```
=============== cmdline_params.py ===================
#!/usr/bin/python

import sys, getopt
def main(argv):
    inputfile = ''
    outputfile = ''
    try:
        opts, args = getopt.getopt(argv,"hi:o:",["ifile=","ofile="])
    except getopt.GetoptError:
        print 'test.py -i <inputfile> -o <outputfile>'
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print 'test.py -i <inputfile> -o <outputfile>'
            sys.exit()
        elif opt in ("-i", "--ifile"):
            inputfile = arg
        elif opt in ("-o", "--ofile"):
            outputfile = arg
    print 'Input file is :', inputfile
    print 'Output file is :', outputfile

if __name__ == "__main__":
    main(sys.argv[1:])


=============== conditional_if.py ===================
#if statement
#Write a function that includes each of the following:

#    An if statement with an elif clause and an else clause.
#    Write several if statements that use the logical operators: and, or, and not. Or, maybe a single
if statement that uses them all.
#    Use the range() built-in function to create a list, for example: numbers = range(10). Write an if
statement that tests a value to determine if it is in a list.
#    Write an if statement that tests whether the value is not in the list (above).
#    Create a small dictionary, for example: fruit = {'watermelon': 44}. Write an if statement that
tests whether a key is in the dictionary and tests whether the value (associated with that key) is
equal to some value (such as 44).
#    Assign a value to a variable, then write an if statement to determine whether that value is the
same as None and another if statement to test whether the value is not None.



#
# if statement
#
def test_if():
    list1 = ['apple', 'banana', 'cherry', 'date', ]
    for item in list1:
        if item == 'apple':
            print "it's an apple"
        elif item == 'banana':
            print "it's a banana"
        elif item == 'cherry':
            print "it's a cherry"
        else:
            print "it's an other --", item
    v1 = 3
    v2 = 5
    v3 = 7
    if not (v1 < v2 and v2 < v3) or v1 != v2:
        print 'yes'
    else:
        print 'no'
    list2 = range(10)
    value1 = 5
    value2 = 20
    if value1 in list2:
        print "it's in -- %d" % value1
    if value2 not in list2:
        print "it's not -- %d" % value2
    dict1 = {'watermelon': 4}
    key1 = 'watermelon'
    if key1 in dict1 and dict1[key1] == 4:
        print "it's good"
    key2 = 'cantaloupe'
    if key2 in dict1 and dict1[key2] == 4:
        print "it's bad"


=============== conditional_while.py ===================
#Qns:
# Use a while statement to loop through a list of words and to find the first word with a specific
number of characters.
# Use a while statement to loop through a list; pop each item off the right end of the list and print
the item, until the list is empty.
# Write a function that uses a while statement and str.find(pat, pos) to print out all the positions
of a pattern string in a target string.
# Convert the above function into a function that returns a generator that produces all the positions
of the pattern string in a target string.




#
# while statement
#
def test_while(word_length):
    words = ['a', 'bb', 'ccc', 'dddd', ]
    idx = 0
    word = None
    while idx < len(words):
        if len(words[idx]) == word_length:
            word = words[idx]
            break
        idx += 1
    print 'word: "%s"' % word
    print '-' * 20
    word_length += 20
    while idx < len(words):
        if len(words[idx]) == word_length:
            word = words[idx]
            break
        idx += 1
    else:
        word = None
    print 'word: "%s"' % word
    print '-' * 20
    items = range(10, 14)
    while items:
        item = items.pop()
        print 'popped item:', item
    return word


def search_positions(pat, target):
    pos = 0
    while True:
        pos = target.find(pat, pos)
        if pos < 0:
            break
        print 'search pos:', pos
        pos += 1


def generate_positions(pat, target):
    pos = 0
    while True:
        pos = target.find(pat, pos)
        if pos < 0:
            break
        yield pos
        pos += 1


def apply_at_positions(pat, target, func):
    pos = 0
    while True:
        pos = target.find(pat, pos)
        if pos < 0:
            break
        func(pos, target[pos:pos + len(pat)])
        pos += 1


def func1(pos, str_val):
    print 'apply at pos: %d  "%s"' % (pos, str_val.upper(), )




=============== count_comment_code2.py ===================
# coding=utf-8

import os
import re

def source_statistic(file_name):
    total = 0
    blank = 0
    comment = 0

    with open(file_name) as f:
        lines = f.readlines()
        total += len(lines)

        pattern1 = re.compile(r'^\s*$')
        pattern2 = re.compile(r'^\s*\#+')

        for line in lines:
            if pattern1.match(line):
                blank += 1
            elif pattern2.match(line):
                comment += 1

    return total, blank, comment


def walk_dir(image_path):
    total = 0
    blank = 0
    comment = 0

    for root, dirs, files in os.walk(image_path):
        for f in files:
            if f.lower().endswith('.py'):
                full_path = os.path.join(root, f)
                #resize_image(full_path)
                print full_path
                result = source_statistic(full_path)
                total += result[0]
                blank += result[1]
                comment += result[2]

    return total, blank, comment


if __name__ == '__main__':
    result = walk_dir("source")
    print 'Total Lines: %d, Blank Lines: %d, Comment Lines: %d' % result


=============== count_comment_code.py ===================
import os
```

```
codes_path = "codes/"

def line_counter(dir):
    codes = os.listdir(dir)
    print codes
    code_lines = 0
    empty_lines = 0
    comment_lines = 0
    for i in codes:
        with open(dir + i) as code_file:
            codes_lines = code_file.readlines()

            for line in codes_lines:
                line = line.strip()
                if line.startswith("#"):
                    comment_lines += 1
                elif line == "":
                    empty_lines += 1
                else:
                    code_lines += 1
    print("There are " +
          str(code_lines) + " code lines, " +
          str(comment_lines) + " comment lines, and " +
          str(empty_lines) + " empty lines.")

if __name__ == '__main__':
    line_counter(codes_path)


=============== cp1252_encoding.py ===================
# coding: cp1252
print('Olá, Mundo!')


=============== create_file.py ===================

import os

def creat_file(path):
    for i in range(26):
        n = str(i).zfill(4)
        sub_path = path + n
        os.mkdir(sub_path)
    return path

if __name__ == '__main__':
    path = raw_input("enter path: ")
    creat_file(path)


=============== default_encodings.py ===================
import sys, locale

expressions = """
        locale.getpreferredencoding()
        type(my_file)
        my_file.encoding
        sys.stdout.isatty()
        sys.stdout.encoding
        sys.stdin.isatty()
        sys.stdin.encoding
        sys.stderr.isatty()
        sys.stderr.encoding
        sys.getdefaultencoding()
        sys.getfilesystemencoding()
    """

my_file = open('dummy', 'w')

for expression in expressions.split():
    value = eval(expression)
    print(expression.rjust(30), '->', repr(value))


=============== dict1.py ===================
#!/usr/bin/env
'''
the following code is a  preliminary word processing,
it won't split abbrevations, like "I'd", nor it won't split words
concatenated with underscore, like "bad_game"
'''
import os
import re
def word_count(file_path):
    word_dict = {}
    with open(file_path) as txt:
        for line in txt:
            words = re.findall(r"\w+", line.strip())
            for word in words:
                word = word.lower()
                word_dict[word] = word_dict.get(word,0) + 1
    return word_dict

result = word_count(os.path.join(os.path.dirname(os.path.abspath(__file__)), "sampletext.txt"))

##################################################################################

'''
"How to "sort" a dict by its key/value"
Obviously, a dict can not be sorted, since it is orderless in nature.
However, what we can do is to sort a representation(a list of tuples or a list of keys) of dict.
link: http://stackoverflow.com/questions/613183/sort-a-python-dictionary-by-value
'''

import operator
d = {"what": 3, "I": 19, "the":30}
# to get a list of tuples with sorted items by the value
sorted_d_by_value = sorted(d.items(), key=operator.itemgetter(1))
# to get a list of tuples with sorted key
sorted_d_by_key = sorted(d.items(), key=operator.itemgetter(0))
print sorted_d_by_key





=============== dict_to_xls_converter2.py ===================
import json, xlwt

with open('city.txt','r')as f:
    data = f.read().decode('gbk')
data = json.loads(data)

book = xlwt.Workbook(encoding='utf-8')
sheet = book.add_sheet('city')

for i in range(len(data)):
    sheet.write(i,0,i)
    sheet.write(i,1,data[str(i+1)])
book.save('city.xls')



=============== dict_to_xls_converter.py ===================
#coding:utf-8
import json
import xlwt

with open('student.txt','r')as f:
    data = f.read().decode('gbk')
data = json.loads(data)

book =xlwt.Workbook(encoding = 'utf-8')
```

```
sheet =book.add_sheet('student')

for i in range(len(data)):
    d = data[str(i+1)]
    sheet.write(i,0,i+1)
    for j in range(len(d)):
        sheet.write(i,j+1,d[j])
book.save('student.xls')


=============== email_gmail.py ===================
# Code : Send attachment "capture.jpg" using gmail

import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEBase import MIMEBase
from email.MIMEText import MIMEText
from email import Encoders
import os

send_user = "YOUR_GMAIL_ID@gmail.com"       #sender_email
send_pwd = "YOUR_GMAIL_PASSWORD"        #sender_password
recv_user = "YOUR_GMAIL_ID@gmail.com"       #receiver_email
subject = "Test Email !!!!"

def mail(to, subject, text, attach):
   msg = MIMEMultipart()

   msg['From'] = send_user
   msg['To'] = recv_user
   msg['Subject'] = subject

   msg.attach(MIMEText(text))

   part = MIMEBase('application', 'octet-stream')
   part.set_payload(open(attach, 'rb').read())
   Encoders.encode_base64(part)
   part.add_header('Content-Disposition','attachment; filename="%s"' % os.path.basename(attach))
   msg.attach(part)

   mailServer = smtplib.SMTP("smtp.gmail.com", 587)
   mailServer.ehlo()
   mailServer.starttls()
   mailServer.ehlo()
   mailServer.login(send_user, send_pwd)
   mailServer.sendmail(send_user, to, msg.as_string())
   mailServer.close()

mail(recv_user, subject,"imaeg snapshot !!!","capture.jpg")


=============== encode_decode4.py ===================
#http://stackoverflow.com/questions/1207457/convert-a-unicode-string-to-a-string-in-python-containing-
extra-symbols

##
import binascii
text1 = "tushar"
text2 = binascii.hexlify(text1)
text3 = binascii.unhexlify(text2)
print "original",text1
print "string to hex",text2
print "hex to string",text3

#web input unicode to ascii
raw_data.encode('ascii','ignore')
unicodedata.normalize('NFKD', raw_data).encode('ascii','ignore')


##
unicodestring = u"Hello world"
```

```
# Convert Unicode to plain Python string: "encode"
utf8string = unicodestring.encode("utf-8")
asciistring = unicodestring.encode("ascii")
isostring = unicodestring.encode("ISO-8859-1")
utf16string = unicodestring.encode("utf-16")

# Convert plain Python string to Unicode: "decode"
plainstring1 = unicode(utf8string, "utf-8")
plainstring2 = unicode(asciistring, "ascii")
plainstring3 = unicode(isostring, "ISO-8859-1")
plainstring4 = unicode(utf16string, "utf-16")
assert plainstring1 == plainstring2 == plainstring3 == plainstring4



