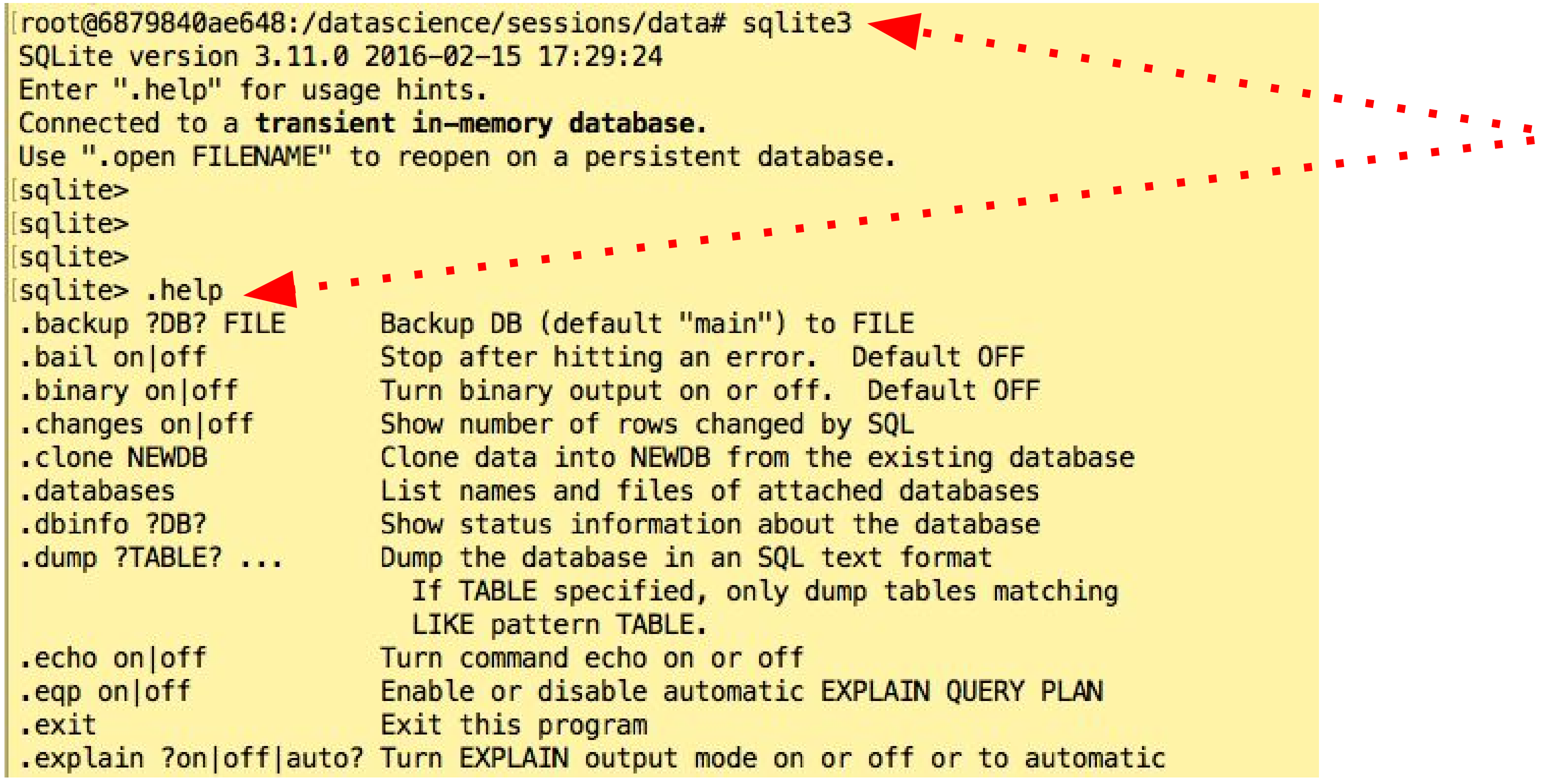
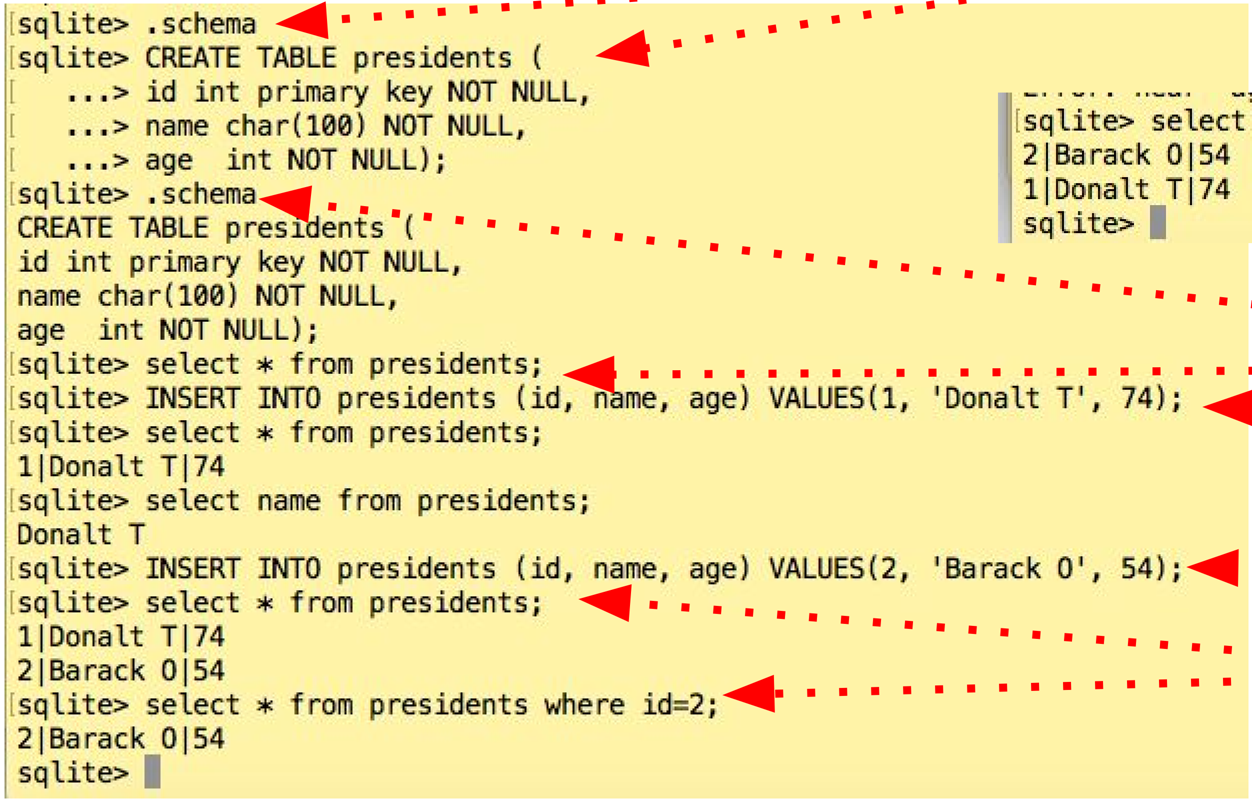
Hello - Database & Graphics

Databases - SQLite

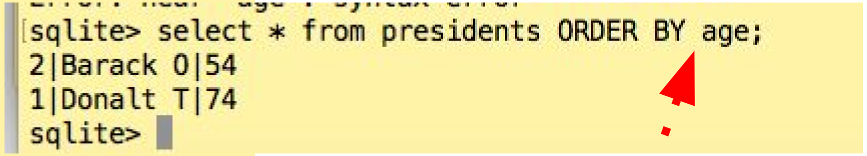


```
root@6879840ae648:/datascience/sessions/data# sqlite3
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
sqlite>
sqlite>
sqlite> .help
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail on|off           Stop after hitting an error.  Default OFF
.binary on|off         Turn binary output on or off.  Default OFF
.changes on|off        Show number of rows changed by SQL
.clone NEWDB           Clone data into NEWDB from the existing database
.databases             List names and files of attached databases
.dbinfo ?DB?           Show status information about the database
.dump ?TABLE? ...      Dump the database in an SQL text format
                         If TABLE specified, only dump tables matching
                         LIKE pattern TABLE.
.echo on|off           Turn command echo on or off
.eqp on|off            Enable or disable automatic EXPLAIN QUERY PLAN
.exit                  Exit this program
.explain ?on|off|auto? Turn EXPLAIN output mode on or off or to automatic
```

Databases - SQLite (Cont..)

