Day - Four

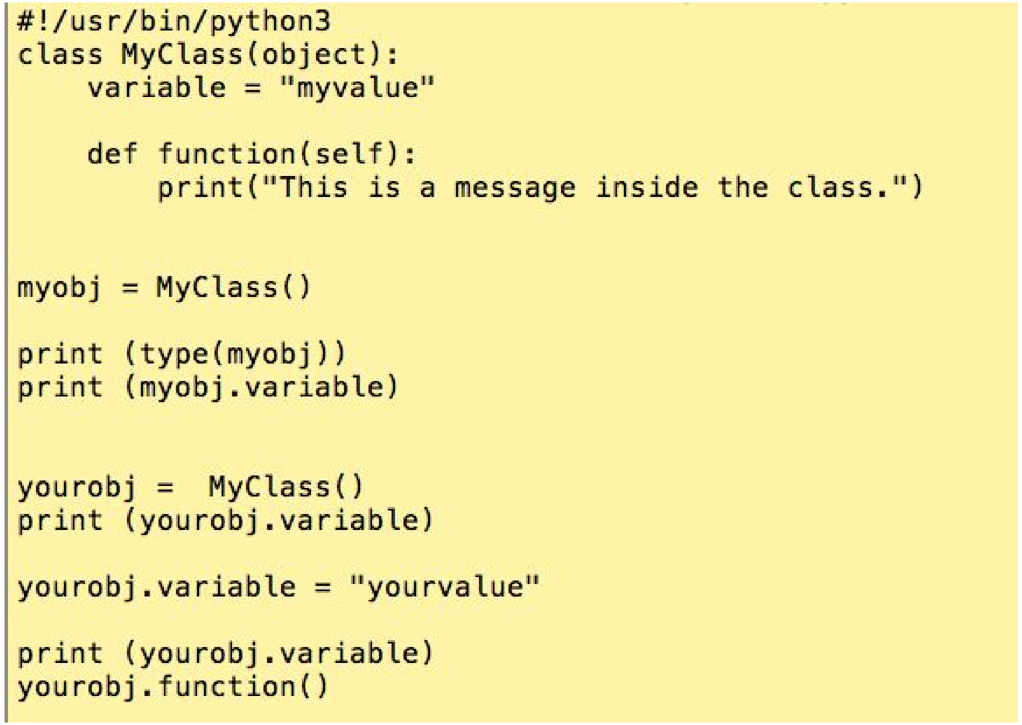
Aegis

SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

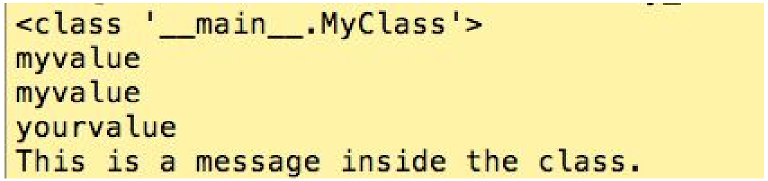
Classes and OOPs, Logging, Regular Expressions

- Defining Class and Objects
- Properties and Example related to use of classes, functions etc.
- Logging with python - Error Logging/Info Logging etc.
- Accessing data available on internet - urllib/requests library
- Regular Expressions

Class



```
#!/usr/bin/python3
class MyClass(object):
    variable = "myvalue"

    def function(self):
        print("This is a message inside the class.")


myobj = MyClass()

print (type(myobj))
print (myobj.variable)


yourobj =  MyClass()
print (yourobj.variable)

yourobj.variable = "yourvalue"

print (yourobj.variable)
yourobj.function()
```



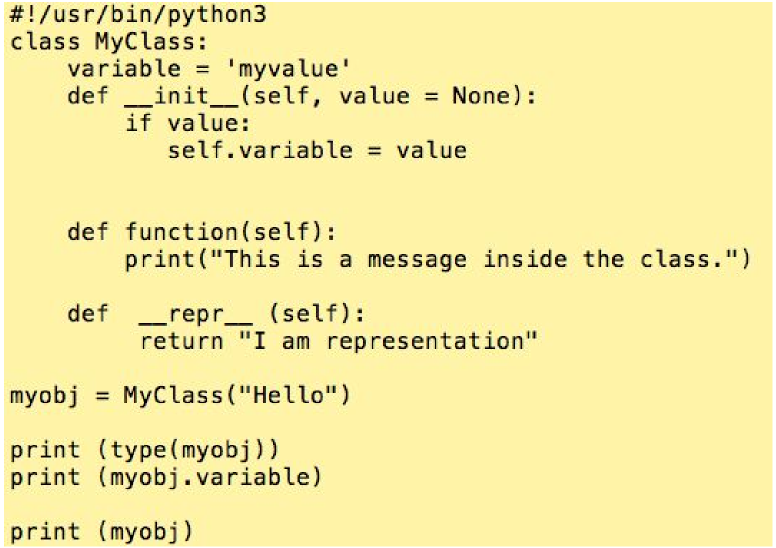
```
<class '__main__.MyClass'>
myvalue
myvalue
yourvalue
This is a message inside the class.
```

my_class.py

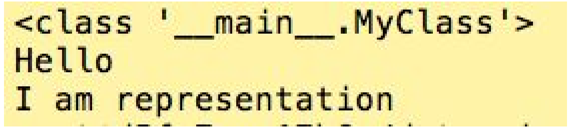
Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Class (Cont …)

- Classes are essentially a template to create your objects.
- init is the constructor for a class.
  - The self parameter refers to the instance of the object



```
#!/usr/bin/python3
class MyClass:
    variable = 'myvalue'
    def __init__(self, value = None):
        if value:
           self.variable = value


    def function(self):
        print("This is a message inside the class.")

    def  __repr__ (self):
         return "I am representation"

myobj = MyClass("Hello")

print (type(myobj))
print (myobj.variable)

print (myobj)
```



```
<class '__main__.MyClass'>
Hello
I am representation
```

my_advance_class.py

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Class (Cont …)

```
class Person(object):
    def __init__(self, name, age):
          self.name = name
          self.age  = age

    def __repr__(self):
        return "Name: %s , Age: %d \n"  % (self.name, self.age)
```

Person.py

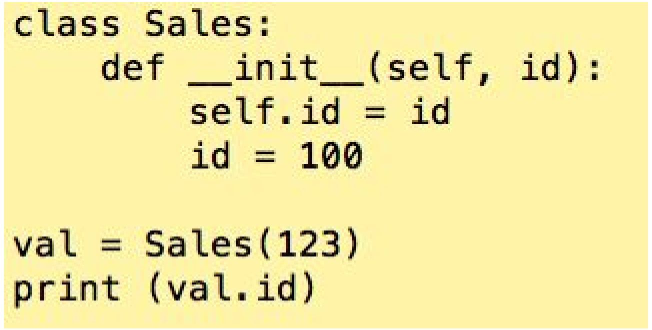
```
#!/usr/bin/python3
from  Person import  Person


def byAge(Person):
   return Person.age

p1  = Person("Doland Trump", 70)
p2  = Person("Barack Obama", 55)
p3  = Person("G Bush", 62)
p4  = Person("Bill Clinton", 54)
p5  = Person("Ronald Reagan", 77)



presidents  = [p1, p2, p3, p4, p5]

print (presidents)

sorted_presidents = sorted(presidents,  key=byAge)

print  (sorted_presidents)
```

sort_person.py

```
[Name: Doland Trump , Age: 70
, Name: Barack Obama , Age: 55
, Name: G Bush , Age: 62
, Name: Bill Clinton , Age: 54
, Name: Ronald Reagan , Age: 77
]
[Name: Bill Clinton , Age: 54
, Name: Barack Obama , Age: 55
, Name: G Bush , Age: 62
, Name: Doland Trump , Age: 70
, Name: Ronald Reagan , Age: 77
]
```

Quiz

What will be the output of following code?



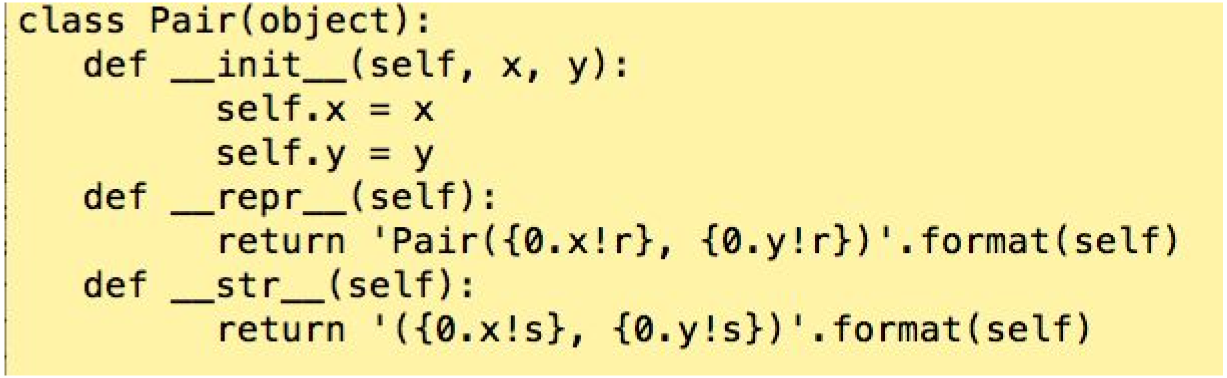
```
class Sales:
    def __init__(self, id):
        self.id = id
        id = 100

val = Sales(123)
print (val.id)
```

Task - 1

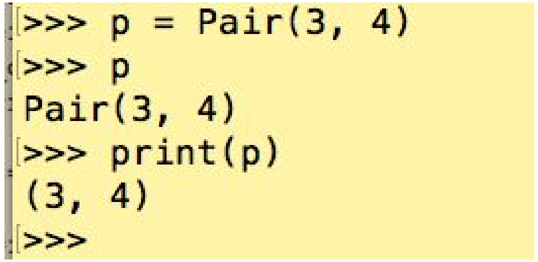
- Create Class - "City"
- Attributes
  a. Population
  b. Country
  c. Name
- Develop Script - to sort Cities (City instances) by Population
- Add GDP to City Class
  a. sort by GDP

Class (Cont...)

__str__ and __repr__ important methods



```
class Pair(object):
   def __init__(self, x, y):
          self.x = x
          self.y = y
   def __repr__(self):
          return 'Pair({0.x!r}, {0.y!r})'.format(self)
   def __str__(self):
          return '({0.x!s}, {0.y!s})'.format(self)
```

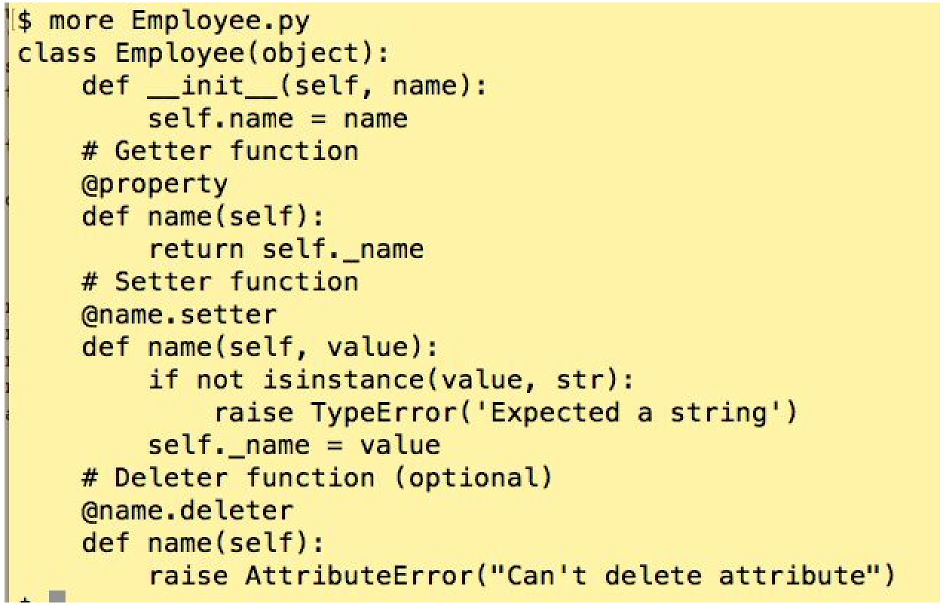


```
>>> p = Pair(3, 4)
>>> p
Pair(3, 4)
>>> print(p)
(3, 4)
>>>
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Class (Cont ...)

Setter and Getter Method discussion - how @property is used?



```
$ more Employee.py
class Employee(object):
    def __init__(self, name):
        self.name = name
    # Getter function
    @property
    def name(self):
        return self._name
    # Setter function
    @name.setter
    def name(self, value):
        if not isinstance(value, str):
            raise TypeError('Expected a string')
        self._name = value
    # Deleter function (optional)
    @name.deleter
    def name(self):
        raise AttributeError("Can't delete attribute")
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Task 2

Develop class - “Student” - with attributes as:

1. Name (it should be string)
2. Branch (e.g. Year_2017_18, Year_2016_17)
3. Age (it should be Integer)

Define setter & getter methods for each attribute.

Example - Following should work.

S = Student(‘Hari’, ‘Year_2017_18’, 22)

S.Age = 25

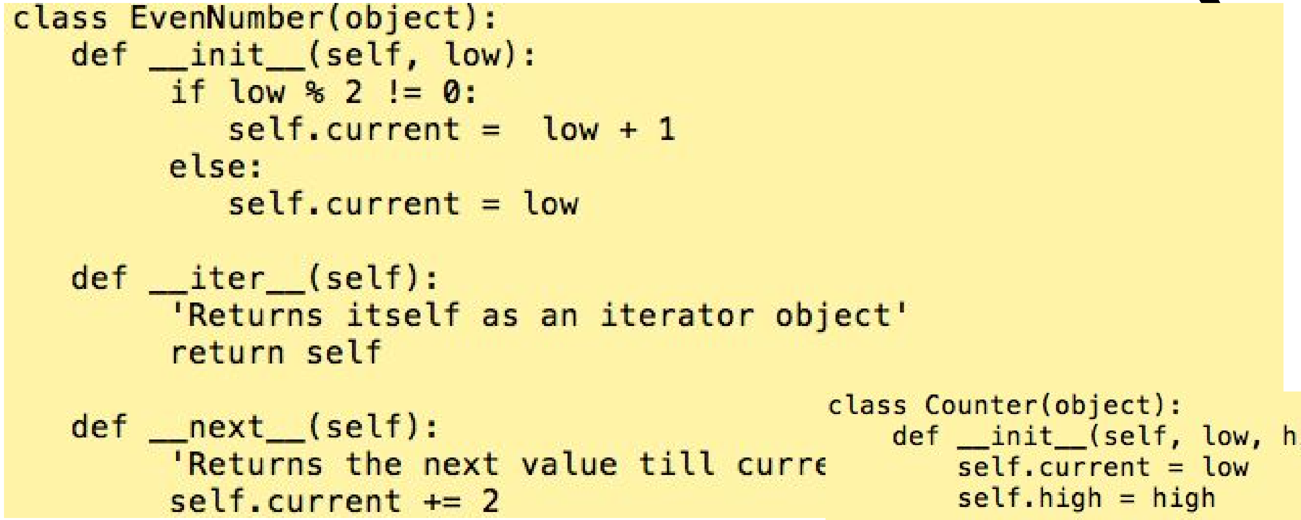
Print (s.age) # This should print 25

Revisit Iterators

Python iterator objects are required to support two methods while following the iterator protocol.

- *iter* returns the iterator object itself. This is used in **for** and **in** statements.
- *next* method returns the next value from the iterator. If there is no more items to return then it should raise StopIteration exception.

