notebook.community

EDIT AND RUN

Python and MySQL tutorial

Author: Cheng Nie

Check chengnie.com for the most recent version

Current Version: Feb 18, 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. You can accept the default options during installation. To see if your Windows is 32 bit or 64 bit, check here

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

This is a tutorial based on the official Python Tutorial for Python 3.5.1. If you need a little more motivation to learn this programming language, consider reading this article.

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)
```

List

In [ ]:

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

In [ ]:

```
a[0]
```

In [ ]:

```
a[3]
```

```
In [ ]:

a[2:4]
```

```
In [ ]:

sum(a[2:4])
```

Built-in functions like `sum` and `len` are explained in the document too. Here is a link to it.

Mutable

```
In [ ]:

a
```

```
In [ ]:

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

x=(1,2,3,4)
x[0]
```

```
In [ ]:

x[0]=7 # it will raise error since tuple is immutable
```

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 [ ]:

```
# Is dan in the dict?
'dan' in tel
```

In [ ]:

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

In [ ]:

```
import collections
od = collections.OrderedDict(sorted(tel.items()))
```

In [ ]:

```
od
```

Quiz: how to print the tel dict sorted by the 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 [ ]:

```
# multiple assignment to assign two variables at the same time
a, b = 0, 1
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))
```

Crawl the reviews for UT Dallas at Yelp.com

The University of Texas at Dallas is reviewed on Yelp.com. It shows on this page that it attracted 38 reviews so far from various reviewers. You learn from the webpage that Yelp displays at most 20 recommended reviews per page and we need to go to page 2 to see the review 21 to review 38. You notice that the URL in the address box of your browser changed when you click on the Next page. Previouly, on page 1, the URL is:

http://www.yelp.com/biz/university-of-texas-at-dallas-richardson

On page 2, the URL is:

http://www.yelp.com/biz/university-of-texas-at-dallas-richardson?start=20

You learn that probably Yelp use this `?start=20` to skip(or `offset` in MySQL language) the first 20 records to show you the next 18 reviews. You can use this pattern of going to the next page to enumerate all pages of a business in Yelp.com.

In this exmaple, we are going get the rating (number of stars) and the date for each of these 38 reviews.

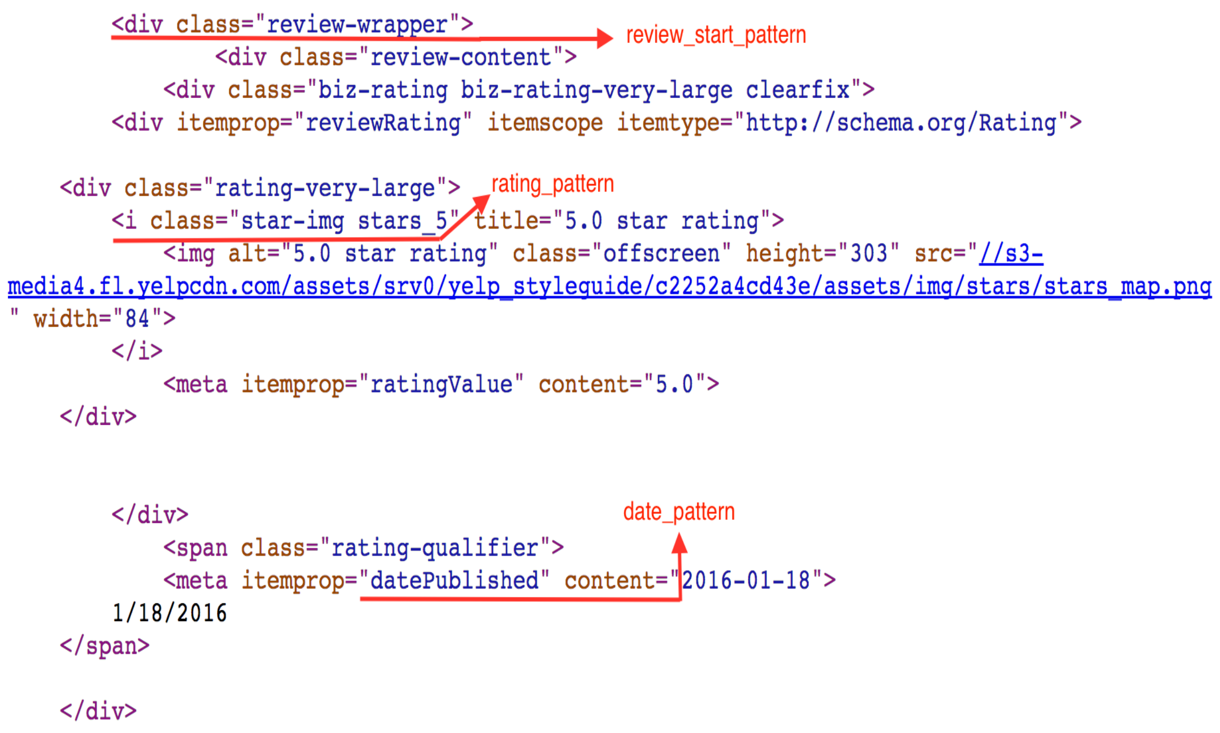
The general procedure to crawl any web page is the following:

1. Look for the string patterns proceeding and succeeding the information you are looking for in the source code of the page (the html file).
2. Write a program to enumerate (`for` or `while` loop) all the pages.

For this example, I did a screenshot with my annotation to illustrate the critical patterns in the Yelp page for UTD reviews.

- `review_start_pattern` is a variable to stroe the string of `'<div class="review-wrapper">'` to locate the beginning of an individual review.
- `rating_pattern` is a variable to stroe the string of `'<i class="star-img stars_'` to locate the rating.
- `date_pattern` is a variable to stroe the string of `'"datePublished" content="'` to locate date of the rating.

It takes some trails and errors to figure out what are good string patterns to use to locate the information you need in an html. For example, I found that `'<div class="review-wrapper">'` appeared exactly 20 times in the webpage, which is a good indication that it corresponds to the 20 individual reviews on the page (the `review-wrapper` tag seems to imply that too).



