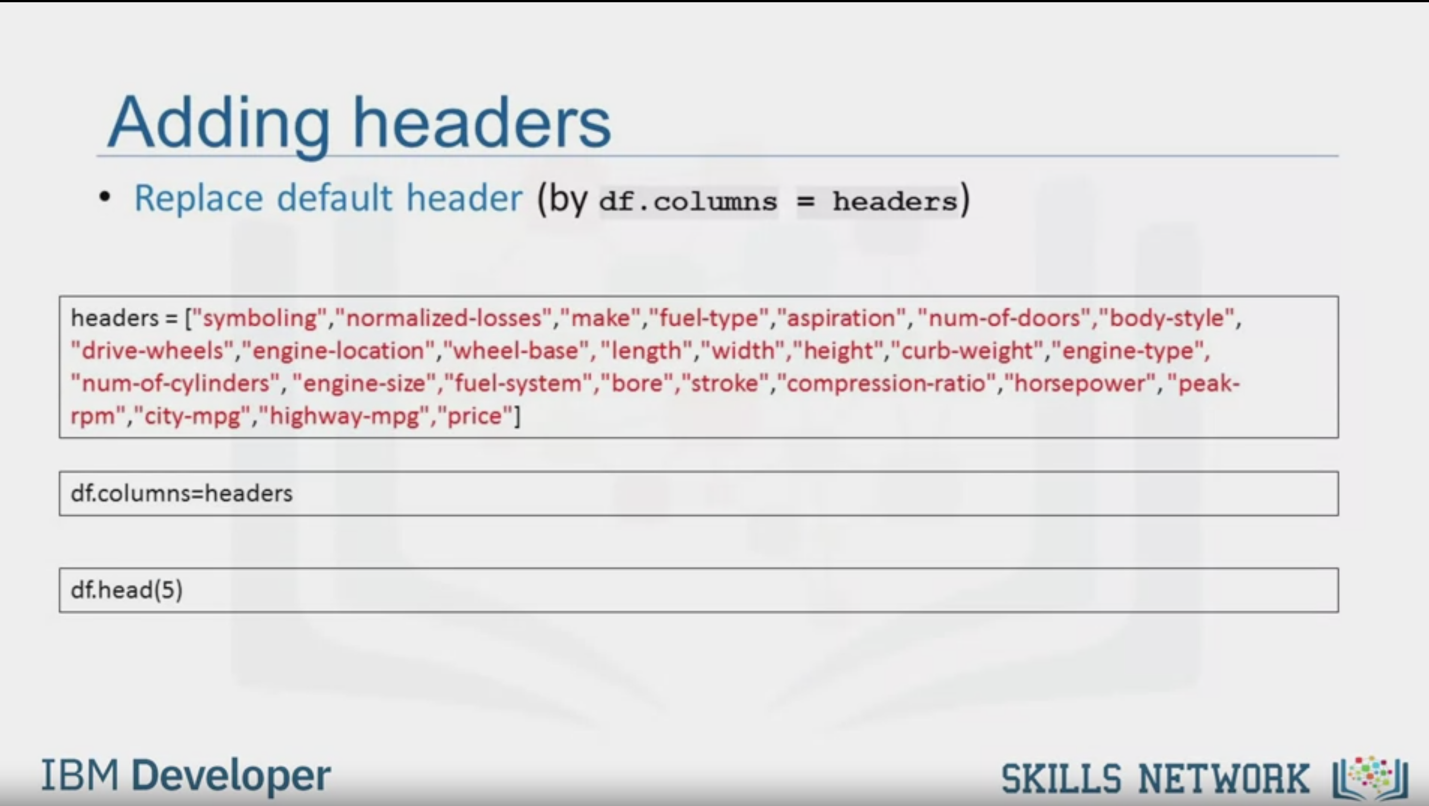
Header

n=5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 0 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 | 111 | 5000 | 21 | 27 | 13495 |
| 1 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 | 111 | 5000 | 21 | 27 | 16500 |
| 2 | 1 | ? | alfa- romero | gas | std | two | hatchback | rwd | front | 94.5 | ... | 152 | mpfi | 2.68 | 3.47 | 9.0 | 154 | 5000 | 19 | 26 | 16500 |
| 3 | 2 | 164 | audi | gas | std | four | sedan | fwd | front | 99.8 | ... | 109 | mpfi | 3.19 | 3.40 | 10.0 | 102 | 5500 | 24 | 30 | 13950 |
| 4 | 2 | 164 | audi | gas | std | four | sedan | 4wd | front | 99.4 | ... | 136 | mpfi | 3.19 | 3.40 | 8.0 | 115 | 5500 | 18 | 22 | 17450 |