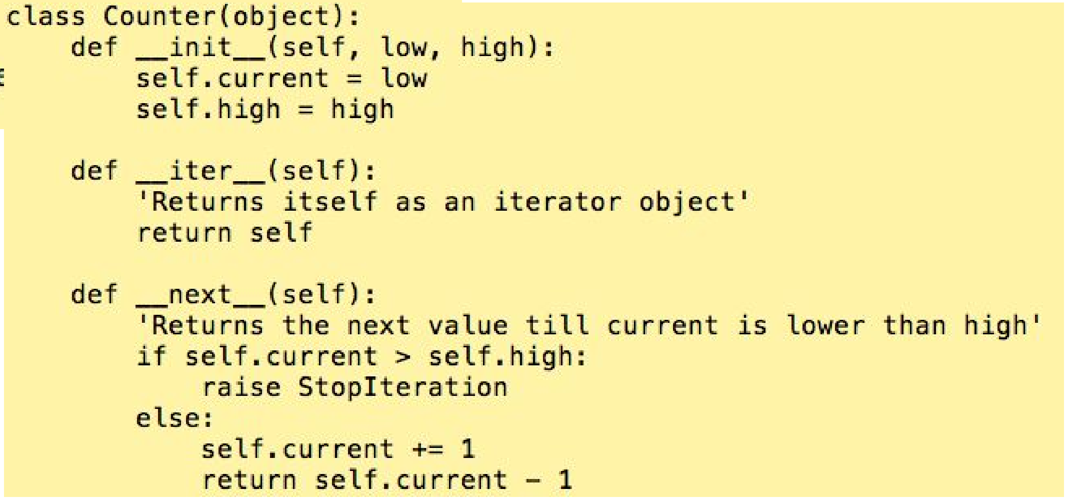
Iterator Revisits (Cont …)



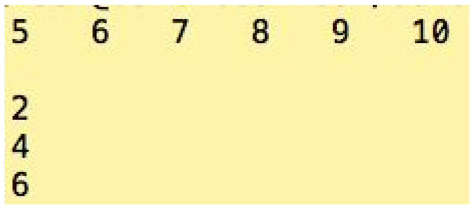
```
class EvenNumber(object):
   def __init__(self, low):
        if low % 2 != 0:
           self.current =  low + 1
        else:
           self.current = low

   def __iter__(self):
        'Returns itself as an iterator object'
        return self

   def __next__(self):
        'Returns the next value till curre
        self.current += 2
```

EvenNumber.py

Counter.py



```
class Counter(object):
    def __init__(self, low, high):
        self.current = low
        self.high = high

    def __iter__(self):
        'Returns itself as an iterator object'
        return self

    def __next__(self):
        'Returns the next value till current is lower than high'
        if self.current > self.high:
            raise StopIteration
        else:
            self.current += 1
            return self.current - 1
```



```
5   6   7   8   9   10

2
4
6
```

test_counter_evennumber.py

SCHOOL OF TELECOMMUNICATION

Regular Expressions

Let us define some rules to form some strings:

- Write a letter "a" at least once
- Append to this the letter "b" exactly five times
- Append to this the letter "c" any even number of times
- Optionally, write the letter "d" at the end

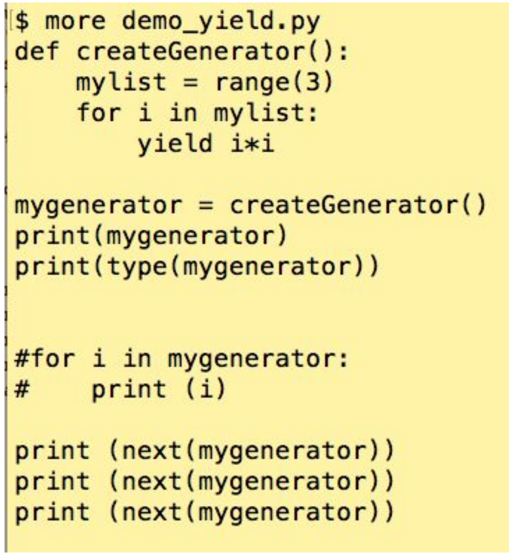
Examples of such strings are:

```
aaaabbbbbccccd
```

```
aabbbbbcc
```

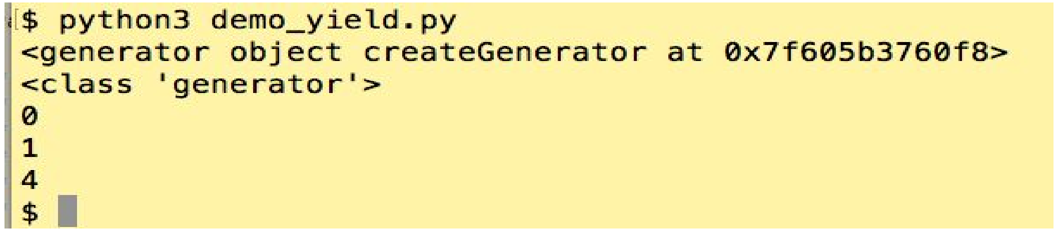
```
...
```

Yield - Generator



```
$ more demo_yield.py
def createGenerator():
    mylist = range(3)
    for i in mylist:
        yield i*i

mygenerator = createGenerator()
print(mygenerator)
print(type(mygenerator))


#for i in mygenerator:
#    print (i)

print (next(mygenerator))
print (next(mygenerator))
print (next(mygenerator))
```

demo_yeild.py



```
$ python3 demo_yield.py
<generator object createGenerator at 0x7f605b3760f8>
<class 'generator'>
0
1
4
$ 
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Regular Expression (Cont …)

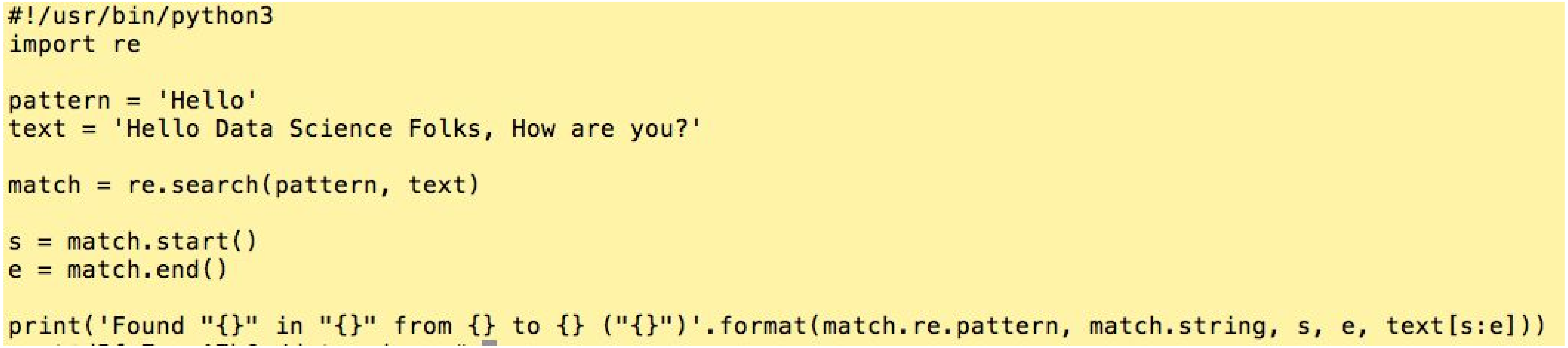
There are infinitely many such strings which satisfy above rules.

Regular Expressions are merely a shorthand way of expressing these sets of rules

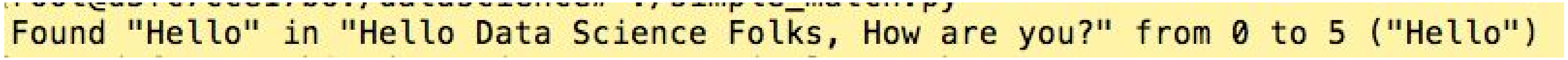
- Regex are text matching patterns described with a formal syntax
- The patterns which are executed on text as input to produce either matching subset or modified version of original text
- Regular Expression is kind of programming language itself
- "re" module provides this functionality in Python Programming

Your friendship with re will always add advantage to your skills if you ever need to deal with Text Processing in your project.

Regular Expressions (Cont ...)



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

pattern = 'Hello'
text = 'Hello Data Science Folks, How are you?'

match = re.search(pattern, text)

s = match.start()
e = match.end()

print('Found "{}" in "{}" from {} to {} ("{}")'.format(match.re.pattern, match.string, s, e, text[s:e]))
```



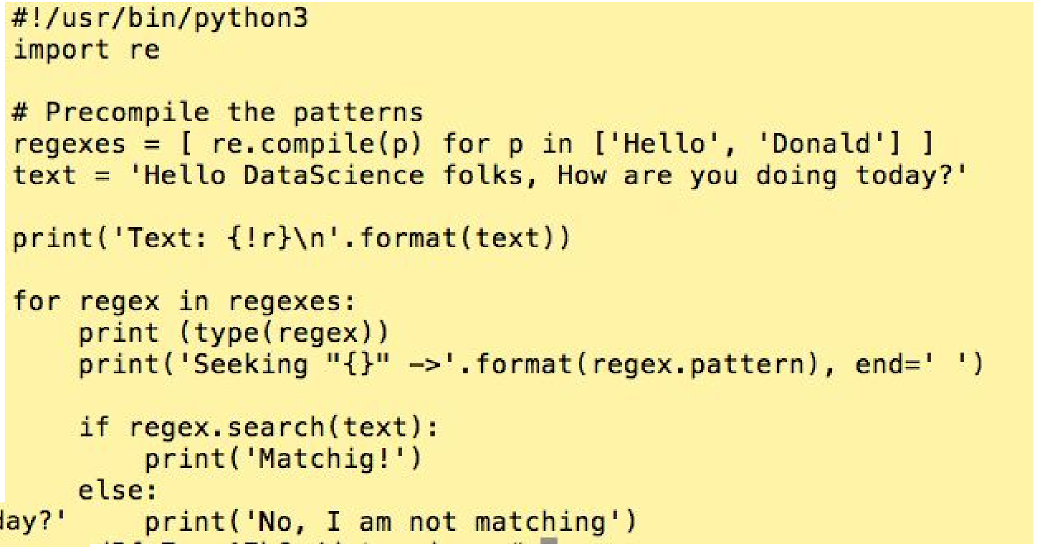
```
Found "Hello" in "Hello Data Science Folks, How are you?" from 0 to 5 ("Hello")
```

simple_match.py

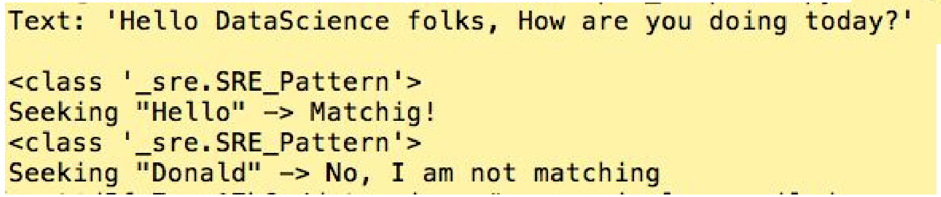
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Regular Expression (Cont …)

Python supports compilation of pattern - it's more efficient to compile the pattern and use it. The compile() function converts an expression string into a RegexObject.



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

# Precompile the patterns
regexes = [ re.compile(p) for p in ['Hello', 'Donald'] ]
text = 'Hello DataScience folks, How are you doing today?'

print('Text: {!r}\n'.format(text))

for regex in regexes:
    print (type(regex))
    print('Seeking "{}" ->'.format(regex.pattern), end=' ')

    if regex.search(text):
        print('Matchig!')
    else:
        print('No, I am not matching')
```

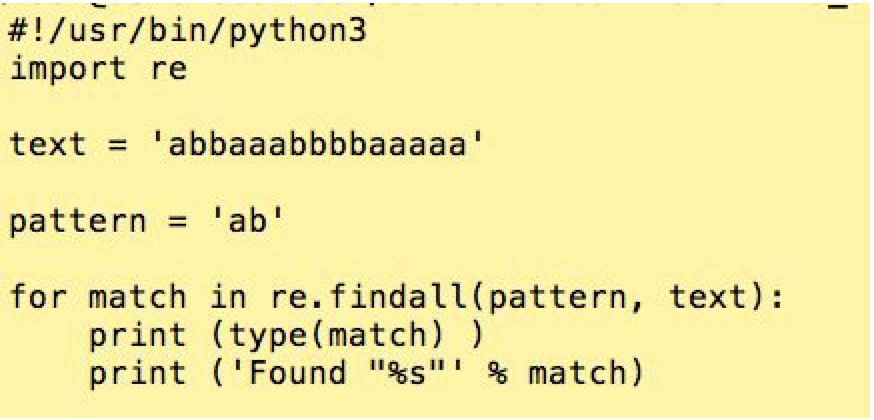


```
Text: 'Hello DataScience folks, How are you doing today?'

<class '_sre.SRE_Pattern'>
Seeking "Hello" -> Matchig!
<class '_sre.SRE_Pattern'>
Seeking "Donald" -> No, I am not matching
```

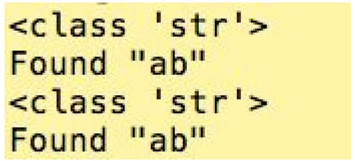
simple_compiled.py

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Regular Expression (Cont …)

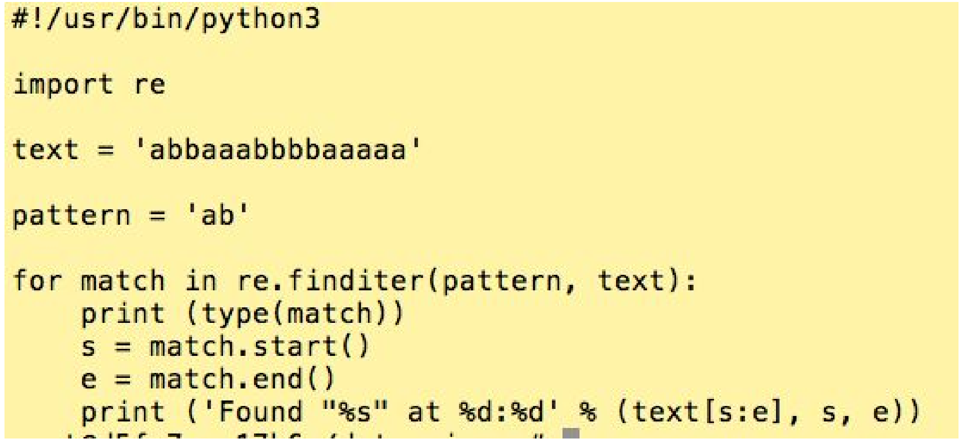


```
#!/usr/bin/python3
import re

text = 'abbaaabbbbaaaaa'

pattern = 'ab'

for match in re.findall(pattern, text):
    print (type(match) )
    print ('Found "%s"' % match)
```

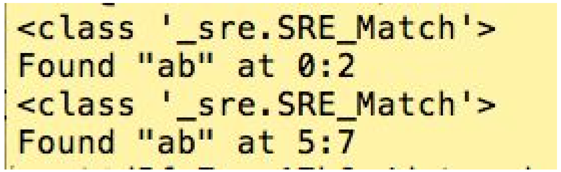


```
<class 'str'>
Found "ab"
<class 'str'>
Found "ab"
```

find_iter.py



```
#!/usr/bin/python3

import re

text = 'abbaaabbbbaaaaa'

pattern = 'ab'

for match in re.finditer(pattern, text):
    print (type(match))
    s = match.start()
    e = match.end()
    print ('Found "%s" at %d:%d' % (text[s:e], s, e))
```

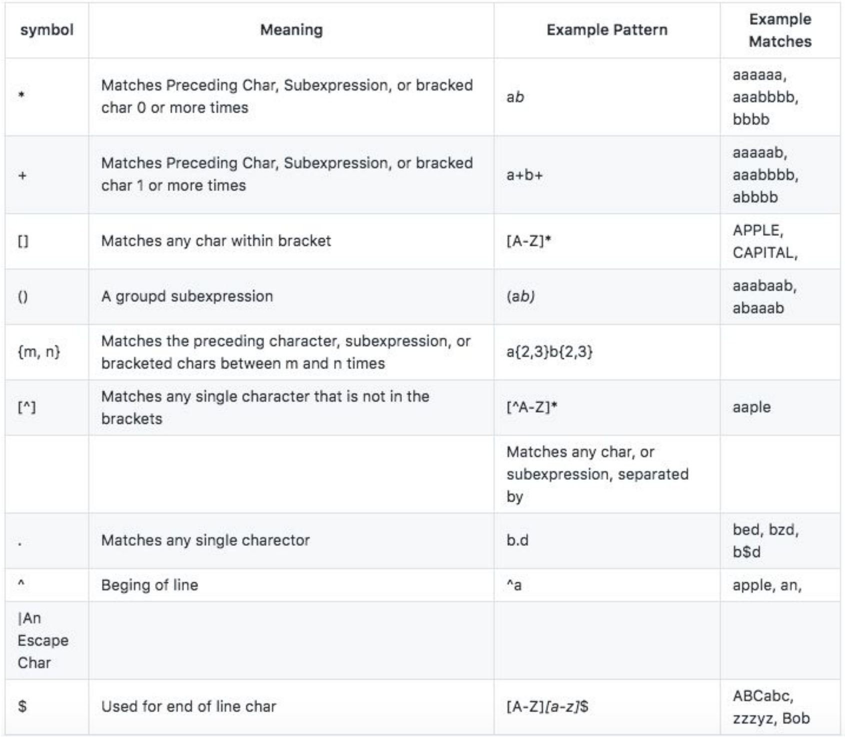