=============== encode_decode_file.py ===================
#base64 encoding/decoding file
import base64
file1 = open("file1","w")
file1.write("how r u")
file1.close()
base64.encode(open("file1"),open("file1.64","w"))
base64.decode(open("file1.64"),open("file2","w"))


=============== encode_decode_image.py ===================

#base64 encoding/decoding image/picture/photo
import base64
imageFile = open("failure.png", "rb")
str = base64.b64encode(imageFile.read())
fh = open("file1", "w")
fh.write(str)
fh.close()

fh = open("file1", "r")
str3 = fh.read()
fh.close()
fh = open("image2.jpg", "wb")
fh.write(str3.decode('base64'))
fh.close()


=============== encode_decode_simple.py ===================
##unicode/encode/decode

import binascii

convert text nbyte string.
def to_hex(t, nbytes):
    chars_per_item = nbytes * 2
    hex_version = binascii.hexlify(t)
    return ' '.join(
        hex_version[start:start + chars_per_item]
        for start in xrange(0, len(hex_version), chars_per_item)
        )

if __name__ == '__main__':
    print to_hex('abcdef', 1)
    print to_hex('abcdef', 3)


=============== encode_decode_text.py ===================


#base64 encoding/decoding text
import base64
string_orig = "tushar"
string_enc = base64.encodestring(string_orig)
string_dec = base64.decodestring(string_enc)
print string_orig
print string_enc
print string_dec



=============== exception_try_catch_.py ===================
import os
fd1 = open("file1","r+")
while True:
    try:
        content = file.next(fd1)
    except StopIteration:
        print "EOF reached"
        break
    else:
        if content is None:
            print "EOF"
            break
        else:
            print content
    finally:
        print "exception or not, i will execute"

print "STOP"
fd1.close


##################################################################
import sys
try:
    sys.exit(0)
except SystemExit,err:
    print 'Tried to exit with code', err.code


##################################################################





=============== file_operations.py ===================

#open-close check
fd = open("file1","r+")
print "name     ",fd.name
print "closed   ",fd.closed
fd.close()
print "closed   ",fd.closed

#append
fd = open("file1","a+")
fd.write("cassidy");
fd.close()

#read file1 & write to file2
fd1 = open("file1","a+")
fd2 = open("file2","r")
content2 = fd2.read();
fd1.write(content2);
fd1.close()
fd2.close()

#cursor position
fd1 = open("file1","r+")
position = fd1.tell()
print position
position = fd1.seek(4,0)
print position
content = fd1.read()
print content
fd1.close()
```

```
#read file line by line
with open("symbols",'rb') as f:
    while True:
        line=f.readline()   ##f.readline(5) read 5 lines
        if not line: break
        process(line)




######file operations---------------------------------
import os
os.rmdir("dir1")
os.mkdir("dir1")
fd1 = open("file1","r+")

while True:
    try:
        content = file.next(fd1)
    except StopIteration:
        print "EOF reached"
        break
    else:
        if content is None:
            print "EOF"
            break
        else:
            print content
    finally:
        print "exception or not, i will execute"

print "STOP"
fd1.close

#-------------------------------------------------------

###############################################################################

with open('file1') as f:
    lines = f.readlines()
    line = lines.rstrip('\n')
    print line

#lines = [line.rstrip('') for line in open('file1')]
#lines = [line.strip('\t').strip('\n') for line in lines]





=============== filtered_words2.py ===================
#coding:utf-8
with open('filtered_words.txt','r')as f:
    data = f.read().decode('gbk')

filt = data.split('\n')

while True:
    text = raw_input("please input:")
    text  = text.decode('gbk')
    for x in filt:
        if text.find(x) != -1:
            text = text.replace(x,'*'*len(x))
    print text


=============== filtered_words.py ===================
#coding:utf-8
with open('filtered_words.txt','r')as f:
    data = f.read()
```

```
filt = data.split('\n')

while True:
    flag = False
    text = raw_input("please input:")
    for x in filt:
        if text.find(x) != -1:
            flag = True
    if flag:
        print "Match Found"
    else:
        print "Match Failed"


=============== find_keywords_in_diary.py ===================
import os
import re

# set diaries path
diaries_path = "diaries/"
diaries = os.listdir(diaries_path)

# set stop words to make informative keywords
stop_words = open("StopWords.txt", 'r').read()
stop_words_list = stop_words.split(" ")


# Find top 5 keywords in a txt
def find_keywords(words):
    words_dictionary = {}
    for word in words:
        if word.lower() not in words_dictionary and word.lower() not in stop_words_list:
            # Put word in dictionary
            words_dictionary[word] = 0
            for item in words:
                if item == word:
                    words_dictionary[word] += 1
    # Find 5 keywords which by highest frequency
    keywords = sorted(
        words_dictionary, key=words_dictionary.__getitem__, reverse=True)[0:5]
    return keywords

for diary in diaries:
    # Coding by utf-8
    with open(diaries_path + diary, "r") as content:
        diary_words_list = re.findall(r"[\w']+", content.read())
        print("The keywords of diary " + diary + " is: ")
        print(find_keywords(diary_words_list))


=============== formatted_print.py ===================
metro_areas = [
    ('Tokyo', 'JP', 36.933, (35.689722, 139.691667)),
    ('Delhi NCR', 'IN', 21.935, (28.613889, 77.208889)),
    ('Mexico City', 'MX', 20.142, (19.433333, -99.133333)),
    ('New York-Newark', 'US', 20.104, (40.808611, -74.020386)),
    ('Sao Paulo', 'BR', 19.649, (-23.547778, -46.635833)),
]

print('{:15} | {:^9} | {:^9}'.format('', 'lat.', 'long.'))
fmt = '{:15} | {:9.4f} | {:9.4f}'
