```
'''
A simple web crawler base on 'wget' and 'BeautifulSoup'
https://code.google.com/p/simple-web-crawler/source/browse/crawler.py
'''

#!/usr/bin/evn python
#coding=utf-8

import os
import sys, time
import logging
from bs4 import BeautifulSoup as BS

class Crawler(object):
    def __init__(self, argv):
     default_maxurls = 5
     self.depth = None
     self.outputDir = None
     self.maxurls = None
     self.seed_file = None

     self.seed_list = []
     self.next_round_seed_list = []

     self.count = 1
     self.domain = None
     self.crawl_url_list_file = 'crawl_url_list'

     self.log_file = 'crawler.log'
         self.logger = self.initLogger()
     while 1:
         if argv[0] == '-d' or argv[0] == '-depth':
             self.depth = int(argv[1])
             elif argv[0] == '-s' or argv[0] == '-seed':
             self.seed_file = argv[1]
             elif argv[0] == '-m' or argv[0] == '-max':
             self.maxurls = int(argv[1])
         elif argv[0] == '-o' or argv[0] == '-output':
             self.outputDir = argv[1]
         if os.path.exists(self.outputDir) is False:
             self.logger.critical('Specified dir name does not exist, exit!')
             sys.exit()
         elif argv[0] == '-r' or argv[0] == '-region':
             self.domain = self.chopDomainName(argv[1])
         argv = argv[2:]
         if len(argv) == 0:
             break

     if self.outputDir is None:
         self.logger.critical('output_directory is required, exit!')
         sys.exit()
     elif self.seed_file is None:
         self.logger.critical('seed_list is required, exit!')
```

```
        sys.exit()
    if self.maxurls is None:
        self.maxurls = default_maxurls

    self.getOriginalSeedList()

def initLogger(self):
logger = logging.getLogger()
fhandler = logging.FileHandler(self.log_file)
formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')
fhandler.setFormatter(formatter)
    logger.addHandler(fhandler)
logger.setLevel(logging.DEBUG)

return logger

def getHtmlContent(self, url):
 output_file_name = self.outputDir + os.sep + str(self.count) + '.html'
 try:
     comd = 'wget --html-extension -O %s %s'%(output_file_name, url)
     os.system(comd)
 except:
     pass
 if os.path.exists(output_file_name) is False:
     return None
 else:
     fp = open(output_file_name, 'r')
     content = fp.read()
     fp.close()
     self.count += 1
     return content

def chopDomainName(self, url):
try:
        if 'https://' in url:
            url = url.replace('https://','')
        elif 'http://' in url:
            url = url.replace('http://','')
        domain_name = url.split('/')[0].strip()
    return domain_name
except:
    self.logger.error('cannot chop domain name from url %s'%url)
    sys.exit()

def getDomainName(self, url):
try:
    domain_name = self.chopDomainName(url)
    self.logger.info('Get domain name is: %s'%domain_name)
    if self.domain is not None:
        if self.domain == domain_name:
            return True
        else:
            return False
```

```
        else:
            return True
    except:
        self.logger.error('cannot extract domain name from url %s'%url)
        sys.exit()

    def getNextRoundSeed(self, page_content):
     if page_content is None:
         return []
     seed_list = []
     bs = BS(page_content)
         for href in bs.find_all('a'):
         link = href.get('href')
         if link.startswith('http') is False:
         continue
         self.logger.info('Get link: %s'%link)
         if self.getDomainName(link):
         seed_list.append(link)
         return seed_list

    def getOriginalSeedList(self):
     assert self.seed_file is not None
     with open(self.seed_file, 'r') as fp:
             self.seed_list = fp.readlines()

    def crawl(self):
     next_round_seed_list = []
     terminated = False
         fpo = open(self.crawl_url_list_file, 'w')
         depth = 1
     while self.count <= self.maxurls:
         for seed in self.seed_list:
             seed = seed.rstrip()
             page_content = self.getHtmlContent(seed)
             if page_content is not None:
             fpo.write('%s\n'%seed)
             if self.count > self.maxurls:
                 terminated = True
                 break
             next_round_seed_list.extend(self.getNextRoundSeed(page_content))

         depth += 1
         if self.depth is not None and depth > self.depth:
             break
         if terminated is True:
         break
         self.seed_list = next_round_seed_list
         fpo.close()

# Just for testing
if __name__ == '__main__':
    if len(sys.argv)==1 or len(sys.argv[1:])%2!=0:
        print 'Usage: python %s -d/-depth depth_num -s/-seed seed_file -m/-max maxurl_num
```

```
        -r/-region domain_name -o/-output output_directory'%(__file__)
    sys.exit()
    obj = Crawler(sys.argv[1:])
    obj.crawl()
```