```
<class '_sre.SRE_Match'>
Found "ab" at 0:2
<class '_sre.SRE_Match'>
Found "ab" at 5:7
```

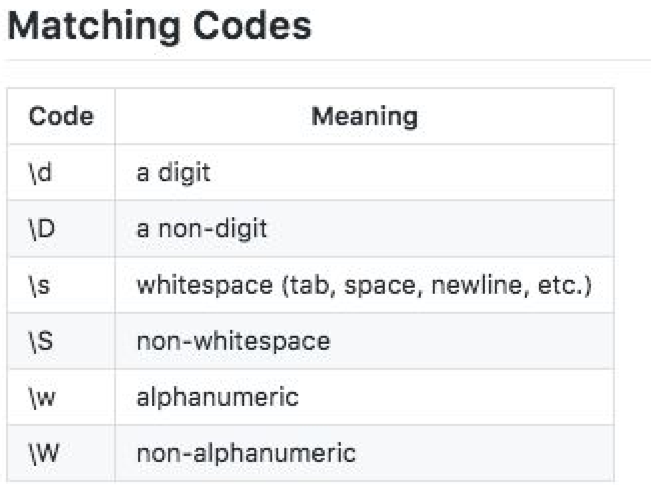
find_all.py

Regular Expression (Cont …)

A Few Rules: Commonly Used RegEx symbols



symbol	Meaning	Example Pattern	Example Matches
*	Matches Preceding Char, Subexpression, or bracked char 0 or more times	ab	aaaaaa, aaabbbb, bbbb
+	Matches Preceding Char, Subexpression, or bracked char 1 or more times	a+b+	aaaaab, aaabbbb, abbbb
[]	Matches any char within bracket	[A-Z]*	APPLE, CAPITAL,
()	A groupd subexpression	(ab)	aaabaab, abaaab
{m, n}	Matches the preceding character, subexpression, or bracketed chars between m and n times	a{2,3}b{2,3}	
[^]	Matches any single character that is not in the brackets	[^A-Z]*	aaple
		Matches any char, or subexpression, separated by	
.	Matches any single charector	b.d	bed, bzd, b$d
^	Beging of line	^a	apple, an,
|An Escape Char			
$	Used for end of line char	[A-Z][a-z]$	ABCabc, zzzyz, Bob



Matching Codes

Code	Meaning
\d	a digit
\D	a non-digit
\s	whitespace (tab, space, newline, etc.)
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric

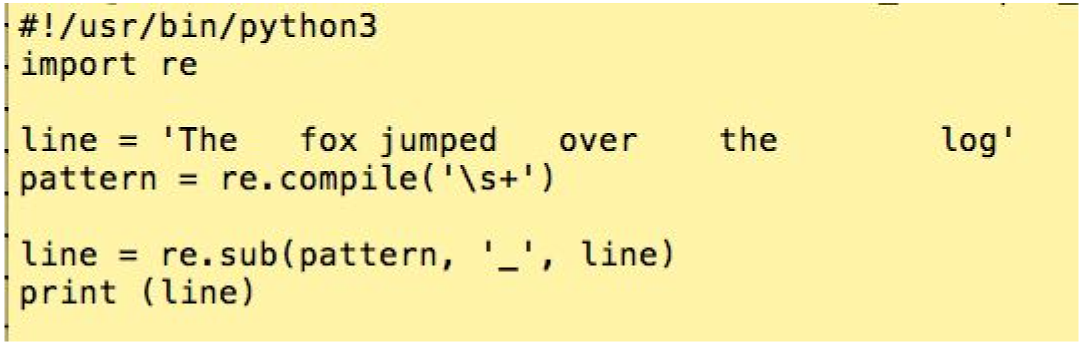
Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Regular Expression (Cont …)

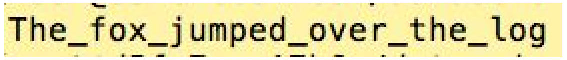
Mostly used functions from re module,

- compile(pattern, flags=0) – it compiles a regular expression pattern into a regular expression object, which can be used for matching using the match and search methods.
- match(pattern, string, flags=0) – if zero or more characters from the beginning of the string match, it returns a Match object, otherwise, it returns None.
- search(pattern, string, flags=0) – similar to match(), but it scans all the string, not only it's beginning.
- sub(pattern, repl, string, count=0, flags=0) - Return the string obtained by replacing the leftmost non-overlapping occurrences of the pattern in string by the replacement repl. repl can be either a string or a callable; if a string, backslash escapes in it are processed. If it is a callable, it's passed the match object and must return a replacement string to be used.

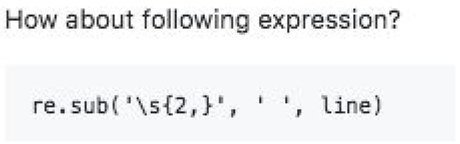
Regular Expression (Cont …)



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

line = 'The    fox jumped    over     the          log'
pattern = re.compile('\s+')

line = re.sub(pattern, '_', line)
print (line)
```



```
The_fox_jumped_over_the_log
```



How about following expression?

```
re.sub('\s{2,}', ' ', line)
```

remove_multiple_spaces.py

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Task - 3

- Remove starting spaces in line
- Remove ending spaces in line
- How about removing all digits from line?

Develop Regular Expression to extract numbers (float, integers) from given text string. The numbers can be in any of the following format

```
#   '10.5', '-10.5', '- 10.5', '+ .2e10', ' 1.01e-2', '      1.01', '-.2', ' 456', ' .123'
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Quiz

https://doughellmann.com/presentations/regexes-fear

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Databases

What is Database?

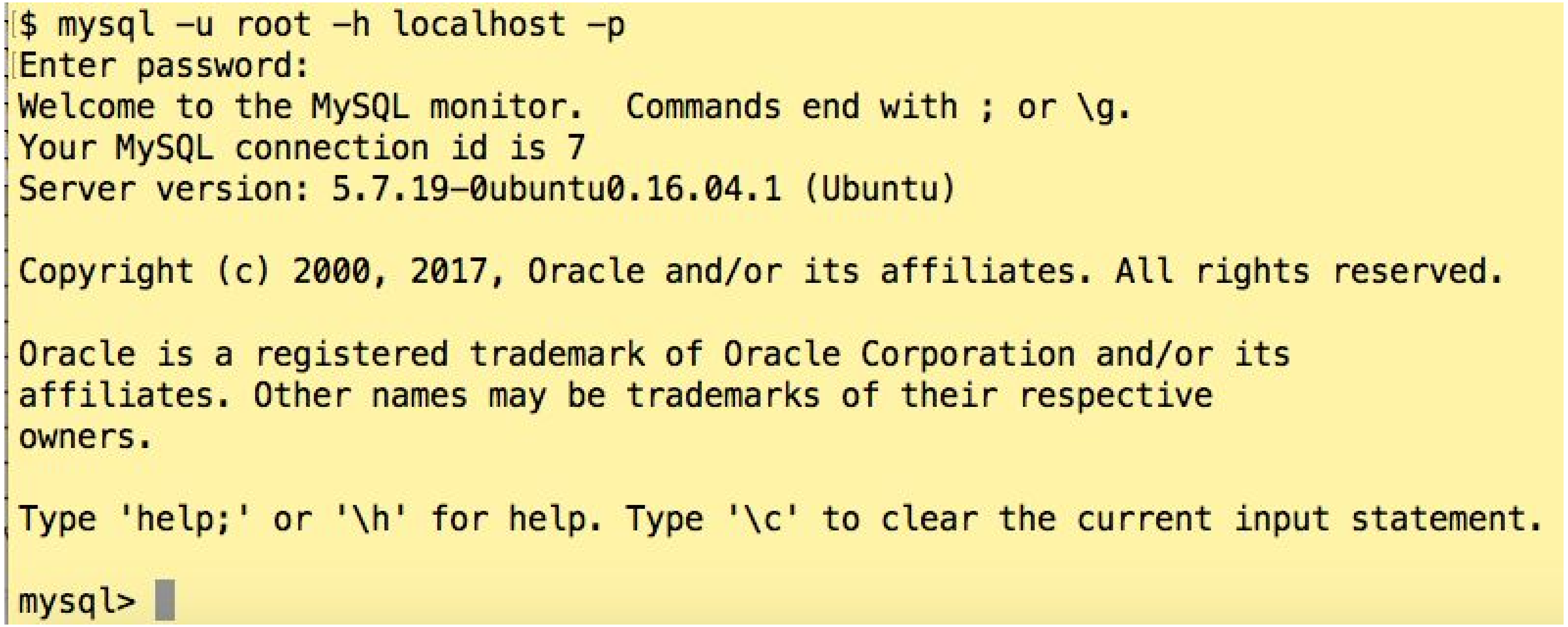
Language: SQL is a standard language for storing, manipulating and retrieving data in databases

Examples - MySQL, PostgreSQL, Oracle, SQLite

Reference - https://dev.mysql.com/doc/connector-python/en/

MySQL

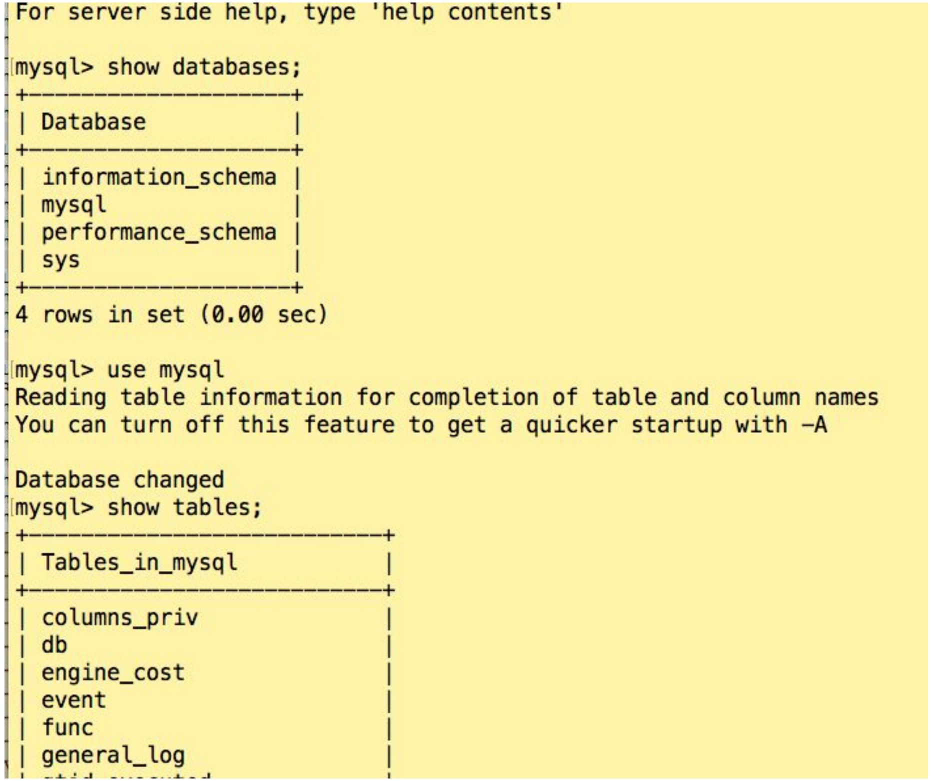
Let us focus on MySQL



```
$ mysql -u root -h localhost -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

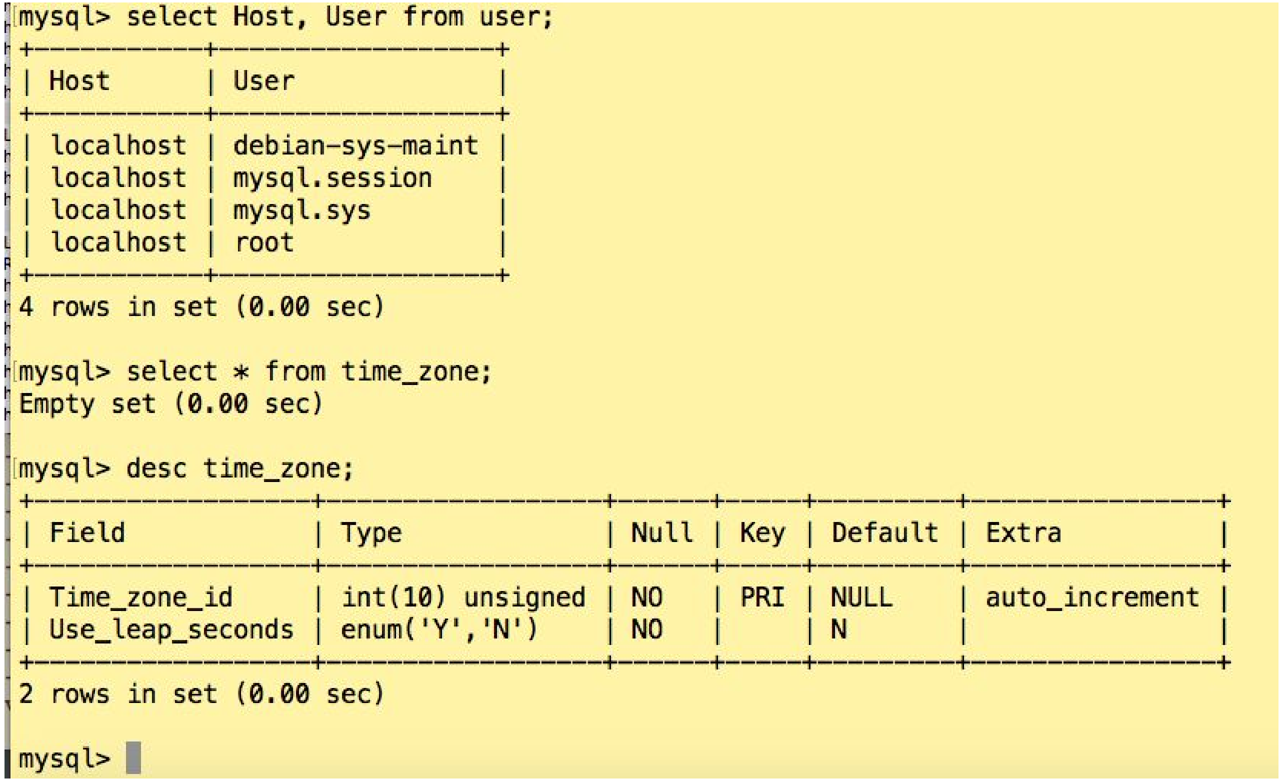
MySQL



```
For server side help, type 'help contents'

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> use mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| engine_cost               |
| event                     |
| func                      |
| general_log               |
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

MySQL



```
mysql> select Host, User from user;
+-----------+------------------+
| Host      | User             |
+-----------+------------------+
| localhost | debian-sys-maint |
| localhost | mysql.session    |
| localhost | mysql.sys        |
| localhost | root             |
+-----------+------------------+
4 rows in set (0.00 sec)

mysql> select * from time_zone;
Empty set (0.00 sec)

mysql> desc time_zone;
+------------------+------------------+------+-----+---------+----------------+
| Field            | Type             | Null | Key | Default | Extra          |
+------------------+------------------+------+-----+---------+----------------+
| Time_zone_id     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| Use_leap_seconds | enum('Y','N')    | NO   |     | N       |                |
+------------------+------------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql>
```

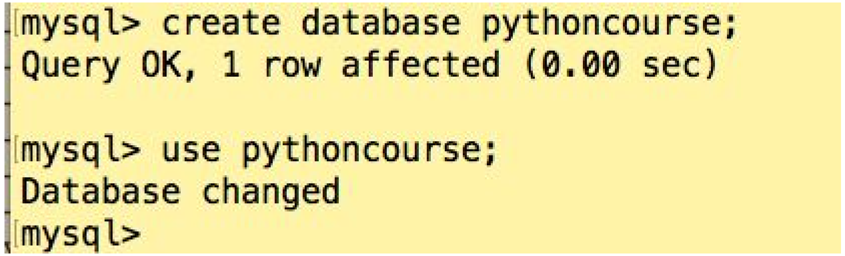
Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

MySQL

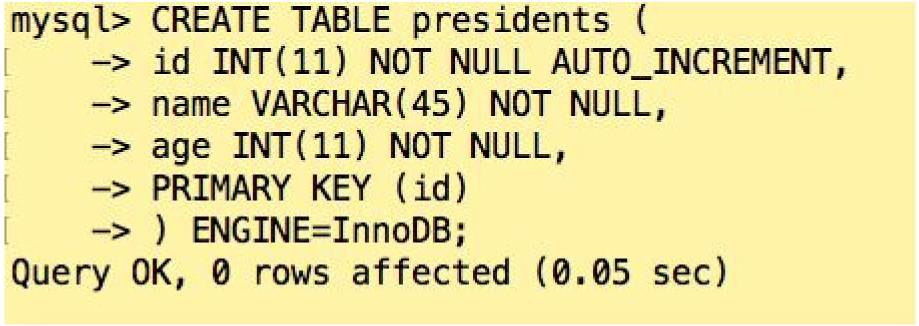
Python Library/Connector Required to connect, retrieve, update data with MySQL database

- MySQL Connector/Python - from Official MySQL Site
- https://pypi.python.org/pypi/mysqlclient (mysqlclient)
- https://github.com/farcepest/moist - Moist
- PyMySQL

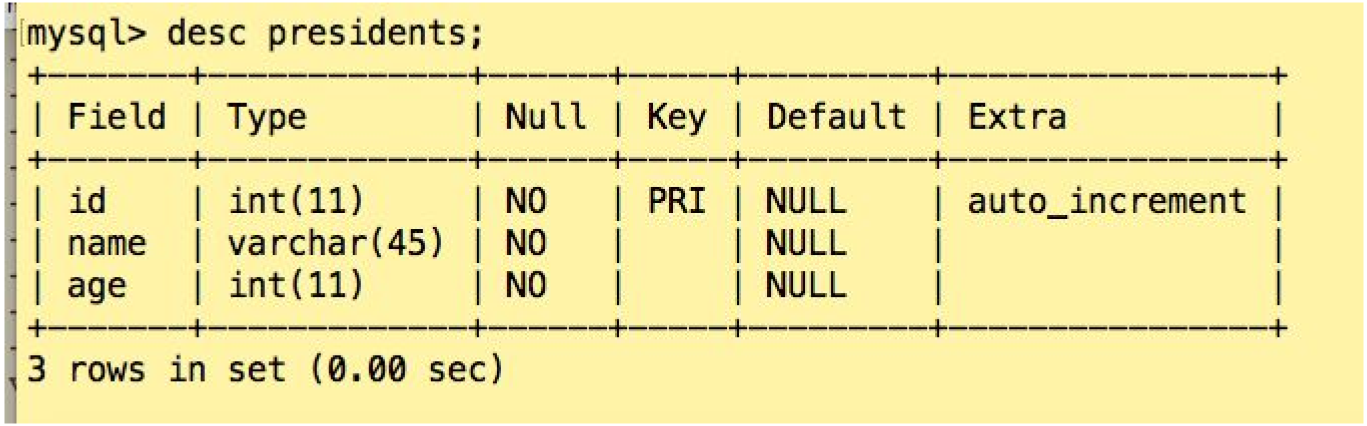
MySQL