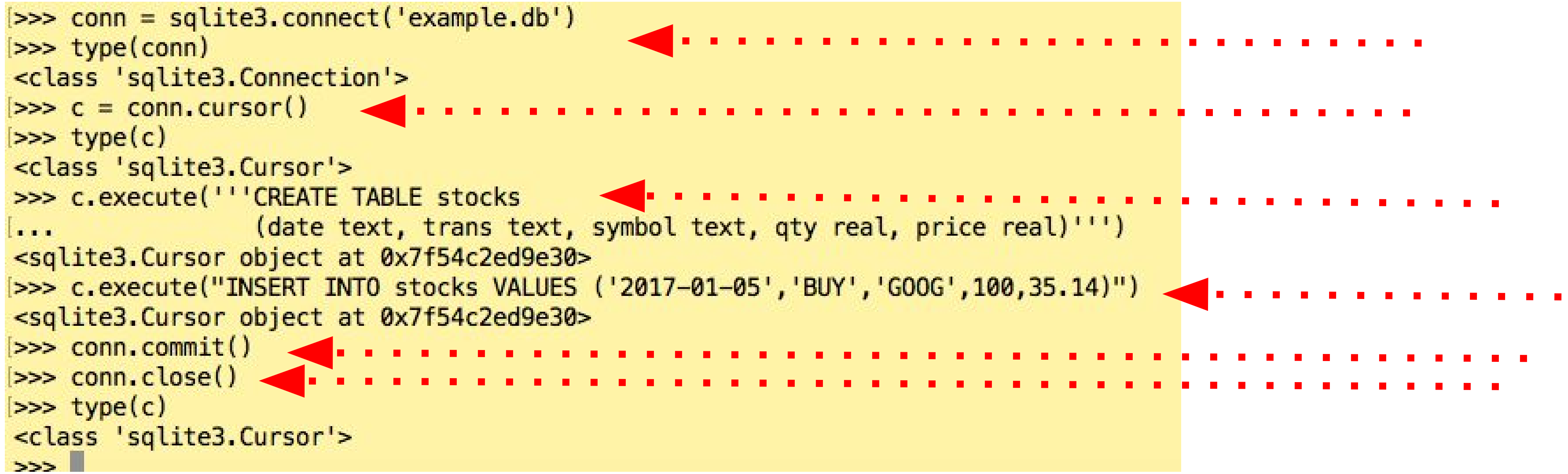


```
[sqlite> .schema
[sqlite> CREATE TABLE presidents (
[   ...> id int primary key NOT NULL,
[   ...> name char(100) NOT NULL,
[   ...> age  int NOT NULL);
[sqlite> .schema
CREATE TABLE presidents (
id int primary key NOT NULL,
name char(100) NOT NULL,
age  int NOT NULL);
[sqlite> select * from presidents;
[sqlite> INSERT INTO presidents (id, name, age) VALUES(1, 'Donalt T', 74);
[sqlite> select * from presidents;
1|Donalt T|74
[sqlite> select name from presidents;
Donalt T
[sqlite> INSERT INTO presidents (id, name, age) VALUES(2, 'Barack O', 54);
[sqlite> select * from presidents;
1|Donalt T|74
2|Barack O|54
[sqlite> select * from presidents where id=2;
2|Barack O|54
sqlite> 
```



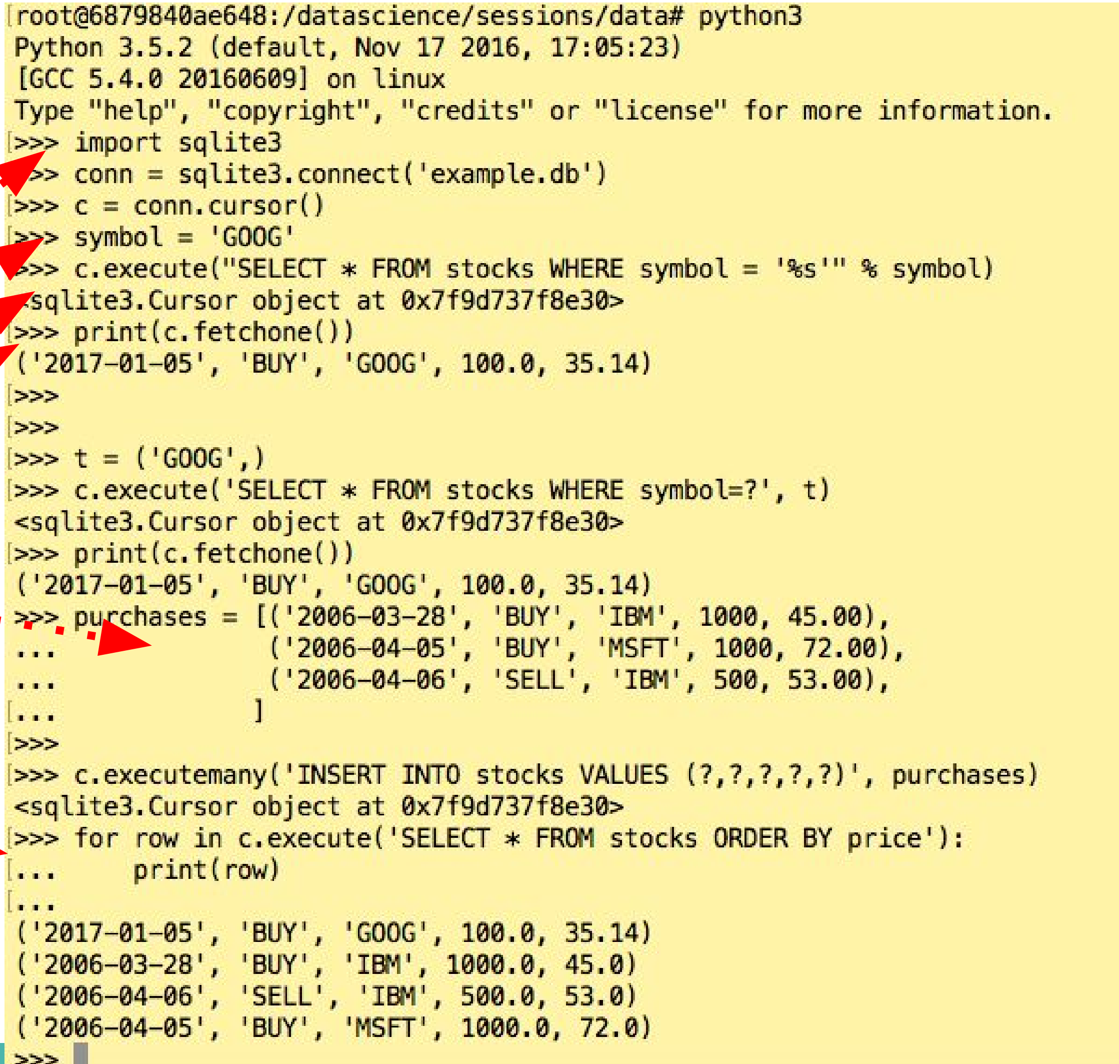
```
[sqlite> select * from presidents ORDER BY age;
2|Barack O|54
1|Donalt T|74
sqlite> 
```

Database - Python Module



```
[>>> conn = sqlite3.connect('example.db')
[>>> type(conn)
<class 'sqlite3.Connection'>
[>>> c = conn.cursor()
[>>> type(c)
<class 'sqlite3.Cursor'>
>>> c.execute('''CREATE TABLE stocks
[...                 (date text, trans text, symbol text, qty real, price real)''')
<sqlite3.Cursor object at 0x7f54c2ed9e30>
[>>> c.execute("INSERT INTO stocks VALUES ('2017-01-05','BUY','GOOG',100,35.14)")
<sqlite3.Cursor object at 0x7f54c2ed9e30>
[>>> conn.commit()
[>>> conn.close()
[>>> type(c)
<class 'sqlite3.Cursor'>
>>> 
```

