Importing text file in numpy

In [ ]:
```
import numpy as np

# Load a text file into a NumPy array
data_array = np.loadtxt('data.txt')

print(data_array)
```

Importing text file in pandas

In [ ]:
```
import pandas as pd

# Load a text file into a pandas DataFrame
data_df = pd.read_csv('data.txt', delimiter='\t')

print(data_df)
```

Importing SAS files

SAS files typically refer to datasets that are created and used within the SAS software environment. SAS datasets are stored in a proprietary binary format that is optimized for performance and data integrity. These files have the extension ".sas7bdat" and contain structured data organized into rows and columns.

In [ ]:
```
from sas7bdat import SAS7BDAT

# Read a SAS dataset
with SAS7BDAT('data.sas7bdat') as file:
    data_df = file.to_data_frame()

# Display the DataFrame
print(data_df)
```

Importing HDF5 files

HDF5 (Hierarchical Data Format 5) is a versatile file format designed to store and manage large and complex datasets. It is commonly used in scientific computing, data analysis, and other fields where handling substantial amounts of structured data is required. HDF5 files can store various types of data, including numerical data, text, images, and more

In [ ]:
```
import h5py

# Open an HDF5 file in read mode
with h5py.File('data.h5', 'r') as file:
    # Access datasets within the file
    dataset = file['my_dataset']
    data_array = dataset[:]

# Display the data array
print(data_array)
```

Importing MATLAB files

MATLAB files typically refer to files associated with MATLAB, which is a high-level programming and numerical computing environment widely used for various tasks, including data analysis, scientific research, engineering simulations, and more.

In [ ]:
```
from scipy.io import loadmat

# Load a MATLAB file
data_dict = loadmat('data.mat')

# Access variables within the dictionary
variable = data_dict['my_variable']

# Display the variable
print(variable)
```

SQL queries in python

In [ ]:

```
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('my_database.db')

# Create a cursor
cursor = conn.cursor()

# Execute an SQL query
cursor.execute("SELECT * FROM my_table WHERE column_name = ?", ('value',))

# Fetch the results
results = cursor.fetchall()

# Display the results
for row in results:
    print(row)

# Close the cursor and connection
cursor.close()
conn.close()
```

Importing the sqlite3 Module:

1. import sqlite3: This line imports the sqlite3 module, which provides the tools necessary for working with SQLite databases in Python. Connecting to the Database:
2. conn = sqlite3.connect('my_database.db'): This line establishes a connection to an SQLite database file named my_database.db. If the file doesn't exist, SQLite will create it. Creating a Cursor:
3. cursor = conn.cursor(): A cursor is a control structure that allows you to execute SQL queries and retrieve results from the database. It acts as a handle to interact with the database. Executing an SQL Query:
4. cursor.execute("SELECT * FROM my_table WHERE column_name = ?", ('value',)): This line executes an SQL query using the cursor. The query selects all rows from a table named my_table where the value in the column named column_name matches the provided value 'value'. The ? is a placeholder for a parameter value, which is provided as a tuple in the second argument of the execute() function.
5. Fetching Results: results = cursor.fetchall(): This line retrieves the results of the executed query and stores them in the results variable. The results are returned as a list of tuples, where each tuple represents a row from the database.
6. Displaying Results: for row in results: print(row): This loop iterates through the results list and prints each row (tuple) to the console. Each row corresponds to a retrieved record from the database.
7. Closing Cursor and Connection:cursor.close(): This line closes the cursor, releasing the resources associated with it. conn.close(): This line closes the database connection, freeing up resources and ensuring proper termination of the database connection.

Importing files from the web

In [ ]:

```python
# Assign url of file: url
url = 'https://s3.amazonaws.com/assets.datacamp.com/production/course_1606/

# Read file into a DataFrame: df
df = pd.read_csv(url, sep=';')
```

In [7]:

```python
from urllib.request import urlopen, Request

# Specify the url
url = "https://en.wikipedia.org/wiki/Pakistan"

# This packages the request: request
request = Request(url)

# Sends the request and catches the response: response
response = urlopen(request)


# Extract the response: html
html= response.read()
response.close()
# Print the html
print(html)
```

```
b'<!DOCTYPE html>\n<html class="client-nojs vector-feature-language-in-
header-enabled vector-feature-language-in-main-page-header-disabled vec
tor-feature-sticky-header-disabled vector-feature-page-tools-pinned-dis
abled vector-feature-toc-pinned-enabled vector-feature-main-menu-pinned
-disabled vector-feature-limited-width-clientpref-1 vector-feature-limi
ted-width-content-enabled vector-feature-zebra-design-disabled" lang="e
n" dir="ltr">\n<head>\n<meta charset="UTF-8">\n<title>Pakistan - Wikipe
dia</title>\n<script>(function(){var className="client-js vector-featur
e-language-in-header-enabled vector-feature-language-in-main-page-heade
r-disabled vector-feature-sticky-header-disabled vector-feature-page-to
ols-pinned-disabled vector-feature-toc-pinned-enabled vector-feature-ma
in-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vecto
r-feature-limited-width-content-enabled vector-feature-zebra-design-dis
abled";var cookie=document.cookie.match(/(?:^|; )enwikimwclientpreferen
ces=([^;]+)/);if(cookie){cookie[1].split(\'%2C\').forEach(function(pre
f){className=className.replace(new RegExp(\'(^| )\'+pref.replace(/-clie
ntpref-\\w+$|[^\\w-]+/g,\'\')+\'-clientpref-\\\\w+( |$)\'),\'$1\'+pref+
\'$2\');});}document.documentElement.className=className;}());RLCONF=
{"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTran
```

