Python and MySQL tutorial

Author: Cheng Nie
Check chengnie.com (http://www.chengnie.com/teaching) for the most recent version
Current Version: Feb 10, 2016

Python Setup

Since most students in this class use Windows 7, I will use Windows 7 for illustration of the setup. Setting up the environmnet in Mac OS and Linux should be similar. Please note that the code should produce the same results whichever operating system (even on your smart phone) you are using because Python is platform independent.

Download the Python 3.5 version of Anaconda that matches your operating system from this link (https://www.continuum.io/downloads). You can accept the default options during installation. To see if your Windows is 32 bit or 64 bit, check here (http://windows.microsoft.com/en-us/windows7/find-out-32-or-64-bit)

You can save and run this document using the Jupyter notebook (previously known as IPython notebook). Another tool that I recommend would be PyCharm (https://www.jetbrains.com/pycharm/), which has a free community edition.

This is a tutorial based on the official Python Tutorial for Python 3.5.1 (https://docs.python.org/3.5/tutorial/index.html). If you need a little more motivation to learn this programming language, consider reading this article (https://docs.python.org/3.5/tutorial/appetite.html).

Old ways of running Python code

- In Python shell together with text editor like notepad++ (https://notepad-plus-plus.org/) or VIM.

Numbers

In [ ]:
```
width = 20
height = 5*9
width * height
```

Calculator

In [ ]:
```
tax = 8.25 / 100
price = 100.50
price * tax
```

In [ ]:
```
price + _
```

In [ ]:
```
round(_, 2)
```

Strings

In [ ]:
```
print('spam email')
```

show ' and " in a string

In [ ]:
```
# This would cause error
print('doesn't')
```

In [ ]:
```
# One way of doing it correctly
print('doesn\'t')
```

In [ ]:
```
# Another way of doing it correctly
print("doesn't")
```

span multiple lines

In [ ]:
```
print('''
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
''')
```

In [ ]:
```
print('''Cheng highly recommends Python programming language''')
```

slice and index

In [ ]:
```
word = 'HELP' + 'A'
word
```

Index in the Python way

```
In [ ]: word[0]
```

```
In [ ]: word[4]
```

```
In [ ]: # endding index not included
        word[0:2]
```

```
In [ ]: word[2:4]
```

```
In [ ]: # length of a string
        len(word)
```

```
In [ ]: # first index default to 0 and second index default to the size
        word[:2]
```

```
In [ ]: # It's equivalent to
        word[0:2]
```

```
In [ ]: # Everything except the first two characters
        word[2:]
```

```
In [ ]: # It's equivalent to
        word[2:len(word)]
```

List

```
In [ ]: a = ['spam', 'eggs', 100, 1234]
        a
```

```
In [ ]: a[0]
```

```
In [ ]: a[3]
```

```
In [ ]: a[-2]
```

```
In [ ]: a[1:-1]
```

```
In [ ]: a[:2] + ['bacon', 2*2]
```

```
In [ ]: 3*a[:3] + ['Boo!']
```

mutable

```
In [ ]: a
```

```
In [ ]: a[2] = a[2] + 23
        a
```

versatile features

```
In [ ]: # Replace some items:
        a[0:2] = [1, 12]
        a
```

```
In [ ]: # Remove some:
        a[0:2] = [] # or del a[0:2]
        a
```

```
In [ ]: # Insert some:
        a[1:1] = ['insert', 'some']
        a
```

```
In [ ]: # inserting at one position is not the same as changing one element
        # a=[1, 12, 100, 1234]
        a = [123, 1234]
        a[1] = ['insert', 'some']
        a
```

Nest lists

```
In [ ]: q = [2, 3]
        p = [1, q, 4]
        p
```

```
In [ ]: len(p)
```

```
In [ ]: p[1]
```

```
In [ ]: p[1][0]
```

tuple

similar to list, but immutable (element cannot be changed)

```
In [ ]:
```

In [ ]:
```
x=(1,2,3,4)
x[0]=7 # it will raise error since tuple is immutable
x[0]
```

dict

In [ ]:
```
tel = {'jack': 4098, 'sam': 4139}
tel['dan'] = 4127
tel
```

In [ ]:
```
tel['jack']
```

In [ ]:
```
del tel['sam']
tel
```

In [ ]:
```
tel['mike'] = 4127
tel
```

In [ ]:
```
# tel.keys()
# tel.values()

'dan' in tel
```

In [ ]:
```
for key in tel:
    print('key:', key, '; value:', tel[key])
```

Control of flow

if

Ask a user to input a number, if it's negative, x=0, else if it's 1

In [ ]:
```
x = int(input("Please enter an integer for x: "))
if x < 0:
     x = 0
     print('Negative; changed to zero')
elif x == 0:
     print('Zero')
elif x == 1:
     print('Single')
else:
     print('More')
```

while

Fibonacci series: the sum of two elements defines the next with the first two elements to be 0 and 1.

```
In [ ]: a, b = 0, 1 # multiple assignment
        while a < 10:
         print(a)
         a, b = b, a+b
```

for

```
In [ ]: # Measure some strings:
        words = ['cat', 'window', 'defenestrate']
        for i in words:
            print(i, len(i))
```

Define function

```
In [2]: def fib(n):    # write Fibonacci series up to n
            """Print a Fibonacci series up to n."""
            a, b = 0, 1
            while a < n:
                print(a)
                a, b = b, a+b
```

```
In [ ]: fib(200)
```

```
In [ ]: fib(2000000000000000) # do not need to worry about the type of a,b
```

Data I/O

Create some data in Python and populate the database with the created data. We want to create a table with 3 columns: id, name, and age to store information about 50 kids in a day care.

