notebook.community

EDIT AND RUN

Lesson 2: Setup Jupyter Notebook for Data Analysis

Learning Objectives:

1. Create Python tools for data analysis using Jupyter Notebooks

2. Learn how to access data from MySQL databases for data analysis

Exercise 1: Install Anaconda

Access https://conda.io/miniconda.html and download the Windows Installer.

Run the following commands on the Anaconda command prompt:

```
conda install numpy, pandas, matplotlib
conda update conda
```

Sometimes data analysis requires previous versions of Python or other tools for a project.

Next we will setup three environments that can be used with various project requirements.

Exercise 2: Configure conda environments for Python 2 and Python 3 data analysis

To create a **Python 2** enviroment run the following from the Anaconda command prompt:

```
conda update conda -y
conda create -n py2 python=2 anaconda jupyter notebook -y
```

To activate the environment:

```
source activate py2
```

On MacOS or Linux:

```
source activate py2
```

To deactivate the environment:

```
source deactivate py2
```

On MacOS or Linux:

```
source deactivate py2
```

To create the **Python 3** environment run the following from the Anaconda command prompt:

```
conda create -n py3 python=3 anaconda jupyter notebook -y
```

To activate the environment:

```
activate py3
```

On MacOS or Linux:

```
source activate py3
```

To deactivate the enviroment:

```
deactivate py3
```

On MacOS or Linux:

```
source deactivate py3
```

Setup Jupyter Notebook to access data from MySQL databases

Exercise 3: Load the mysql libraries into the environment and access data from MySQL database

Run the following commands from the Anaconda command line:

```
pip install ipython-sql
conda install mysql-python
```

This will install sql magic capabilities to Jupyter Notebook

Load the sql magic jupyter notebook extension:

In [178]:

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

Configure sql magic to output queries as pandas dataframes:

In [179]:

```
%config SqlMagic.autopandas=True
```

Import the data analysis libraries:

In [180]:

```
import pandas as pd
import numpy as np
```

Import the MySQLdb library

In [181]:

```
import MySQLdb
```

Connect to the MySQL database using sql magic commands

The connection to the MySQL database uses the following format:

```
mysql://username:password@hostname/database
```

To start a sql command block type:

```
%%sql
```

Note: Make sure the %%sql is on the top of the cell

Then the remaining lines can contain SQL code.

Example: to connect to **pidata** database and select records from the **temps** table:

```
In [182]:

%%sql mysql://pilogger:foobar@172.20.101.81/pidata
SELECT * FROM temps LIMIT 10;
```

```
10 rows affected.

Out[182]:
```

	device	datetime	temp	hum
0	pi223	2017-07-15 23:24:42	72.86	44.5
1	pi223	2017-07-15 23:25:44	72.86	44.9
2	pi223	2017-07-15 23:26:45	72.86	44.0
3	pi223	2017-07-15 23:27:46	73.04	44.5
4	pi223	2017-07-15 23:28:48	73.04	45.0
5	pi223	2017-07-15 23:29:49	73.04	45.0
6	pi223	2017-07-15 23:30:50	73.22	44.5
7	pi223	2017-07-15 23:31:51	73.04	44.0

	device	datetime	temp	hum
8	pi223	2017-07-15 23:32:52	73.04	44.6
9	pi223	2017-07-15 23:33:54	73.04	44.6

Example to create a pandas dataframe using the results of a mysql query

```
In [183]:

df = %sql SELECT * FROM temps WHERE datetime > date(now());
```

```
376 rows affected.
```

```
In [184]:

df
```

Out[184]:

	device	datetime	temp	hum
0	pi223	2017-07-23 00:00:46	74.30	45.6
1	pi223	2017-07-23 00:03:50	74.48	45.7
2	pi223	2017-07-23 00:06:52	74.48	45.7
3	pi223	2017-07-23 00:09:53	74.48	46.4
4	pi223	2017-07-23 00:12:54	74.48	45.7
5	pi223	2017-07-23 00:15:56	74.48	45.7