Database Python Module

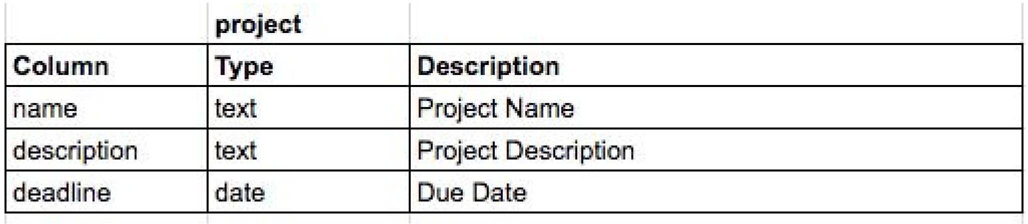


```
root@6879840ae648:/datascience/sessions/data# python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sqlite3
>>> conn = sqlite3.connect('example.db')
>>> c = conn.cursor()
>>> symbol = 'GOOG'
>>> c.execute("SELECT * FROM stocks WHERE symbol = '%s'" % symbol)
<sqlite3.Cursor object at 0x7f9d737f8e30>
>>> print(c.fetchone())
('2017-01-05', 'BUY', 'GOOG', 100.0, 35.14)
>>>
>>>
>>> t = ('GOOG',)
>>> c.execute('SELECT * FROM stocks WHERE symbol=?', t)
<sqlite3.Cursor object at 0x7f9d737f8e30>
>>> print(c.fetchone())
('2017-01-05', 'BUY', 'GOOG', 100.0, 35.14)
>>> purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
...              ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
...              ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
...             ]
>>>
>>> c.executemany('INSERT INTO stocks VALUES (?,?,?,?,?)', purchases)
<sqlite3.Cursor object at 0x7f9d737f8e30>
>>> for row in c.execute('SELECT * FROM stocks ORDER BY price'):
...     print(row)
...
('2017-01-05', 'BUY', 'GOOG', 100.0, 35.14)
('2006-03-28', 'BUY', 'IBM', 1000.0, 45.0)
('2006-04-06', 'SELL', 'IBM', 500.0, 53.0)
('2006-04-05', 'BUY', 'MSFT', 1000.0, 72.0)
>>>
```

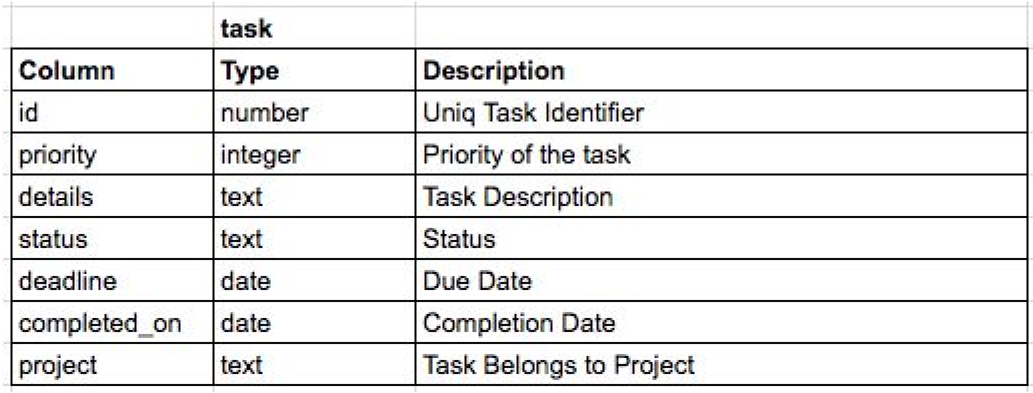
create_db.py

```
#!/usr/bin/python3
import os
import sqlite3

db_filename = 'todo.db'
new_db = not os.path.exists(db_filename)

conn = sqlite3.connect(db_filename)

if new_db:
    print('Please create Schema')
else:
    print('Database already created - mostly schema exists')
conn.close()
```

Database - Project TODO

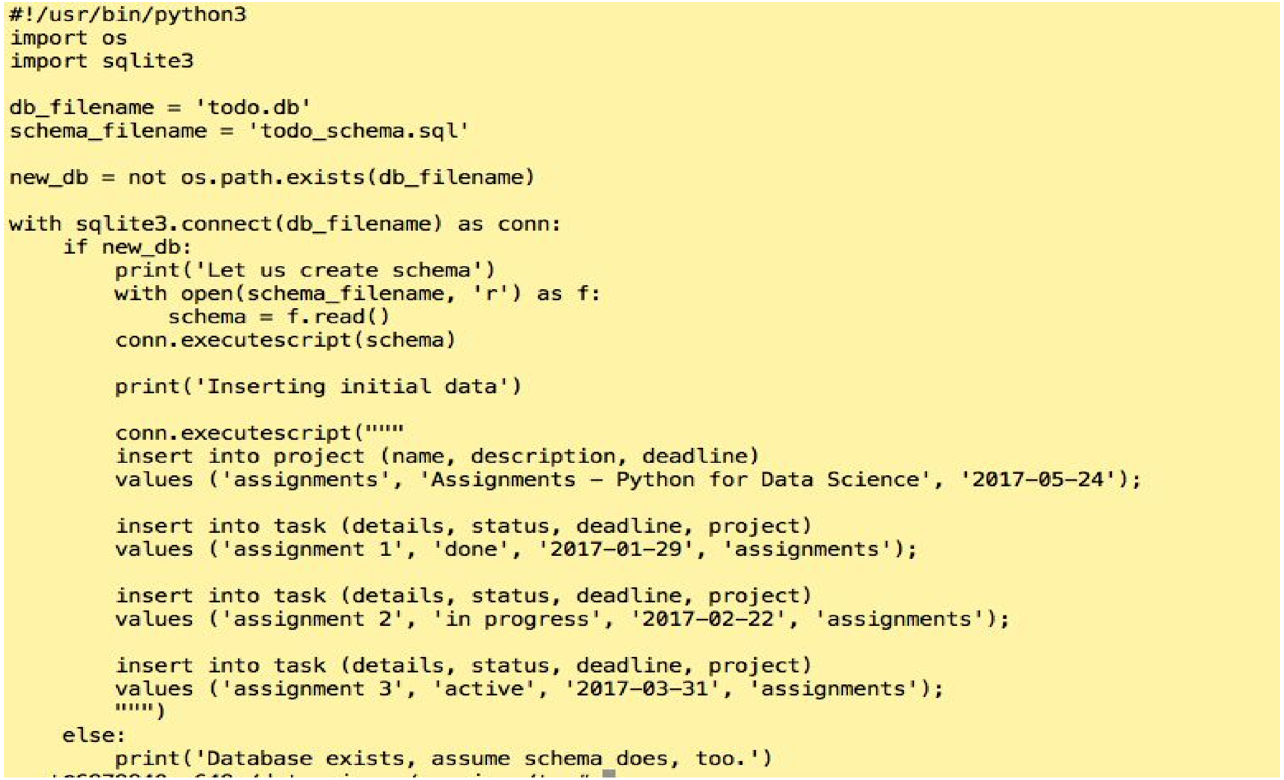


	project	
Column	**Type**	**Description**
name	text	Project Name
description	text	Project Description
deadline	date	Due Date