The various modules that extend the basic Python funtions are indexed here (https://docs.python.org/3.5/py-modindex.html).

```
In [ ]: # output for viewing first

        import string
        import random

        # fix the pseudo-random sequences for easy replication
        # It will generate the same random sequences of nubmers/letters with th
        random.seed(123)

        for i in range(50):
        # Data values separated by comma(csv file)
            print(i+1,random.choice(string.ascii_uppercase),random.choice(range
```

```
In [ ]: # write the data to a file
        random.seed(123)
        out_file=open('data.csv','w')
        columns=['id','name','age']
        out_file.write(','.join(columns)+'\n')
        for i in range(50):
            row=[str(i+1),random.choice(string.ascii_uppercase),str(random.choi
            out_file.write(','.join(row)+'\n')
        else:
            out_file.close()
```

MySQL

Install MySQL 5.7 following this link (https://dev.mysql.com/downloads/workbench/). You might also need to install the prerequisites listed here (http://dev.mysql.com/resources/wb62_prerequisites.html) before you can install the workbench.

The documentation for MySQL is here (http://dev.mysql.com/doc/refman/5.7/en/).

You can accept the default options during installation. Later, you will connect to MySQL using the password you set during the installation.

To get comfortable with it, you might find this tutorial (http://www.w3schools.com/sql/) of Structured Query Language(SQL) to be helpful.

```
show databases;
These commands are executed in MySQL query tab, not in Python.

In mysql, you need to end all commands with ;


----------------------- In MySQL ------------------

show databases;

create database test;

use test;

show tables;

create table example(
id int not null,
name varchar(30),
age tinyint,
primary key(id));

show tables;

desc example;

load data local infile
"C:\\Users\\cnie\\Dropbox\\Sp16_TA\\Sarkar\\MIS\ 7310
KM\\2016\\python3\\data.csv" into table test.example FIELDS
TERMINATED BY ',' lines terminated by '\r\n' ignore 1 lines;
```

More about index

Target: "HLA", select every other character

In [ ]:
```
# start: end: step
word[0::2]
```

In [ ]:
```
word[0:len(word):2]
```

Negative index

In [ ]:
```
word[-1]     # The last character
```

In [ ]:
```

```

```
In [ ]: word[-2]    # The last-but-one character
        word[-2:]    # The last two characters
```

```
In [ ]: word[:-2]    # Everything except the last two characters
```

More about list

Target: Get the third power of integers between 0 and 10.

```
In [ ]: # loop way
        cubes = []
        for x in range(11):
                cubes.append(x**3)
        cubes
```

```
In [ ]: # map way
        def cube(x):
            return x*x*x

        list(map(cube, range(11)))
```

```
In [ ]: # list comprehension way
        x_list=[x**3 for x in range(11)]
        x_list
```

Use `if` in list comprehension

Target: find the even number below 10

```
In [3]: result = []
        for i in range(11):
            if i%2 == 0:
                result.append(i)
        else:
            print(result)

        [0, 2, 4, 6, 8, 10]
```

```
In [4]: [i for i in range(11) if i%2==0]

Out[4]: [0, 2, 4, 6, 8, 10]
```

```
In [ ]: l=[1,3,5,6,8,10]
        [i for i in l if i%2==0]
```

Interact MySQL with Python

Since the official MySQL 5.7.11 provides support for Python upto Version 3.4, we need to install a package to provide to support the Python 3.5. Execute the following line to install it.

In [ ]:
```
!conda  install mysql-connector-python
```

In [ ]:
```
#
# ----------------------- In Python ------------------

# access table from Python

# connect to MySQL in Python
import mysql.connector
cnx = mysql.connector.connect(user='root',password='pythonClass',databa
# All DDL (Data Definition Language) statements are executed using a ha
cursor = cnx.cursor()

#cursor.execute("")
# write the same data to the example table
query0 = '''insert into example (id, name, age) \
values ({id_num},"{c_name}",{c_age});'''
random.seed(123)
for i in range(50):
    query1 = query0.format(id_num = i+1,
                           c_name = random.choice(string.ascii_uppercas
                           c_age = random.choice(range(6)))
    print(query1)
    cursor.execute(query1)
    cnx.commit()
```