for name, cc, pop, (latitude, longitude) in metro_areas:  # <2>
    if longitude <= 0:  # <3>
        print(fmt.format(name, latitude, longitude))


=============== html_extract_links.py ===================
from HTMLParser import HTMLParser
import urllib2

class MyHTMLParser(HTMLParser):
    def __init__(self):
        HTMLParser.__init__(self)
        self.links = []

    def handle_starttag(self, tag, attrs):
        if tag == "a":
            if len(attrs) == 0: pass
            else:
                for (variable, value)  in attrs:
                    if variable == "href":
                        self.links.append(value)

if __name__ == "__main__":
    html_code = """<a href="www.google.com">google.com</a><a HREF ="www.sina.com.cn">Sina</a>"""
    url = 'http://www.google.com'
    html = urllib2.urlopen(url)
    html_code = html.read()
    hp = MyHTMLParser()
    hp.feed(html_code)
    hp.close()
    for link in hp.links:
        if link.find("http")==0:
            print link




============== html_extract_text.py ==================
from HTMLParser import HTMLParser
from re import sub
from sys import stderr
from traceback import print_exc
import urllib2

class _DeHTMLParser(HTMLParser):
    def __init__(self):
        HTMLParser.__init__(self)
        self.__text = []

    def handle_data(self, data):
        text = data.strip()
        if len(text) > 0:
            text = sub('[ \t\r\n]+', ' ', text)
            self.__text.append(text + ' ')

    def handle_starttag(self, tag, attrs):
        if tag == 'p':
            self.__text.append('\n\n')
        elif tag == 'br':
            self.__text.append('\n')

    def handle_startendtag(self, tag, attrs):
        if tag == 'br':
            self.__text.append('\n\n')

    def text(self):
        return ''.join(self.__text).strip()


def dehtml(text):
    try:
        parser = _DeHTMLParser()
        parser.feed(text)
        parser.close()
        return parser.text()
    except:
        print_exc(file=stderr)
        return text


if __name__ == '__main__':
    url = "http://SOMEURL"
    html = urllib2.urlopen(url)
    html_code = html.read()
    html_code = sub('<script>(.*?)</script>','',html_code)
    print dehtml(html_code).decode('gbk').encode('utf-8')
    with open('result.txt','w') as f:
        f.write(dehtml(html_code).decode('gbk').encode('utf-8'))


=============== important_word.py ===================
# -*- coding: utf-8 -*-

import re
import os

# Get all files in designated path
def get_files(path):
    filepath = os.listdir(path)
    files = []
    for fp in filepath:
        fppath = path + '/' + fp
        if(os.path.isfile(fppath)):
            files.append(fppath)
        elif(os.path.isdir(fppath)):
            files += get_files(fppath)
    return files

# Get the most popular word in designated files
def get_important_word(files):
    worddict = {}
    for filename in files:
        f = open(filename, 'rb')
        s = f.read()
        words = re.findall(r'[a-zA-Z0-9]+', s)
        for word in words:
            worddict[word] = worddict[word] + 1 if word in worddict else 1
        f.close()
    wordsort = sorted(worddict.items(), key=lambda e:e[1], reverse=True)
    return wordsort

if __name__ == '__main__':
    files = get_files('.')
    print files
    wordsort = get_important_word(files)
    maxnum = 1
    for i in range(len(wordsort) - 1):
        if wordsort[i][1] == wordsort[i + 1][1]:
            maxnum += 1
        else:
            break
    for i in range(maxnum):
        print wordsort[i]


=============== inheritance.py ===================
class A:
    def ping(self):
        print('ping:', self)


class B(A):
    def pong(self):
        print('pong:', self)


class C(A):
    def pong(self):
        print('PONG:', self)


class D(B, C):

    def ping(self):
        super().ping()
        print('post-ping:', self)

    def pingpong(self):
        self.ping()
        super().ping()
        self.pong()
        super().pong()
        C.pong(self)


=============== keyword_extractor1.py ===================
# coding=UTF-8
import nltk
from nltk.corpus import brown
import os

# This is a fast and simple noun phrase extractor (based on NLTK)
# Feel free to use it, just keep a link back to this post
# http://thetokenizer.com/2013/05/09/efficient-way-to-extract-the-main-topics-of-a-sentence/
# http://www.sharejs.com/codes/
# Create by Shlomi Babluki
# May, 2013


# This is our fast Part of Speech tagger
#############################################################################
brown_train = brown.tagged_sents(categories='news')
regexp_tagger = nltk.RegexpTagger(
    [(r'^-?[0-9]+(.[0-9]+)?$', 'CD'),
     (r'(-|:|;)$', ':'),
     (r'\'*$', 'MD'),
     (r'(The|the|A|a|An|an)$', 'AT'),
     (r'.*able$', 'JJ'),
     (r'^[A-Z].*$', 'NNP'),
     (r'.*ness$', 'NN'),
     (r'.*ly$', 'RB'),
     (r'.*s$', 'NNS'),
     (r'.*ing$', 'VBG'),
     (r'.*ed$', 'VBD'),
     (r'.*', 'NN')
     ])
unigram_tagger = nltk.UnigramTagger(brown_train, backoff=regexp_tagger)
bigram_tagger = nltk.BigramTagger(brown_train, backoff=unigram_tagger)
#############################################################################


# This is our semi-CFG; Extend it according to your own needs
#############################################################################
cfg = {}
cfg["NNP+NNP"] = "NNP"
cfg["NN+NN"] = "NNI"
cfg["NNI+NN"] = "NNI"
cfg["JJ+JJ"] = "JJ"
cfg["JJ+NN"] = "NNI"
#############################################################################


class NPExtractor(object):

    def __init__(self, sentence):
        self.sentence = sentence

    # Split the sentence into singlw words/tokens
    def tokenize_sentence(self, sentence):
        tokens = nltk.word_tokenize(sentence)
        return tokens

    # Normalize brown corpus' tags ("NN", "NN-PL", "NNS" > "NN")
    def normalize_tags(self, tagged):
        n_tagged = []
        for t in tagged:
            if t[1] == "NP-TL" or t[1] == "NP":