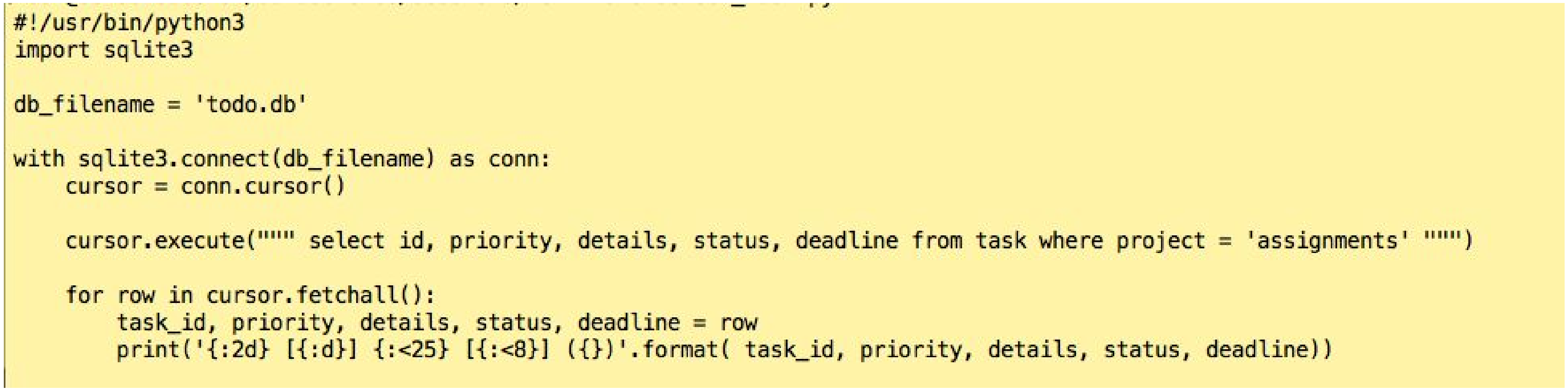
	task	
Column	**Type**	**Description**
id	number	Uniq Task Identifier
priority	integer	Priority of the task
details	text	Task Description
status	text	Status
deadline	date	Due Date
completed_on	date	Completion Date
project	text	Task Belongs to Project

Database - create schema & add data

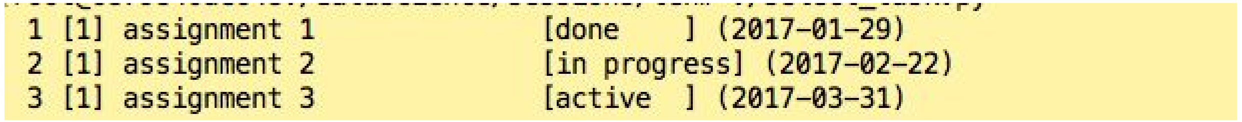


```
#!/usr/bin/python3
import os
import sqlite3

db_filename = 'todo.db'
schema_filename = 'todo_schema.sql'

new_db = not os.path.exists(db_filename)

with sqlite3.connect(db_filename) as conn:
    if new_db:
        print('Let us create schema')
        with open(schema_filename, 'r') as f:
            schema = f.read()
        conn.executescript(schema)

        print('Inserting initial data')

        conn.executescript("""
        insert into project (name, description, deadline)
        values ('assignments', 'Assignments - Python for Data Science', '2017-05-24');

        insert into task (details, status, deadline, project)
        values ('assignment 1', 'done', '2017-01-29', 'assignments');

        insert into task (details, status, deadline, project)
        values ('assignment 2', 'in progress', '2017-02-22', 'assignments');

        insert into task (details, status, deadline, project)
        values ('assignment 3', 'active', '2017-03-31', 'assignments');
        """)
    else:
        print('Database exists, assume schema does, too.')
```

Database - Retrieve Data

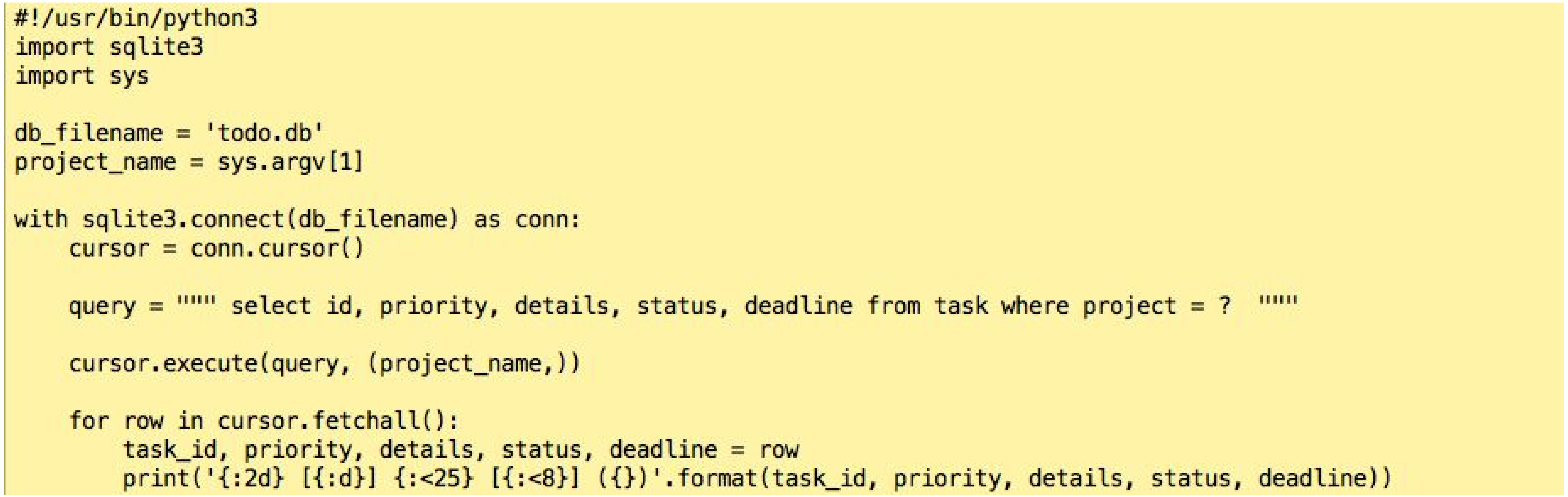


```
#!/usr/bin/python3
import sqlite3

db_filename = 'todo.db'

with sqlite3.connect(db_filename) as conn:
    cursor = conn.cursor()

    cursor.execute(""" select id, priority, details, status, deadline from task where project = 'assignments' """)

    for row in cursor.fetchall():
        task_id, priority, details, status, deadline = row
        print('{:2d} [{:d}] {:<25} [{:<8}] ({})'.format( task_id, priority, details, status, deadline))
```

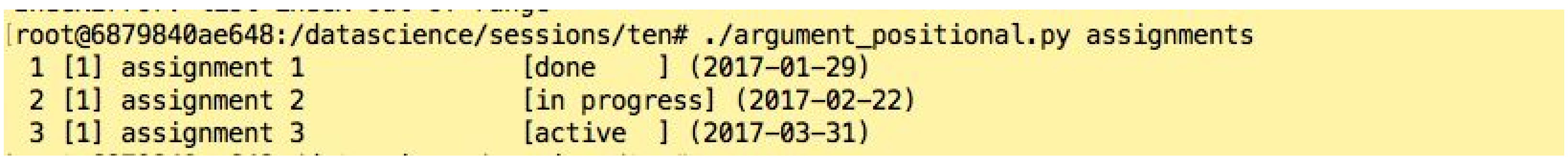


```
 1 [1] assignment 1              [done    ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active  ] (2017-03-31)
```