	device	datetime	temp	hum
6	pi223	2017-07-23 00:18:57	74.48	45.7
7	pi223	2017-07-23 00:21:59	74.48	45.7
8	pi223	2017-07-23 00:25:00	74.30	45.6
9	pi223	2017-07-23 00:28:02	74.30	45.6
10	pi223	2017-07-23 00:31:03	74.30	45.6
11	pi223	2017-07-23 00:34:05	74.30	45.6
12	pi223	2017-07-23 00:37:06	74.12	45.5
13	pi223	2017-07-23 00:40:07	73.94	45.5
14	pi223	2017-07-23 00:43:09	73.94	45.5
15	pi223	2017-07-23 00:46:10	73.76	45.4
16	pi223	2017-07-23 00:49:12	73.58	45.3
17	pi223	2017-07-23 00:52:13	73.40	45.2
18	pi223	2017-07-23 00:55:15	73.40	45.2
19	pi223	2017-07-23 00:58:16	73.40	45.2
20	pi223	2017-07-23 01:01:17	73.22	45.1
21	pi223	2017-07-23 01:04:19	73.22	45.1
22	pi223	2017-07-23 01:07:20	73.04	45.0
23	pi223	2017-07-23 01:10:22	73.04	45.0
24	pi223	2017-07-23 01:13:23	73.22	45.1
25	pi223	2017-07-23 01:16:24	73.04	45.0
26	pi223	2017-07-23 01:19:26	73.04	45.0

	device	datetime	temp	hum
27	pi223	2017-07-23 01:22:27	73.04	45.0
28	pi223	2017-07-23 01:25:29	72.86	44.9
29	pi223	2017-07-23 01:28:30	73.04	45.0
...	...	...	...	...
346	pi223	2017-07-23 17:27:35	75.74	46.4
347	pi223	2017-07-23 17:30:37	75.74	46.4
348	pi223	2017-07-23 17:33:38	75.74	46.4
349	pi223	2017-07-23 17:36:40	75.74	45.5
350	pi223	2017-07-23 17:39:41	75.56	44.7
351	pi223	2017-07-23 17:42:43	75.38	45.5
352	pi223	2017-07-23 17:45:44	75.56	46.3
353	pi223	2017-07-23 17:48:45	75.38	45.6
354	pi223	2017-07-23 17:51:47	75.20	44.4
355	pi223	2017-07-23 17:54:48	75.20	45.1
356	pi223	2017-07-23 17:57:50	75.02	44.8
357	pi223	2017-07-23 18:00:51	75.02	44.9
358	pi223	2017-07-23 18:03:53	75.02	44.6
359	pi223	2017-07-23 18:06:54	75.02	44.5
360	pi223	2017-07-23 18:09:55	74.84	45.3
361	pi223	2017-07-23 18:12:57	74.84	45.9
362	pi223	2017-07-23 18:15:58	75.02	46.0

	device	datetime	temp	hum
363	pi223	2017-07-23 18:19:00	75.02	46.0
364	pi223	2017-07-23 18:22:01	74.66	45.8
365	pi223	2017-07-23 18:25:03	74.84	45.9
366	pi223	2017-07-23 18:28:04	74.84	45.9
367	pi223	2017-07-23 18:31:06	75.02	47.4
368	pi223	2017-07-23 18:34:10	75.74	47.9
369	pi223	2017-07-23 18:37:13	75.74	47.5
370	pi223	2017-07-23 18:40:15	75.74	47.5
371	pi223	2017-07-23 18:43:16	75.74	47.4
372	pi223	2017-07-23 18:46:18	75.92	47.6
373	pi223	2017-07-23 18:49:19	75.74	47.4
374	pi223	2017-07-23 18:52:21	75.74	46.4
375	pi223	2017-07-23 18:55:25	75.38	46.2

376 rows × 4 columns

Note the data type of the dataframe df:

```
In [185]:

type(df)
```

```
Out[185]:

pandas.core.frame.DataFrame
```

Use %%sql to start a block of sql statements

Example: Show tables in the pidata database

In [186]:

```
%%sql
use pidata;
show tables;
```

```
0 rows affected.
2 rows affected.
```

Out[186]:

	Tables_in_pidata
0	temps
1	temps3

Exercise 4: Another way to access mysql data and load into a pandas dataframe

Connect using the mysqldb python library:

In [187]:

```
#Enter the values for you database connection
database = "pidata"               # e.g. "pidata"
hostname = "172.20.101.81"        # e.g.: "mydbinstance.xyz.us-east-1.rds.ama
port = 3306                       # e.g. 3306
uid = "pilogger"                  # e.g. "user1"
pwd = "foobar"                    # e.g. "Password123"
```

In [188]:

```
conn = MySQLdb.connect( host=hostname, user=uid, passwd=pwd, db=database )
cur = conn.cursor()
```

Create a dataframe from the results of a sql query from the pandas object:

In [189]:

```
new_dataframe = pd.read_sql("SELECT * \
                FROM temps",
                con=conn)
conn.close()
```