```
mysql> create database pythoncourse;
Query OK, 1 row affected (0.00 sec)

mysql> use pythoncourse;
Database changed
mysql>
```



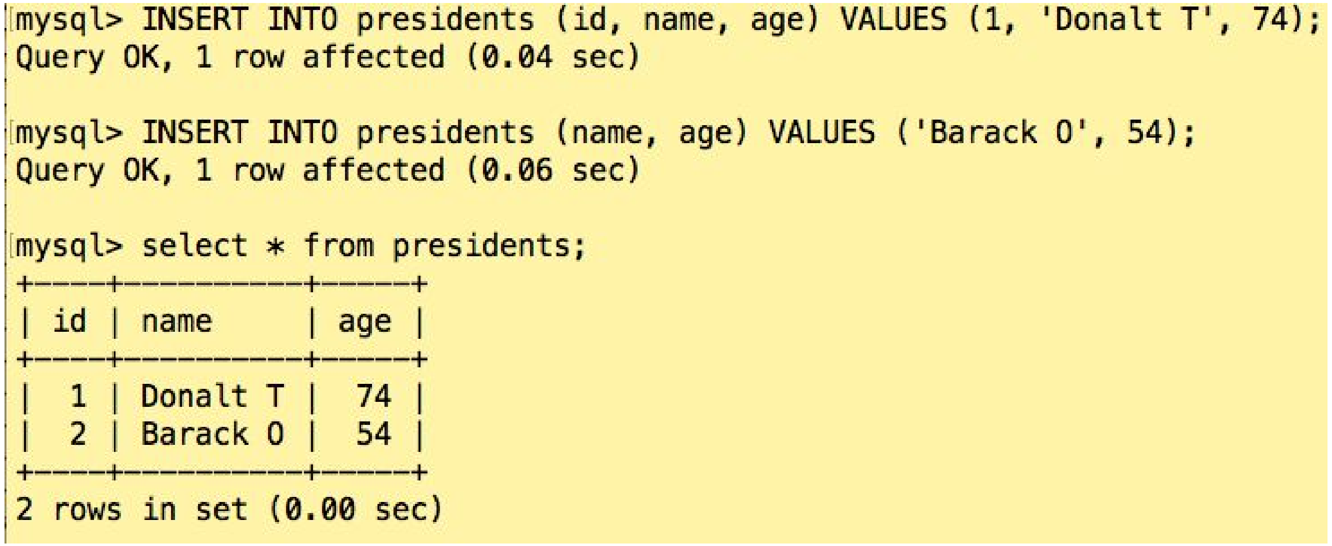
```
mysql> CREATE TABLE presidents (
    -> id INT(11) NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(45) NOT NULL,
    -> age INT(11) NOT NULL,
    -> PRIMARY KEY (id)
    -> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.05 sec)
```



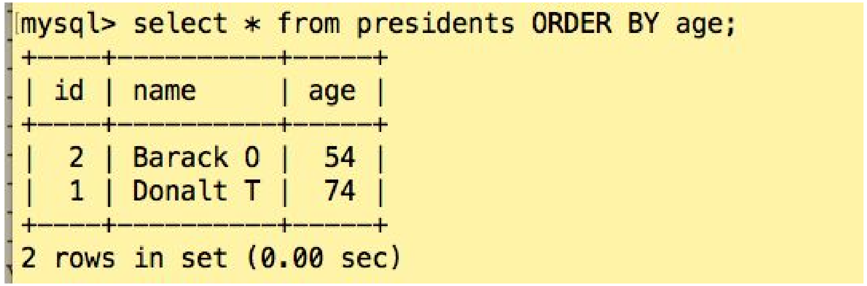
```
mysql> desc presidents;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| name  | varchar(45) | NO   |     | NULL    |                |
| age   | int(11)     | NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

MySQL



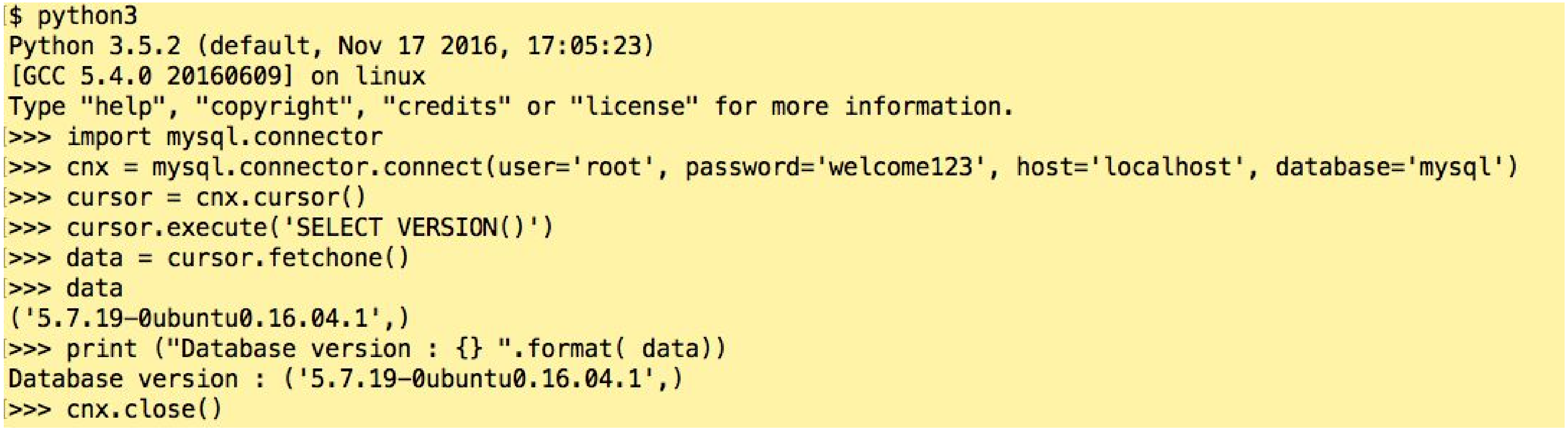
```
mysql> INSERT INTO presidents (id, name, age) VALUES (1, 'Donalt T', 74);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO presidents (name, age) VALUES ('Barack O', 54);
Query OK, 1 row affected (0.06 sec)

mysql> select * from presidents;
+----+----------+-----+
| id | name     | age |
+----+----------+-----+
|  1 | Donalt T |  74 |
|  2 | Barack O |  54 |
+----+----------+-----+
2 rows in set (0.00 sec)
```



```
mysql> select * from presidents ORDER BY age;
+----+----------+-----+
| id | name     | age |
+----+----------+-----+
|  2 | Barack O |  54 |
|  1 | Donalt T |  74 |
+----+----------+-----+
2 rows in set (0.00 sec)
```

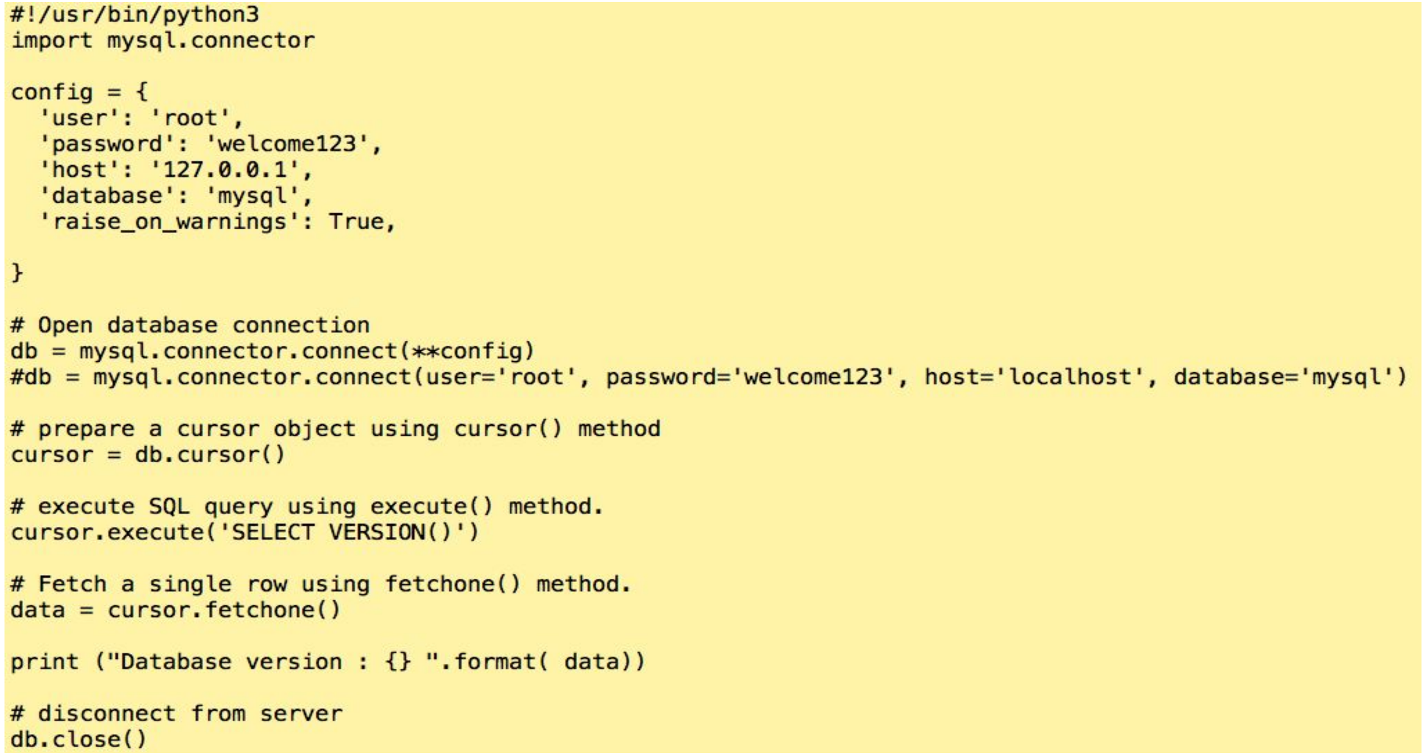
Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

MySQL



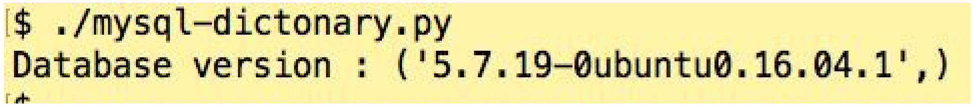
```
$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql.connector
>>> cnx = mysql.connector.connect(user='root', password='welcome123', host='localhost', database='mysql')
>>> cursor = cnx.cursor()
>>> cursor.execute('SELECT VERSION()')
>>> data = cursor.fetchone()
>>> data
('5.7.19-0ubuntu0.16.04.1',)
>>> print ("Database version : {} ".format( data))
Database version : ('5.7.19-0ubuntu0.16.04.1',)
>>> cnx.close()
```

Python + MySQL



```
#!/usr/bin/python3
import mysql.connector

config = {
  'user': 'root',
  'password': 'welcome123',
  'host': '127.0.0.1',
  'database': 'mysql',
  'raise_on_warnings': True,

}

# Open database connection
db = mysql.connector.connect(**config)
#db = mysql.connector.connect(user='root', password='welcome123', host='localhost', database='mysql')

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute('SELECT VERSION()')

# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print ("Database version : {} ".format( data))

# disconnect from server
db.close()
```

mysql-dictionary.py



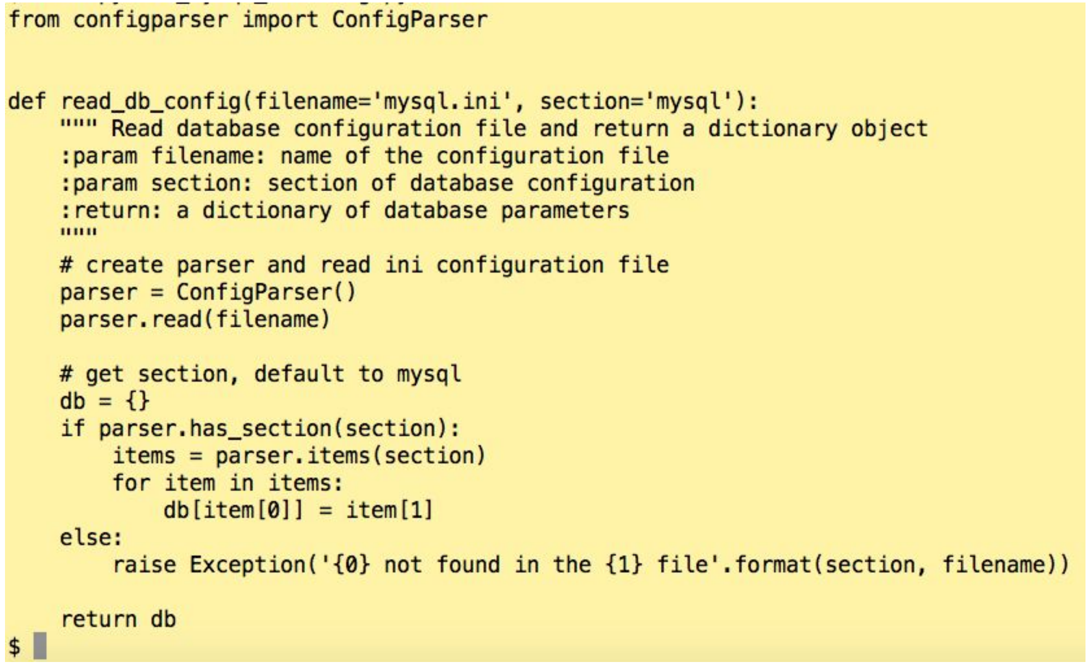
```
[$ ./mysql-dictonary.py
Database version : ('5.7.19-0ubuntu0.16.04.1',)
[$
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

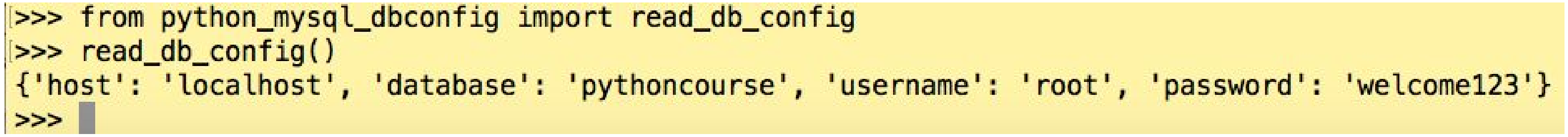
Python + MySQL

Let us use “ConfigParser” Module

pythonmysql-dbconfig.py



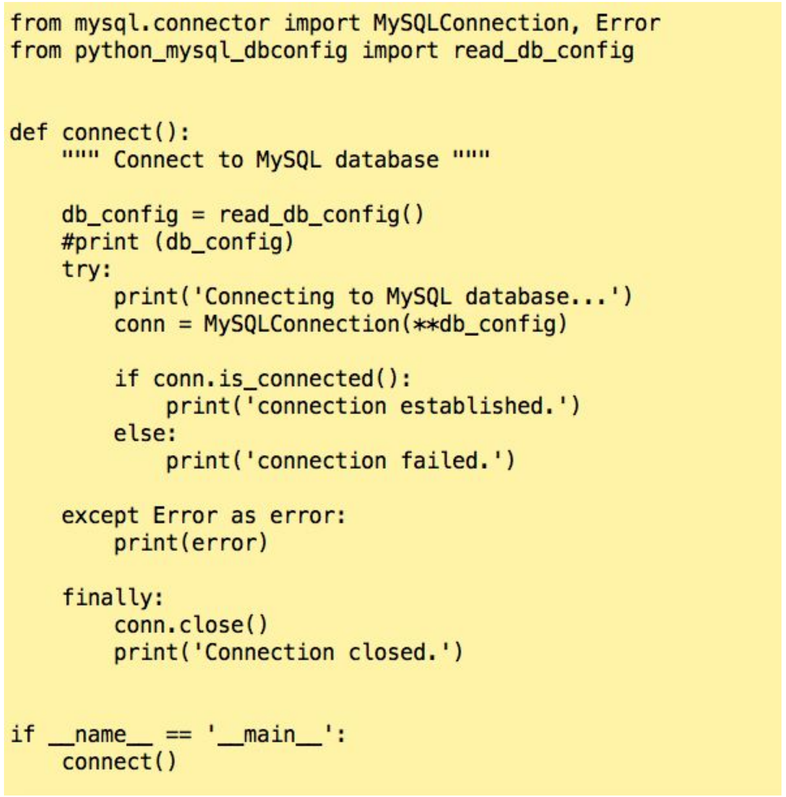
```
from configparser import ConfigParser


def read_db_config(filename='mysql.ini', section='mysql'):
    """ Read database configuration file and return a dictionary object
    :param filename: name of the configuration file
    :param section: section of database configuration
    :return: a dictionary of database parameters
    """
    # create parser and read ini configuration file
    parser = ConfigParser()
    parser.read(filename)