```
        <div class="review-wrapper">
                <div class="review-content">
            <div class="biz-rating biz-rating-very-large clearfix">
        <div itemprop="reviewRating" itemscope itemtype="http://schema.org/Rating">

    <div class="rating-very-large">
        <i class="star-img stars_5" title="5.0 star rating">
            <img alt="5.0 star rating" class="offscreen" height="303" src="//s3-media4.fl.yelpcdn.com/assets/srv0/yelp_styleguide/c2252a4cd43e/assets/img/stars/stars_map.png" width="84">
        </i>
            <meta itemprop="ratingValue" content="5.0">
    </div>


        </div>
            <span class="rating-qualifier">
            <meta itemprop="datePublished" content="2016-01-18">
        1/18/2016
    </span>

    </div>
```

```
In [ ]:

# crawl_UTD_reviews
# Author: Cheng Nie
# Email: me@chengnie.com
# Date: Feb 8, 2016
# Updated: Feb 12, 2016

from urllib.request import urlopen
```

```
num_pages = 2
reviews_per_page = 20
# the file we will save the rating and date
out_file = open('UTD_reviews.csv', 'w')
# the url that we need to locate the page for UTD reviews
url = 'http://www.yelp.com/biz/university-of-texas-at-dallas-\
richardson?start={start_number}'
# the three string patterns we just explained
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)

    # open the url and save the source code string to page_content
    html = urlopen(url.format(start_number = page * reviews_per_page))
    page_content = html.read().decode('utf-8')

    # locate the beginning of an individual review
    review_start = page_content.find(review_start_pattern)

    while review_start != -1:
        # it means there at least one more review to be crawled
        reviews_count += 1

        # get the rating
        cut_front = page_content.find(rating_pattern, review_start) \
                    + len(rating_pattern)
        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_pattern)
        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()
```

You can save the code in the cell above into a notebook file named crawl_BOM.ipynb in the same directory as this Jupyter notebook. Then you can run the Python code from that cell in the new notebook. You can modify it for your homework submission.

Define function

```
In [ ]:

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.

```
In [ ]:
```

```
# output for eyeballing the data

import string
import random

# fix the pseudo-random sequences for easy replication
# It will generate the same random sequences
# of nubmers/letters with the same seed.
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(6)), sep=',')
```

```
In [ ]:

# write the data to a file called data.csv
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.choice(range(6)))]
    out_file.write(','.join(row)+'\n')
else:
    out_file.close()
```

```
In [ ]:

# load data back into Python
for line in open('data.csv', 'r'):
    print(line)
```

```
In [ ]:

# To disable to the new line added for each print
# use the end parameter in print function
```

```
for line in open('data.csv', 'r'):
    print(line, end = '')
```

MySQL

Install MySQL 5.7 Workbench first following this link. You might also need to install the prerequisites listed here before you can install the Workbench. The Workbench is an interface to interact with MySQL database. The actual MySQL database server requires a second step: run the MySQL Installer, then add and intall the MySQL servers using the Installer. You can accept the default options during installation. Later, you will connect to MySQL using the password you set during the installation and configuration. I set the password to be `pythonClass`.

