



Week 1 - Importing Datasets

1. The Problem

Why Data Analysis?

- Data is everywhere.
- Data analysis/data science helps us answer questions from data.
- Data analysis plays an important role in:
 - Discovering useful information
 - Answering questions
 - Predicting future or the unknown

Tom wants to sell his car



How much money should he sell his car for?

The price he sets should not be too high, but not too low either.

So let's begin with our scenario. Let's say we have a friend named Tom. And Tom wants to sell his car. But the problem is he doesn't know how much he should sell his car for. Tom wants to sell his car for as much as he can. But he also wants to set the price reasonably, so someone would want to purchase it. So the price he sets should represent the value of the car.

How can we help Tom determine the best price for his car? Let's think like data scientists and clearly define some of his problems. For example, is there data on the prices of other cars and their characteristics? What features of cars affect their prices? Color? Brand? Does horsepower also effect the selling price, or perhaps something else? As a data analyst or data scientist, these are some of the questions we can start thinking about. To answer these questions, we're going to need some data. In the next videos, we'll be going into how to understand the data, how to import it into Python, and how to begin looking into some basic insights from the data.

2. Understanding the Data

Dataset – Used Automobiles (CSV)

```
3,7,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,13495
3,7,alfa-romero,gas,std,two,convertible,rwd,front,88.60,168.80,64.10,48.80,2548,dohc,four,130,mpfi,3.47,2.68,9.00,111,5000,21,27,16500
1,7,alfa-romero,gas,std,two,hatchback,rwd,front,94.50,171.20,65.50,52.40,2823,dohc,six,152,mpfi,2.46,3.47,9.80,154,5000,19,26,16500
2,164,audi,gas,std,four,sedan,fwd,front,99.40,176.60,66.20,54.30,2337,ohc,four,109,mpfi,3.19,3.40,10.00,102,5500,24,30,13950
2,164,audi,gas,std,four,sedan,fwd,front,99.40,176.60,66.40,54.30,2824,ohc,five,136,mpfi,3.19,3.40,8.00,115,5500,18,22,17450
2,7,audi,gas,std,two,sedan,fwd,front,99.80,177.20,66.30,53.10,2507,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,15250
1,158,audi,gas,std,four,sedan,fwd,front,105.80,192.70,71.40,55.70,2954,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,17710
1,7,audi,gas,std,four,wagon,fwd,front,105.80,192.70,71.40,55.70,2954,ohc,five,136,mpfi,3.19,3.40,8.50,110,5500,19,25,18920
1,158,audi,gas,turbo,four,sedan,fwd,front,105.80,192.70,71.40,55.90,3086,ohc,five,131,mpfi,3.13,3.40,8.30,140,5500,17,20,23875
0,7,audi,gas,turbo,two,hatchback,fwd,front,99.50,178.20,67.90,52.00,3053,ohc,five,131,mpfi,3.13,3.40,7.00,160,5500,16,22,7
2,192,bmw,gas,std,two,sedan,fwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16430
0,192,bmw,gas,std,four,sedan,fwd,front,101.20,176.80,64.80,54.30,2395,ohc,four,108,mpfi,3.50,2.80,8.80,101,5800,23,29,16925
0,189,bmw,gas,std,two,sedan,rwd,front,101.20,176.80,64.80,54.30,2710,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,20970
0,188,bmw,gas,std,four,sedan,rwd,front,101.20,176.80,64.80,54.30,2765,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,21,28,21105
1,7,bmw,gas,std,four,sedan,rwd,front,103.50,189.00,66.90,55.70,3055,ohc,six,164,mpfi,3.31,3.19,9.00,121,4250,20,25,24565
0,7,bmw,gas,std,four,sedan,rwd,front,103.50,189.00,66.90,55.70,3230,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,16,22,30760
0,7,bmw,gas,std,two,sedan,rwd,front,103.50,193.80,67.90,53.70,3380,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,16,22,41315
0,7,bmw,gas,std,four,sedan,rwd,front,110.00,197.00,70.90,56.30,3505,ohc,six,209,mpfi,3.62,3.39,8.00,182,5400,15,20,36880
2,121,chevrolet,gas,std,two,hatchback,fwd,front,88.40,141.10,60.30,53.20,1488,1,three,61,2bbl,2.51,3.03,9.50,48,5100,47,53,5151
```

<https://archive.ics.uci.edu/ml/machine-learning-databases/autos/>

we'll be talking about data analysis and the scenario in which we'll be playing the data analyst or data scientist. But before we begin, talking about the problem, used car prices, we should first understand the importance of data analysis.