IBM Developer

SKILLS NETWORK



Adding headers

• Replace default header (by df.columns = headers)

headers = ["symboling", "normalized-losses", "make", "fuel-type", "aspiration", "num-of-doors", "body-style", "drive-wheels", "engine-location", "wheel-base", "length", "width”, “height", "curb-weight"," engine-type", "num-of-cylinders", "engine-size", "fuel-system", "bore", "stroke", "compression-ratio”, “horsepower", "peak-

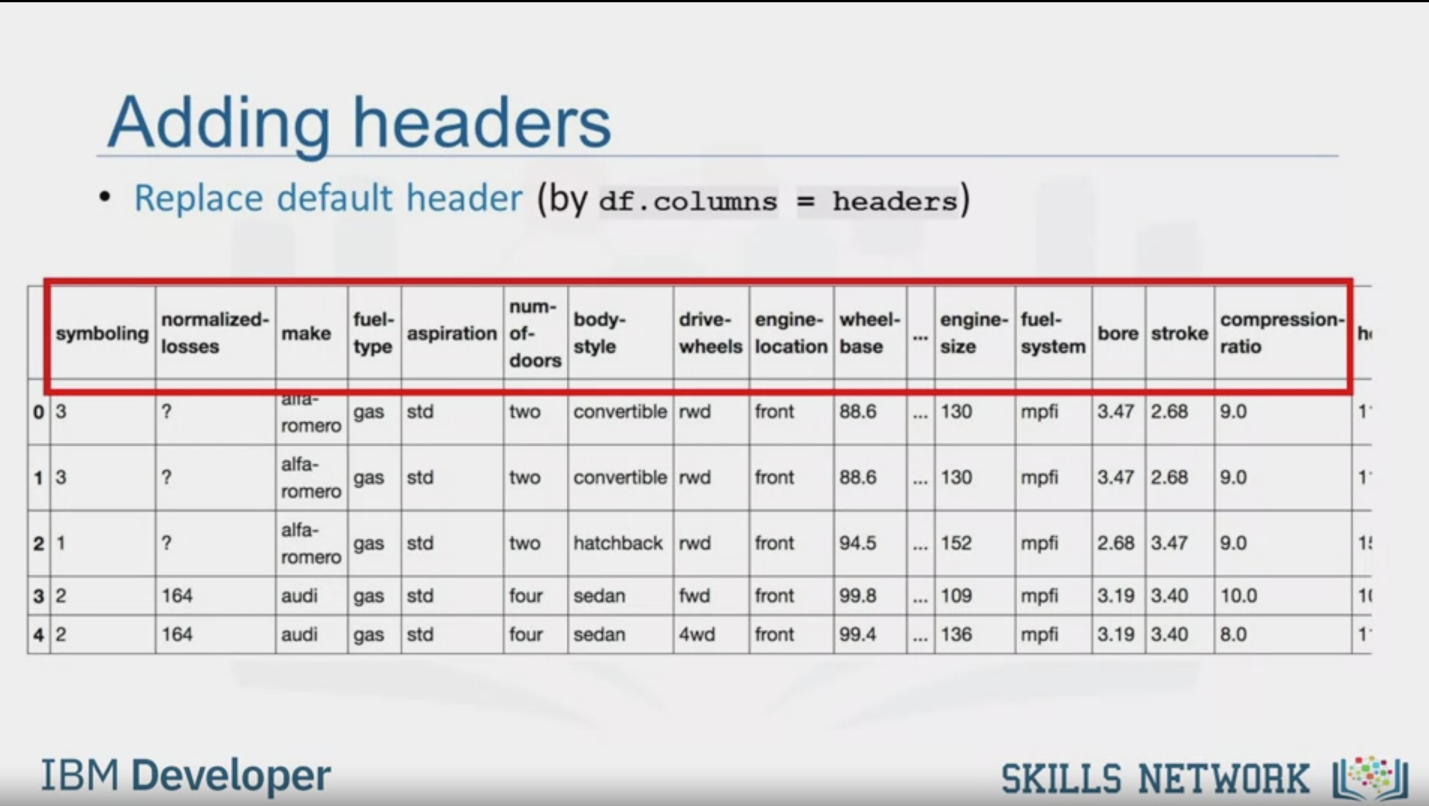
rpm", "city-mpg", "highway-mpg", "price"]

df.columns=headers

df.head(5)

