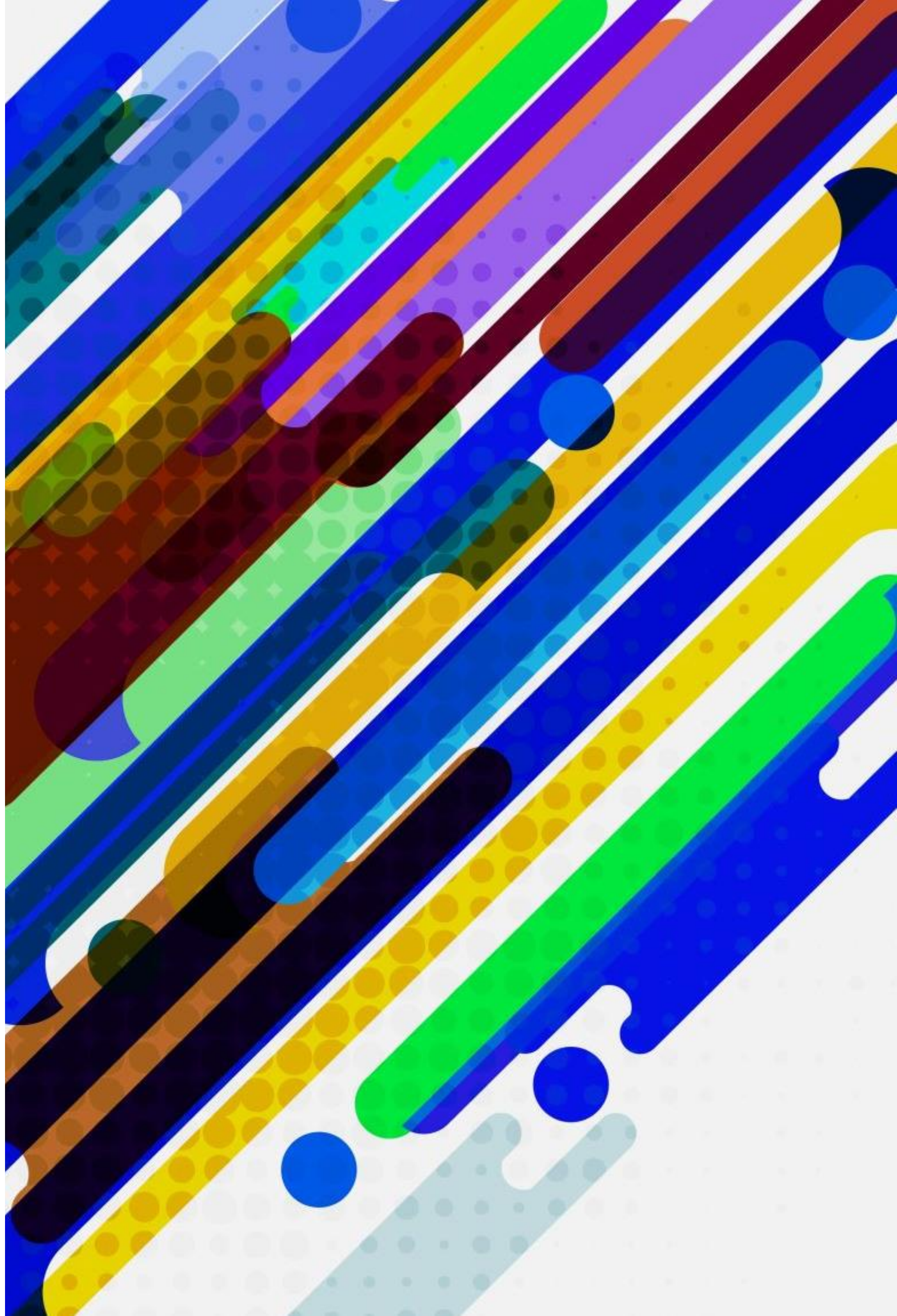
WEEK 10: WEB SCARPING (II) & FILE I/O

Johnny Zhang



VPL Example (Demo):

Example 1:

URL:https://vpl.bibliocommons.com/events/search/index

```
import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from bs4 import BeautifulSoup
from selenium.webdriver.common.by import By

DRIVER_PATH = "/Python/ChromeDriver/chromedriver"
URL = "https://vpl.bibliocommons.com/events/search/index"

browser = webdriver.Chrome(service=Service(DRIVER_PATH))
browser.get(URL)

# Give the browser time to load all content.
time.sleep(3)

content = browser.find_elements(By.CSS_SELECTOR,".event-row-item")
for e in content:
    start = e.get_attribute('innerHTML')
    # Beautiful soup allows us to remove HTML tags from our content if it exists.
    soup = BeautifulSoup(start, features="lxml")
    print(soup.get_text())
    print("***")  # Go to new line.
```

```
Featured

Tuesday, February 1, 2022



  A Musical Celebration for Black History Month


On from Feb 1 - Feb 28


          Tuesday, February 1, 2022

          All Day
```

- First strip out the new line and tab characters.
- Remove all groupings of two or more consecutive spaces.
- Inspect your output after the changes and make custom changes with the *replace()* function.

```
# Remove hidden characters for tabs and new lines.
  rawString = re.sub(r"[\n\t]*", "", rawString)
```

Featured

Tuesday, February 1, 2022

A Musical Celebration for Black History Month

On from Feb 1 - Feb 28

Tuesday, February 1, 2022

All Day

```
Featured    Tuesday, February 1, 2022    A Musical Celebration for Black History Month
On from Feb 1 - Feb 28                    Tuesday, February 1, 2022            All Day
tv    Online eventOnline eventThis February, VPL is excited to present an online concert
experience produced by Joy Bullen:Celebrations and Revelations 2022 for African
Heritage...View        DetailsTime         All Day              tv    Online eventOnline event
162 seat(s) remainingOnlinePerforming ArtsSpecial Events     /          Adults
Seniors      Teens      162 seat(s) remainingOnlinePerforming ArtsSpecial Events      /
Adults       Seniors      Teens
```

***

```python
# Replace two or more consecutive empty spaces with '*'.
    rawString = re.sub('[ ]{2,}', '*', rawString)
```

```
Featured     Tuesday, February 1, 2022     A Musical
Celebration for Black History Month             On from Feb 1 -
Feb 28                        Tuesday, February 1, 2022              All
Day                   tv     Online eventOnline eventThis February,
VPL is excited to present an online concert experience
produced by Joy Bullen:Celebrations and Revelations 2022
for African Heritage...View          DetailsTime            All Day
tv     Online eventOnline event  162 seat(s)
remainingOnlinePerforming ArtsSpecial Events       /
Adults        Seniors        Teens       162 seat(s)
remainingOnlinePerforming ArtsSpecial Events       /
Adults        Seniors        Teens

***
```

```
Featured*Tuesday, February 1, 2022*A Musical Celebration for Black
History Month*On from Feb 1 - Feb 28*Tuesday, February 1, 2022 *All
Day*tv*Online eventOnline eventThis February, VPL is excited to
present an online concert experience produced by Joy
Bullen:Celebrations and Revelations 2022 for African
Heritage...View*DetailsTime*All Day*tv*Online eventOnline
event*162 seat(s) remainingOnlinePerforming ArtsSpecial
Events*/*Adults*Seniors*Teens*162 seat(s)
remainingOnlinePerforming ArtsSpecial
Events*/*Adults*Seniors*Teens
***
```