- As you know, data is collected everywhere around us. Whether it's collected manually by scientists or collected digitally, every time you click on a website, or your mobile device.
- But data does not mean information. Data analysis and, in essence, data science, helps us unlock the information and insights from raw data to answer our questions.
- So data analysis plays an important role by helping us to discover useful information from the data, answer questions, and even predict the future or the unknown.

Estimate used car prices

How can we help Tom determine the best price for his car?

- Is there data on the prices of other cars and their characteristics?
- What features of cars affect their prices?
 - Color? Brand? Horsepower? Something else?
- Asking the right questions in terms of data

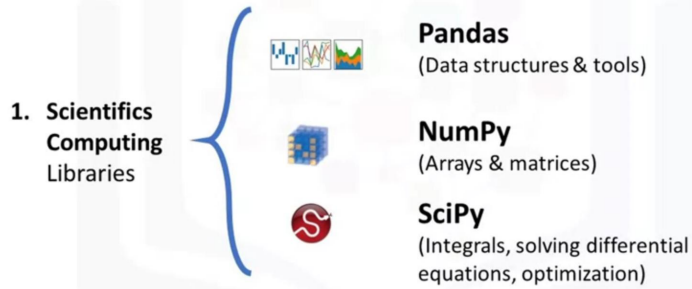
Each of the attributes in the dataset

No.	Attribute name	attribute range	No.	Attribute name	attribute range
1	symboling	-3, -2, -1, 0, 1, 2, 3.	14	curb-weight	continuous from 1488 to 4066.
2	normalized-losses	continuous from 65 to 256.	15	engine-type	dohc, dohc, l, ohc, ohcf, ohcv, rotor.
3	make	audi, bmw, etc.	16	num-of-cylinders	eight, five, four, six, three, twelve, two.
4	fuel-type	diesel, gas.	17	engine-size	continuous from 61 to 326.
5	aspiration	std, turbo.	18	fuel-system	1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
6	num-of-doors	four, two.	19	bore	continuous from 2.54 to 3.94.
7	body-style	hardtop, wagon, etc.	20	stroke	continuous from 2.07 to 4.17.
8	drive-wheels	4wd, fwd, rwd.	21	compression-ratio	continuous from 7 to 23.
9	engine-location	front, rear.	22	horsepower	continuous from 48 to 288.
10	wheel-base	continuous from 86.6 to 120.9.	23	peak-rpm	continuous from 4150 to 6600.
11	length	continuous from 141.1 to 208.1.	24	city-mpg	continuous from 13 to 49.
12	width	continuous from 60.3 to 72.3.	25	highway-mpg	continuous from 16 to 54.
13	height	continuous from 47.8 to 59.8.	26	price	continuous from 5118 to 45400.

Attributes description: <https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.names>

3. Python Packages for Data Science

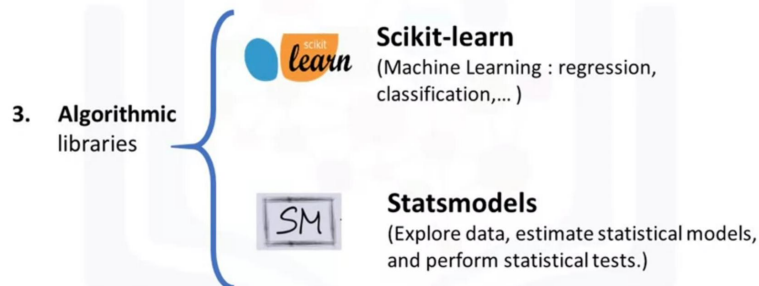
Scientifics Computing Libraries in Python



Visualization Libraries in Python



Algorithmic Libraries in Python



In order to do data analysis in Python, we should first tell you a little bit about the main packages relevant to analysis in Python.

