#　批量讀取文件　－Ｈｒ篩選簡歷 -利用關鍵字篩選簡歷

# 自動生成word

import glob

final\_result = []

def search(path, target):

res = glob.glob(path)

for i in res:

if glob.os.isdir(i):

\_path = glob.os.path.join(i, ‘\*’)

search(\_path, target)

else:

if target in i: # 查文件名

final\_result.append(i)

return final\_result

if \_\_name\_\_ == “\_\_main\_\_”:

path = glob.os.path.join(glob.os.getcwd(), ‘\*’)

search(path, target=’python’)

import glob

final\_result = []

def search(path, target):

result = glob.glob(path)

for \_data in result:

if glob.os.path.isdir(\_data):

\_path = glob.os.path.join(glob.os.getcwd(), ‘\*’)

search(\_path, target)

else:

f = open(\_data, ‘r’, encoding=’utf-8)

content = f.read()

if target in content: # 查文件的內容

final\_result.append(i)

return final\_result

if \_\_name\_\_ == ‘\_\_main\_\_’:

path = glob.os.path.join(glob.os.getcwd(), ‘\*’)

res = search(path, target=’lun’)

print(res)

# coding:utf-8

#data = {'name': {'path/name': 'content', 'path2/name': 'content'}}

# 但是name裡有zip要避開 –要獲取文件路徑最後的字符串

data = {}

def search(path):

res = glob.glob(path)

for \_data in res:

if glob.os.path.isdir(\_data):

\_path = glob.os.path.join(\_data, ‘\*’)

search(\_path, target)

else:

name = glob.os.path.split(\_data)[-1] # ‘.py’

if ‘zip in name:

f = open(\_data, ‘rb’)

else:

f = open(\_data, ‘r’, encoding=’utf-8’)

content = f.read()

f.close()

if name in data: # 要確定文件名在字典內

sub\_data = data[name] # v -‘.py’的內容

for k, v in sub\_data.items():

if v == content: # 如果內容相同

glob.os.remove(\_data) # 直接移除整個路徑

print(f‘{\_data} will be deleted.’)

else:

data[name][\_data] = content #　‘.py’的內容

else:

sub\_data/data[name] = {

\_data: content

}

If \_\_name\_\_ == ‘\_\_main\_\_’:

path = glob.os.path.join(glob.os.getcwd(), ‘\*’)

search(path)

for k, v in data.items(): # 整個字典

print(k, v)

for k, v in v.items(): # 第二層字典

print(k, v)

# update\_name –

文件複製, 內容覆蓋, 裁減(移動, 重命名), 刪除 ?, 壓縮, 解壓縮

* 都必須先獲取origin, target的絕對路徑 -?
* -> os.path.join(os.getcwd(), ‘file\_name’))
* 可用shutil庫 -複製, 內容覆蓋, 裁減(移動, 重命名), 壓縮, 解壓縮

Copy -from shutil import copy(orgin, target) -target:file, filefolder

Cover-from shutil import copyfile(origin, target) -target: file

Move -from shutil import move(origin, target) -target: file, filefoler, 可以不存在

* move -裁減文件進文件夾; rename: target存在, 相當於起新名

make\_archive -from shutil import make\_archive(new\_name, 後綴, origin) -return 生成的壓縮包地址

unpack\_archive- from shutil import unpack\_archive(target, 解壓後的路徑)

* 刪除 – from os import remove -remove(origin, target)

import glob

文件夾複製, 裁減, 刪除

Copy – from shutil import copytree(origin, target) -FileExistError: target不能存在

Cut – from shutil import move(origin, target) -當target不存在, 屬於重命名

Remove -from shutil import rmtree(origin, target) -FileNotFoundError: target要存在

def update\_name(path):

res = glob.glob(path)

for \_data in res:

if glob.os.path.isdir(\_data):

\_path = glob.os.path.join(\_data, ‘\*’)

update\_name(\_path)

else:

\_path\_list = glob.os.path.split(\_data)

Name = \_path\_list[-1]

new\_name = ‘imooc\_%s’ % (len(name)

new\_path = glob.os.path.join(\_path\_list[0], new\_name)

shutil.move(\_data, new\_path)

if \_\_name\_\_ == ‘\_\_main\_\_’:

path = glob.os.path.join(glob.os.getcwd(), ‘\*’)

update\_name(path)

文件生成:

from docx import Document

都要先建立一個Document\_obj

doc = Document()

保存資料: doc.save(‘filename.docx’)

全局樣式: style = doc.styles[‘..’] - style樣式對象

斜體: style.italic = True, 粗體 style.bold = True, 顏色: style.font.color.rgb=RGBColor(),

大小 from docx.shared import Pt -style.font.size = Pt(20)

-> 對追加的內容才有樣式

標題:h = doc.add\_heading(‘..’, level:0-9) 增加: h.add\_run(‘…’)

段落: p = doc.add\_paragraph(‘..’) p.add\_run()

置左, 中, 右: from docx.enum.text import WD\_PARAGRAPH\_ALIGNMENT.LEFT/CENTER/RIGHT

圖片: picture = doc.add\_picture(origin, width=Inches(), height=Inches())

from docx.shared import Inches

置左, 中, 右: from docx.enum.text import WD\_ALIGN\_PARAGRAPH. LEFT/CENTER/RIGHT

增加: picture.add\_run(origin)

表格: table = doc.add\_table(rows列=, cols行=, style=)

cell = table.rows[0].cells -表格列對象

cell[0].text = 當前列0行的內容

cell[1].text = 當前列1行的內容

表格樣式: from docx.enum.text import WD\_STYLE\_TYPE

分頁: doc.add\_page\_break()