    # get section, default to mysql
    db = {}
    if parser.has_section(section):
        items = parser.items(section)
        for item in items:
            db[item[0]] = item[1]
    else:
        raise Exception('{0} not found in the {1} file'.format(section, filename))

    return db
$
```



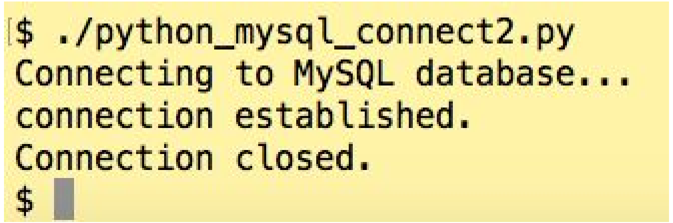
```
>>> from python_mysql_dbconfig import read_db_config
>>> read_db_config()
{'host': 'localhost', 'database': 'pythoncourse', 'username': 'root', 'password': 'welcome123'}
>>>
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Python + MySQL



```
from mysql.connector import MySQLConnection, Error
from python_mysql_dbconfig import read_db_config


def connect():
    """ Connect to MySQL database """

    db_config = read_db_config()
    #print (db_config)
    try:
        print('Connecting to MySQL database...')
        conn = MySQLConnection(**db_config)

        if conn.is_connected():
            print('connection established.')
        else:
            print('connection failed.')

    except Error as error:
        print(error)

    finally:
        conn.close()
        print('Connection closed.')


if __name__ == '__main__':
    connect()
```



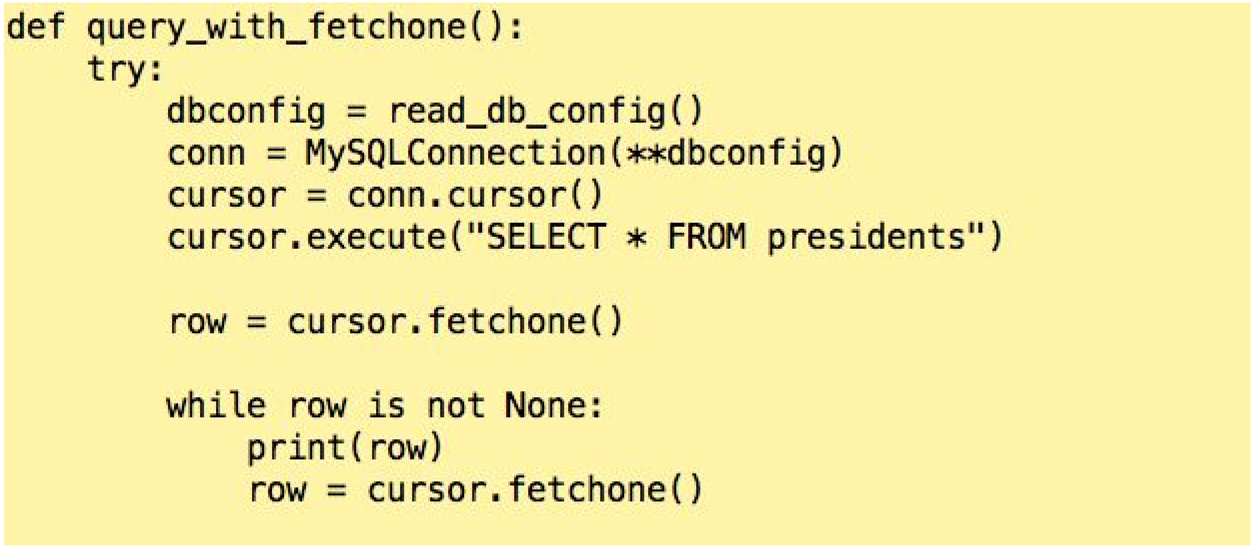
```
$ ./python_mysql_connect2.py
Connecting to MySQL database...
connection established.
Connection closed.
$ 
```

python-mysql-connect2.py

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

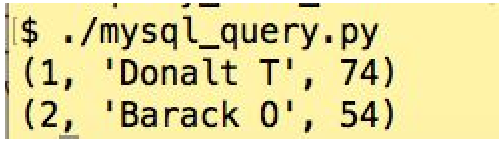
Python+MySQL

Query data - one row at one time. The fetchone() method returns the next row of a query result set or None in case there is no row left.



```python
def query_with_fetchone():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM presidents")

        row = cursor.fetchone()

        while row is not None:
            print(row)
            row = cursor.fetchone()
```

mysql_query.py



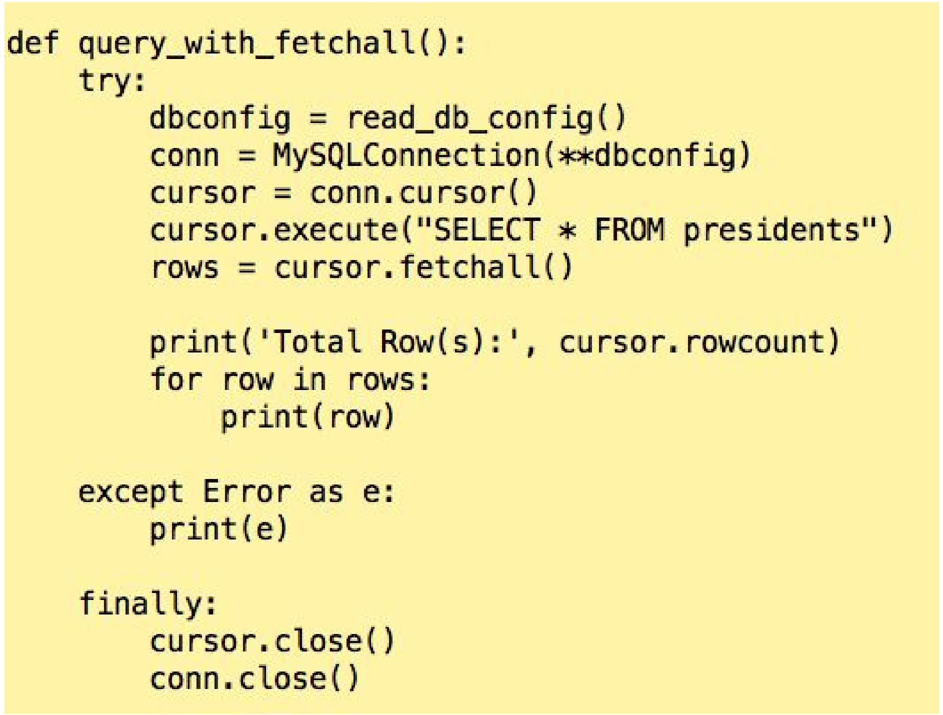
```
$ ./mysql_query.py
(1, 'Donalt T', 74)
(2, 'Barack O', 54)
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

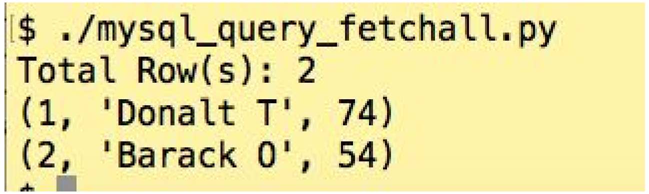
Python + MySQL

Query data - all row in one go. The fetchall() method is used. Use this method only when table

Is small - less number of rows



```
def query_with_fetchall():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM presidents")
        rows = cursor.fetchall()

        print('Total Row(s):', cursor.rowcount)
        for row in rows:
            print(row)

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()
```



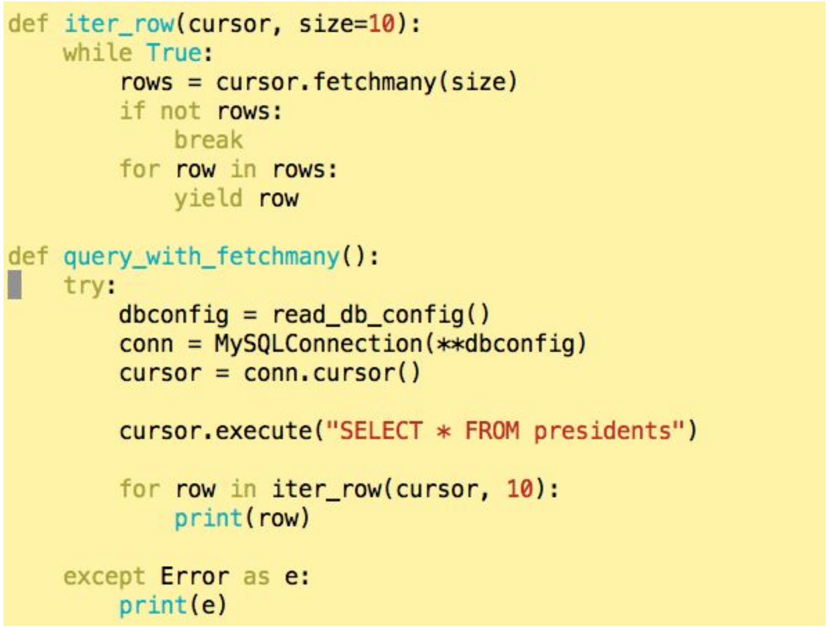
```
$ ./mysql_query_fetchall.py
Total Row(s): 2
(1, 'Donalt T', 74)
(2, 'Barack O', 54)
```

mysql_query_fetchall.py

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Python + MySQL

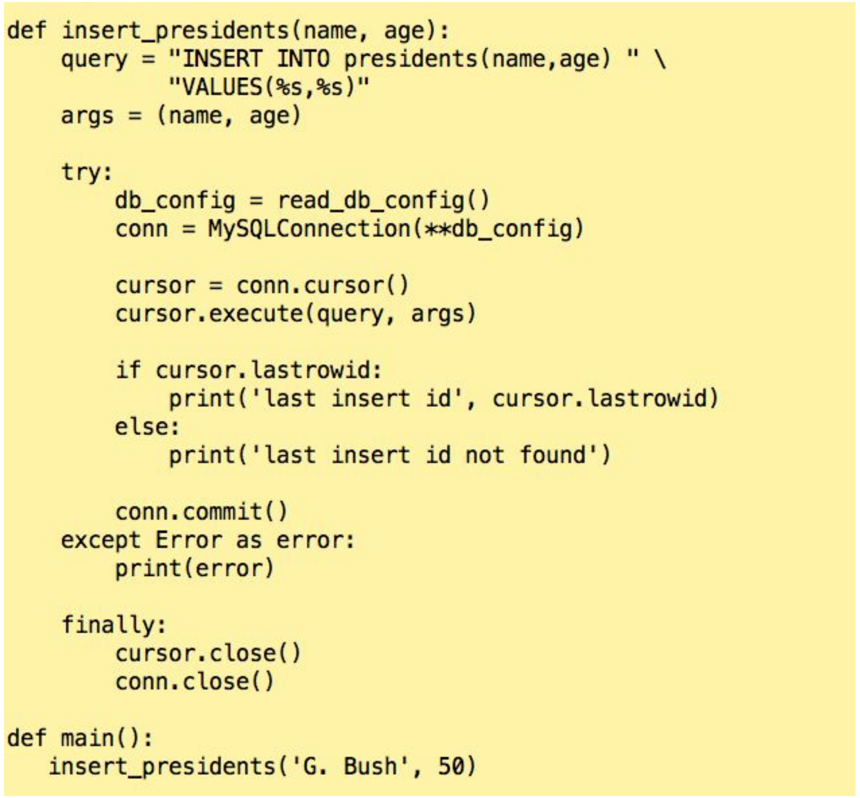
MySQL Connector/Python provides us with the fetchmany() method that returns the next number of rows (n) of the result set, which allows us to balance between time and memory space. Let's take a look at how do we use fetchmany() method.



```python
def iter_row(cursor, size=10):
    while True:
        rows = cursor.fetchmany(size)
        if not rows:
            break
        for row in rows:
            yield row

def query_with_fetchmany():
    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()

        cursor.execute("SELECT * FROM presidents")

        for row in iter_row(cursor, 10):
            print(row)

    except Error as e:
        print(e)
```

mysql-query_fetchmany.py

Insert



```
def insert_presidents(name, age):
    query = "INSERT INTO presidents(name,age) " \
            "VALUES(%s,%s)"
    args = (name, age)

    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.execute(query, args)

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
        else:
            print('last insert id not found')

        conn.commit()
    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

def main():
   insert_presidents('G. Bush', 50)
```

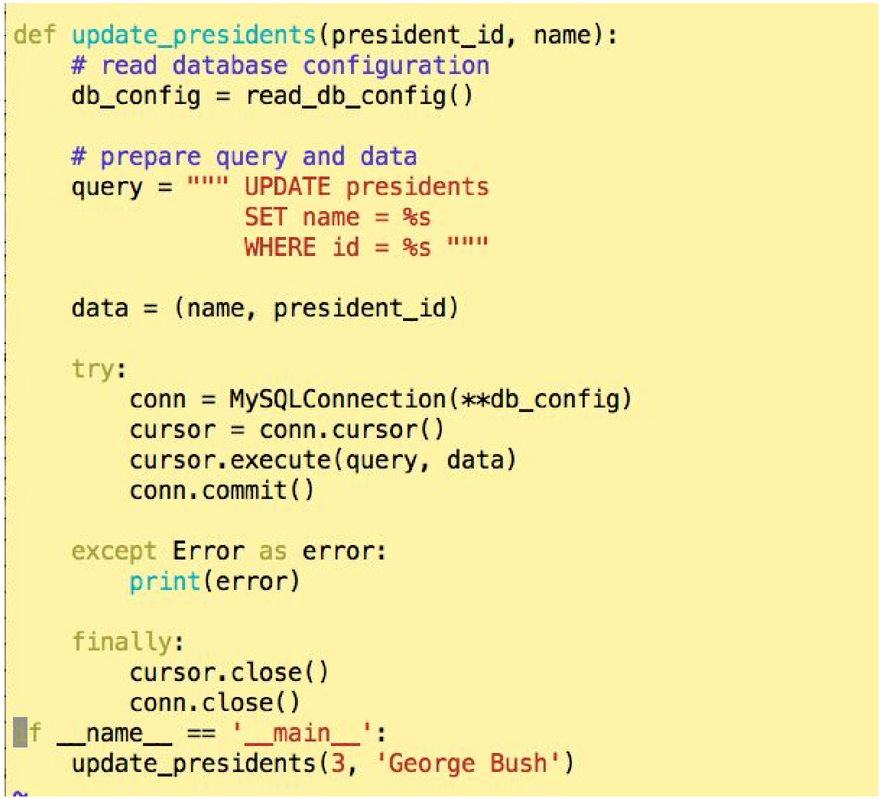
mysql_insert_records.py

insert-multiple

```
def insert_presidents(presidents):
    query = "INSERT INTO presidents(name,age) " \
            "VALUES(%s,%s)"