HTTP requests

In [4]:

```python
# Import package
import requests

# Specify the url: url
url = 'https://en.wikipedia.org/wiki/Saudi_Arabia'

# Packages the request, send the request and catch the response: r
r = requests.get(url)

# Extract the response: text
text = r.text

# Print the html
print(text)
```

```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vect
or-feature-language-in-main-page-header-disabled vector-feature-sticky-
header-disabled vector-feature-page-tools-pinned-disabled vector-featur
e-toc-pinned-enabled vector-feature-main-menu-pinned-disabled vector-fe
ature-limited-width-clientpref-1 vector-feature-limited-width-content-e
nabled vector-feature-zebra-design-disabled" lang="en" dir="ltr">
<head>
<meta charset="UTF-8">
<title>Saudi Arabia - Wikipedia</title>
<script>(function(){var className="client-js vector-feature-language-in
-header-enabled vector-feature-language-in-main-page-header-disabled ve
ctor-feature-sticky-header-disabled vector-feature-page-tools-pinned-di
sabled vector-feature-toc-pinned-enabled vector-feature-main-menu-pinne
d-disabled vector-feature-limited-width-clientpref-1 vector-feature-lim
ited-width-content-enabled vector-feature-zebra-design-disabled";var co
okie=document.cookie.match(/(?:^|; )enwikimwclientpreferences=([^;]
+)/);if(cookie){cookie[1].split('%2C').forEach(function(pref){className
=className.replace(new RegExp('(^| )'+pref.replace(/-clientpref-\w+$|[^
```

BeautifulSoup

In [9]:

```python
# Import packages
import requests
from bs4 import BeautifulSoup
# Specify url: url
url = 'https://en.wikipedia.org/wiki/Saudi_Arabia'
# Package the request, send the request and catch the response: r
r = requests.get(url)

# Extracts the response as html: html_doc
html_doc = r.text

# Create a BeautifulSoup object from the HTML: soup
soup = BeautifulSoup(html_doc)

# Prettify the BeautifulSoup object: pretty_soup
pretty_soup = soup.prettify()

# Print the response
print(pretty_soup)
```

```
y","ext.tmh.player.styles":"ready","codex-search-styles":"ready","skin
s.vector.styles":"ready","skins.vector.icons":"ready","jquery.makeColla
psible.styles":"ready","ext.visualEditor.desktopArticleTarget.noscrip
t":"ready","ext.wikimediaBadges":"ready","ext.uls.interlanguage":"read
y","wikibase.client.init":"ready"};RLPAGEMODULES=["ext.cite.ux-enhancem
ents","mediawiki.page.gallery","ext.tmh.player","mediawiki.page.medi
a","ext.scribunto.logs","site","mediawiki.page.ready","jquery.makeColla
psible","mediawiki.toc","skins.vector.js","ext.visualEditor.desktopArti
cleTarget.init","ext.visualEditor.targetLoader","ext.eventLogging","ex
t.wikimediaEvents","ext.navigationTiming","ext.cx.eventlogging.campaign
s",
"ext.quicksurveys.init","ext.centralNotice.geoIP","ext.centralNotice.st
artUp","ext.gadget.ReferenceTooltips","ext.gadget.charinsert","ext.gadg
et.extra-toolbar-buttons","ext.gadget.switcher","ext.centralauth.centra
lautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.ech
o.centralauth","ext.uls.compactlinks","ext.uls.interface","ext.cx.uls.q
uick.actions","wikibase.client.vector-2022","ext.checkUser.clientHint
s","ext.growthExperiments.SuggestedEditSession"];
  </script>
  <script>
```

Loading and exploring a JSON

In [ ]:

```python
import json

# Load JSON: json_data
with open("c:/blogdown/a_movie.json") as json_file:
    json_data = json.load(json_file)

# Print each key-value pair in json_data
for k in json_data.keys():
    print(k + ': ', json_data[k])
```

JSON–from the web to Python

In [ ]:
```python
# Import package
import requests

# Assign URL to variable: url
url = 'http://www.omdbapi.com/?apikey=72bc447a&t=social+network'

# Package the request, send the request and catch the response: r
r = requests.get(url)

# Decode the JSON data into a dictionary: json_data
json_data = r.json()

# Print each key-value pair in json_data
for k in json_data.keys():
    print(k + ': ', json_data[k])
```

Import Tweets

In [ ]:
```python
import tweepy

API_KEY = 'your_api_key'
API_SECRET_KEY = 'your_api_secret_key'
ACCESS_TOKEN = 'your_access_token'
ACCESS_TOKEN_SECRET = 'your_access_token_secret'

auth = tweepy.OAuthHandler(API_KEY, API_SECRET_KEY)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth)
```

In [ ]:
```python
keyword = 'data science'
tweets = api.search(q=keyword, count=10)

for tweet in tweets:
    print(tweet.text)
```