```
#  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) 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’) # read excel_obj
    booksheet = excel.sheet_by_name(‘sheet_name’) # read worksheet
                excel.sheet_by_index()
                excel.sheets() -gross sheets
    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')
        worksheet = excel.add_sheet('sheet_name')
        for row, data in enumerate(content):
            for cols, v in enumerate(data):
                worksheet.wirte(row, cols, v)
# 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)
# 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(html, ‘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
```