Clean Data

```
# Remove hidden characters for tabs and new lines.
   rawString = re.sub(r"[\n\t]*", "", rawString)

   # Replace two or more consecutive empty spaces with '*'.
   rawString = re.sub('[ ]{2,}', '*', rawString)
   # Replace old string with new string.
   rawString.replace("Old String", "New String*")
```

```
for e in content:
   textContent  = e.get_attribute('innerHTML')
   # Beautiful soup removes HTML tags from our content if it exists.
   soup         = BeautifulSoup(textContent, features="lxml")
   rawString    = soup.get_text().strip()

   # Remove hidden characters for tabs and new lines.
   rawString = re.sub(r"[\n\t]*", "", rawString)

   # Replace two or more consecutive empty spaces with '*'
   rawString = re.sub('[ ]{2,}', '*', rawString)

   # Fine tune the results so they can be parsed.
   rawString = rawString.replace("Location", "Location*")
   rawString = rawString.replace("Registration closed", "Registration closed*")
   rawString = rawString.replace("Registration required", "Registration required*")
   rawString = rawString.replace("In Progress", "*In Progress*")
   rawString = rawString.replace("*/*", "/")
   rawString = rawString.replace("Full*","*Full*")

   print(rawString)
   print("***")
```

```
Featured*Tuesday, February 1, 2022*A Musical Celebration for Black
History Month*On from Feb 1 - Feb 28*Tuesday, February 1, 2022  *All
Day*tv*Online eventOnline eventThis February, VPL is excited to
present an online concert experience produced by Joy
Bullen:Celebrations and Revelations 2022 for African
Heritage...View*DetailsTime*All Day*tv*Online eventOnline
event*162 seat(s) remainingOnlinePerforming ArtsSpecial
Events/Adults*Seniors*Teens*162 seat(s)
remainingOnlinePerforming ArtsSpecial
Events/Adults*Seniors*Teens
***
```

```
Reading Buddies*Friday, November 19, 2021  *4:00 pm - 5:00
pm*Location*Fraserview Branch*In Progress**Registration closed*Reading
Circles/School Age Children*Teens*Registration closed*Reading Circles/School
Age Children*Teens*Time*4:00 pm -5:00 pm*Location*Fraserview Branch
***
```

```
#print(rawString)

eventArray = rawString.split('*')
```

Reading Buddies*Friday, November 19, 2021 *4:00 pm - 5:00 pm*Location*Fraserview Branch*In Progress**Registration closed*Reading Circles/School Age Children*Teens*Registration closed*Reading Circles/School Age Children*Teens*Time*4:00 pm -5:00 pm*Location*Fraserview Branch

```
EVENT_NAME = 0

EVENT_DATE = 1

EVENT_TIME = 2

eventName = eventArray[EVENT_NAME]

eventDate = eventArray[EVENT_DATE].strip() # remove leading and trailing spaces

eventTime = eventArray[EVENT_TIME].strip() # remove leading and trailing spaces

location  = eventArray[len(eventArray)-1]

print("Name:    " + eventName)

print("Date:    " + eventDate)

print("Time:    " + eventTime)

print("Location: " + location)

print("***")
```

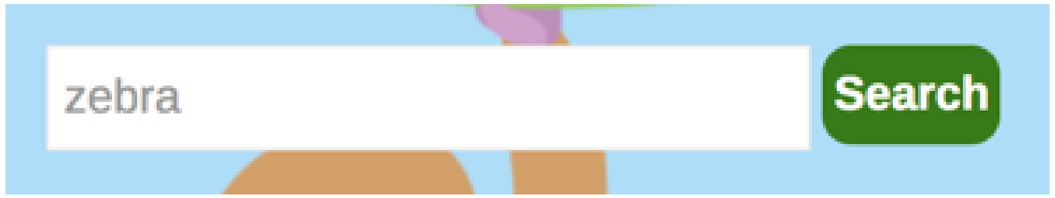
```
>>> str1="Reading Buddies*Friday, November 19, 2021  *4:00 pm - 5:00 pm*Location*Fraserview Branch*In Progress**Registration closed*Reading Circles/School Age Children*Teens*Registration closed*Reading Circles/School Age Children*Teens*Time*4:00 pm -5:00 pm*Location*Fraserview Branch "
>>> str1.split("*")
['Reading Buddies', 'Friday, November 19, 2021', '4:00 pm - 5:00 pm', 'Location', 'Fraserview Branch', 'In Progress', '', 'Registration closed', 'Reading Circles/School Age Children', 'Teens', 'Registration closed', 'Reading Circles/School Age Children', 'Teens', 'Time', '4:00 pm -5:00 pm', 'Location', 'Fraserview Branch ']
>>>
```

Sending Keys

How to automatically search for a specific topic on a website?