The documentation for MySQL is here.

To get comfortable with it, you might find this tutorial of Structured Query Language(SQL) to be helpful.

```
In [ ]:

These commands are executed in MySQL query tab, not in Python.

In mysql, you need to end all commands with ;

#
# ----------------------- In MySQL ------------------

# display the database
show databases;

# create a database named test
create database test;

# choose a database for future commands
use test;

# display the tables in test database
show tables;

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

# now we should have the example table
show tables;

# how was the table example defined again?
desc example;

# is there anything in the example table?
select * from example;

# import csv file into MySQL database
load data local infile "C:\\Users\\cxn123430\\Downloads\\data.csv" into table

# is there anything now?
select * from example;

# drop the table
drop table example;

# does the example table still exist?
show tables;
```

Quiz: import the crawled Yelp review file `UTD_reviews.csv` into a table in your database.

Use Python to access MySQL database

Since the official MySQL 5.7 provides support for Python upto Version 3.4 as of writing this tutorial, we need to install a package named `mysql-connector-python` to provide support for the cutting-edge Python 3.5. Execute the following line in Windows command line to install it. This is relatively easy since you have the Anancoda installed. We can use the `conda` command to intall that package in the Windows command line.

In [ ]:

```
#
# ---------------------- In Windows command line(cmd) ------------------


conda  install mysql-connector-python
```

Remember that we use Python to save 50 kids' infomation into a csv file named `data.csv` first and then use the `load` command in MySQL to import the data? We don't actually need to save the `data.csv` file to hard disk. And we can "load" the same data into database without leaving Python.

```
In [ ]:

#
# ----------------------- In Python ------------------

# access table from Python

# connect to MySQL in Python
import mysql.connector
cnx = mysql.connector.connect(user='root',
                              password='pythonClass',
                              database='test')
# All DDL (Data Definition Language) statements are
# executed using a handle structure known as a cursor
cursor = cnx.cursor()

# create a table named example
cursor.execute('''create table example(
id int not null,
name varchar(30),
age tinyint,
primary key(id));''')
cnx.commit()

# write the same data  to the example table without saving a csv file
query0_template = '''insert into example (id, name, age) \
values ({id_num},"{c_name}",{c_age});'''
random.seed(123)
for i in range(50):
    query0 = query0_template.format(id_num = i+1,
                           c_name = random.choice(string.ascii_uppercase),
                           c_age = random.choice(range(6)))
    print(query0)
    cursor.execute(query0)
    cnx.commit()
```

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

```
In [ ]:

#
# ----------------------- In MySQL ------------------

# To get the totoal 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.
create table e_copy select * from example;
select * from e_copy;

# note that the primary key is not copied to the e_copy table
desc e_copy;

# add the primary key to e_copy table using the alter command
alter table e_copy add primary key(id);

# is it done correctly?
desc e_copy;

# does MySQL take the primary key seriously?
insert into e_copy (id, name, age) values (null,'P',6);
insert into e_copy (id, name, age) values (3,'P',6);

# alright, let's insert something else
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,'S',null),(55,'T',null);
insert into e_copy (id, name) values (53,'R');

# who is the child with id of 53?
select * from e_copy where id = 53;

# update the age for this child.
update e_copy set age=3 where id=53;
select * from e_copy where id = 53;

# what's inside the table now?
select * from e_copy;
```

Again, you can actually do everything in Python without going to the MySQL workbench.

```
In [ ]:

#
# ----------------------- In Python ------------------

# query all the content in the e_copy table
cursor.execute('select * from e_copy;')
for i in cursor:
    print(i)
```

