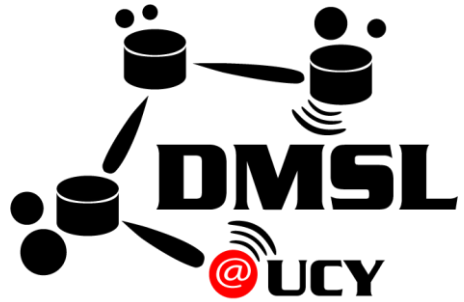
EPL646 – Advanced Topics in Databases
Panda, Dask & Parquet

Christoforos Panayiotou
http://www.cs.ucy.ac.cy/~dzeina/courses/epl646/labs/lab.html



Python Installation

- Installing with.exe and .msi files may be blocked (as it needs admin rights), thus you will not be able to install Python that way
  - There is another way though that doesn't require you to download an installer
- Go to https://python.org
  - Instead of going straight for the main download button, go to “All Releases”
  - Select the Python version you want under “Looking for a specific release?”
  - Scroll down to where it says “Files” and you'll see “Windows embeddable package (32-bit)” and “Windows embeddable package (64-bit)”
  - Download the one corresponding to which bitness you require
  - Once that file is downloaded, simply un-zip it wherever you want Python to be installed to

Python Installation

- Adjust the installation
  - By default the embedable package severely limits the search-path using a ***_pth*** file;
    - We disable that and also create a missing directory (which is fine to be empty)
    - In a terminal (PowerShell in windows) inside the directory you used for python:
    - Rename the ***_pth*** file: *mv python3??._pth python??.pth*
    - Create the directory DLLs: mkdir DLLs
- Get get-pip
  - We want to use pip to install everything else, which we can get and install by:
    - In a terminal inside the directory you used for python give the following commands:

      wget https://bootstrap.pypa.io/get-pip.py -o get-pip.py
      .\python.exe get-pip.py

Python Installation

- Get virtualenv and setup a test environment
  - We can now get virtualenv and create a new environment and install packages into it
  - There is one wrinkle after a new environment is created a file (*python3??.zip*) needs to be manually copied into it
    - In a terminal inside the directory you used for python give the following commands:
      .\python.exe -m pip install virtualenv
      .\python.exe -m virtualenv ..\testenv
      cp .\python3??.zip ..\testenv\Scripts
- Run python in your test environment
  - Open a terminal (PowerShell in windows) in the directory that contains your test environment and run *Activate.ps1*: *.\testenv\Scripts\Activate.ps1*
    - Running PowerShell scripts must be enabled by allowing execution of remotely signed scripts (probably will require admin rights):
      Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
  - Alternatively open a command prompt (not a terminal) and run *Activate.bat*:
    testenv\Scripts\Activate.bat
  - In the opened terminal (or command prompt) you can now run python or pip
    - Verify installation by running *python --version* and *pip --version*

Installing packages (Panda, Dask and Parquet)

- To use Panda, Dask or Parquet in our python installation we need to add the relevant packages (*pandas*, *dask* and *pyarrow*)
- Within your test environment and run the following:
  - *pip install pandas*
  - *pip install dask*
  - *pip install pyarrow*
- Test your installation by running the lecture examples
  - Download the data file:
    wget https://media.geeksforgeeks.org/wp-content/uploads/nba.csv -o nba.csv
  - Start python and run the code of the lecture slides 3-43, 3-44

Practice

- Modify the example code to convert the nba.csv file to parquet format

Using SQLite from python

- First you need to import the relevant module (no need to install any packages):
  import sqlite3
- Then you must connect to your SQLite file
  - Creating a brand-new SQLite database is as easy as growing a connection to the usage of the sqlite3 module inside the Python preferred library
  - To establish a connection, all you have to do is to pass the file path to the connect(…) method in the sqlite3 module
  - If the database represented by the file does not exist, it will be created under this path

```
try:
    conn = sqlite3.connect("Sqlite3.db")
    print("Database Sqlite3.db formed.")
except:
    print("Database Sqlite3.db not formed.")
```

Using SQLite from python

- Adding data to SQLite using Panda
  - The Panda library makes it very easy to import data to SQLite,
  - We just need call the *to_sql()* method on a Dataframe

```
df = pandas.read_csv("datafile.csv")
df.to_sql(table_name, conn, if_exists='append', index=False)
```

- Don’t forget to close your database

```
if conn:
        conn.close()
```

Practice

- Download a large csv file (you can find some examples here: https://github.com/datablist/sample-csv-files)
- Transform the downloaded csv to parquet and SQLite formats

Adding performance

- You can use different libraries to read data in python
  - Pyarrow for example provides modules for reading csv files
    - First you need to import the correct module (*csv*):
      from pyarrow import csv
    - Then you use the *read_csv()* method:
      table = csv.read_csv("datafile.csv")
    - You can then easily write the data in parquet format (don't forget to first import the *parquet* module):
      parquet.write_table(table, "datafile.parquet")
  - The same goes for the Dask library
    - First you need to import the correct module (*read_csv*):
      from dask.dataframe import read_csv
    - Use the read_csv() method for reading:
      dask_df = read_csv("datafile.csv", dtype={'column_xpto': 'float64'})
    - And finally write to parquet format using the to_parquet() method:
      dask_df.to_parquet("datafile.parquet")
    - Note that, by default, the Dask library will create a distributed parquet file

Practice

- Convert the csv file you downloaded before to parquet format using the Pyarrow and Dask libraries
- Compare the created files as well as the execution times for each method (using Panda, Pyarrow and Dask)

Questions?

http://www.cs.ucy.ac.cy/~dzeina/courses/epl646/labs/lab.html