In [190]:

```
new_dataframe
```

Out[190]:

	device	datetime	temp	hum
0	pi223	2017-07-15 23:24:42	72.86	44.5
1	pi223	2017-07-15 23:25:44	72.86	44.9
2	pi223	2017-07-15 23:26:45	72.86	44.0
3	pi223	2017-07-15 23:27:46	73.04	44.5
4	pi223	2017-07-15 23:28:48	73.04	45.0
5	pi223	2017-07-15 23:29:49	73.04	45.0
6	pi223	2017-07-15 23:30:50	73.22	44.5
7	pi223	2017-07-15 23:31:51	73.04	44.0

	device	datetime	temp	hum
8	pi223	2017-07-15 23:32:52	73.04	44.6
9	pi223	2017-07-15 23:33:54	73.04	44.6
10	pi223	2017-07-15 23:34:55	72.86	44.2
11	pi223	2017-07-15 23:35:56	72.86	44.5
12	pi223	2017-07-15 23:36:57	72.68	44.4
13	pi223	2017-07-15 23:37:58	72.68	44.6
14	pi223	2017-07-15 23:39:00	72.68	44.8
15	pi223	2017-07-15 23:40:01	72.50	44.7
16	pi223	2017-07-15 23:41:02	72.86	44.9
17	pi223	2017-07-15 23:42:03	72.68	44.8
18	pi223	2017-07-15 23:43:05	72.68	44.8
19	pi223	2017-07-15 23:44:06	72.68	44.8
20	pi223	2017-07-15 23:45:07	72.68	44.8
21	pi223	2017-07-15 23:46:08	72.68	44.8
22	pi223	2017-07-15 23:47:10	72.68	44.8
23	pi223	2017-07-15 23:48:11	72.68	44.8
24	pi223	2017-07-15 23:49:12	72.50	44.7
25	pi223	2017-07-15 23:50:13	72.68	44.8
26	pi223	2017-07-15 23:51:14	72.68	44.8
27	pi223	2017-07-15 23:52:16	72.50	44.7
28	pi223	2017-07-15 23:53:17	72.50	44.7

	device	datetime	temp	hum
29	pi223	2017-07-15 23:54:18	72.50	44.7
...	...	...	...	...
3594	pi223	2017-07-23 17:27:35	75.74	46.4
3595	pi223	2017-07-23 17:30:37	75.74	46.4
3596	pi223	2017-07-23 17:33:38	75.74	46.4
3597	pi223	2017-07-23 17:36:40	75.74	45.5
3598	pi223	2017-07-23 17:39:41	75.56	44.7
3599	pi223	2017-07-23 17:42:43	75.38	45.5
3600	pi223	2017-07-23 17:45:44	75.56	46.3
3601	pi223	2017-07-23 17:48:45	75.38	45.6
3602	pi223	2017-07-23 17:51:47	75.20	44.4
3603	pi223	2017-07-23 17:54:48	75.20	45.1
3604	pi223	2017-07-23 17:57:50	75.02	44.8
3605	pi223	2017-07-23 18:00:51	75.02	44.9
3606	pi223	2017-07-23 18:03:53	75.02	44.6
3607	pi223	2017-07-23 18:06:54	75.02	44.5
3608	pi223	2017-07-23 18:09:55	74.84	45.3
3609	pi223	2017-07-23 18:12:57	74.84	45.9
3610	pi223	2017-07-23 18:15:58	75.02	46.0
3611	pi223	2017-07-23 18:19:00	75.02	46.0
3612	pi223	2017-07-23 18:22:01	74.66	45.8

	device	datetime	temp	hum
3613	pi223	2017-07-23 18:25:03	74.84	45.9
3614	pi223	2017-07-23 18:28:04	74.84	45.9
3615	pi223	2017-07-23 18:31:06	75.02	47.4
3616	pi223	2017-07-23 18:34:10	75.74	47.9
3617	pi223	2017-07-23 18:37:13	75.74	47.5
3618	pi223	2017-07-23 18:40:15	75.74	47.5
3619	pi223	2017-07-23 18:43:16	75.74	47.4
3620	pi223	2017-07-23 18:46:18	75.92	47.6
3621	pi223	2017-07-23 18:49:19	75.74	47.4
3622	pi223	2017-07-23 18:52:21	75.74	46.4
3623	pi223	2017-07-23 18:55:25	75.38	46.2

3624 rows × 4 columns

Now let's create the tables to hold the sensor data from our Raspberry Pi

Logon using an admin account and create a table called temps3 to hold sens

```
The table contains the following fields:
device       -- VARCHAR, Name of the device that logged the data
datetime     -- DATETIME, Date time in ISO 8601 format YYYY-MM-DD HH:MM:SS
temp         -- FLOAT, temperature data
hum          -- FLOAT, humidity data
```