IBM Developer

SKILLS NETWORK



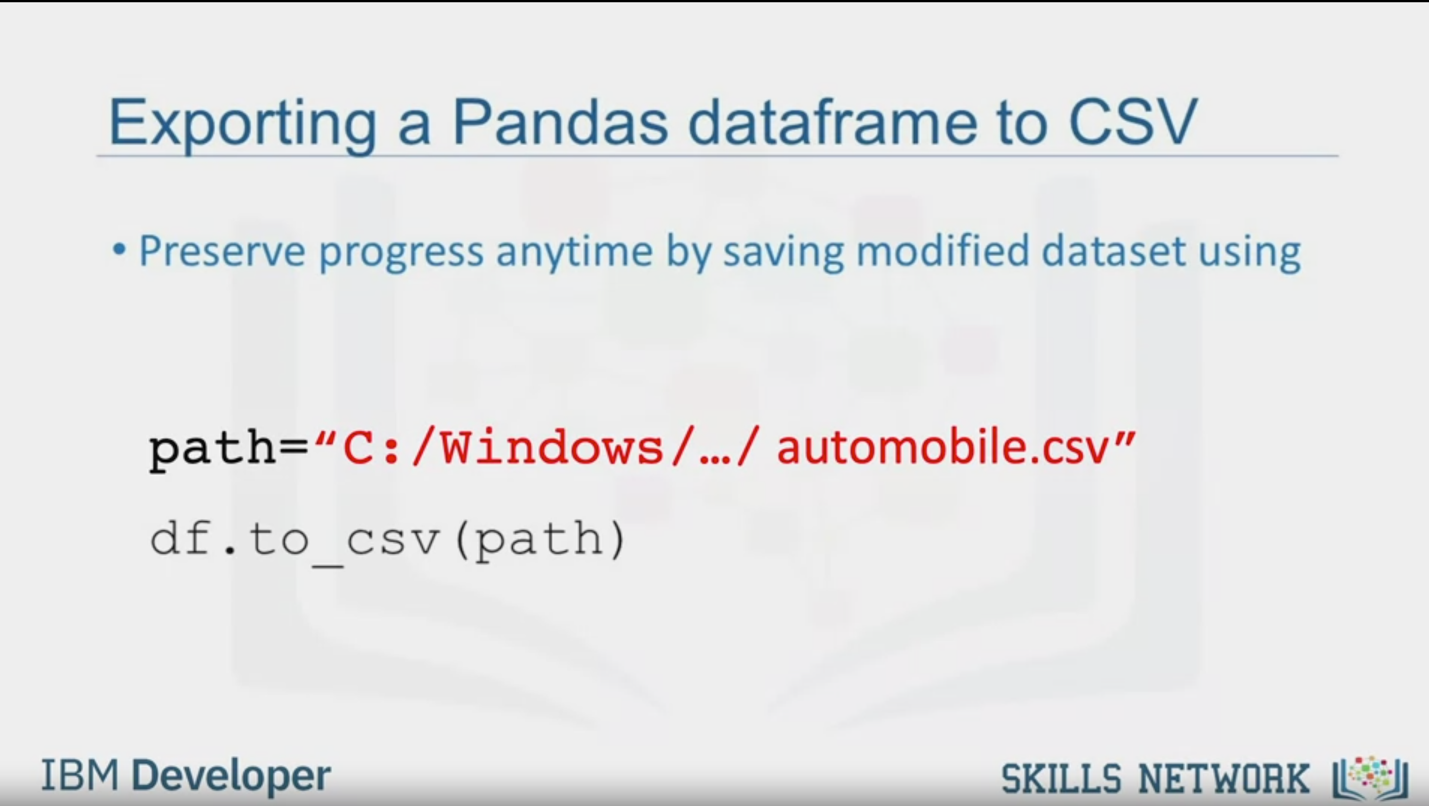
Adding headers

• Replace default header (by df.columns = headers)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | symboling | normalized-losses | make | fuel-type | aspiration | num-of-doors | body-  style | drive-wheels | engine-location | wheel-base | … | engine-size | fuel-system | bore | stroke | compression-ratio |
| 0 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 |
| 1 | 3 | ? | alfa- romero | gas | std | two | convertible | rwd | front | 88.6 | ... | 130 | mpfi | 3.47 | 2.68 | 9.0 |