- find the search textbox with the CSS selector and
- use Selenium's *send_keys()* function to input your search topic

```
# Find the search input.
search = browser.find_element(By.CSS_SELECTOR,"#s")
search.send_keys("Zebra")
```



After sending keys we can then click the button to trigger the search:

```
# Find the search button - this is only enabled when a search query is entered
button = browser.find_element(By.CSS_SELECTOR,"#searchsubmit")
button.click()  # Click the button.
```

Clicking

- Selenium allows us automate clicks to elements. Notice here that the method used to locate the element is singular because we are only identifying only one element.
- find_elements_by_css_selector is not used:

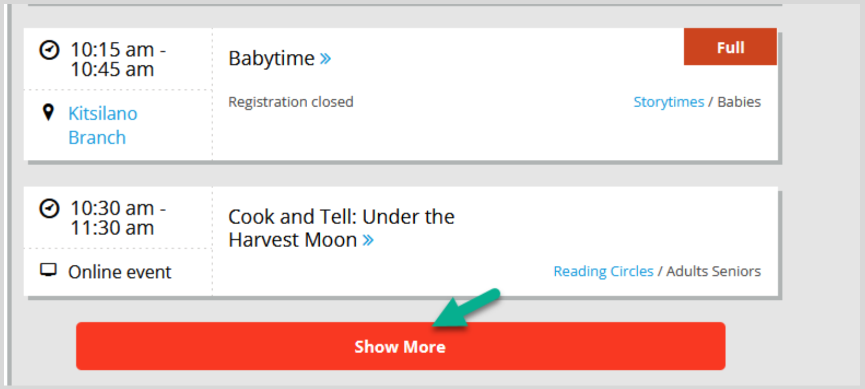
```
button = browser.find_element(By.CSS_SELECTOR,".btn-lg")
```

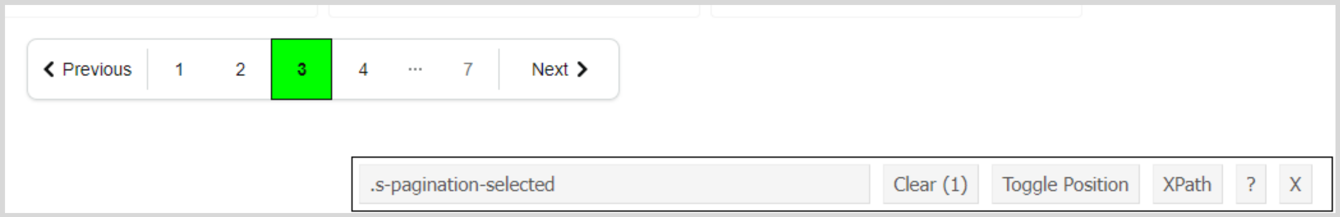
```
button = browser.find_element_by_css_selector(".btn-lg")

#button.click()

for i in range(0,20):

  button.click()
```


10:15 am - 10:45 am
Kitsilano Branch
Babytime »
Registration closed
Full
Storytimes / Babies
10:30 am - 11:30 am
Online event
Cook and Tell: Under the Harvest Moon »
Reading Circles / Adults Seniors

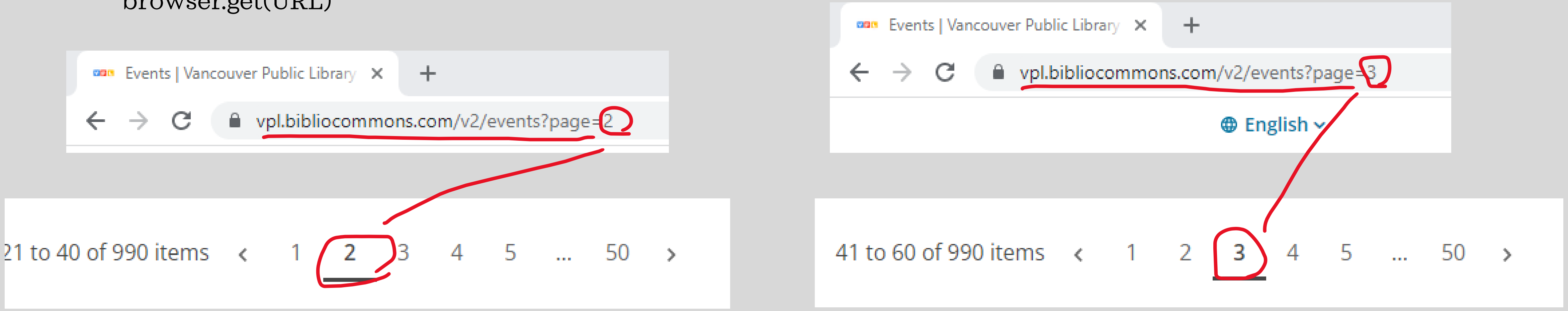

.s-pagination-selected

Next Page

https://vpl.bibliocommons.com/v2/events

For Loop:

```
for pageNum in range(1, 3):
    URL = "https://vpl.bibliocommons.com/v2/events?page=" + str(pageNum)
    print("Show Page ",pageNum)
    browser.get(URL)
```