In [191]:

```
%%sql mysql://admin:admin@172.20.101.81/pidata

DROP TABLE if exists temps3;

CREATE TABLE temps3 (
    device varchar(20) DEFAULT NULL,
    datetime datetime DEFAULT NULL,
    temp float DEFAULT NULL,
    hum float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
0 rows affected.
0 rows affected.

Out[191]:
```

Next we will create a user to access the newly created table that will be used by the Raspberry Pi program

Example: Start a connection using an admin account, create a new user called user1. Grant limited privileges to the pidata.temps3 table

Note: Creating a user with @'%' allows the user to access the database from any host

```
In [143]:

%%sql mysql://admin:admin@172.20.101.81
CREATE USER 'user1'@'%' IDENTIFIED BY 'logger';
GRANT SELECT, INSERT, DELETE, UPDATE ON pidata.temps3 TO 'user1'@'%';
FLUSH PRIVILEGES;
```

```
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[143]:

In [144]:

```
%sql select * from mysql.user;
```

11 rows affected.

Out[144]:

	Host	User	Select_priv	Insert_priv	Update
0	localhost	root	Y	Y	Y
1	localhost	mysql.sys	N	N	N
2	localhost	debian-sys-maint	Y	Y	Y
3	localhost	rmj	Y	Y	Y
4	localhost	phpmyadmin	N	N	N
5	192.168.8.131	pilogger	N	N	N
6	192.168.8.131	rmj	Y	Y	Y
7	%	pilogger	N	N	N
8	%	rmj	Y	Y	Y
9	%	admin	Y	Y	Y
10	%	user1	N	N	N

11 rows × 45 columns

Next we will test access to the newly created table using the new user Start a new connection using the new user

```
In [146]:

%%sql mysql://user1:logger@172.20.101.81/pidata
select * from temps3;
```

```
0 rows affected.

Out[146]:
```

Let's add some test data to make sure we can insert using the new user

```
In [174]:

for x in range(10):
    %sql INSERT INTO temps3 (device,datetime,temp,hum) VALUES('pi222',date(no
```

```
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
1 rows affected.
```

```
In [175]:

%sql SELECT * FROM temps3;
```

20 rows affected.

Out[175]:

	device	datetime	temp	hum
0	pi222	2017-07-23	73.2	22.0
1	pi222	2017-07-23	73.2	22.0
2	pi222	2017-07-23	73.2	22.0
3	pi222	2017-07-23	73.2	22.0
4	pi222	2017-07-23	73.2	22.0
5	pi222	2017-07-23	73.2	22.0
6	pi222	2017-07-23	73.2	22.0
7	pi222	2017-07-23	73.2	22.0
8	pi222	2017-07-23	73.2	22.0
9	pi222	2017-07-23	73.2	22.0
10	pi222	2017-07-23	73.2	22.0
11	pi222	2017-07-23	73.2	22.0
12	pi222	2017-07-23	73.2	22.0
13	pi222	2017-07-23	73.2	22.0
14	pi222	2017-07-23	73.2	22.0
15	pi222	2017-07-23	73.2	22.0
16	pi222	2017-07-23	73.2	22.0
17	pi222	2017-07-23	73.2	22.0

	device	datetime	temp	hum
18	pi222	2017-07-23	73.2	22.0
19	pi222	2017-07-23	73.2	22.0

Now we will delete the rows in the database

```
In [166]:

%sql DELETE FROM temps3;
```

```
10 rows affected.

Out[166]:
```

```
In [168]:

%sql SELECT * FROM temps3;
```

```
0 rows affected.

Out[168]:
```

```
In [133]:

%%sql mysql://admin:admin@172.20.101.81
drop user if exists 'user1'@'%';
```

```
0 rows affected.

Out[133]:
```

```
In [134]:

%sql select * from mysql.user;
```

```
10 rows affected.

Out[134]:
```

	Host	User	Select_priv	Insert_priv	Update_
0	localhost	root	Y	Y	Y
1	localhost	mysql.sys	N	N	N
2	localhost	debian-sys-maint	Y	Y	Y
3	localhost	rmj	Y	Y	Y
4	localhost	phpmyadmin	N	N	N
5	192.168.8.131	pilogger	N	N	N
6	192.168.8.131	rmj	Y	Y	Y
7	%	pilogger	N	N	N
8	%	rmj	Y	Y	Y
9	%	admin	Y	Y	Y

```
10 rows × 45 columns
```

Content source: richjimenez/mysql-data-raspberry-pi

Similar notebooks:

- lesson-2-jupyter-notebook-for-data-analysis
- Environment setup
- conda_tutorial
- 00-Installing-Python-pyradi
- lesson-3-mysql-database-for-raspberry-pi-sensor-ata
- 0.common_questions
- NewishPythonTools
- requirements
- Build_and_test_environment
- ee_python_api_instructions