Database - positional argument

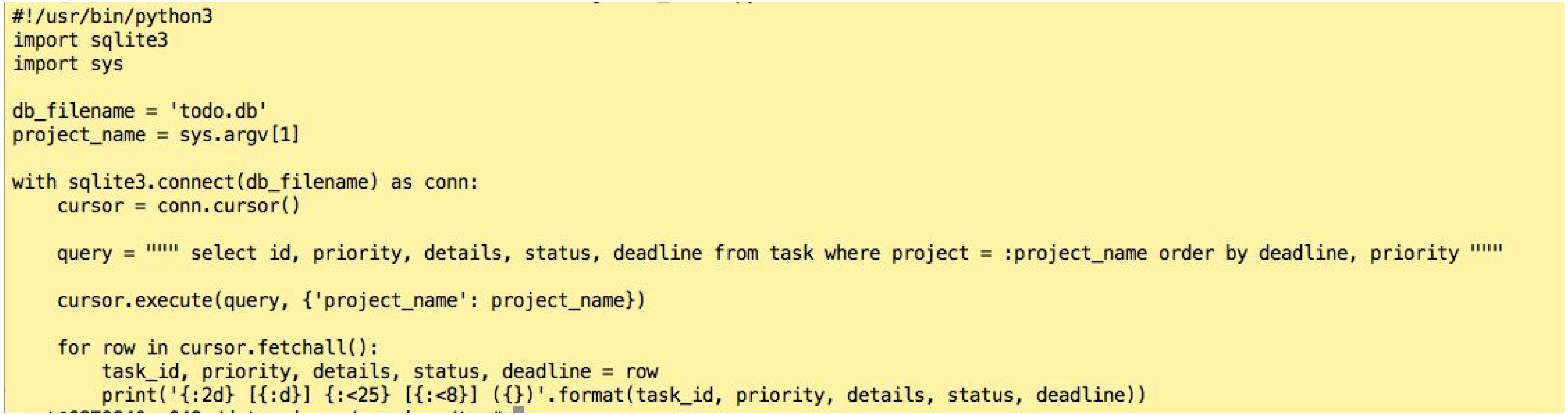


```
#!/usr/bin/python3
import sqlite3
import sys

db_filename = 'todo.db'
project_name = sys.argv[1]

with sqlite3.connect(db_filename) as conn:
    cursor = conn.cursor()

    query = """ select id, priority, details, status, deadline from task where project = ?  """

    cursor.execute(query, (project_name,))

    for row in cursor.fetchall():
        task_id, priority, details, status, deadline = row
        print('{:2d} [{:d}] {:<25} [{:<8}] ({})'.format(task_id, priority, details, status, deadline))
```

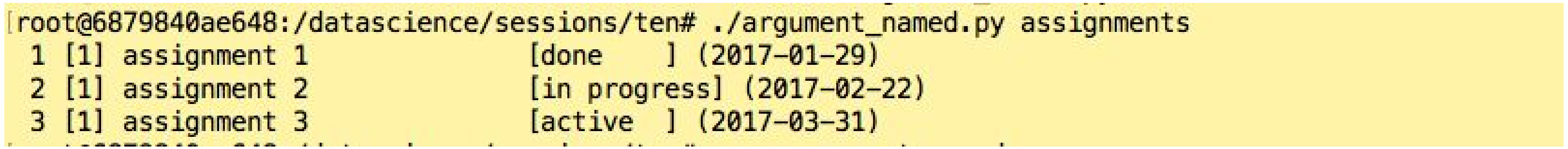


```
root@6879840ae648:/datascience/sessions/ten# ./argument_positional.py assignments
 1 [1] assignment 1              [done     ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active   ] (2017-03-31)
```

argument_named.py

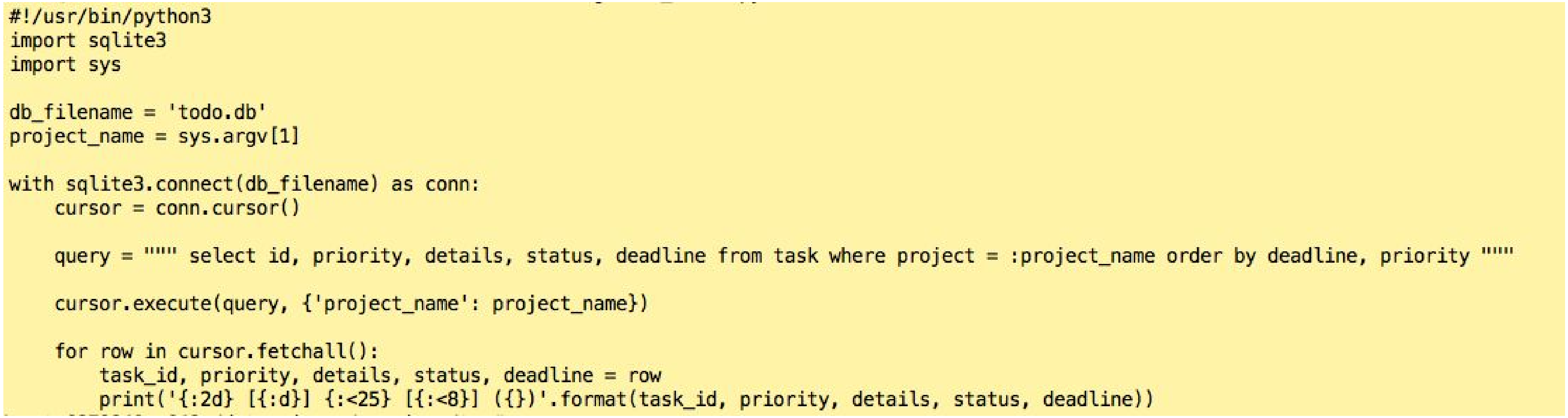


```
#!/usr/bin/python3
import sqlite3
import sys

db_filename = 'todo.db'
project_name = sys.argv[1]

with sqlite3.connect(db_filename) as conn:
    cursor = conn.cursor()

    query = """ select id, priority, details, status, deadline from task where project = :project_name order by deadline, priority """

    cursor.execute(query, {'project_name': project_name})

    for row in cursor.fetchall():
        task_id, priority, details, status, deadline = row
        print('{:2d} [{:d}] {:<25} [{:<8}] ({})'.format(task_id, priority, details, status, deadline))
```

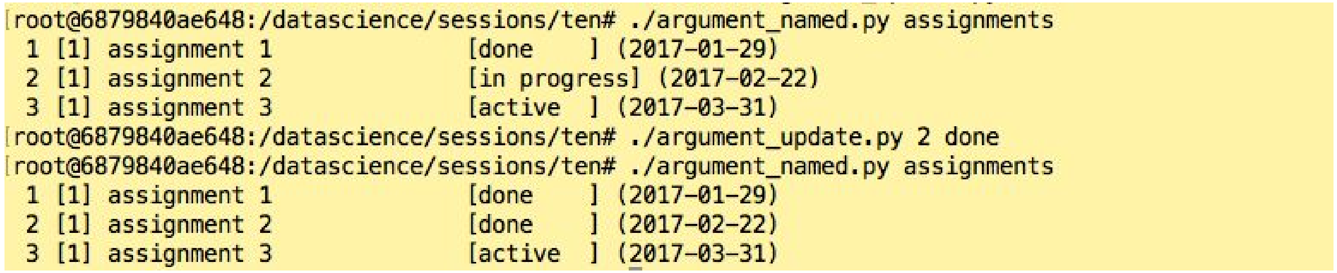


```
root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1              [done     ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active   ] (2017-03-31)
```

argument_update.py

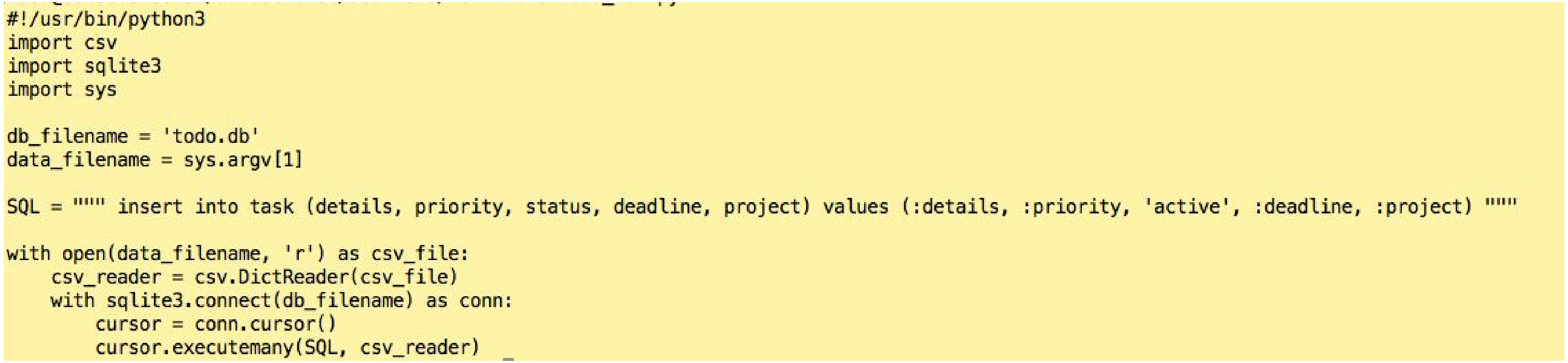


```
#!/usr/bin/python3
import sqlite3
import sys

db_filename = 'todo.db'
project_name = sys.argv[1]

with sqlite3.connect(db_filename) as conn:
    cursor = conn.cursor()

    query = """ select id, priority, details, status, deadline from task where project = :project_name order by deadline, priority """

    cursor.execute(query, {'project_name': project_name})

    for row in cursor.fetchall():
        task_id, priority, details, status, deadline = row
        print('{:2d} [{:d}] {:<25} [{:<8}] ({})'.format(task_id, priority, details, status, deadline))
```