    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.executemany(query, presidents)

        conn.commit()
    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

def main():
   presidents = [ ('A', 10), ('B', 20), ('C', 30)]
   insert_presidents(presidents)
```

mysql_insert_multiple_records.py

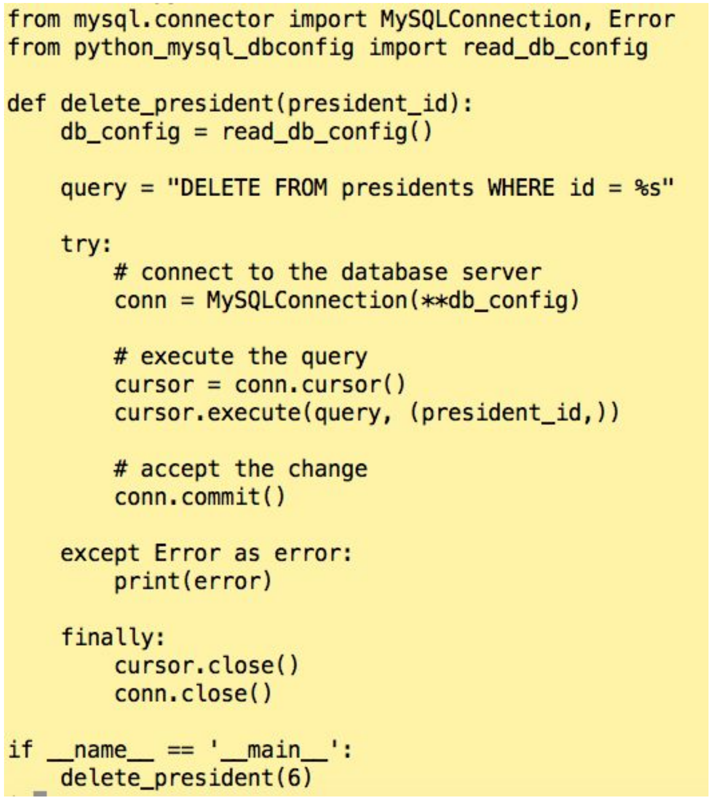
Update Record



```python
def update_presidents(president_id, name):
    # read database configuration
    db_config = read_db_config()

    # prepare query and data
    query = """ UPDATE presidents
                SET name = %s
                WHERE id = %s """

    data = (name, president_id)

    try:
        conn = MySQLConnection(**db_config)
        cursor = conn.cursor()
        cursor.execute(query, data)
        conn.commit()

    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()
if __name__ == '__main__':
    update_presidents(3, 'George Bush')
```

mysql_update_record.py

Delete Record



```
from mysql.connector import MySQLConnection, Error
from python_mysql_dbconfig import read_db_config

def delete_president(president_id):
    db_config = read_db_config()

    query = "DELETE FROM presidents WHERE id = %s"

    try:
        # connect to the database server
        conn = MySQLConnection(**db_config)

        # execute the query
        cursor = conn.cursor()
        cursor.execute(query, (president_id,))

        # accept the change
        conn.commit()

    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()

if __name__ == '__main__':
    delete_president(6)
```

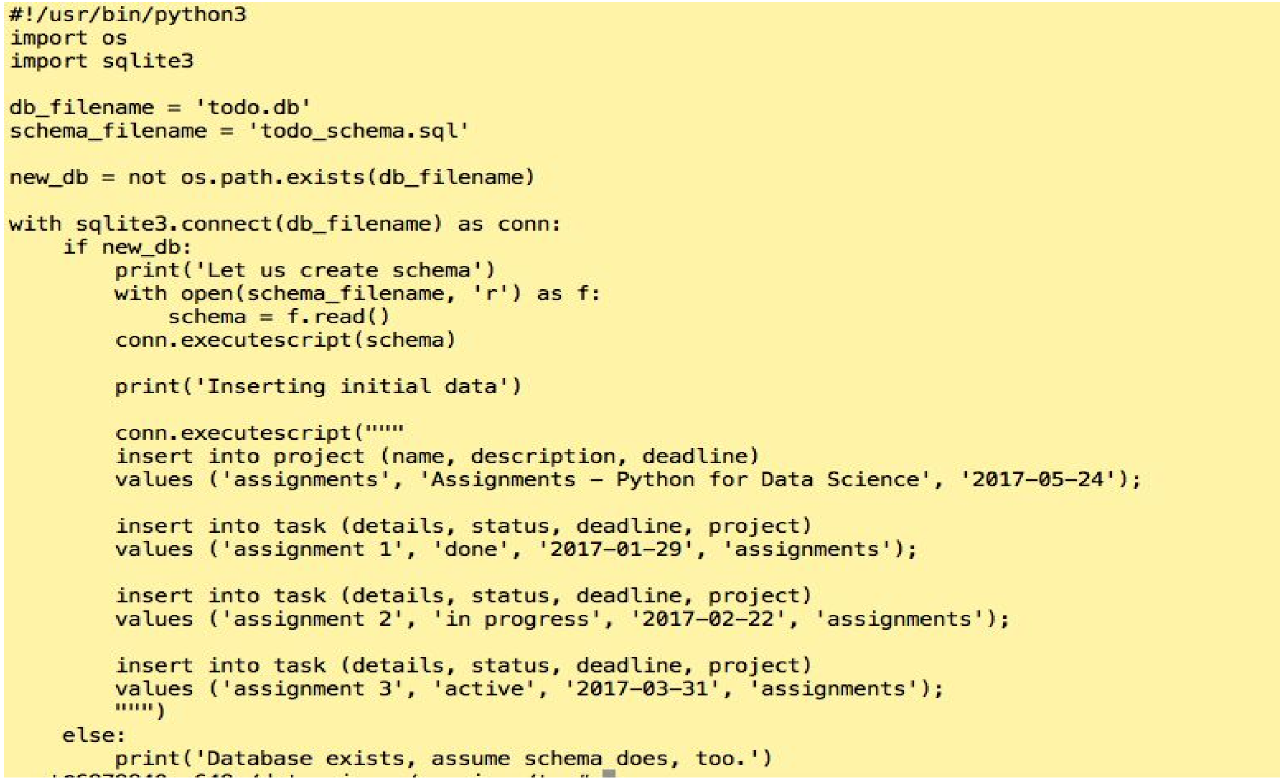
mysql_delete_record.py

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

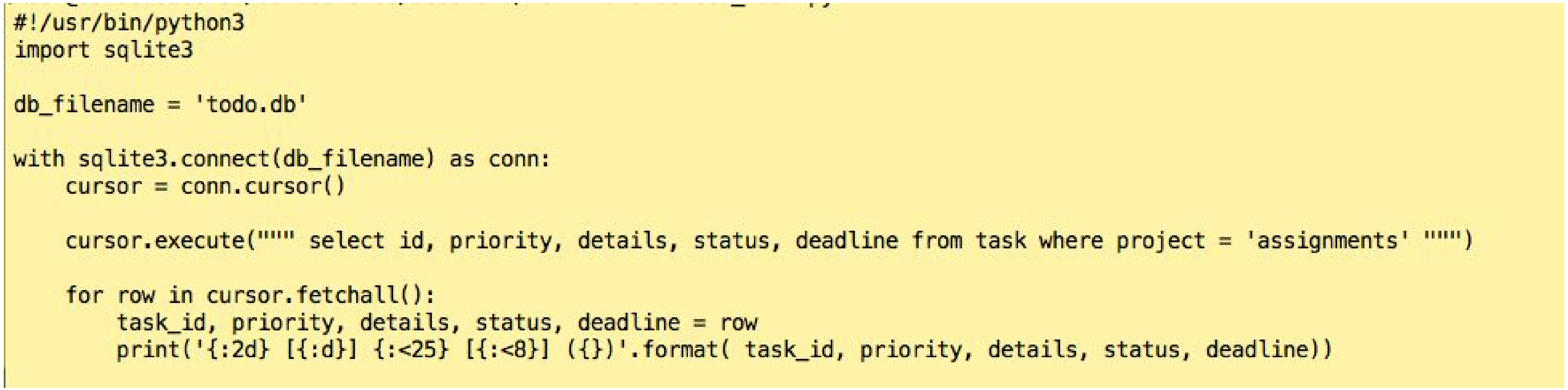
Database - create schema & add data



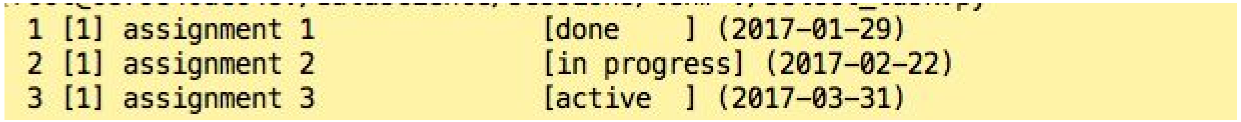
```
#!/usr/bin/python3
import os
import sqlite3

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.')
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Database - Retrieve Data



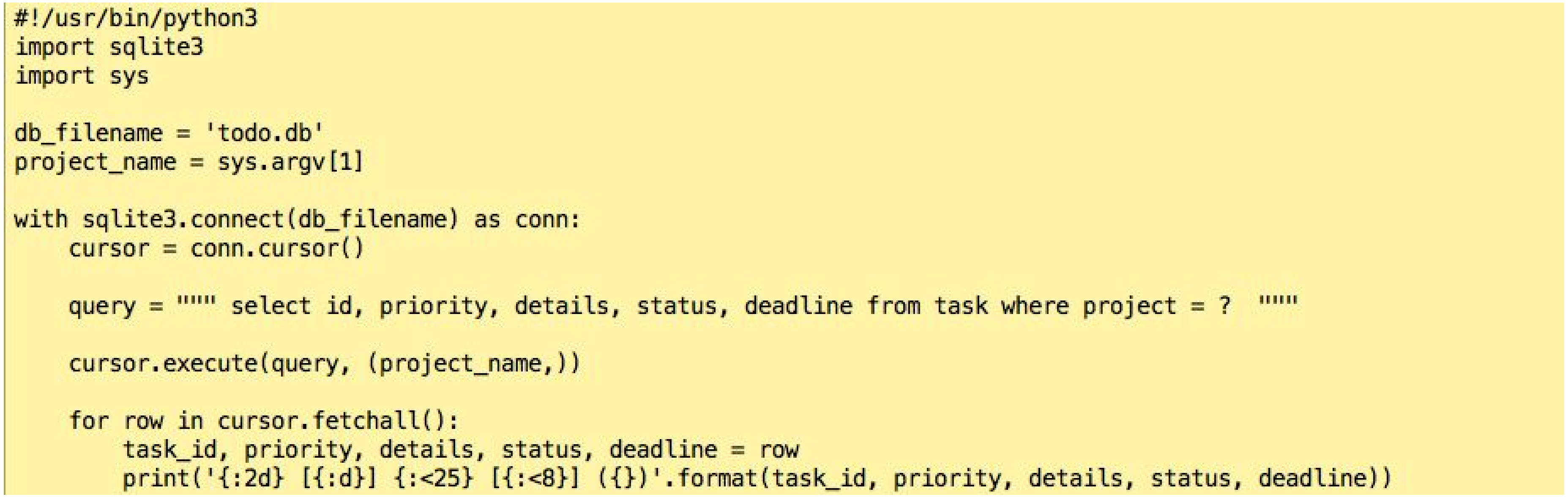
```
#!/usr/bin/python3
import sqlite3

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))
```



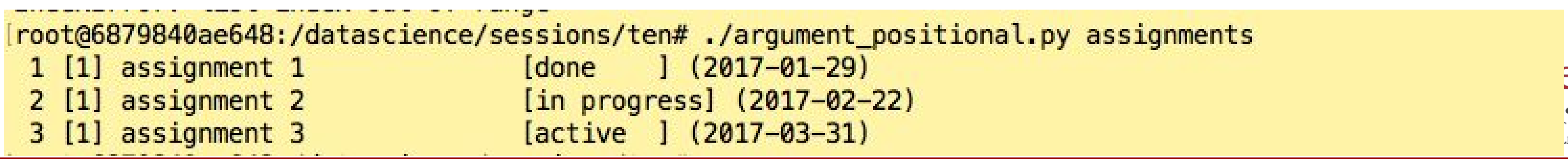
```
 1 [1] assignment 1              [done    ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active  ] (2017-03-31)
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Database - positional argument

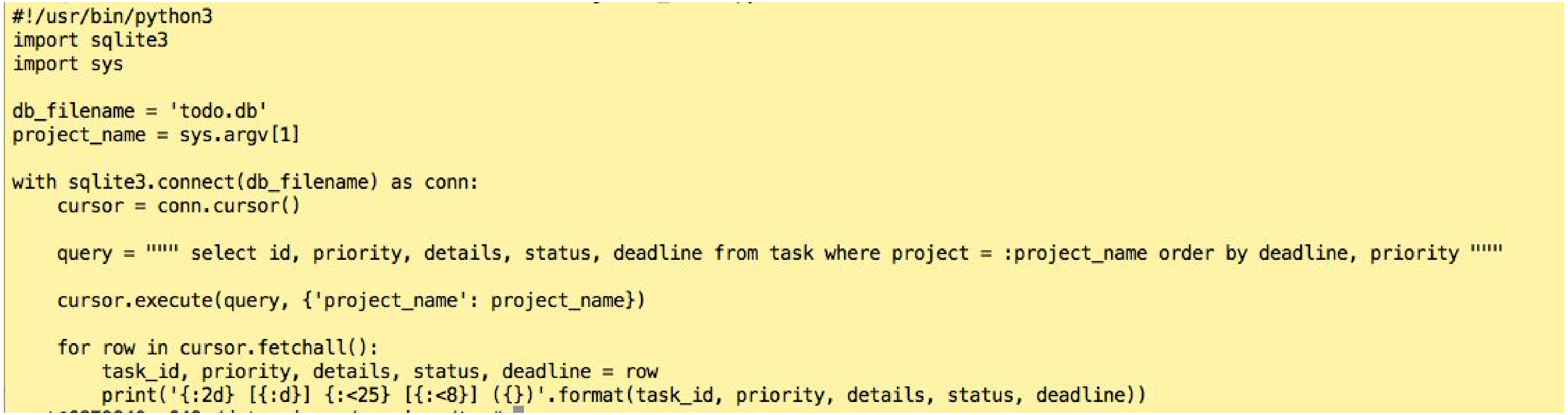


```
#!/usr/bin/python3
import sqlite3
import sys

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))
```

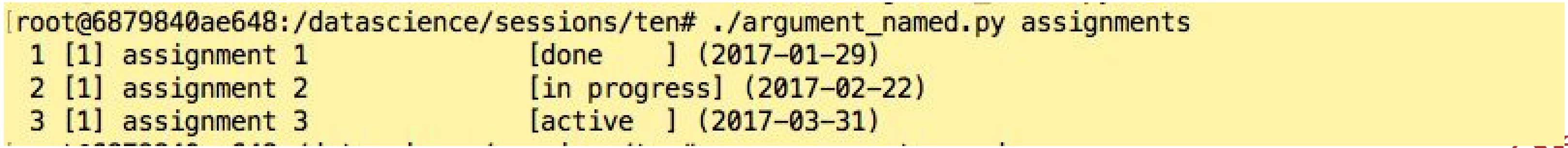


```
root@6879840ae648:/datascience/sessions/ten# ./argument_positional.py assignments
 1 [1] assignment 1              [done    ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active  ] (2017-03-31)
```

argument_named.py



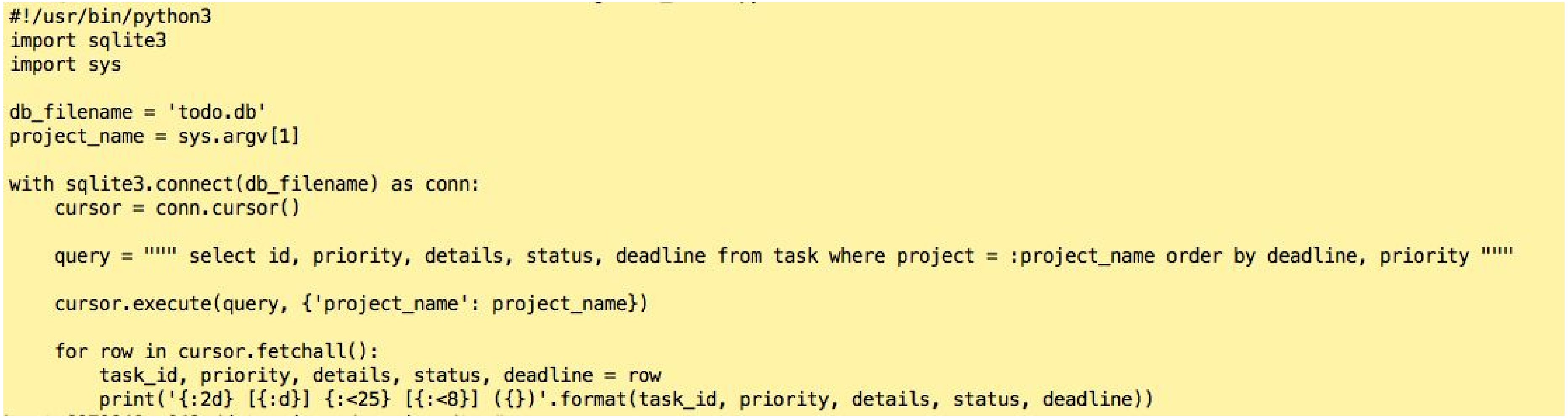
```
#!/usr/bin/python3
import sqlite3
import sys