| 2 | 1 | ? | alfa- romero | gas | std | two | hatchback | rwd | front | 94.5 | ... | 152 | mpfi | 2.68 | 3.47 | 9.0 |
| 3 | 2 | 164 | audi | gas | std | four | sedan | fwd | front | 99.8 | ... | 109 | mpfi | 3.19 | 3.40 | 10.0 |
| 4 | 2 | 164 | audi | gas | std | four | sedan | 4wd | front | 99.4 | ... | 136 | mpfi | 3.19 | 3.40 | 8.0 |

IBM Developer

SKILLS NETWORK



Exporting a Pandas dataframe to CSV

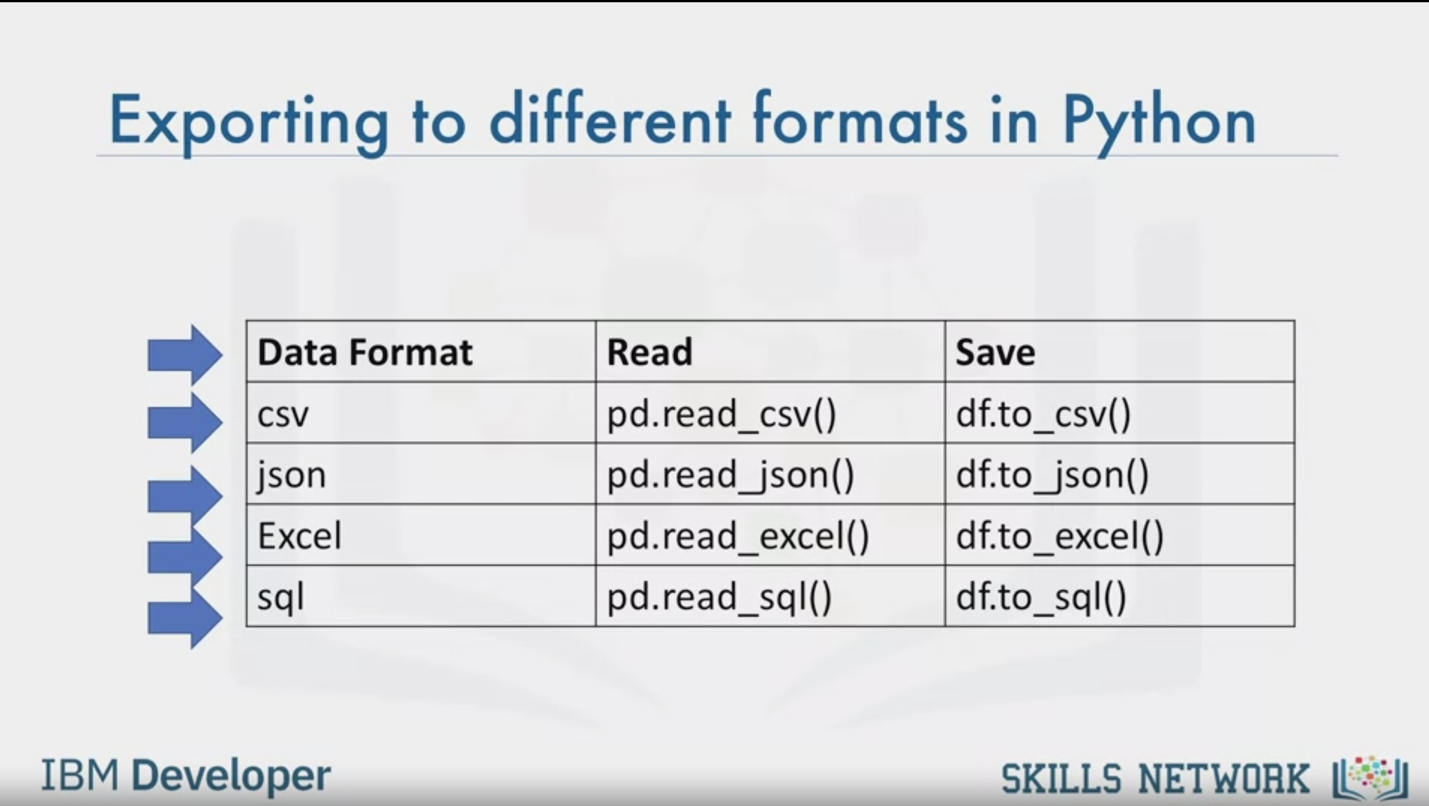
• Preserve progress anytime by saving modified dataset using