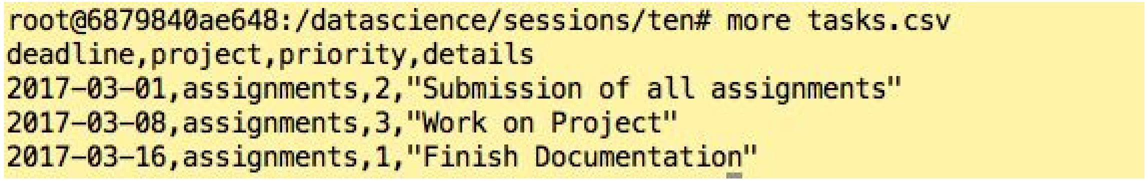


```
[root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1              [done     ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active   ] (2017-03-31)
[root@6879840ae648:/datascience/sessions/ten# ./argument_update.py 2 done
[root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1              [done     ] (2017-01-29)
 2 [1] assignment 2              [done     ] (2017-02-22)
 3 [1] assignment 3              [active   ] (2017-03-31)
```

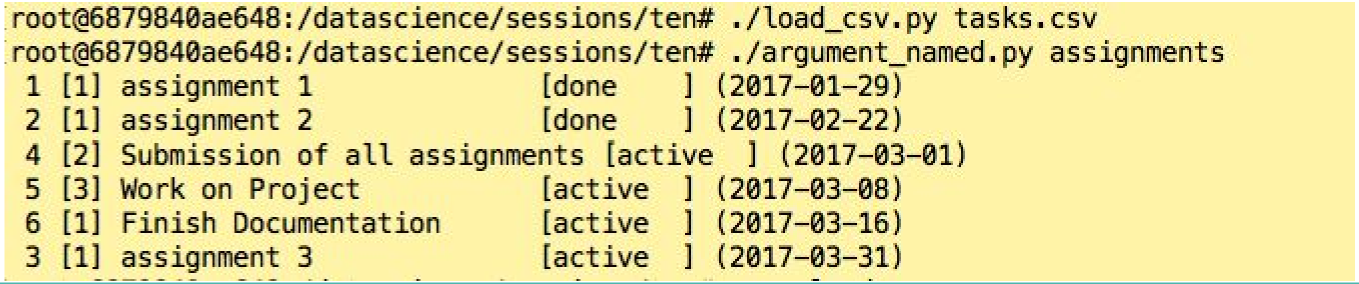
load_csv.py



```python
#!/usr/bin/python3
import csv
import sqlite3
import sys

db_filename = 'todo.db'
data_filename = sys.argv[1]

SQL = """ insert into task (details, priority, status, deadline, project) values (:details, :priority, 'active', :deadline, :project) """

with open(data_filename, 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    with sqlite3.connect(db_filename) as conn:
        cursor = conn.cursor()
        cursor.executemany(SQL, csv_reader)
```



```
root@6879840ae648:/datascience/sessions/ten# more tasks.csv
deadline,project,priority,details
2017-03-01,assignments,2,"Submission of all assignments"
2017-03-08,assignments,3,"Work on Project"
2017-03-16,assignments,1,"Finish Documentation"
```



```
root@6879840ae648:/datascience/sessions/ten# ./load_csv.py tasks.csv
root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1                 [done     ] (2017-01-29)
 2 [1] assignment 2                 [done     ] (2017-02-22)
 4 [2] Submission of all assignments [active   ] (2017-03-01)
 5 [3] Work on Project              [active   ] (2017-03-08)
 6 [1] Finish Documentation         [active   ] (2017-03-16)
 3 [1] assignment 3                 [active   ] (2017-03-31)
```

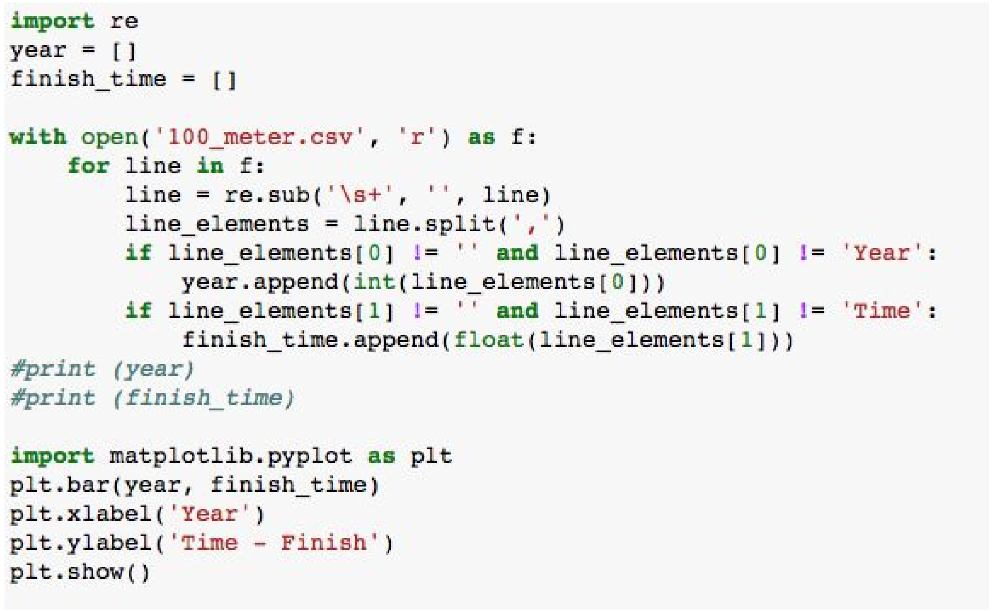
Data Visualization - Plotting

matplotlib.pyplot is a collection of command style functions that helps draw graphs

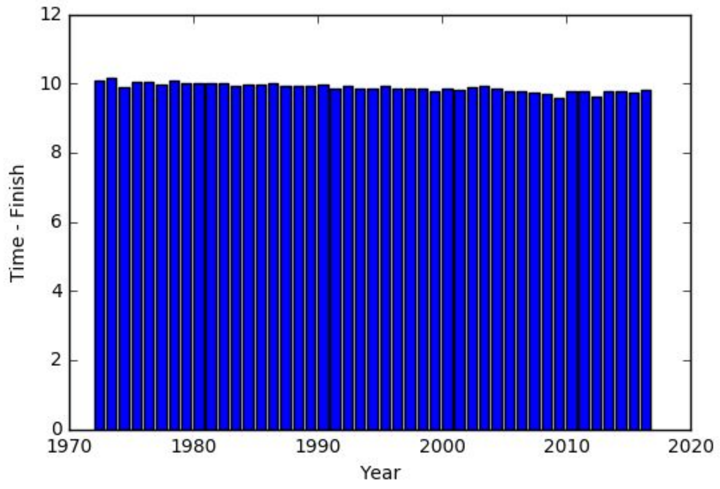
Types of Graph

- Bar Graphs
- Box and Whiskers (Boxplots)
- Frequency Distribution
- Histogram
- Line Graphs
- Pie Graphs
- Scatter Graphs
- Stemplots