                n_tagged.append((t[0], "NNP"))
                continue
            if t[1].endswith("-TL"):
                n_tagged.append((t[0], t[1][:-3]))
                continue
            if t[1].endswith("S"):
                n_tagged.append((t[0], t[1][:-1]))
                continue
            n_tagged.append((t[0], t[1]))
        return n_tagged

    # Extract the main topics from the sentence
    def extract(self):

        tokens = self.tokenize_sentence(self.sentence)
        tags = self.normalize_tags(bigram_tagger.tag(tokens))

        merge = True
        while merge:
            merge = False
            for x in range(0, len(tags) - 1):
                t1 = tags[x]
                t2 = tags[x + 1]
                key = "%s+%s" % (t1[1], t2[1])
                value = cfg.get(key, '')
                if value:
                    merge = True
                    tags.pop(x)
                    tags.pop(x)
                    match = "%s %s" % (t1[0], t2[0])
                    pos = value
                    tags.insert(x, (match, pos))
                    break

        matches = []
        for t in tags:
            if t[1] == "NNP" or t[1] == "NNI":
                # if t[1] == "NNP" or t[1] == "NNI" or t[1] == "NN":
                matches.append(t[0])
        return matches


# Main method, just run "python np_extractor.py"
def main():
    path = '.'
    for file in os.listdir(path):
        text = []
        if file.endswith('.text'):
            with open(file, 'rt') as f:
                for line in f:
                    words = line.split()
                    text += words
            str_text=' '.join(text)
            np_extractor = NPExtractor(str_text)
            result = np_extractor.extract()
            print("This file is about: %s" % ", ".join(result))
if __name__ == '__main__':
    main()


=============== keyword_extractor.py ===================
import os,re
from collections import Counter
import nltk
from nltk.corpus import stopwords

useless_words=['the','I','and','']
def main():
    for file in os.listdir("."):
        result=Counter()
        if file.endswith('.text'):
            with open(file,'rt') as f:
                for line in f:
                    # delete the stopwords in note
                    words=line.split()
                    words= [w for w in words if not w in stopwords.words('english')]
                    result+=Counter(words)
            print('The most important word in %s is %s',(file,result.most_common(2)))

if __name__ == '__main__':
    main()


=============== link_retrieval_bs.py ===================
import requests as req
from bs4 import BeautifulSoup

url = "http://www.theplantlist.org/browse/-/"
response = req.get(url=url)

soup = BeautifulSoup(response.content,"lxml")
urls = [a_tag.get('href') for a_tag in soup.find_all('a') if a_tag.get('href').startswith('http')]

print(urls)


=============== list_files_directory.py ===================
import os

# Get all files in designated path
def get_files(path):
    filepath = os.listdir(path)
    files = []
    for fp in filepath:
        fppath = path + '/' + fp
        if(os.path.isfile(fppath)):
            files.append(fppath)
        elif(os.path.isdir(fppath)):
            files += get_files(fppath)
    return files

list1 = get_files(".")

for i in list1:
    print i


=============== list_pdf_files.py ===================
import os
import glob

def main():
    list_pdf = glob.glob("*.pdf")
    print "pdf files in directory:",list_pdf

main()


=============== mysql1.py ===================
#insert data

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');

with con:

    cur = con.cursor()
    cur.execute("DROP TABLE IF EXISTS Writers")
    cur.execute("CREATE TABLE Writers(Id INT PRIMARY KEY AUTO_INCREMENT, \
                 Name VARCHAR(25))")
    cur.execute("INSERT INTO Writers(Name) VALUES('abc')")
    cur.execute("INSERT INTO Writers(Name) VALUES('def')")
    cur.execute("INSERT INTO Writers(Name) VALUES('feh')")
    cur.execute("INSERT INTO Writers(Name) VALUES('fet')")
    cur.execute("INSERT INTO Writers(Name) VALUES('geh')")


=============== mysql2.py ===================
# retrieve data (all at once)
import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');

with con:

    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")

    rows = cur.fetchall()

    for row in rows:
        print row



=============== mysql3.py ===================
# retrieve data (one-by-one)

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');

with con:

    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")

    for i in range(cur.rowcount):

        row = cur.fetchone()
        print row[0], row[1]
#close database
    con.close()


=============== object_reference1.py ===================

class bus1:
    """remove passengers """

    def __init__(self, passengers=None):
        if passengers is None:
            self.passengers = []  # <1>
        else:
            self.passengers = passengers  #<2>

    def pick(self, name):
        self.passengers.append(name)

    def drop(self, name):
        self.passengers.remove(name)  # <3>

passengers = ['ajit', 'nikhil', 'sam', 'srini', 'jagdish']
print "actual: ",passengers
bus = bus1(passengers)
bus.drop('nikhil')
bus.drop('srini')
print "remaining: ",passengers


=============== pass_by_value_reference.py ===================
#!/usr/bin/python

#pass by value----------------------------------------
```

```
# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str;
   return;

# Now you can call printme function
printme("I'm first call to user defined function!");
printme("Again second call to the same function");


#pass by reference----------------------------------------

# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist.append([1,2,3,4]);
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist



=============== password1.py ===================
import os
from hashlib import sha256
from hmac import HMAC

def encrypt_password(password, salt=None):
    if salt is None:
        salt = os.urandom(8)
    assert 8 == len(salt)
    assert isinstance(salt, str)

    if isinstance(password, unicode):
        password = password.encode('utf-8')
    assert isinstance(password, str)

    for i in range(10):
        encrypted = HMAC(password, salt, sha256).digest()
    return salt + encrypted


def validate_password(hashed, password):
    return hashed == encrypt_password(password, hashed[:8])


if __name__ == "__main__":
    password_new = raw_input("Set your password\n")
    password_saved = encrypt_password(password_new)
    password_again = raw_input("Now,type in your password\n")
    print "Password match." if validate_password(password_saved, password_again) else "wrong password."


=============== pdf_page_count_using_re.py ===================
import re
import os

rxcountpages = re.compile(r"/Type\s*/Page([^s]|$)", re.MULTILINE|re.DOTALL)

def count_pages(filename):
    data = file(filename,"rb").read()
    print data
    return len(rxcountpages.findall(data))

if __name__=="__main__":
    filename = 'input.pdf'
    print count_pages(filename)


=============== random_code_generator1.py ===================
import random
import string

coupon_number = 20
coupon_size = 15

for i in range(coupon_number):
    coupon = ''.join(
        random.sample(string.digits + string.ascii_uppercase, coupon_size))
    print(coupon)


=============== random_code_generator.py ===================
# -*- coding: utf-8 -*-

from random import Random

def codeGenerator(number, codeLength = 8):
    print '**** Code Generator ****'
    codeFile = open('codes.txt', 'w')
    if number <= 0:
        return 'invalid number of codes'
    else:
        chars = 'abcdefghijklmnopgrstuvwxyzABCDEFGHIJKLMNOPGRSTUVWXYZ1234567890'
        random = Random()
        for j in range(1, number+1):
            str = ''
            for i in range(1, codeLength+1):
                index = random.randint(1, len(chars))
                str = str + chars[index-1]
            codeFile.write(str+'\n')

print codeGenerator(20)


=============== retrieve_links_from_html.py ===================
#coding:utf-8
import re

with open('Gaana.com.html','rb')as f:
    data = f.read()

data = data.replace('\r','').replace('\b','').replace('\n','')
find = re.compile(r'href="(.*?)"')
result = find.findall(data)
for x in result:
    if x.find("http") > -1:
        print x


=============== retrieve_links_webpage2.py ===================
#!/usr/bin/env python3

import re
import requests

r = requests.get('https://github.com')
print type(r)
matches = re.findall('(?:https?|ftp)://[^\s/$\.?#].[^\s]+', r.text)

for i in range(0, len(matches)):
    print('matche >>>: {} \n'.format(matches[i]))


=============== retrieve_link_webpage_http_string_match.py ===================
import requests
import re
from bs4 import BeautifulSoup
```

```
url = "http://stackoverflow.com"
html = requests.get(url)

soup = BeautifulSoup(html.text,"html.parser")
find_href = soup.findAll('a')
for x in find_href:
    print(x['href'])


=============== selenium_save_complete_page.py ===================
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
import time
from lxml import html
import os
import urllib2
from bs4 import BeautifulSoup as bsp
from selenium.webdriver.support.ui import WebDriverWait
import timeit
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()
driver.get("https://google.in")
time.sleep(2)
ActionChains(driver) \
    .key_down(Keys.CONTROL) \
    .key_down('s') \
    .key_up(Keys.CONTROL) \
    .key_up('s') \
    .perform()

#element = driver.find_element_by_id('lst-ib')
#ActionChains(driver) \
#    .key_down(Keys.CONTROL) \
#    .click(element) \
#    .key_up(Keys.CONTROL)

raw_input("ctrl+s triggered...")

#htmltext  = (driver.page_source).encode('utf-8')
#f = open("google1", "w")
#print >> f, htmltext
#f.close()
time.sleep(5)

driver.close()
print "Exiting.."



=============== shallow_deep_copy.py ===================
import copy

class Bus:
    def __init__(self, passengers=None):
        if passengers is None:
            self.passengers = []
        else:
            self.passengers = list(passengers)

    def pick(self, name):
        self.passengers.append(name)

    def drop(self, name):
        self.passengers.remove(name)


bus1 = Bus(['Alice', 'Bill', 'Claire', 'David'])
```

```
bus2 = copy.copy(bus1)
bus3 = copy.deepcopy(bus1)
bus1.drop('Bill')
print bus1.passengers
print bus2.passengers
print bus3.passengers


=============== shuffled_list_.py ===================
#pick items from shuffled list

import random

class BingoCage:

    def __init__(self, items):
        self._items = list(items)  # <1>
        random.shuffle(self._items)  # <2>

    def pick(self):  # <3>
        try:
            return self._items.pop()
        except IndexError:
            raise LookupError('pick from empty shuff list')  # <4>

    def __call__(self):  # <5>
        return self.pick()


print BingoCage(range(3))
print BingoCage(range(3)).pick()
print BingoCage(range(3))
print callable(BingoCage)


=============== sockets_basics.py ===================
#!/usr/bin/python

#SERVER
import socket               # Import socket module

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname()
s.bind((host, 1870))

s.listen(10)
while True:
   c, addr = s.accept()
   print 'Got connection from', addr
   c.send('Thank you for connecting')
   c.close()

#CLIENT
import socket               # Import socket module

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = socket.gethostname() # Get local machine name
s.connect((host, 1870))
print s.recv(1024)
s.close                     # Close the socket when done




=============== sqlite_example.py ===================

import os
import sqlite3

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