path="C: /Windows/.../automobile.csv"

df.to\_cv (path)

IBM Developer

SKILLS NETWORK



Exporting to different formats in Python

|  |  |  |
| --- | --- | --- |
| Data Format | Read | Save |
| csv | pd.read\_csv() | df.to\_csv() |
| json | pd.read json() | df.to\_json() |
| Excel | pd.read\_excel() | df.to\_excel() |
| sql | pd.read\_sql() | df.to\_sql() |

IBM Developer

SKILLS NETWORK



SKILLS NETWORK

IBM Developer

In this video, we'll look at how to read

any data using python's pandas package.

Once we have our data in Python,

then we can perform all the subsequent data analysis procedures we need.

Data acquisition is a process of loading

and reading data into notebook from various sources.

To read any data using Python's pandas package,

there are two important factors to consider,

format and file path.

Format is the way data is encoded.

We can usually tell different encoding schemes

by looking at the ending of the file name.

Some common encodings are: CSV,

JSON, XLSX, HDF and so forth.

The path tells us where the data is stored.

Usually, it is stored either on the computer

we are using or online on the internet.

In our case, we found a dataset of used cars

which was obtained from the web address shown on the slide.

When Jerry entered the web address in his web browser,

he saw something like this.

Each row is one datapoint.

A large number of properties are associated with each datapoint.

Because the properties are separated from each other by commas,

we can guess the data format is CSV,

which stands for comma separated values.

At this point, these are just numbers and don't mean much to humans,

but once we read in this data we can try to make more sense out of it.

In pandas, the read\_CSV method can read in files with

columns separated by commas into a pandas data frame.

Reading data in pandas can be done quickly in three lines.

First, import pandas, then define a variable with

a file path and then use the read\_ CSV method to import the data.

However, read\_CSV assumes the data contains a header.

Our data on used cars has no column headers.

So, we need to specify read\_CSV to

not assign headers by setting header to none.

After reading the dataset,

it is a good idea to look at the data frame to get

a better intuition and to ensure that

everything occurred the way you expected.

Since printing the entire dataset may take up

too much time and resources to save time,

we can just use dataframe.head to show the first n rows of the data frame.

Similarly, dataframe.tail shows the bottom end rows of data frame.

Here, we printed out the first five rows of data.

It seems that the dataset was read successfully.

We can see that pandas automatically set the column header as a list of

integers because we set header equals none when we read the data.

It is difficult to work with the data frame

without having meaningful column names.

However, we can assign column names in pandas.

In our present case,

it turned out that we have the column names in a separate file online.

We first put the column names in a list called headers,

then we set df.columns equals

headers to replace the default integer headers by the list.

If we use the head method

introduced in the last slide to check the dataset,

we see the correct headers inserted at the top of each column.

At some point in time,

after you've done operations on your dataframe you may

want to export your pandas dataframe to a new CSV file.

You can do this using the method to\_CSV.

To do this, specify the file path

which includes the file name that you want to write to.

For example, if you would like to save dataframe

df as automobile.CSV to your own computer,

you can use the syntax df.to\_CSV.

For this course, we will only read and save CSV files.

However, pandas also supports importing and

exporting of most data file types with different dataset formats.

The code syntax for reading and saving other data formats

is very similar to read or save CSV file.

Each column shows a different method to read and

save files into a different format.