ReadExcel 🡪 WriteExcel

import xlrd

import xlsxwriter

def read():

excel = xlrd.open\_workbook(‘Excel\_name.xlsx’) # read excel\_obj

booksheet = excel.sheet\_by\_name(‘sheet\_name’) # read worksheet

excel.sheet\_by\_index()

excel.sheets() -gross sheets

# read worksheet <-> xlsxwriter.Wordbook(‘excel\_name’).add\_sheet(‘sheet\_name’)

booksheet.nrows()

booksheet.ncols()

for i in booksheet.get\_rows():

i == [text:'姓名', text:'性别', text:'年龄', text:'成绩', text:'等级']

content = []

for j in i:

j == text:'姓名', text:'性别', text:'年龄', text:'成绩', text:'等级'

content.append(j.value)

print(content)

['姓名', '性别', '年龄', '成绩', '等级']

['小慕', '男', 10.0, 90.0, '优']

def write():

excel = xlsxwriter.Workbook(‘Excel\_name’) <-> xlrd.wordbook(‘excel\_name’)

worksheet = excel.add\_sheet(‘sheet\_name’)

for row, data in enumerate(content): # 先按列開始填

for cols, v in enumerate(data): # 再按行填

worksheet.wirte(row, cols, v) # 最後填值

# 重頭寫一個新的工作簿

book1 = excel.add\_sheet(‘new\_sheet\_name’)

data = [

(‘excellent’, ‘good’, ‘soso’, ‘bad’),

(10, 7, 5, 3)

]

book1.write\_column(‘A1’, data[0])

book1.write\_column(‘B1’, data[1])

# 圖表製作

chart = book1.add\_chart(‘type’: ‘pie’) # pie圖

# 數據 -title, data, name

chart.add\_series(

categories: new\_sheet\_name!$A$1:$A$4,

values: new\_sheet\_name!$B$1: $B$4,

name: pie\_chart # 定義數據名稱

)

chart.set\_title(‘pie\_title’) # 定義圖表名稱

excel.close() # 保存資料

if \_\_name\_\_ == ‘\_\_main\_\_’:

result = read()

print(result)

write()

# word -> html -> pdf

import pdfkit – pip install htmltopdf

# html -> pdf : pdfkit.from\_file(html, save\_pdf\_path)

# str -> html: pdfkit.from\_string(html based str, save\_pdf\_path)

html = ‘’’

<html>

<head>

</head>

<meta charset=’utf-8’ /> # 頭部注釋

<body>

<p>你好</p> # 內文

</body>

</html>

# url -> pdf: pdfkit.from\_url(url, save\_pdf\_path)

# word -> html : pydocx

from pydocx import PyDocX – pip install pydocx

html = PyDocX.to\_html(word.docx)

f = open(new\_html\_name, ‘w’)

f.write(html)

f.close()

## html -> pdf: pdfkit – pip install htmltopdf

pdfkit.from\_string(html, save\_path\_pdf)

# Reminder -需要手動給予htmltopdf的絕對路徑加入到本地文件中:

path\_wk = r’ C:\Users\user\Desktop\wkhtmltox\bin\wkhtmltopdf.exe'

config = pdfkit.configuration(wkhtmltopdf = path\_wk)

configuration = config

send\_email: smtp, pop3 協議, email內容

Route: log-in email -> write accepter -> write title and content, including 附件 -> send-email -> 是否定時

Ps:

host: smtp -ip, website

port: based on your host like gmail 465, sena 25

local\_hostname: 如果host在主機上, use host as local\_host

Package: smtplib; 定時: schedule

mail\_host = 'ssl://smtp.gmail.com:465'

mail\_user = 'husenior11123@gmail.com'

mail\_password = 'wdycjpatwlswsqdo'

sender = 'husenior11123@gmail.com'

receivers = ['s1041026@gm.ncue.edu.tw']

# 信件格式

message[‘From’] = Header(sender)

message[‘Subject’] = Header(‘python腳本測試’, ‘utf-8’)

def send():

try:

smtp\_obj = smtp.SMTP()

smtp\_obj.connect(ip\_address, 465)

smtp\_obj.login(mail\_name, mail\_passord)

smtp\_obj.sendmail(sender, receivers, message.as\_string())

except smtplib.SMTPException as e:

print(‘error as %s’ %e)

from email.mime.text import MIMEText

MIMEText(‘這是測試’, ‘plain’, ‘utf-8’)

# html

message = MIMEText(‘<p style=’color:red;’>這是測試</p>’, ‘html’, ‘utf-8’)

# 附件

from email.mime.multipart import MIMEMultipart

message = MIMEMultipart()

message = MIMEText(open(‘send.py’, ‘rb’).read(), ‘base64’, ‘utf-8’)

# 定義發送的內容 -使用附件先將內容讀取 -> 定義附件類型(流類型) -> 對附件進行名稱定義 -> 把附件附加到message內

attr = MIMEText(open(‘py\_name’, ‘rb’).read(), ‘base64’, ‘utf-8’)

attr[‘Content\_type’] = ‘application/octet-stream’ # 流的協議

attr[‘Content\_disposition’] = ‘attachment;filename=’”py\_name”’

# 將附件附加在郵件上

message.attach(attr)

message.attach(MIMEText(‘這是一個附帶附件的郵件’, ‘plain’, ‘utf-8’))

# 定時

schedule.every(num\*).seconds.do(func\*)

\*\* 使用while loop, 因為shedule.run\_pending() 沒監測到要執行的程式會結束, 要使它不斷地執行

while 1:

schedule.run\_pending()

time.sleep(1) # 要使它不要如此高頻的監測, 否則造成cpu消耗