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))
```



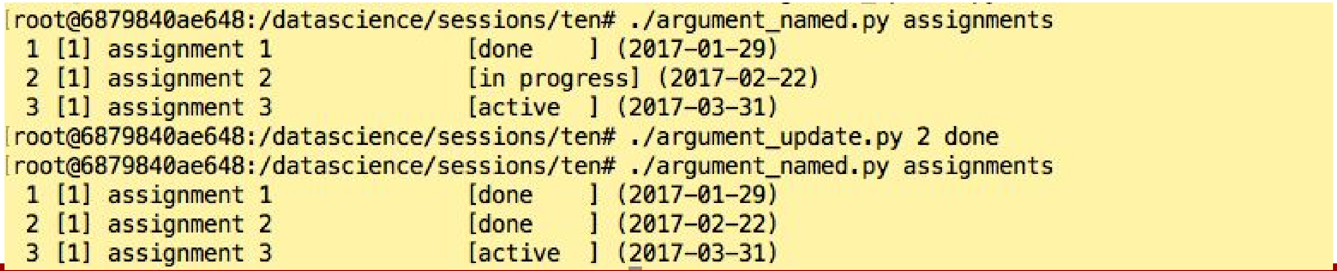
```
root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1              [done    ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active  ] (2017-03-31)
```

Regis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

argument_update.py

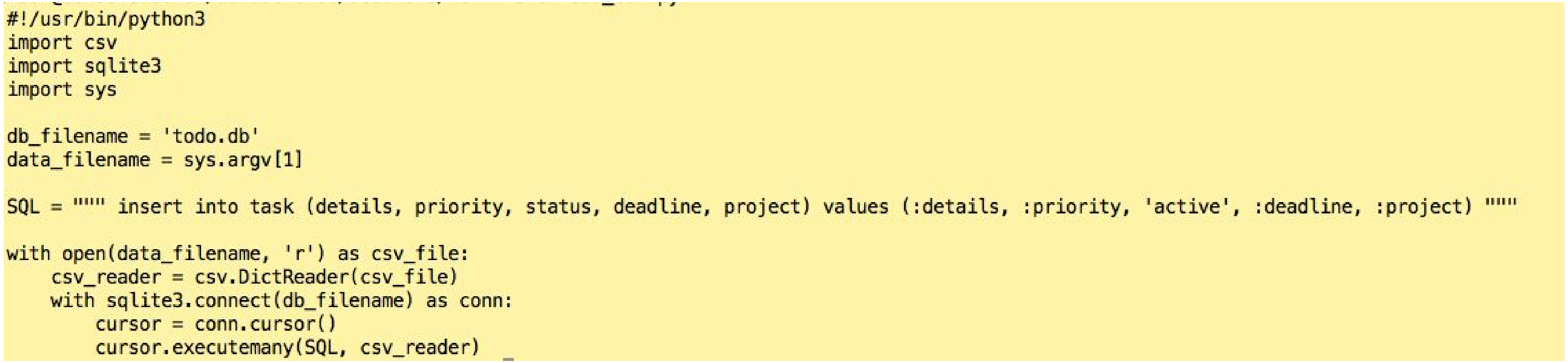


```
#!/usr/bin/python3
import sqlite3
import sys

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))
```

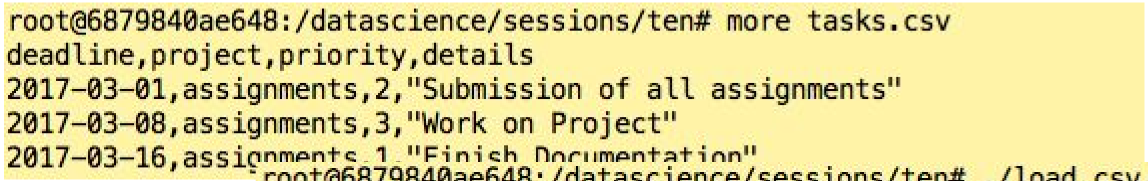


```
[root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1              [done    ] (2017-01-29)
 2 [1] assignment 2              [in progress] (2017-02-22)
 3 [1] assignment 3              [active  ] (2017-03-31)
[root@6879840ae648:/datascience/sessions/ten#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

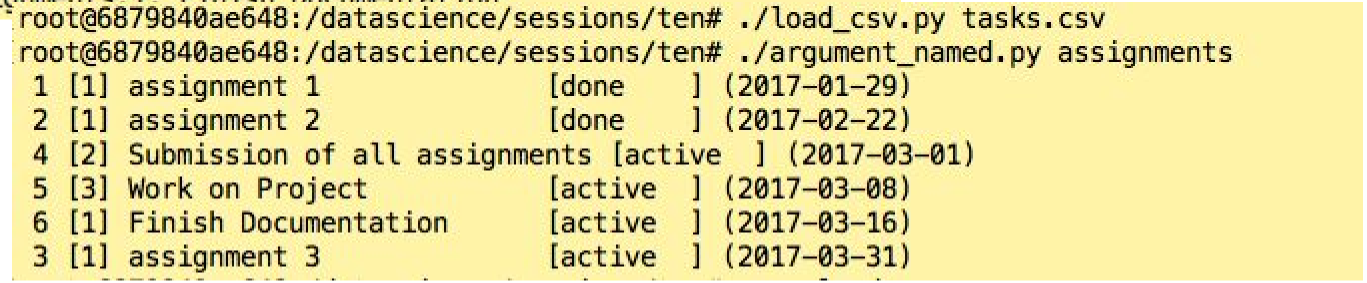
load_csv.py



```
#!/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"
```

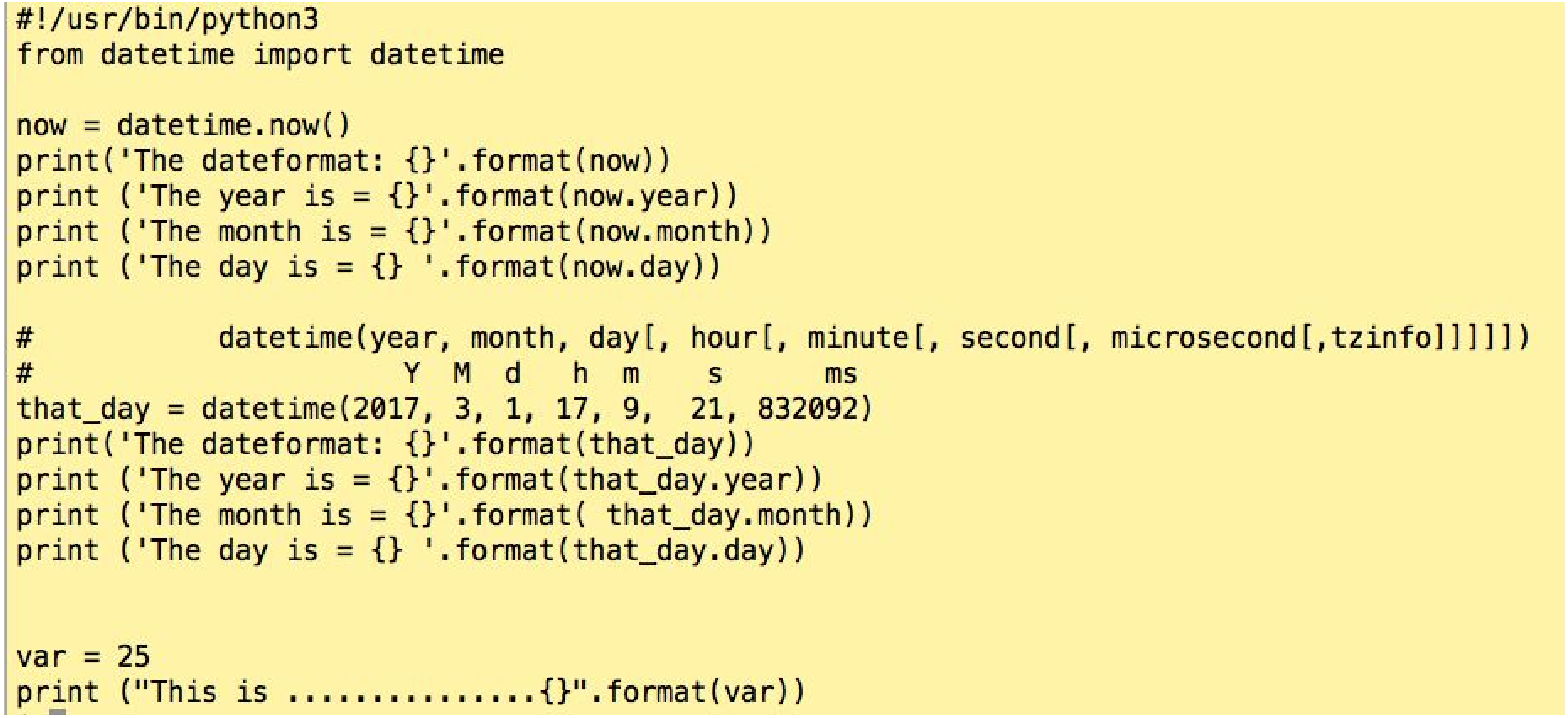


```
root@6879840ae648:/datascience/sessions/ten# ./load_csv.py tasks.csv
root@6879840ae648:/datascience/sessions/ten# ./argument_named.py assignments
 1 [1] assignment 1                [done    ] (2017-01-29)
 2 [1] assignment 2                [done    ] (2017-02-22)
 4 [2] Submission of all assignments [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)
```

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

datetime Library

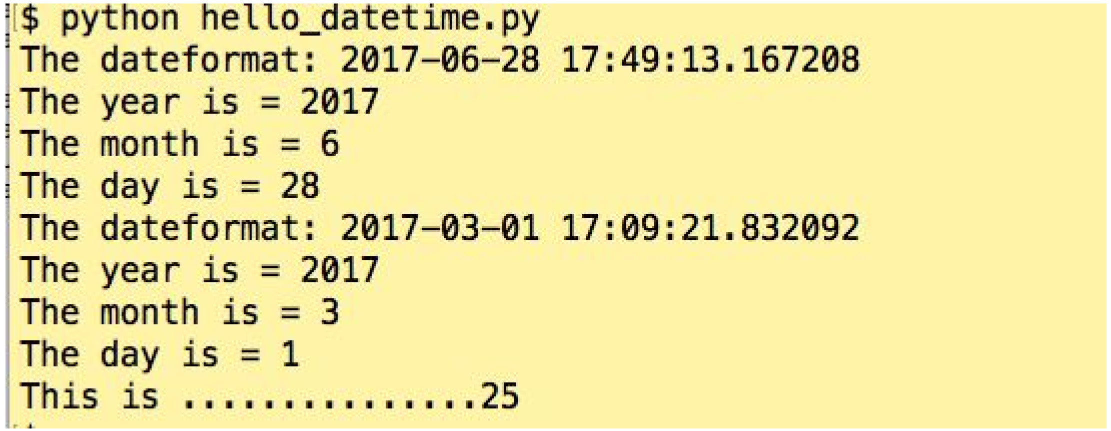
hello_datetime.py



```
#!/usr/bin/python3
from datetime import datetime

now = datetime.now()
print('The dateformat: {}'.format(now))
print ('The year is = {}'.format(now.year))
print ('The month is = {}'.format(now.month))
print ('The day is = {} '.format(now.day))

#           datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinfo]]]]])
#                      Y  M  d   h  m    s      ms
that_day = datetime(2017, 3, 1, 17, 9,  21, 832092)
print('The dateformat: {}'.format(that_day))
print ('The year is = {}'.format(that_day.year))
print ('The month is = {}'.format( that_day.month))
print ('The day is = {} '.format(that_day.day))


var = 25
print ("This is ...............{}".format(var))
```

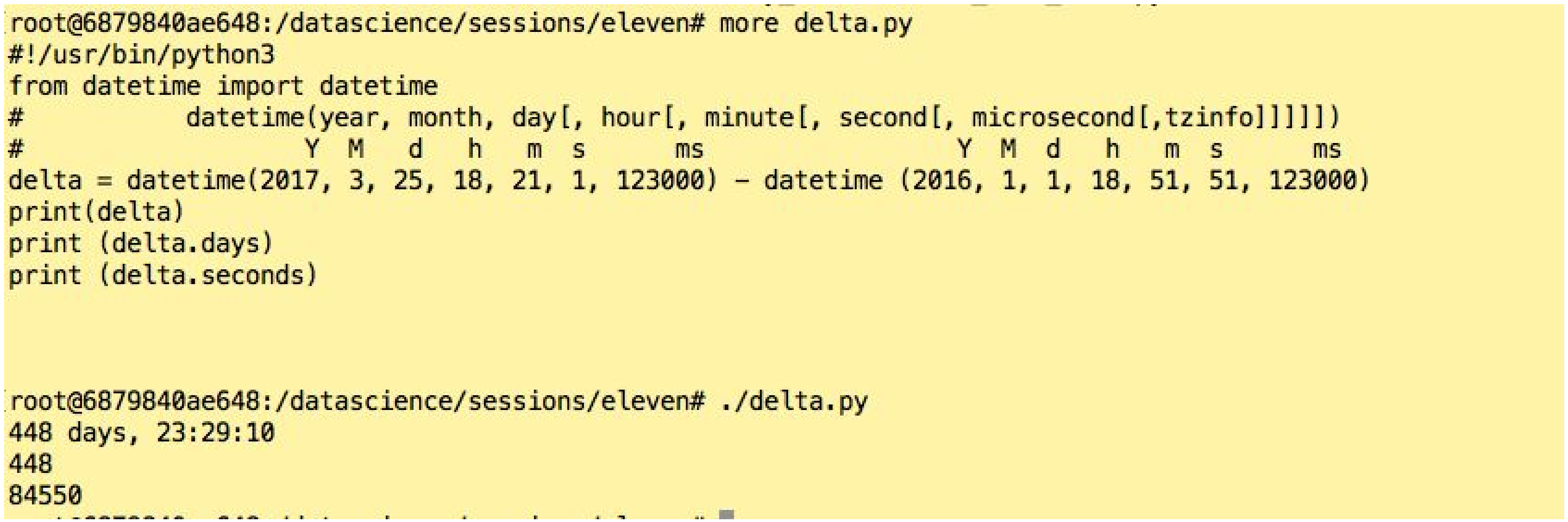
datetime - Library (Cont ...)

hello_datetime.py



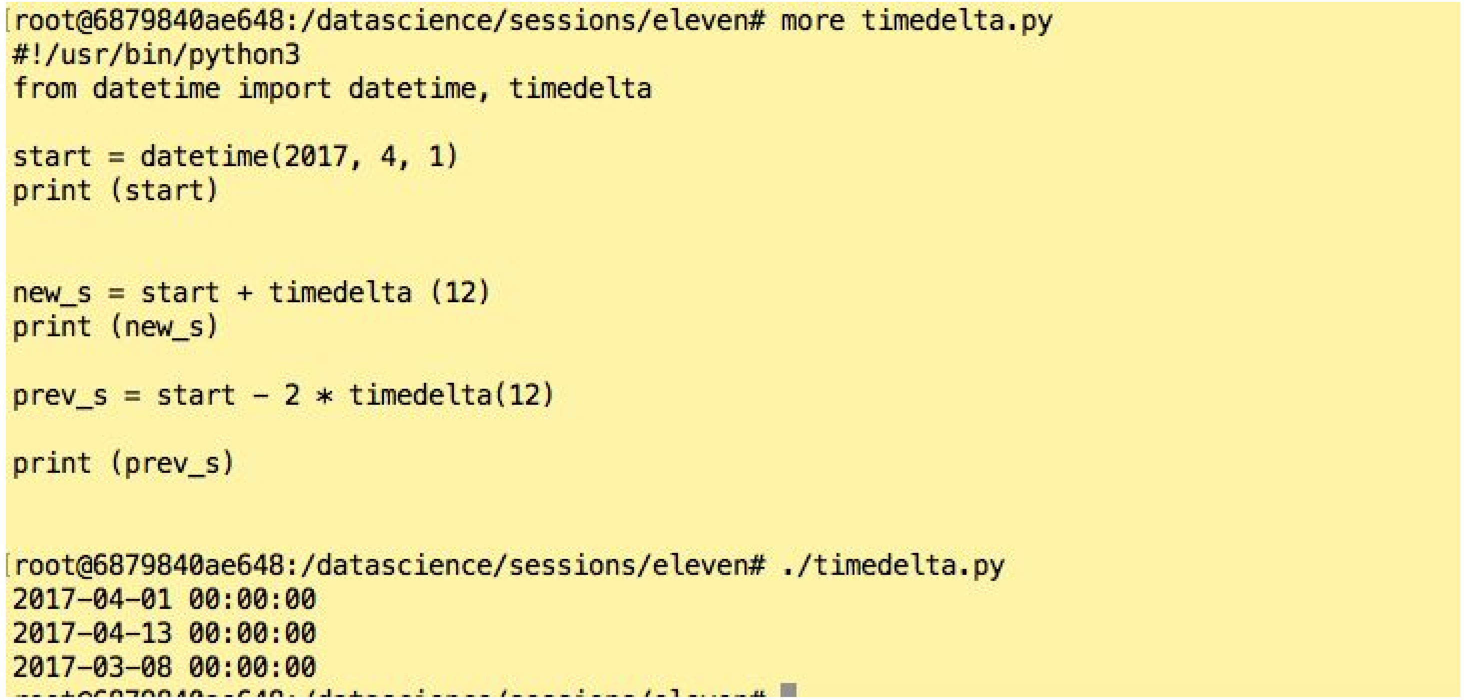
```
$ python hello_datetime.py
The dateformat: 2017-06-28 17:49:13.167208
The year is = 2017
The month is = 6
The day is = 28
The dateformat: 2017-03-01 17:09:21.832092
The year is = 2017
The month is = 3
The day is = 1
This is ...............25
```

