chap6-ret_proc_storeData

June 18, 2022

[5]:
```
# import genfromtxt function
from numpy import genfromtxt

# read comma separated file
product_data = genfromtxt('demo.csv', delimiter=',')

# display initial 5 records
print(product_data)
```

```
[[14. 32. 33.]
 [24. 45. 26.]
 [27. 38. 39.]]
```

[13]:
```
# import numpy
import numpy as np

# create a sample array
sample_array = np.asarray([[1,2,3], [4,5,6], [7,8,9]])

# write sample array to CSV file
np.savetxt("my_first_demo.csv", sample_array, delimiter=",")
```

[14]:
```
# import pandas
import pandas as pd

# read CSV file
sample_df = pd.read_csv('demo.csv', sep=',', header=None)

# display initial 5 records
sample_df.head()
```

[14]:

	0	1	2
0	14	32	33
1	24	45	26
2	27	38	39

[15]:
```
# check my_first_demo.csv
check_first_demo = pd.read_csv('my_first_demo.csv', sep=',', header=None)

# display initial 5 records - in this case all array
check_first_demo.head()
```

[15]:

	0	1	2
0	1.0	2.0	3.0
1	4.0	5.0	6.0
2	7.0	8.0	9.0

[16]:
```
# save DataFrame to CSV file
sample_df.to_csv('demo_sample_df.csv')
```

[17]:
```
# check what you have saved
check_saved = pd.read_csv('demo_sample_df.csv', sep=',', header=None)

# display initial 5 records - data overwrited (in this case doesn't matter)
check_saved.head()
```

[17]:

	0	1	2	3
0	NaN	0	1	2
1	0.0	14	32	33
2	1.0	24	45	26
3	2.0	27	38	39

[2]:
```
# import pandas
import pandas as pd

# reading and writing from Excel

# read excel file
df = pd.read_excel('employee.xlsx', sheet_name='performance')

# display initial 5 records
df.head()
```

[2]:

	name	performance_score
0	Allen Smith	723
1	S Kumar	520
2	Jack Morgan	674
3	Ying Chin	556
4	Dheeraj Patel	711

[3]:
```
# reading and writing data from JSON

# reading JSON file
```

```
df = pd.read_json('employee.json')

# display initial 5 records
df.head()
```

[3]:

	name	age	income	gender	department	grade
0	Allen Smith	45.0	NaN	None	Operations	G3
1	S Kumar	NaN	16000.0	F	Finance	G0
2	Jack Morgan	32.0	35000.0	M	Finance	G2
3	Ying Chin	45.0	65000.0	F	Sales	G3
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2

[4]:
```
# writing DataFrame to JSON file
df.to_json('employee_demo.json', orient="columns")
```

[5]:
```
# write DataFrame to hdf5
df.to_hdf('employee.h5', 'table', append=True)
```

[18]:
```
# read a hdf5 file
df = pd.read_hdf('employee.h5', 'table')

# display initial 5 records
df.head()
```

[18]:

	name	age	income	gender	department	grade
0	Allen Smith	45.0	NaN	NaN	Operations	G3
1	S Kumar	NaN	16000.0	F	Finance	G0
2	Jack Morgan	32.0	35000.0	M	Finance	G2
3	Ying Chin	45.0	65000.0	F	Sales	G3
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2

[19]:
```
# reading and writing data from HTML tables

# read HTML table from given URL
table_url = 'https://en.wikipedia.org/wiki/
 ↪List_of_sovereign_states_and_dependent_territories_in_North_America'
df_list = pd.read_html(table_url)
```

[20]:
```
print("Number of DataFrame: ", len(df_list))
```

```
Number of DataFrame:  8
```

[21]:
```
# check first DataFrame
df_list[0].head()
```

[21]:

	Flag	English short name	English long name	\
0	NaN	Antigua and Barbuda[n 1]	Antigua and Barbuda	

```
1   NaN         Bahamas, The[n 1]  Commonwealth of The Bahamas
2   NaN             Barbados[n 1]                     Barbados
3   NaN          Belize[n 1][n 2]                       Belize
4   NaN               Canada[n 3]                       Canada

          Domestic short name(s)     Capital               Currency  \
0   English: Antigua and Barbuda  St. John's  East Caribbean dollar
1              English: Bahamas      Nassau        Bahamian dollar
2            English: Barbados  Bridgetown       Barbadian dollar
3              English: Belize    Belmopan          Belize dollar
4  English: CanadaFrench: Canada      Ottawa        Canadian dollar

   Population
0       97118
1      389482
2      287025
3      390353
4    35151728
```