Now we want to add one new column of `mother_name` to record the mother's name for each child in the child care.

```
In [ ]:

#
# ----------------------- In Python ------------------



# # example for adding new info for existing record
cursor.execute('alter table e_copy add mother_name varchar(1) default null')
cnx.commit()

query1_template='update e_copy set mother_name="{m_name}" where id={id_num};'
random.seed(333)

for i in range(55):
    query1=query1_template.format(m_name = random.choice(string.ascii_upperca
    print(query1)
    cursor.execute(query1)
    cnx.commit()
```

```
In [ ]:

#
# ----------------------- In Python ------------------
# example for insert new records
query2_template='insert into e_copy (id, name,age,mother_name) \
values ({id_num},"{c_name}",{c_age},"{m_name}")'
for i in range(10):
    query2=query2_template.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(query2)
    cursor.execute(query2)
    cnx.commit()
```

Check if you've updated the data successfully in MySQL database from Python

```
In [ ]:

#
# ----------------------- In Python ------------------

# query all the content in the e_copy table
cursor.execute('select * from e_copy;')
for i in cursor:
    print(i)
```

```
In [ ]:

#
# ----------------------- In MySQL ------------------

Use the GUI to export the database into a self-contained file (the extension
```

Regular expression in Python

Before you run this part, you need to download the digits.txt and spaces.txt files to the same folder as this notebook

What's in the `digits.txt` file?

```
In [ ]:

import re
infile=open('digits.txt','r')
content=infile.read()
print(content)
```

How can I find all the numbers in a file like `digits.txt` ?

```
In [ ]:

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

How can I find all the equations?

```
In [ ]:

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

The equations seem to be incorrect, how can I correct them without affecting other text information?

```
In [ ]:

# subsitute equations to correct them
# use the left hand side number
print(re.sub('(\d+)=\d+','\1=\1',content))
```

```
In [ ]:
```

```
# another way to subsitute equations to correct them
# use the right hand side number
print(re.sub('\d+=(\d+)','\\1=\\1',content))
```

In [ ]:

```
# Save to file
print(re.sub('(\d+)=\d+','\\1=\\1',content), file = open('digits_corrected.tx
```

Preprocessing a text file with various types of spaces.

In [ ]:

```
infile=open('spaces.txt','r')
content=infile.read()
print(content)
```

In [ ]:

```
print(re.sub('[\t ]+','\t',content))
```

In [ ]:

```
print(re.sub('[\t ]+','\t',content), file = open('spaces_corrected.txt', 'w')
```

More about index

In [ ]:

```
word = 'HELP' + 'A'
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)]
```

How about selecting every other character?

In [ ]:

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

In [ ]:

```
# It's equivalent to
word[0:len(word):2]
```

Negative index

In [ ]:

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

In [ ]:

```
word[-2]     # The last-but-one character
```

In [ ]:

```
word[-2:]    # The last two characters
```

In [ ]:

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

More about list

In [ ]:

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

In [ ]:

```
a[-2]
```

```
In [ ]:

a[1:-1]
```

```
In [ ]:

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

```
In [ ]:

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

Versatile features of a list

```
In [ ]:

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

```
In [ ]:

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

```
In [ ]:

# create some copies for change
b = a.copy()
c = a.copy()
```

```
In [ ]:

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

```
In [ ]:

# inserting at one position is not the same as changing one element
c[1] = ['insert', 'some']
c
```

How to 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**3 for x in range(11)]
```

Target: find the even number below 10

```
In [ ]:

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

```
In [ ]:

# Use if in list comprehension
[i for i in range(11) if i%2==0]
```

```
In [ ]:

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

What's next?

Online help by searching in Google

- Sort dict by value

More topics

- Jupyter Notebook
- sqlite3
- Debug

- BeautifulSoup
- Scikit-learn for machine learning
- networkx for social network analysis
- matplotlib for Python graphics
- Books published in Jupyter notebooks

Books that I recommend

All three books are available online legally through the UTD library. The Amazon links are given because the reviews might help you decide wheter or not to read them.

- Python Essential Reference (4th Edition)
- Web Scraping with Python
- Python for Data Analysis

Courses that I recommend

- Google Python Class
- Programming for Everybody (Getting Started with Python) University of Michigan
- Code Academy course

Content source: cniedotus/Python_scrape

Similar notebooks:

- Python3_tutorial
- Python3_tutorial-checkpoint
- Python docker
- python3_vs_python2
- 2016-03-22-Shallow-and-Deep-Copy
- 2016-05-01-Class-and-Metaclass-i
- how_to_define_function_module_class

- Ders-01-Programlamaya-Giris
- Python Basic Lesson 01 - 简介
- 02_python_tutorial_basics2