Difference - datetime



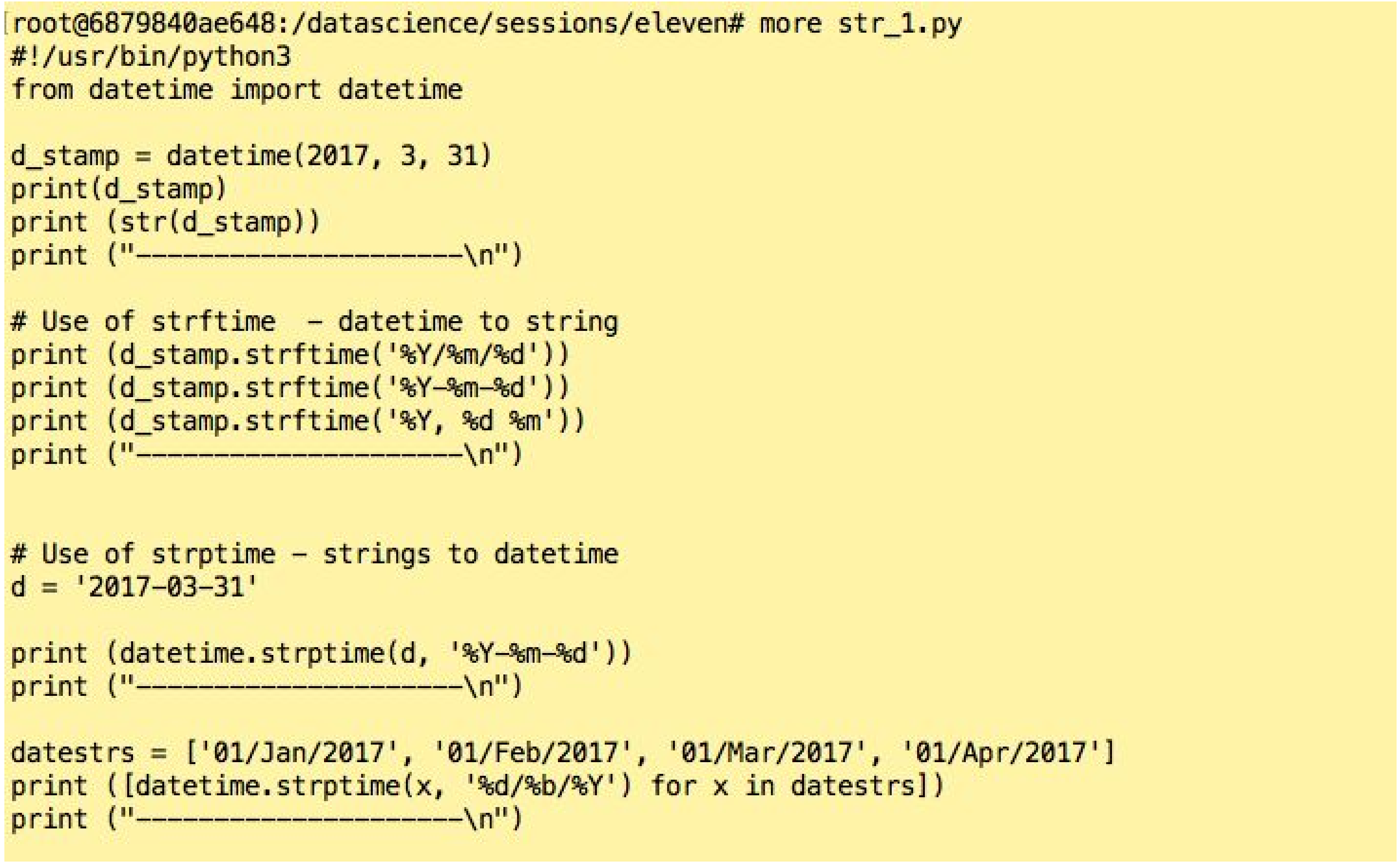
```
root@6879840ae648:/datascience/sessions/eleven# more delta.py
#!/usr/bin/python3
from datetime import datetime
#           datetime(year, month, day[, hour[, minute[, second[, microsecond[,tzinfo]]]]])
#                   Y  M   d   h   m  s     ms                  Y  M  d   h   m  s     ms
delta = datetime(2017, 3, 25, 18, 21, 1, 123000) - datetime (2016, 1, 1, 18, 51, 51, 123000)
print(delta)
print (delta.days)
print (delta.seconds)




root@6879840ae648:/datascience/sessions/eleven# ./delta.py
448 days, 23:29:10
448
84550
```

timedelta



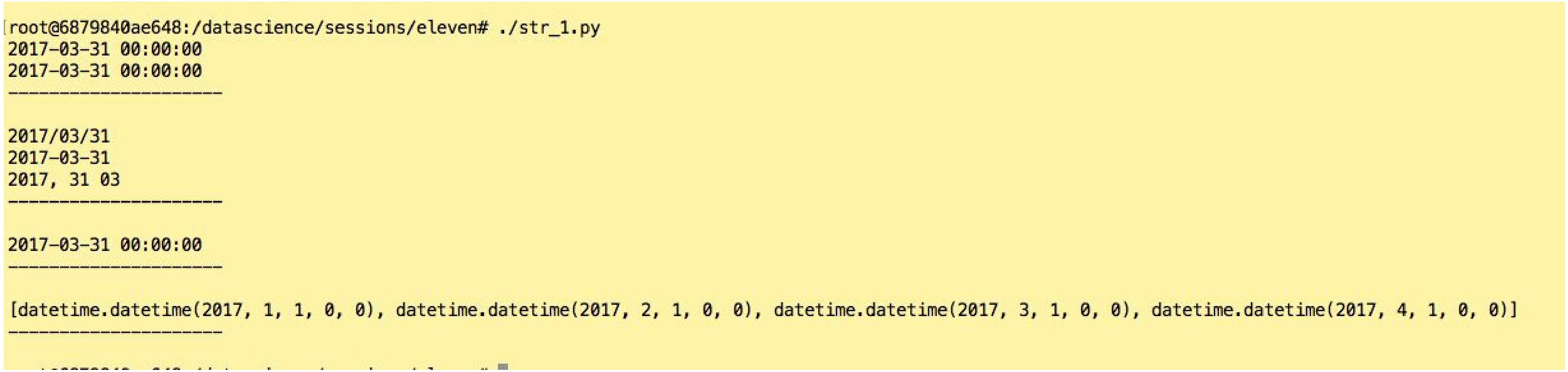
```
[root@6879840ae648:/datascience/sessions/eleven# more timedelta.py
#!/usr/bin/python3
from datetime import datetime, timedelta

start = datetime(2017, 4, 1)
print (start)


new_s = start + timedelta (12)
print (new_s)

prev_s = start - 2 * timedelta(12)

print (prev_s)


[root@6879840ae648:/datascience/sessions/eleven# ./timedelta.py
2017-04-01 00:00:00
2017-04-13 00:00:00
2017-03-08 00:00:00
```

strings and datetime



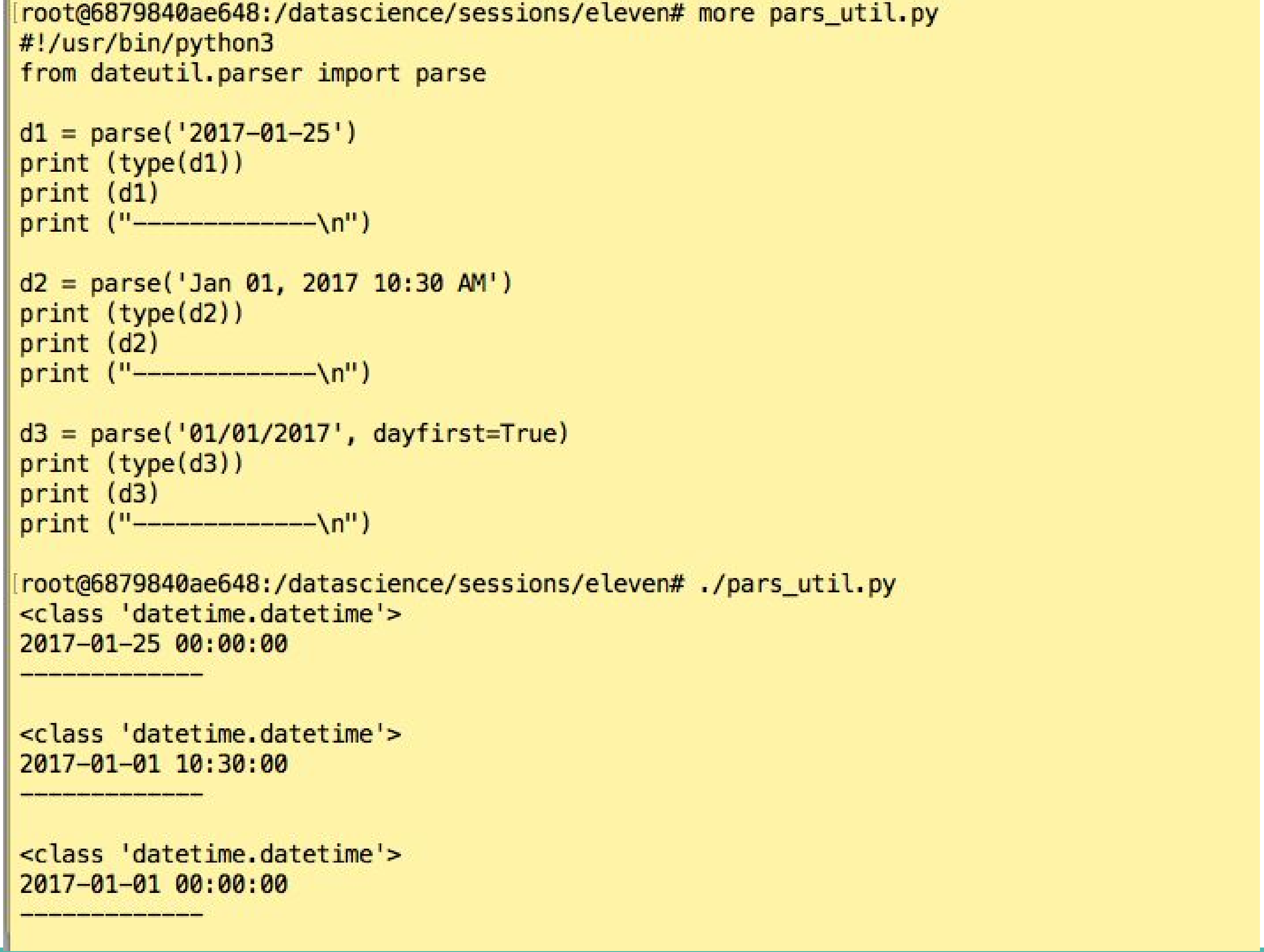
```
root@6879840ae648:/datascience/sessions/eleven# more str_1.py
#!/usr/bin/python3
from datetime import datetime

d_stamp = datetime(2017, 3, 31)
print(d_stamp)
print (str(d_stamp))
print ("---------------------\n")

# Use of strftime  - datetime to string
print (d_stamp.strftime('%Y/%m/%d'))
print (d_stamp.strftime('%Y-%m-%d'))
print (d_stamp.strftime('%Y, %d %m'))
print ("---------------------\n")


# Use of strptime - strings to datetime
d = '2017-03-31'

print (datetime.strptime(d, '%Y-%m-%d'))
print ("---------------------\n")

datestrs = ['01/Jan/2017', '01/Feb/2017', '01/Mar/2017', '01/Apr/2017']
print ([datetime.strptime(x, '%d/%b/%Y') for x in datestrs])
print ("---------------------\n")
```

strings and datetime (cont...)



```
[root@6879840ae648:/datascience/sessions/eleven# ./str_1.py
2017-03-31 00:00:00
2017-03-31 00:00:00
---------------------

2017/03/31
2017-03-31
2017, 31 03
---------------------

2017-03-31 00:00:00
---------------------

[datetime.datetime(2017, 1, 1, 0, 0), datetime.datetime(2017, 2, 1, 0, 0), datetime.datetime(2017, 3, 1, 0, 0), datetime.datetime(2017, 4, 1, 0, 0)]
---------------------
```

dateutil



```
root@6879840ae648:/datascience/sessions/eleven# more pars_util.py
#!/usr/bin/python3
from dateutil.parser import parse

d1 = parse('2017-01-25')
print (type(d1))
print (d1)
print ("------------\n")

d2 = parse('Jan 01, 2017 10:30 AM')
print (type(d2))
print (d2)
print ("------------\n")

d3 = parse('01/01/2017', dayfirst=True)
print (type(d3))
print (d3)
print ("------------\n")

root@6879840ae648:/datascience/sessions/eleven# ./pars_util.py
<class 'datetime.datetime'>
2017-01-25 00:00:00
------------

<class 'datetime.datetime'>
2017-01-01 10:30:00
------------

<class 'datetime.datetime'>
2017-01-01 00:00:00
------------
```

LAB Assignment

Q. 1 From TWiki_Application Log - find out the duration of log file.

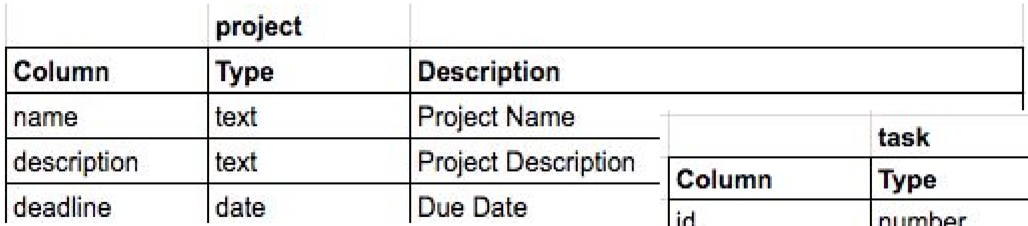
Q.2 Demonstration of dictionary - caching mechanism. Finding Factors!

Q.3 - Work on lab_three.py

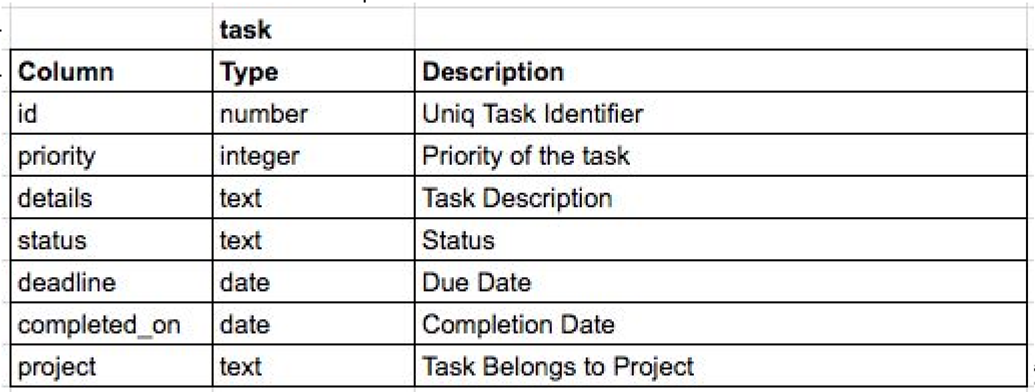
Q. 4. Work on lab_four.py

LAB Assignment

Q. 5 You need to create database table by “firstname_lastname” after connecting to database. We have provided “todo_schema.sql” in “assignment directory. Use that file to create tables described below.   Deliver the screen-shots of your work.



	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