Events | Vancouver Public Library
vpl.bibliocommons.com/v2/events?page=2
21 to 40 of 990 items
1 2 3 4 5 ... 50
Events | Vancouver Public Library
vpl.bibliocommons.com/v2/events?page=3
English
41 to 60 of 990 items
1 2 3 4 5 ... 50

Reading From CSV Files

Pandas.read_csv() function imports a CSV file to DataFrame format.

- ❑ FilePath: full directory of the file location
- ❑ header (int, list): this allow you to specify which row will be used as columns names for your DataFrame.
  - ➢ Default value is header=0;
  - ➢ if your file doesn’t have a header, simply set header=None
- ❑ names(array): List of column names to use
- ❑ sep: Specify a custom delimiter for the CSV imput, the default is a comma.
- ❑ index_col: This allow you to set which columns to be used as the index of the dataframe. The default value is None.
- ❑ usecol: Specify which columns to import to the dataframe.
- ❑ nrows: only read the number of first rows from the file
- ❑ skiprows: Line numbers to skip or number of lines to skip at the start of the file
- ❑ encoding: encoding to use for UTF when reading/writing (https://docs.python.org/3/library/codecs.html#standard-encodings)

Example-1

```
import pandas as pd
# Import data into a DataFrame.
path = "/Users/pm/Desktop/DayDocs/2019_2020/PythonForDataAnalytics/workingData/babysamp-98.txt"
df = pd.read_csv(path, skiprows=1,sep='\t', names=('MomAge', 'DadAge', 'MomEduc', 'MomMarital', 'numlive',
                    "dobmm", 'gestation', 'sex', 'weight', 'prenatalstart','orig.id', 'preemie'))
```

Example -2

```
import pandas as pd
# The data file path and file name need to be configured.
PATH     = "/Users/pm/Desktop/DayDocs/2019_2020/PythonForDataAnalytics/workingData/"
CSV_DATA = "phone_data.csv"
# Note this has a comma separator.
df = pd.read_csv(PATH + CSV_DATA, skiprows=1,  encoding = "ISO-8859-1", sep=',',
             names=('index', 'date', 'duration', 'item', 'month','network',
                 'network_type' ))
```

Pandas.read_table

Read general delimited file into Datafile.

Parameters:

- filepath:
- Sep: str, default ‘\t’
- Header
- Names
- Skiprows: line numbers to skip (0-indexed) or number of lines to skip at the start of the file.

Example

T/F

```
import pandas as pd
# Import data into a DataFrame.
Path = "/Pythonb/DataSets/babysamp-98.txt"
df = pd.read_table(Path)
df
```

```
import pandas as pd

# Import data into a DataFrame.

path = "/Users/pm/Desktop/DayDocs/2019_2020/PythonForDataAnalytics/workingData/babysamp-98.txt"

df = pd.read_csv(path, skiprows=1,sep='\t', names=('MomAge', 'DadAge', 'MomEduc', 'MomMarital', 'numlive',
                "dobmm", 'gestation', 'sex', 'weight', 'prenatalstart','orig.id', 'preemie'))
```

↓

```
df = pd.read_table(path, skiprows=1,names=('MomAge', 'DadAge', 'MomEduc', 'MomMarital', 'num
live', 'dobmm', 'gestation', 'sex', 'weight', 'prenatalstart','orig.id', 'preemie'))
```

Write DataFrame to CSV

How to write DataFrame content to a CSV file with the ***to_csv()*** function:

- ❑ path
- ❑ sep
- ❑ header
- ❑ Index
- ❑ encoding

Example-3

```
import pandas as pd
PATH        = "/Users/pm/desktop/"
CSV_FILE    = 'grades.csv'
dataset     = { "NumericGrade":[99,98,84], "LetterGrade":['A+', 'A', 'B']}
dfOut       = pd.DataFrame( data = dataset)

# Here I have decided to use a tab separator.
# The default separator is a comma which also could work.
dfOut.to_csv(PATH + CSV_FILE, sep='\t')

# Since I saved the file with a tab separator the instruction
# that reads the content must also use a tab separator.
dfIn        = pd.read_csv(DRIVER_PATH + CSV_FILE, sep='\t')
print(dfIn.head(2))
```

Exercise 1 & 2

Connecting to Databases (SQL Alchemy)

◦ **Creating a Database Table with a DataFrame**

```
# Create the database at the specified path.
DB_FILE    = 'forestFire.db'
engine     = create_engine('sqlite:///' + PATH + DB_FILE, echo=False)
# engine = create_engine('postgresql://scott:tiger@localhost:5432/mydatabase’)
#engine = create_engine('mysql://scott:tiger@localhost/foo')
connection = engine.connect()
```

Then the table is created and given data by using the DataFrame instruction *to_sql()*.

```
# Store data in database in a table named 'brazilForest'.
df.to_sql(name='table_name', con=connection, if_exists='replace', index=False)
```

https://docs.sqlalchemy.org/en/14/core/engines.html

Convert a DataFram to DB

```
import pandas as pd
from   sqlalchemy import create_engine

# The data file path and file name need to be configured.
PATH     = "C:\\datasets\\"
CSV_DATA = "brazil_forestFires.csv"

# Note this has a comma separator.
df = pd.read_csv(PATH + CSV_DATA, skiprows=1,  encoding = "ISO-8859-1", sep=',',
                 names=('year', 'state',  'month', 'number','date', ))
print(df.tail(2))

 # Create the database at the specified path.
DB_FILE    = 'forestFire.db'
engine     = create_engine('sqlite:///' + PATH + DB_FILE, echo=False)
connection = engine.connect()

# Store data in database in a table named 'brazilForest'.
df.to_sql(name='brazilForest', con=connection, if_exists='replace', index=False)
```

DB Brower for Sqlite from https://sqlitebrowser.org/

Exercise 3

Read from DB: Example 3 (Modified)

```
import pandas as pd
from sqlalchemy import create_engine
PATH     = "C:\\Python\\DataSets\\"
DB_FILE = 'forestFire.db'
engine = create_engine('sqlite:///' + PATH + DB_FILE, echo=False)
connection = engine.connect()
def showQueryResult(sql, connection):
    print("\n*** Showing SQL statement")
    subDf = pd.read_sql(sql, connection)
    print("\n*** Showing dataframe summary")
    return subDf
# Get DataFrame contents for 'Rio' and 'Sao Paulo' only.
sql = "SELECT * FROM brazilForest WHERE state = 'Rio' OR state='Sao Paulo' ORDER BY date"
newDf = showQueryResult(sql, connection)
print(newDf.tail())
```

```
# Placed query in this function to enable code re-usuability.
def showQueryResult(sql, dbconnection):
    print("\n*** Showing SQL statement")
    print(sql)
    # Perform query
    subDf = pd.read_sql(sql, dbconnection)
    print("\n*** Showing dataframe summary")
    return subDf

# Get DataFrame contents for 'Rio' and 'Sao Paulo' only.
sql    = "SELECT * FROM " + "brazilForest" \
       + " WHERE state = 'Rio' OR state='Sao Paulo' " \
       + " ORDER BY date"

newDf = showQueryResult(sql, connection)
print(newDf.tail())
```

Exercise 4 & 5

Read from Excel File

How to read from an Excel document into a dataframe

❑ the following packages must be installed in either the Spyder console or in the PyCharm terminal:

- pip install xlrd
- pip install openpyxl

```
import openpyxl
import pandas as pd

PATH      = "C:\\datasets\\"
FILE_NAME  = "Tides.xlsx"
df        = pd.read_excel(PATH + FILE_NAME, sheet_name='Sheet1')
df.to_excel(PATH + "NewFile.xlsx", sheet_name='Sheet1')
print (df)
```