- A Python library is a collection of functions and methods that allow you to perform lots of actions without writing any code.
- The libraries usually contain built in modules providing different functionalities which you can use directly. And there are extensive libraries offering a broad range of facilities.

We have divided the Python data analysis libraries into three groups.

- The first group is called **scientific computing libraries**.
 - **Pandas** offers data structure and tools for effective data manipulation and analysis. It provides facts, access to structured data. The primary instrument of Pandas is the two dimensional table consisting of column and row labels, which are called a dataframe. It is designed to provide easy indexing functionality.
 - The **NumPy** library uses arrays for its inputs and outputs. It can be extended to objects for matrices and with minor coding changes, developers can perform fast array processing.
 - **SciPy** includes functions for some advanced math problems as listed on this slide, as well as data visualization.
- Using **data visualization methods** is the best way to communicate with others, showing them meaningful results of analysis. These libraries enable you to create graphs, charts and maps.
 - The **Matplotlib** package is the most well known library for data visualization. It is great for making graphs and plots. The graphs are also highly customizable.
 - Another high level visualization library is **Seaborn**. It is based on Matplotlib. It's very easy to generate various plots such as heat maps, time series and violin plots.
- With machine learning algorithms, we're able to develop a model using our data set and obtain predictions. The **algorithmic libraries** tackle the machine learning tasks from basic to complex. Here we introduce two packages:
 - The **Scikit-learn** library contains tools statistical modeling, including regression, classification, clustering, and so on. This library is built on NumPy, SciPy and Matplotlib.
 - **Statsmodels** is also a Python module that allows users to explore data, estimate statistical models and perform statistical tests.

4. Importing and Exporting Data in Python

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>

Importing a CSV into Python

```
import pandas as pd

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

df = pd.read_csv(url)
```

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)
```

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.

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()

	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

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)
```