[22]:
```
# write DataFrame to raw HTML
df_list[1].to_html('country.html')
```

[29]:
```
# import numpy
import numpy as np

# import pandas
import pandas as pd

# import genfromtxt function
from numpy import genfromtxt

from pandas.io.parquet import to_parquet
```

[33]:
```
# reading and writing data from a pickle pandas object

# import pandas
import pandas as pd

# read CSV file
df = pd.read_csv('demo.csv', sep = ',', header=None)

# save DataFrame object in pickle file
df.to_pickle('demo_obj.pkl')
```

[34]:
```
# read DataFrame object from pickle file
pickle_obj = pd.read_pickle('demo_obj.pkl')
```

```
# display initial 5 records
pickle_obj.head()
```

[34]:

	0	1	2
0	14	32	33
1	24	45	26
2	27	38	39

[36]:
```
# import sqlite3
import sqlite3

# create connection. This will create the connection with employee
# database. If the database does not exist it will create the database

# crete database if not exists
conn = sqlite3.connect('employee.db')

# create cursor
cur = conn.cursor()
```

[39]:
```
# execute SQL query and create the database table
# cur.execute("create table emp(eid int, salary int)")

# execute SQL query and write the data into database
# cur.execute("insert into emp values(105, 57000)")

# commit the transaction
# conn.commit()
```

[40]:
```
# execute SQL query and Read the data from the database
cur.execute('select * from emp')
```

[40]:
```
<sqlite3.Cursor at 0x203e90eef10>
```

[41]:
```
# fetch records
print(cur.fetchall())
```

```
[(105, 57000)]
```

[42]:
```
# close the Database connection
conn.close()
```

[59]:
```
# reading and writing from MySQL

# import pymysql connector module
import pymysql
```

[60]:
```
# create a connection object using connect() method

# connection = pymysql.connect(host='localhost',
# user='root', password='password', db='employee',
# charset='utf8mb4', cursorclass=pymysql.cursors.DictCursor)
# try:
#     with connection.cursor() as cur:
#         # inject a record in database
#         sql_query = "INSERT INTO 'emp' ('aid', 'salary') VALUES (%s, %s)"
#         cur.execute(sql_query, (104, 43000))

# commit the record insertion explicitly
# connection.commit()

# with connection.cursor() as cur:
#     # read records from employee table
#     sql_query = "SELECT * FROM 'emp'"
#     cur.execute(sql_query)
#     table_data = cur.fetchall()
#     print(table_data)

# except:
#     print("Exception Occured")
# finally:
#     connection.close()
```

[68]:
```
# reading and writing data from MongoDB

# create mongo client
# client = pymongo.MongoClient()

# get database
# db = client.employee

# get the collection from database
# collection = db.employee

# write the data using insert_one() method
# employee_salary = {"eid":114, "salary":25000}
# collection.insert_one(employee_salary)

# create a dataframe with fetched data
# data = pd.DataFrame(list(collection.find()))
```

[71]:
```
# reading and writing data from cassandra

# import the cluster
```

```
# from cassandra.cluster import Cluster

# Creating a cluster object
# cluster = Cluster()

# Create connections by calling Cluster.connect():
# conn = cluster.connect()

# Execute the insert query
# session.execute(
# """ INSERT INTO users (eid, ename, age) VALUES (%(eid)s, %(ename)s, %(age)s,␣
 ↪%(name)s)""",
# {'eid':101, 'ename': "Steve smith", 'age': 42})

# Execute the select query
# rows = conn.execute('SELECT * FROM users')

# Print the results
# for emp_row in rows:
#    print(emp_row.eid, emp_row.ename, emp_row.age)

# Create a dataframe and assign fetched data to DataFrame
# data = pd.DataFrame(rows)
```

[ ]:
```
# reading and writing data from redis
# Import module
# import redis

# Create connection
# r = redis.Redis(host='localhost', port=6379, db=0)

# Setting key-value pair
# r.set('eid', '101')

# Get value for given key
# value=r.get('eid')

# Print the value
# print(value)
```