Simple Bar Graph

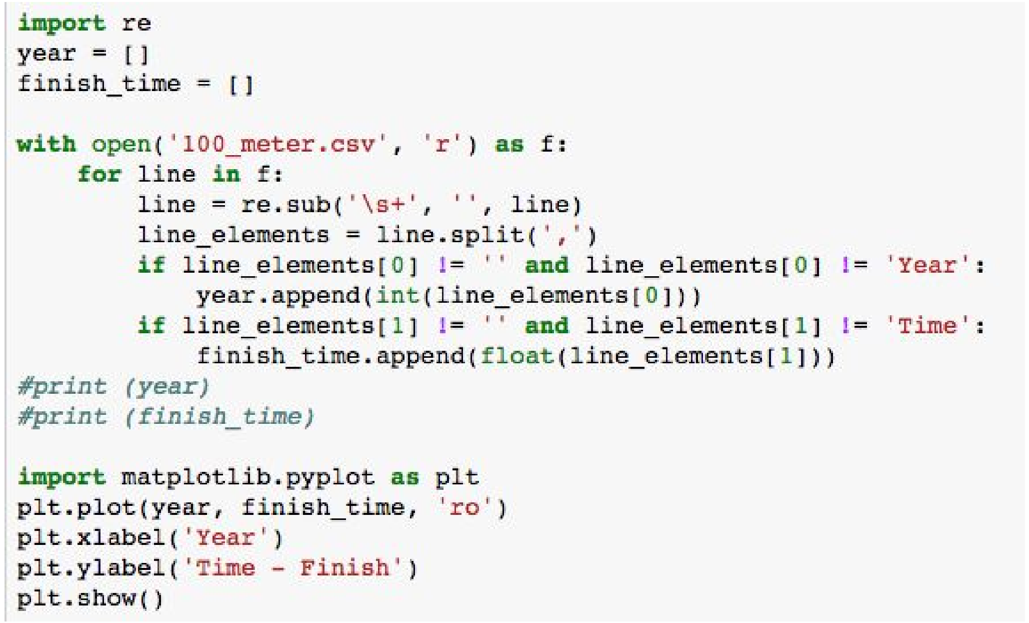


```python
import re
year = []
finish_time = []

with open('100_meter.csv', 'r') as f:
    for line in f:
        line = re.sub('\s+', '', line)
        line_elements = line.split(',')
        if line_elements[0] != '' and line_elements[0] != 'Year':
            year.append(int(line_elements[0]))
        if line_elements[1] != '' and line_elements[1] != 'Time':
            finish_time.append(float(line_elements[1]))
#print (year)
#print (finish_time)

import matplotlib.pyplot as plt
plt.bar(year, finish_time)
plt.xlabel('Year')
plt.ylabel('Time - Finish')
plt.show()
```

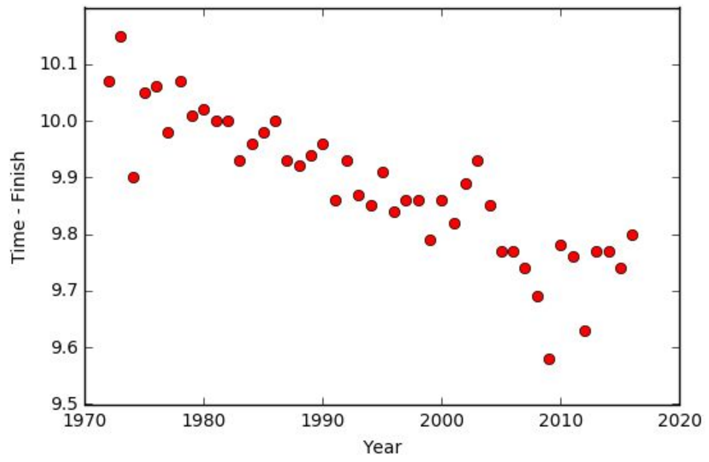


running_simple_bar.ipynb

Simple Plot



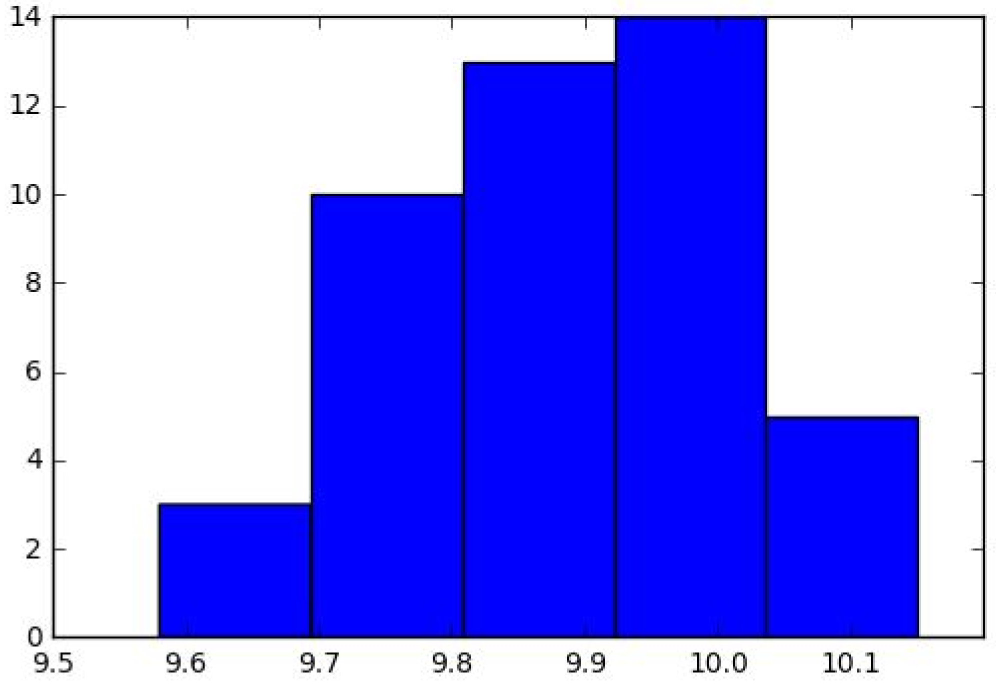
```python
import re
year = []
finish_time = []

with open('100_meter.csv', 'r') as f:
    for line in f:
        line = re.sub('\s+', '', line)
        line_elements = line.split(',')
        if line_elements[0] != '' and line_elements[0] != 'Year':
            year.append(int(line_elements[0]))
        if line_elements[1] != '' and line_elements[1] != 'Time':
            finish_time.append(float(line_elements[1]))
#print (year)
#print (finish_time)

import matplotlib.pyplot as plt
plt.plot(year, finish_time, 'ro')
plt.xlabel('Year')
plt.ylabel('Time - Finish')
plt.show()
```



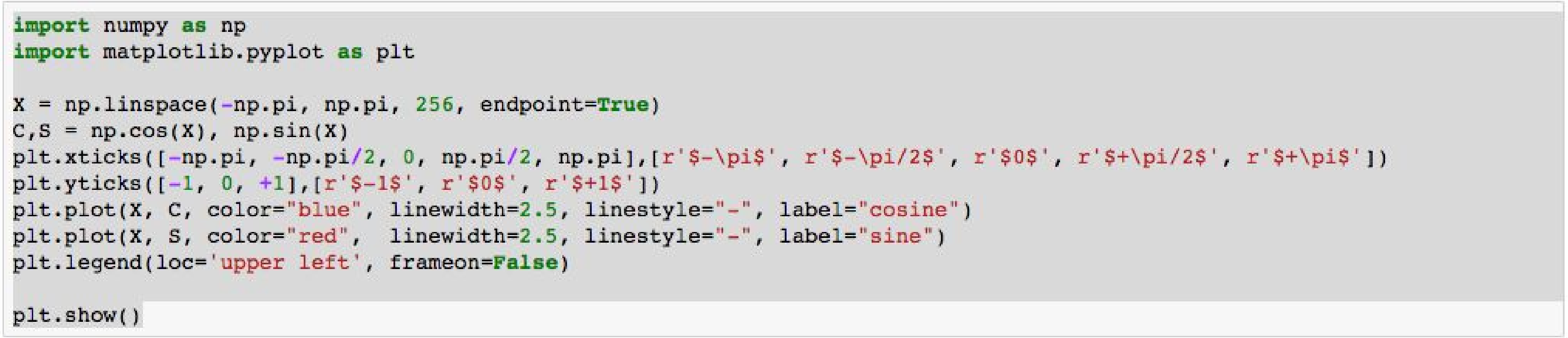
running_simple_plot.ipynb

Histogram - Finish Time

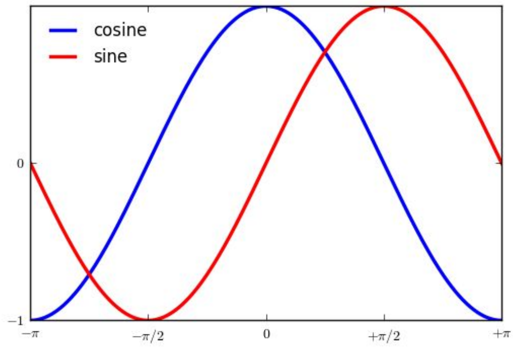
```
plt.hist(finish_time, bins=5)
plt.show()
```