Exporting a Pandas dataframe to CSV

- Preserve progress anytime by saving modified dataset using

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

df.to_csv(path)
```

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()

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. 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.

总结:

5. Getting Started Analyzing Data in Python

Basic insights from the data

- Understand your data before you begin any analysis
- Should check:
 - Data Types
 - Data Distribution
- Locate potential issues with the data

In this video, we introduce some simple Pandas methods that all data scientists and analysts should know when using Python, Pandas and data. At this point, we assume that the data has been loaded. It's time for us to explore the dataset. Pandas has several built-in methods that can be used to understand the datatype or features or to look at the distribution of data within the dataset. Using these methods, gives an overview of the dataset and also point out potential issues such as the wrong data type of features which may need to be resolved later on.

Data has a variety of types. The main types stored in Pandas' objects are object, float, Int, and datetime. The data type names are somewhat different from those in native Python. This table shows the differences and similarities between them. Some are very similar such as the numeric data types, int and float. The object pandas type function's similar to string in Python, save for the change in name. While the datetime Pandas type, is a very useful type for handling time series data.

There are two reasons to check data types in a dataset.

- Pandas automatically assigns types based on the encoding it detects from the original data table. For a number of reasons, this assignment may be incorrect. For example, it should be awkward if the car price column which we should expect to contain continuous numeric numbers, is assigned the data type of object. It would be more natural for it to have the float type. Jerry may need to manually change the data type to float.
- The second reason, is that allows an experienced data scientists to see which Python functions can be applied to a specific column. For example, some math functions can only be applied to numerical data. If these functions are applied to non-numerical data an error may result.

Basic Insights of Dataset - Data Types

Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the <code>datetime</code> module in Python's standard library)	time data.

Why check data types?

- potential info and type mismatch
- compatibility with python methods

Basic Insights of Dataset - Data Types

- In pandas, we use `dataframe.dtypes` to check data types

df.dtypes

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

dtype: object

dataframe.describe()

- Returns a statistical summary

df.describe()

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

dataframe.describe(include="all")

- Provides full summary statistics
- df.describe(include="all")

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	engine-size	fuel-system	bore	stroke
count	205.000000	205	205	205	205	205	205	205	205	205.000000	205.000000	205	205	205
unique	NaN	52	22	2	2	3	5	3	2	NaN	NaN	8	39	37
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	NaN	mpfi	3.62	3.40
freq	NaN	41	32	185	168	114	96	120	202	NaN	NaN	94	23	20
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.756585	126.907317	NaN	NaN	NaN
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	41.642693	NaN	NaN	NaN
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	61.000000	NaN	NaN	NaN
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	97.000000	NaN	NaN	NaN
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	120.000000	NaN	NaN	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	141.000000	NaN	NaN	NaN
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	326.000000	NaN	NaN	NaN

Basic Insights of Dataset - Info

dataframe.info() provides a concise summary of your DataFrame.

df.info()

Row Number

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

When the dtype method is applied to the data set, the data type of each column is returned in a series. A good data scientists intuition tells us that most of the data types make sense. The make of cars for example are names. So, this information should be of type object. The last one on the list could be an issue. As bore is a dimension of an engine, we should expect a numerical data type to be used. Instead, the object type is used. In later sections, Jerry will have to correct these type mismatches.

Now, we would like to check the statistical summary of each column to learn about the distribution of data in each column. **The statistical metrics can tell the data scientist if there are mathematical issues that may exist such as extreme outliers and large deviations. The data scientists may have to address these issues later.** To get the quick statistics, we use **the describe method**. It returns the number of terms in the column as count, average column value as mean, column standard deviation as std, the maximum minimum values, as well as the boundary of each of the quartiles.

By default, the dataframe.describe functions skips rows and columns that do not contain numbers. It is possible to make the describe method worked for object type columns as well. To enable a summary of all the columns, we could add an argument. Include equals all inside the describe function bracket. Now, the outcome shows the summary of all the 26 columns, including object typed attributes. We see that **for the object type columns, a different set of statistics is evaluated, like unique, top, and frequency.**

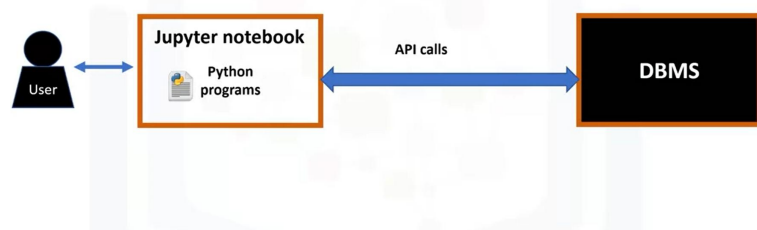
- **unique** is the number of distinct objects in the column.
- **top** is most frequently occurring object,
- **freq** is the number of times the top object appears in the column.

Some values in the table are shown here as **NaN** which stands for not a number. This is because that particular statistical metric cannot be calculated for that specific column data type.

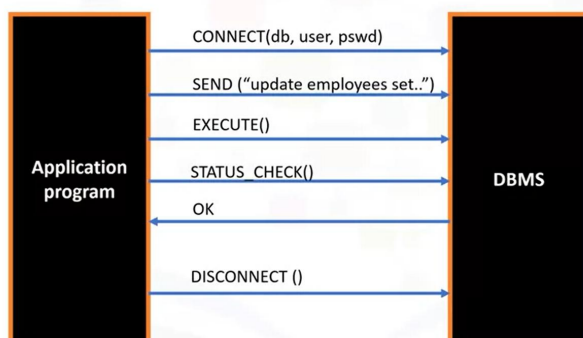
Another method you can use to check your dataset, is **the dataframe.info function**. This function shows the top 30 rows and bottom 30 rows of the data frame.

6. Accessing Databases with Python

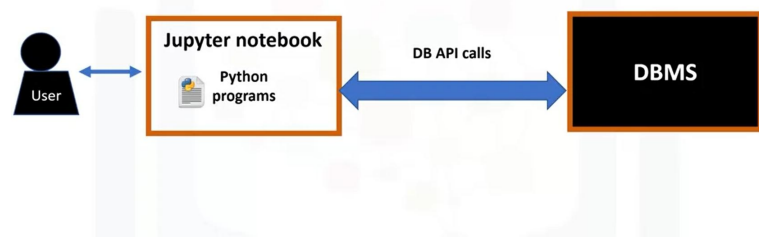
Accessing databases using Python



What is a SQL API?



What is a DB-API?



in this video you will learn how to access databases using Python. Databases are powerful tools for data scientists. After completing this module, you'll be able to explain the basic concepts related to using Python to connect to databases.

This is how a typical user accesses databases using Python code written on a Jupyter notebook, a web based editor. There is a mechanism by which the Python program communicates with the DBMS. The Python code connects to the database using API calls. We will explain the basics of **SQL APIs** and **Python DB APIs**. An **application programming interface** is a set of functions that you can call to get access to some type of service.

- **SQL API** consists of library function calls as an application programming interface, API, for the DBMS. To pass SQL statements to the DBMS, an application program calls functions in the API, and it calls other functions to retrieve query results and status information from the DBMS. The basic operation of a typical SQL API is illustrated in the figure. The application program begins its database access with one or more API calls that connect the program to the DBMS. To send the SQL statement to the DBMS, the program builds the statement as a text string in a buffer and then makes an API call to pass the buffer contents to the DBMS. The application program makes API calls to check the status of its DBMS request and to handle errors. The application program ends its database access with an API call that disconnects it from the database.
- **DB-API** is Python's standard API for accessing relational databases. It is a standard that allows you to write a single program that works with multiple kinds of relational databases instead of writing a separate program for each one. So, if you learn the DB-API functions, then you can apply that knowledge to use any database with Python.

总结:

Concepts of the Python DB API

Connection Objects

- Database connections
- Manage transactions

Cursor Objects

- Database Queries

What are Connection methods?

- `cursor()`
- `commit()`
- `rollback()`
- `close()`

What are cursor methods?

- `.callproc()`
- `.execute()`
- `.executemany()`
- `.fetchone()`
- `.fetchmany()`
- `.fetchall()`
- `.nextset()`
- `.arraysize()`
- `.close()`

Writing code using DB-API