```
if db_is_new:
    print "Creating database"
    conn = sqlite3.connect(db_filename)
    cur = conn.cursor()
    cur.execute('SELECT SQLITE_VERSION()')
    data1 = cur.fetchone() #fetchall will retrieve all available
    cur.execute("CREATE TABLE Cars(Id INT, Name TEXT, Price INT)")
    cur.execute("INSERT INTO Cars VALUES(1,'Audi',52642)")
    cur.execute("INSERT INTO Cars VALUES(2,'Mercedes',57127)")
    cur.execute("INSERT INTO Cars VALUES(3,'Mercedes',34888)")
    cur.execute("INSERT INTO Cars VALUES(4,'omni',10000)")
    cur.execute("INSERT INTO Cars VALUES(5,'hyundai',9000)")
    conn.close()
    print "SQLite version: %s" %data1
    print "DONE"
else:
    print "database exists"
    conn = sqlite3.connect(db_filename)
    cur = conn.cursor()
    cur.execute("SELECT * FROM Cars")
    row = cur.fetchone()
    print row
    conn.close()



=============== string2.py ===================
#    Create a string. Print it.
#    Create a multi-line string. Print it.
#    Concatenate two strings together. Print the result.
#    Print the length of one of your strings.
#    Search one string for a sub-string. Print the location of the sub-string.
#    Format a string using the string formatting operator (%). Interpolate into the string: (1) a
string, (2) an integer, and (3) a float.
#    Left justify a short string within a longer field. Print it. Do the same but right justify it.
Print it. Center the string within a longer field. Print it.
#    Strip white-space off (1) the left side, (2) the right side, and (3) both sides of a string.
Print each result.
#    Split a string into a list of words.
#    Join a list of strings together with "::" between each of the sub-strings.
#    Given a word and a list of strings, write a function collect(word, strings) that returns a (new)
list that contains the strings that contain the word.

#What you will learn:

#    How to create and manipulate strings.
#    How to concatenate two strings.
#    How to format a string with the string formatting operator.
#    How to justify a string.
#    How to strip white-space off a string.

##########################################################################################
# vim: set fileencoding=utf-8 :

S1 = """peaches are pink
lemons are yellow
tomatoes are red
and i'm very mellow
"""

S2 = 'Selçuk'


def test():
    """Perform various operations on strings.
    """
    s1 = 'the brown dog is a frisky dog'
    print 's1:', s1
    print 'S1:', S1
    print 'I love tomatoes in the summertime\nI love apples in the fall\n'
    print 's1 + S1:', s1 + S1
    print 'len(s1):', len(s1)
    n1 = s1.find('dog')
    n2 = s1.find('dog', n1 + 1)
    print 'n1, n2:', n1, n2
    name = 'dave'
    category = 3
    intensity = 4.8
    s2 = '"%s" is in category %d with intensity %.3f' % (
        name, category, intensity, )
    print 's2:', s2
    print 'ljust: "%s"' % name.ljust(10)
    print 'rjust: "%s"' % name.rjust(10)
    s3 = '    abc def    \n'
    print 'lstrip: "%s"' % s3.lstrip()
    print 'rstrip: "%s"' % s3.rstrip()
    print 'strip: "%s"' % s3.strip()
    data = ['abc', 'def', 'ghi', 'jkl', ]
    print 'join:', '::'.join(data)
    # notice the encoding definition at the top of this file.
    # can also use:
    #    # coding=utf-8
    print 'len s4:', len(S2)
    s5 = S2.decode('utf-8')
    print 'len s5:', len(s5)
    s6 = s5.encode('utf-8')
    print 's6:', s6
    s7 = u'Sel\xe7uk'
    s8 = s7.encode('utf-8')
    print 's8:', s8


def collect(word, strings):
    acc = []
    for string in strings:
        if string.find(word) > -1:
            acc.append(string)
    return acc


def main():
    test()
    accum = collect('dog', ['each dog has his day', 'the cat is furry'])
    print 'accum from collect:', accum


if __name__ == '__main__':
    main()





=============== string_search.py.py ===================
"""
Synopsis:
    Search a target string for a pattern.
    Print all the positions of the occurance of the pattern string
        within the target string.
    Implement the above in two different ways:
        1. Use a while: statement.
        2. Implement a generator function (a function that uses the
           yield statement).
Usage:
    python search_str.py
"""

import sys


def search_while(pat, target):
    pos = 0
    while True:
        pos = target.find(pat, pos)
        if pos < 0:
            break
        print pos
        pos += 1


def generate_positions(pat, target):
    pos = 0
    while True:
        pos = target.find(pat, pos)
        if pos < 0:
            break
        yield pos
        pos += 1


def test():
    args = sys.argv[1:]
    if len(args) != 0:
        sys.exit(__doc__)
    search_while('xxx', 'aaaxxxbbbxxxcccxxxdddxxxeee')
    print '=-' * 30
    generator1 = generate_positions('xxx', 'aaaxxxbbbxxxcccxxxdddxxxeee')
    for pos in generator1:
        print 'pos:', pos


if __name__ == '__main__':
    test()


=============== string_word_count.py.py ===================
import sys

def count(filename):
    infile = open(filename, 'r')
    accum = {}
    for line in infile:
        line = line.rstrip()
        words = line.split()
        for word in words:
##             accum.setdefault(word, 0)
##             accum[word] += 1
            if word in accum:
                accum[word] += 1
            else:
                accum[word] = 1
    infile.close()
    return accum


def test():
    args = sys.argv[1:]
    filename = args[0]
    accum = count(filename)
    words = accum.keys()
    words.sort()
    for word in words:
        print 'word: "%s" -- count: %d' % (word, accum[word], )

test()


=============== sum_range_ex.py ===================
#sum of numbers lesser than 1000 & divisible by 3 or 5

print sum([i for i in range(1000) if i % 3 == 0 or i % 5 == 0])


=============== time_date_cal.py ===================
import time
```

```
#current time
print "time  : ", time.asctime(time.localtime(time.time()))
#time since 1970
print time.time()
print "time : hour ", time.localtime(time.time())[3]
print "time : minute ", time.localtime(time.time())[4]
print "time : sec ", time.localtime(time.time())[5]
print time.clock()

import calendar
print calendar.month(2021,4)


=============== varargs.py ===================
def printme(argone, *tuple1 ):
    if (not tuple1):
        print "1 argument passed"
        print argone
    else:
        counter = 0
        for var in tuple1:
            counter = counter+1
        print counter+1,"arguments passed"
        print argone,tuple1
        return;

printme(10);
printme(10,'cat',50);



=============== word_count.py ===================

def calcWords(path):
    file = open(path, 'r')
    inputStr = file.read()
    wordsNum = 0
    for i in inputStr:
        if i == ' ' or i == '\n':
            wordsNum += 1

    file.close()
    print wordsNum + 1

if __name__ == '__main__':
    calcWords('input.txt')


=============== word_count_using_regex.py ===================
import re
with open('test.txt','r')as f:
    data = f.read()
result = re.split(r"[^a-zA-Z]",data)
print len([x for x in result if x!= ''])


=============== word_frequency1.py ===================
import re
import operator

myfile = open('english_text.txt', 'r')
words = re.findall(r"[\w']+", myfile.read())
words_dictionary = {}

for word in words:
    if word not in words_dictionary:
        words_dictionary[word] = 0
        for item in words:
            if item == word:
                words_dictionary[word] += 1

sort_words_dictionary = sorted(words_dictionary.items(), key=operator.itemgetter(1), reverse=True)
```

```
print(sort_words_dictionary)


=============== word_frequency.py ===================
with open('test1.txt', 'r') as f:
    from collections import Counter
    c = Counter()
    for line in f:
        words = line.split()
        # print(words)
        c += Counter(words)
    for words in c.most_common():
        print(words)


=============== zipfile_create.py ===================
import sys
import zipfile

if __name__ == '__main__':
    zf = zipfile.PyZipFile('t1.zip', mode='w')
    try:
        zf.writepy('.')
        zf.write('input.txt')
        zf.write('test.txt')
    finally:
        zf.close()
    for name in zf.namelist():
        print name
```