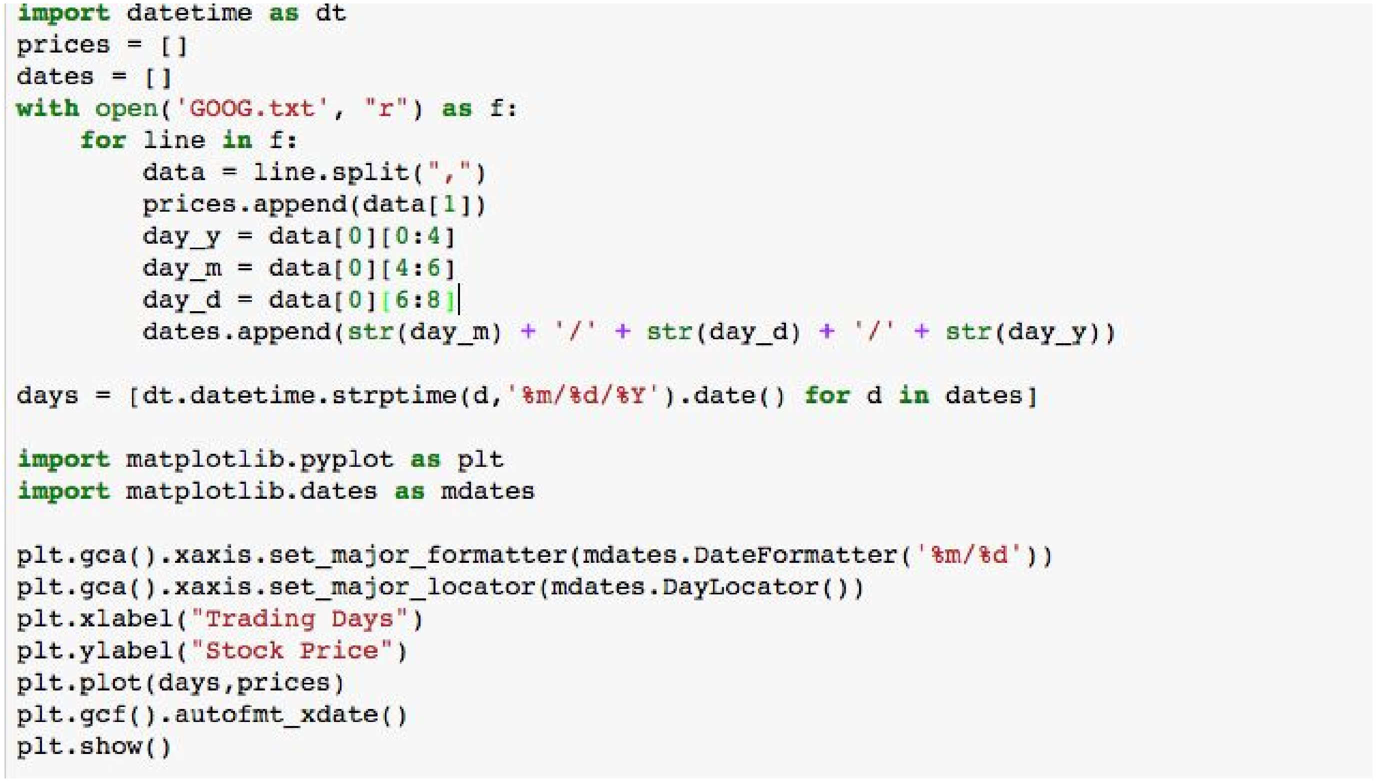
Customizing Graphs



```python
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C,S = np.cos(X), np.sin(X)
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],[r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])
plt.yticks([-1, 0, +1],[r'$-1$', r'$0$', r'$+1$'])
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine")
plt.legend(loc='upper left', frameon=False)

plt.show()
```

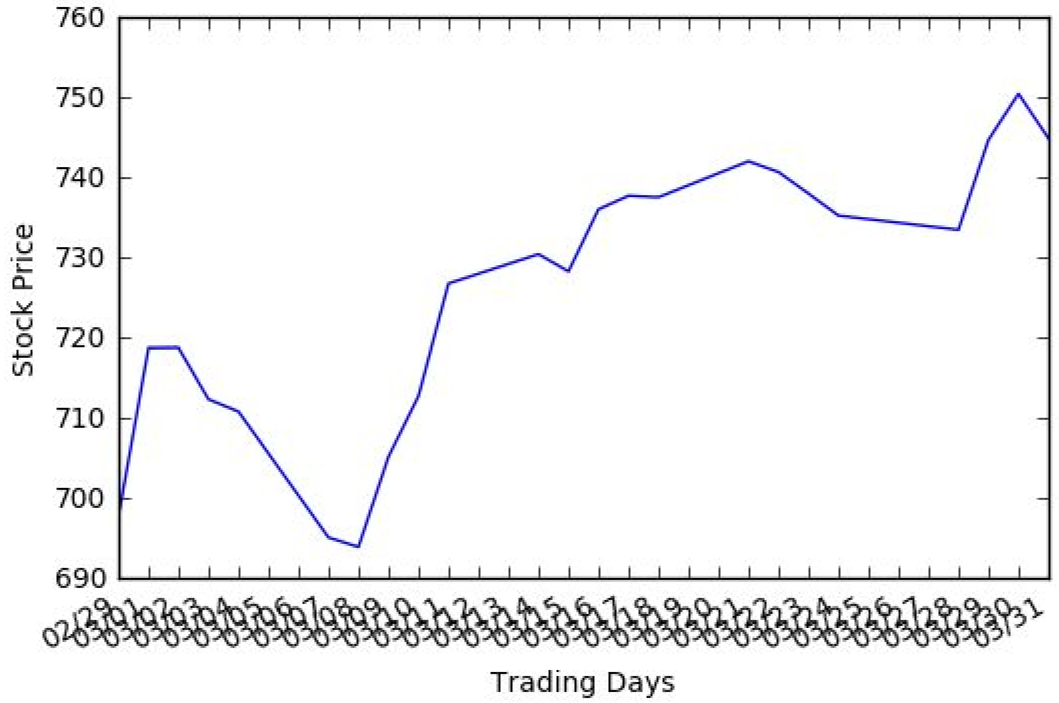


Google Stock Price



```
import datetime as dt
prices = []
dates = []
with open('GOOG.txt', "r") as f:
    for line in f:
        data = line.split(",")
        prices.append(data[1])
        day_y = data[0][0:4]
        day_m = data[0][4:6]
        day_d = data[0][6:8]
        dates.append(str(day_m) + '/' + str(day_d) + '/' + str(day_y))

days = [dt.datetime.strptime(d,'%m/%d/%Y').date() for d in dates]

import matplotlib.pyplot as plt
import matplotlib.dates as mdates

plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator())
plt.xlabel("Trading Days")
plt.ylabel("Stock Price")
plt.plot(days,prices)
plt.gcf().autofmt_xdate()
plt.show()
```

Google Stock Price (cont..)



Thank you