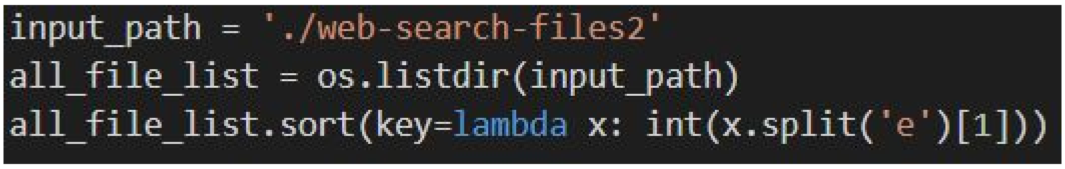
資料結構 PA1

B07901103 電機三 陳孟宏

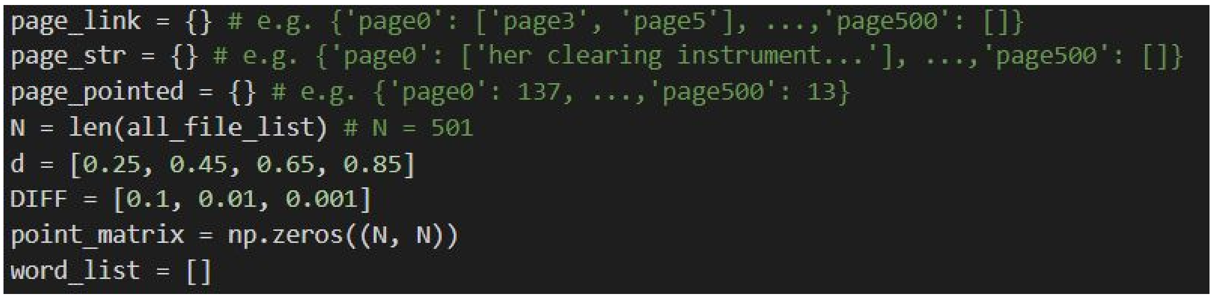
(1) 吃進 input file, 並按照 page 順序輸出



```
input_path = './web-search-files2'
all_file_list = os.listdir(input_path)
all_file_list.sort(key=lambda x: int(x.split('e')[1]))
```

(2) 全域變數

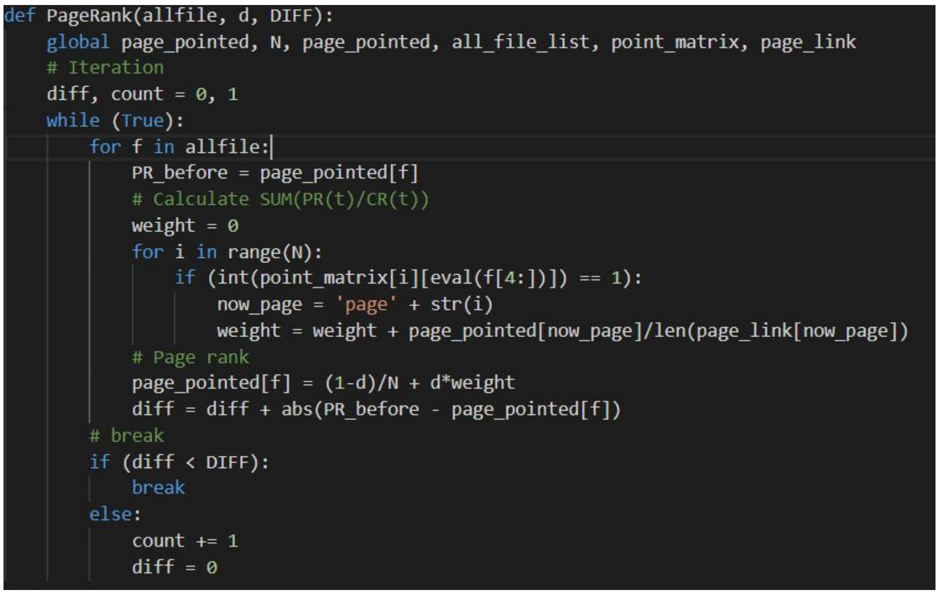
- **page_link: page對應他有連接的頁數**
- **page_str: page中的句子**
- **page_pointed: page的page rank, 每次迭代完都不一樣**
- **N: 總共的頁數**
- **point_matrix: (row, column) = 1 指的是row指向column**
- **word_list: 存入每個單字**



```
page_link = {} # e.g. {'page0': ['page3', 'page5'], ...,'page500': []}
page_str = {} # e.g. {'page0': ['her clearing instrument...'], ...,'page500': []}
page_pointed = {} # e.g. {'page0': 137, ...,'page500': 13}
N = len(all_file_list) # N = 501
d = [0.25, 0.45, 0.65, 0.85]
DIFF = [0.1, 0.01, 0.001]
point_matrix = np.zeros((N, N))
word_list = []
```

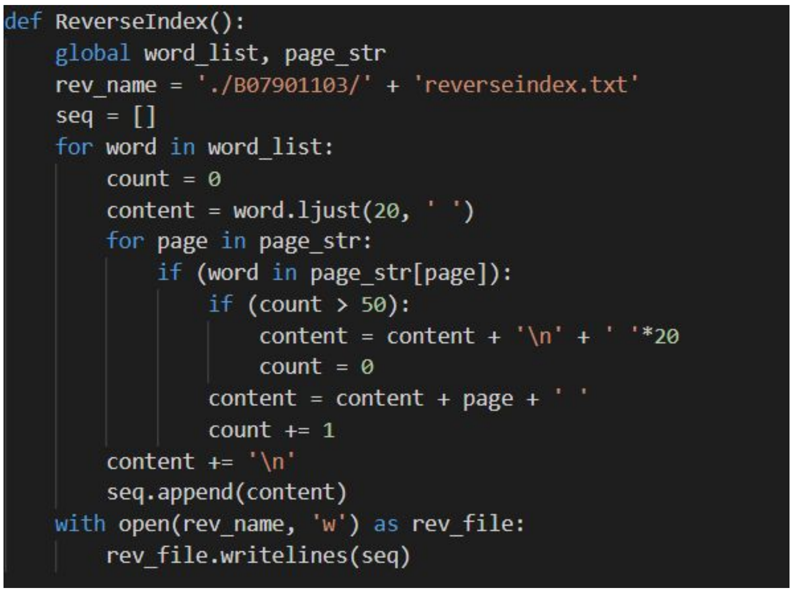
(3) 函數

- **PageRank: 吃進所有頁數, 按照page rank演算法開始迭代, 直到diff 小於一個定值(DIFF)時跳出迴圈, 此時的page_pointed就是每個頁數的page rank組成的dictionary**



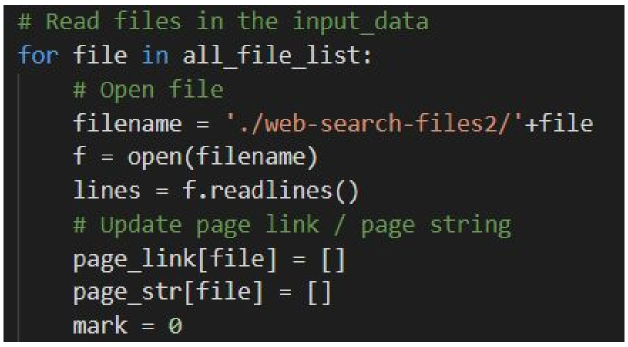
```
def PageRank(allfile, d, DIFF):
    global page_pointed, N, page_pointed, all_file_list, point_matrix, page_link
    # Iteration
    diff, count = 0, 1
    while (True):
        for f in allfile:
            PR_before = page_pointed[f]
            # Calculate SUM(PR(t)/CR(t))
            weight = 0
            for i in range(N):
                if (int(point_matrix[i][eval(f[4:])]) == 1):
                    now_page = 'page' + str(i)
                    weight = weight + page_pointed[now_page]/len(page_link[now_page])
            # Page rank
            page_pointed[f] = (1-d)/N + d*weight
            diff = diff + abs(PR_before - page_pointed[f])
        # break
        if (diff < DIFF):
            break
        else:
            count += 1
            diff = 0
```

- **ReverseIndex: 按照ASCII(大小寫)及字母順序輸出word_list中每個字被哪幾頁提及的檔案, 為了格式美觀, 若超過50頁就跳下一行輸出**



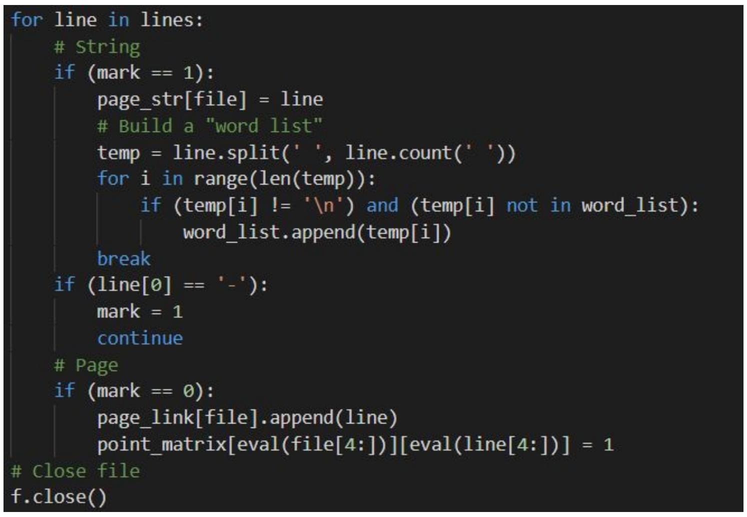
```
def ReverseIndex():
    global word_list, page_str
    rev_name = './B07901103/' + 'reverseindex.txt'
    seq = []
    for word in word_list:
        count = 0
        content = word.ljust(20, ' ')
        for page in page_str:
            if (word in page_str[page]):
                if (count > 50):
                    content = content + '\n' + ' '*20
                    count = 0
                content = content + page + ' '
                count += 1
        content += '\n'
        seq.append(content)
    with open(rev_name, 'w') as rev_file:
        rev_file.writelines(seq)
```

- **main(主程式)**
  - **吃進input file**

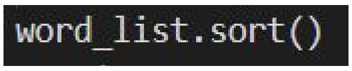


```
# Read files in the input_data
for file in all_file_list:
    # Open file
    filename = './web-search-files2/'+file
    f = open(filename)
    lines = f.readlines()
    # Update page link / page string
    page_link[file] = []
    page_str[file] = []
    mark = 0
```

  - **初始化全域變數的值 (並把word_list排序)**

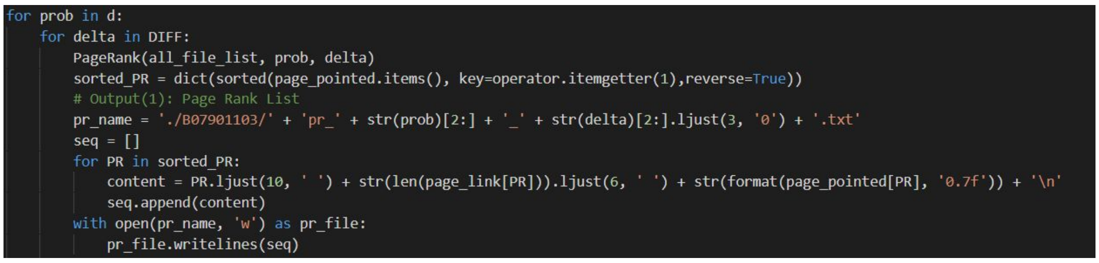


```
for line in lines:
    # String
    if (mark == 1):
        page_str[file] = line
        # Build a "word list"
        temp = line.split(' ', line.count(' '))
        for i in range(len(temp)):
            if (temp[i] != '\n') and (temp[i] not in word_list):
                word_list.append(temp[i])
        break
    if (line[0] == '-'):
        mark = 1
        continue
    # Page
    if (mark == 0):
        page_link[file].append(line)
        point_matrix[eval(file[4:])][eval(line[4:])] = 1
# Close file
f.close()
```



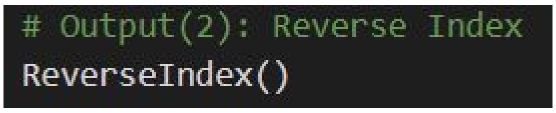
```
word_list.sort()
```

- 第一份輸出檔, 呼叫PageRank()函數, 計算完(d, DIFF)組合的page rank後輸出檔案



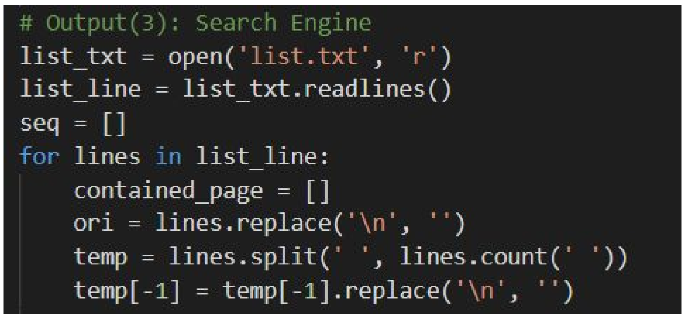
```python
for prob in d:
    for delta in DIFF:
        PageRank(all_file_list, prob, delta)
        sorted_PR = dict(sorted(page_pointed.items(), key=operator.itemgetter(1),reverse=True))
        # Output(1): Page Rank List
        pr_name = './B07901103/' + 'pr_' + str(prob)[2:] + '_' + str(delta)[2:].ljust(3, '0') + '.txt'
        seq = []
        for PR in sorted_PR:
            content = PR.ljust(10, ' ') + str(len(page_link[PR])).ljust(6, ' ') + str(format(page_pointed[PR], '0.7f')) + '\n'
            seq.append(content)
        with open(pr_name, 'w') as pr_file:
            pr_file.writelines(seq)
```

- 第二份輸出檔, 呼叫ReverseIndex()函數



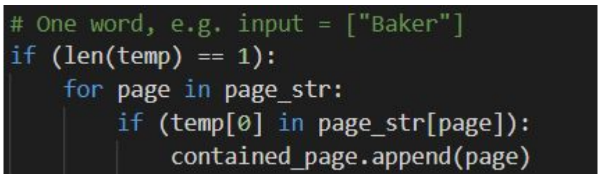
```python
# Output(2): Reverse Index
ReverseIndex()
```

- 第三份輸出檔
    - 首先先把字尾的換行符號('\n')刪除



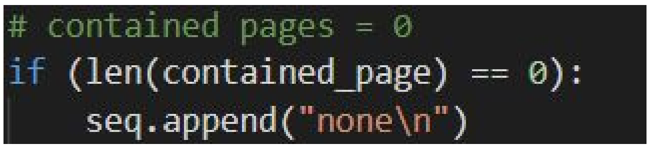
```python
# Output(3): Search Engine
list_txt = open('list.txt', 'r')
list_line = list_txt.readlines()
seq = []
for lines in list_line:
    contained_page = []
    ori = lines.replace('\n', '')
    temp = lines.split(' ', lines.count(' '))
    temp[-1] = temp[-1].replace('\n', '')
```

    - 情況一: 搜尋字數為一個單字



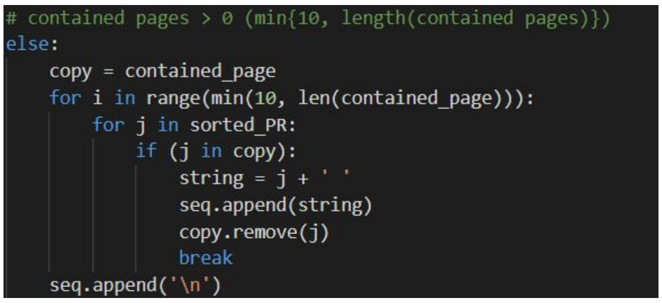
```python
# One word, e.g. input = ["Baker"]
if (len(temp) == 1):
    for page in page_str:
        if (temp[0] in page_str[page]):
            contained_page.append(page)
```

    -
        - 1. 該單字沒有在任何一頁中, 輸出"none"



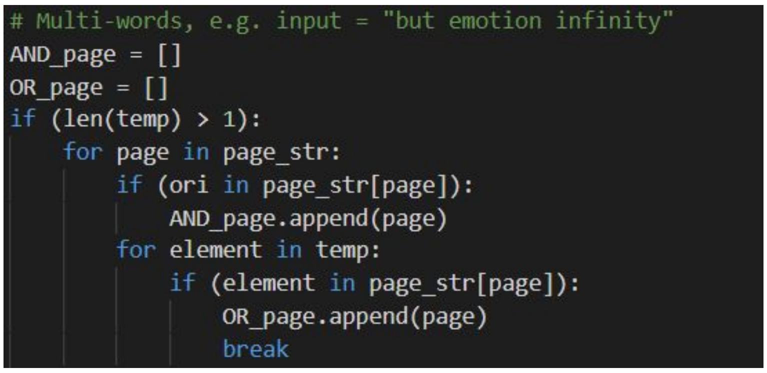
```python
# contained pages = 0
if (len(contained_page) == 0):
    seq.append("none\n")
```

        - 2. 該單字在某些頁中, 按照page rank輸出最多10頁



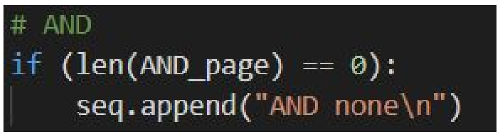
```python
# contained pages > 0 (min{10, length(contained pages)})
else:
    copy = contained_page
    for i in range(min(10, len(contained_page))):
        for j in sorted_PR:
            if (j in copy):
                string = j + ' '
                seq.append(string)
                copy.remove(j)
                break
    seq.append('\n')
```

- **情況二: 搜尋字數超過一個單字**

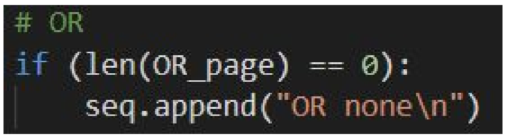


```python
# Multi-words, e.g. input = "but emotion infinity"
AND_page = []
OR_page = []
if (len(temp) > 1):
    for page in page_str:
        if (ori in page_str[page]):
            AND_page.append(page)
        for element in temp:
            if (element in page_str[page]):
                OR_page.append(page)
                break
```

- **1. AND / OR, 該單字沒有在任何一頁中, 輸出 "none"**

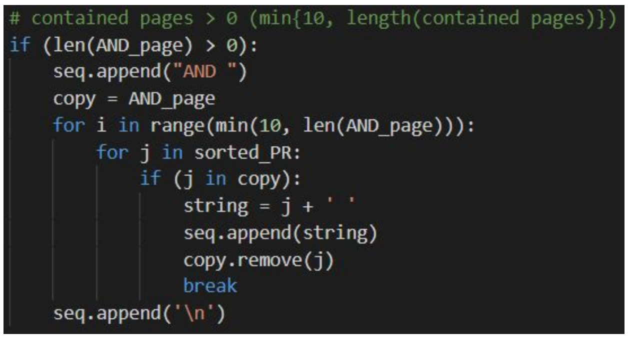


```python
# AND
if (len(AND_page) == 0):
    seq.append("AND none\n")
```

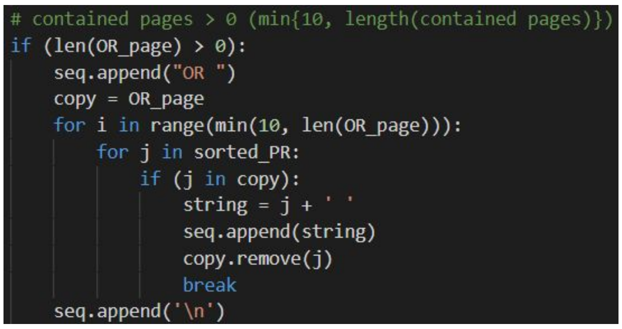


```python
# OR
if (len(OR_page) == 0):
    seq.append("OR none\n")
```

- **2. AND / OR, 該單字在某些頁中, 按照page rank 輸出最多10頁**



```python
# contained pages > 0 (min{10, length(contained pages)})
if (len(AND_page) > 0):
    seq.append("AND ")
    copy = AND_page
    for i in range(min(10, len(AND_page))):
        for j in sorted_PR:
            if (j in copy):
                string = j + ' '
                seq.append(string)
                copy.remove(j)
                break
    seq.append('\n')
```



```python
# contained pages > 0 (min{10, length(contained pages)})
if (len(OR_page) > 0):
    seq.append("OR ")
    copy = OR_page
    for i in range(min(10, len(OR_page))):
        for j in sorted_PR:
            if (j in copy):
                string = j + ' '
                seq.append(string)
                copy.remove(j)
                break
    seq.append('\n')
```

(4) 複雜度

- **Time complexity: O(N*W), 其中N為page數, W為page連到的頁數**
- **Space complexity: O(N*N), 在建立page_matrix看誰連到誰的矩陣耗費最大**