To get better understanding of the table we just created. We will use MySQL command line again.

```
----------------------- In MySQL ------------------

# To get the number of records.
select count(*) from example;

# To get age histgram:
select distinct age, count(*) from example group by age;

# create a copy of the example table for modifying.
drop table if exists e_copy;
create table e_copy select * from example;
alter table e_copy add primary key(id);
```

```
insert into e_copy (id, name, age) values (null,'P',6);
insert into e_copy (id, name, age) values (3,'P',6);
insert into e_copy (id, name, age) values (51,'P',6);
insert into e_copy (id, name, age) values (52,'Q',null);
insert into e_copy (id, name, age) values (54,'Q',null),
(55,'Q',null);
insert into e_copy (id, name) values (53,'Q');
update e_copy set age=7 where id=53;
update e_copy set age=DEFAULT where id=53;
update e_copy set age='' where id=53;

select sum(age) from e_copy;
select avg(age) from e_copy;
select * from e_copy where age<3 and name<="C";
```

In [ ]:

```
#
# ----------------------- In Python ------------------
#
cursor.execute('select * from e_copy;')
for i in cursor:
    print(i)
```

In [ ]:

```
#
# ----------------------- In Python ------------------
#
# # example for adding new info for existing record
# cursor.execute('alter table e_copy add mother_name varchar(1) default
query='update e_copy set mother_name="{m_name}" where id={id_num};'
# random.seed(333)

for i in range(50):
    query1=query.format(m_name = random.choice(string.ascii_uppercase),
    print(query1)
    cursor.execute(query1)
    cnx.commit()
```

```
In [ ]: #
        # ----------------------- In Python ------------------
        #

        # example for insert new records
        query2='insert into e_copy (id, name,age,mother_name) \
        values ({id_num},"{c_name}",{c_age},"{m_name}")'
        for i in range(10):
            query3=query2.format(id_num = i+60,
                                 c_name = random.choice(string.ascii_uppercase)
                                 c_age = random.randint(0,6),
                                 m_name = random.choice(string.ascii_uppercase)
            print(query3)
            cursor.execute(query3)
            cnx.commit()

        # check if you've updated the data successfully in MySQL
```

Regular expression in Python

```
In [1]: import re
        # digits
        # find all the numbers
        infile=open('digits.txt','r')
        content=infile.read()
        print(content)
```

```
Sam says 87=66
Lucy says 71=32
20=21, said John
85=78
65=34
86=32
54=86
40=82
49=40
Cheng says 95=47
```

```
In [2]: # Find all the numbers in the file
        numbers=re.findall(r'\d+',content)
        for n in numbers:
                print(n)
```

```
87
66
71
32
20
21
85
78
65
34
86
32
54
86
40
82
49
40
95
47
```

```
In [3]: # find equations
        equations=re.findall(r'(\d+)=\d+',content)
        for e in equations:
                print(e)
```

```
87
71
20
85
65
86
54
40
49
95
```

```
In [4]: # subsitute equations to correct them
        print(re.sub(r'(\d+)=\d+',r'\1=\1',content))
```

```
Sam says 87=87
Lucy says 71=71
20=20, said John
85=85
65=65
86=86
54=54
40=40
49=49
Cheng says 95=95
```

```
In [ ]: # Save to file
        print(re.sub(r'(\d+)=\d+',r'\1=\1',content), file = open('digits_correc
```

Crawl the reviews for UT Dallas at Yelp.com

```
In [ ]: # crawl_UTD_reviews.py
        # Author: Cheng Nie
        # Email: me@chengnie.com
        # Date: Feb 8, 2016

        from urllib.request import urlopen
        from time import sleep
        from random import randint

        num_pages = 2
        reviews_per_page = 20
        out_file = open('UTD_reviews.txt', 'w')
        url = 'http://www.yelp.com/biz/university-of-texas-at-dallas-richardson
        review_start_pattern = '<div class="review-wrapper">'
        rating_pattern = '<i class="star-img stars_'
        date_pattern = '"datePublished" content="'
        reviews_count = 0

        for page in range(num_pages):

            print('processing page', page)
            html = urlopen(url.format(start_number = page * reviews_per_page))
            page_content = html.read().decode('utf-8')
            review_start = page_content.find(review_start_pattern)

            while review_start != -1:
                # it means there are more reviews to be crawled
                reviews_count += 1

                # get the rating
                cut_front = page_content.find(rating_pattern, review_start) + l
                cut_end = page_content.find('" title="', cut_front)
                rating = page_content[cut_front:cut_end]

                # get the date
                cut_front = page_content.find(date_pattern, cut_end) + len(date
                cut_end = page_content.find('">', cut_front)
                date = page_content[cut_front:cut_end]

                # save the data into out_file
                out_file.write(','.join([rating, date]) + '\n')

                review_start = page_content.find(review_start_pattern, cut_end)

            print('crawled', reviews_count, 'reviews so far')


        out_file.close()
```

In [ ]:

```
# Another way to run a piece of code
# You can save the code in the cell above into a text file named
# crawl_UTD_reviews.py in the same directory as this Jupyter notebook.
# Then you can run the Python code from this cell
%run crawl_UTD_reviews.py
```