```
from dbmodule import connect

#Create connection object
connection = connect('databasename','username','pswd')

#Create a cursor object
cursor = connection.cursor()

#Run queries
cursor.execute('select * from mytable')
results = cursor.fetchall()

#Free resources
Cursor.close()
connection.close()
```

The two main concepts in the Python DB-API are **connection objects** and **query objects**.

- You use [connection objects](#) to connect to a database and manage your transactions.
- [Cursor objects](#) are used to run queries. You open a cursor object and then run queries. The cursor works similar to a cursor in a text processing system where you scroll down in your result set and get your data into the application. Cursors are used to scan through the results of a database.

Here are the methods used with [connection objects](#).

- The `cursor()` method returns a new cursor object using the connection.
- The `commit()` method is used to commit any pending transaction to the database.
- The `rollback()` method causes the database to roll back to the start of any pending transaction.
- The `close()` method is used to close a database connection. Let's walk through a Python application that uses the DB-API to query a database.

First, you import your database module by using the connect API from that module. To open a connection to the database, you use the connection function and pass in the parameters that is, the database name, username, and password. The connect function returns connection object.

After this, you create a cursor object on the connection object. The cursor is used to run queries and fetch results. After running the queries using the cursor, we also use the cursor to fetch the results of the query.

Finally, when the system is done running the queries, it frees all resources by closing the connection. Remember that it is always important to close connections to avoid unused connections taking up resources. Thanks for watching this video.

Lesson Summary

In this lesson, you have learned how to:

- **Define the Business Problem:** Look at the data and make some high-level decision on what kind of analysis should be done
- **Import and Export Data in Python:** How to import data from multiple data sources using the Pandas library and how to export files into different formats.
- **Analyze Data in Python:** How to do some introductory analysis in Python using functions like `dataframe.head()` to view the first few lines of the dataset, `dataframe.info()` to view the column names and data types.

总结: