**國立高雄科技大學**

**資訊工程系**

**依使用者需求之**

**公車路線規劃**

指導教授：黃淵科教授

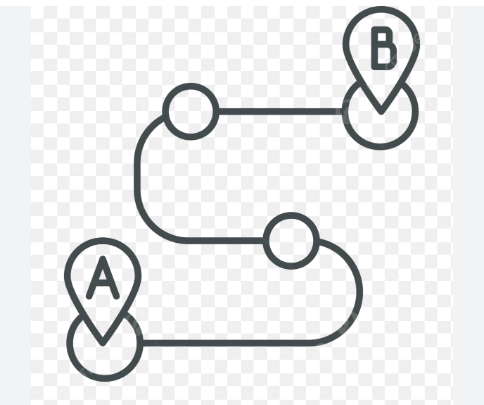
學生：C110151118 王議乾 C110151127 古育加

C110151155 李杰峰 C110151168 許凱雋

**摘要**

現代城市交通擁擠與汙染已成為一大挑戰。公車作為大眾運輸工具,可有效緩解這些問題，透過簡單友好的介面,使用者可輕鬆規劃出最佳路線,避免時間和燃料的浪費。這不僅提高公車系統的吸引力,也間接減少私家車使用,促進城市的可持續發展。該專案整合了演算法、API開發和前端設計,展現了跨領域技術的實用價值。

本研究旨在開發一個路線規劃系統,透過後端的廣度優先搜尋(BFS)演算法,計算出最短路徑,並整合FastAPI框架,提供前端網頁介面,使用戶能夠輕鬆查詢起點和終點之間的最佳公車路線



目錄

[**摘要** 2](#_Toc169709348)

[**緒論** 4](#_Toc169709349)

[(一)理論推演 4](#_Toc169709350)

[1.1 Python 4](#_Toc169709351)

[**1.2 公車資料創建** 5](#_Toc169709352)

[**1.3如何規劃路線** 5](#_Toc169709353)

[**1.4利用python 的fastapi 框架去連接前端網頁** 6](#_Toc169709354)

[(二)實作與模擬過程 6](#_Toc169709355)

[2.1公車路線創建 6](#_Toc169709356)

[2.2站牌資訊創建 7](#_Toc169709357)

[2.3公車路線建立 7](#_Toc169709358)

[2.4 BFS公車路線規劃演算法 8](#_Toc169709359)

[2.5 FastApi框架 10](#_Toc169709360)

[(三)測試與模擬分析 15](#_Toc169709361)

[(四)結論 18](#_Toc169709362)

[(五)參考資料 18](#_Toc169709363)

# 緒論

現代都市生活中,交通擁擠和環境污染已成為一大挑戰。私家車的過度使用不僅導致道路壅塞,延長通勤時間,更產生大量廢氣排放,對環境造成沉重負擔。有效利用大眾運輸系統被視為紓解這些問題的關鍵之道。

其中,公車作為城市主要的大眾運輸工具之一,具有能源效率高、基礎設施成本低等優勢,備受重視。然而,公車路線的複雜性和班次時間的不確定性,卻也給乘客的出行帶來了諸多挑戰。傳統的公車資訊系統大多僅提供靜態路線圖和時刻表,無法根據乘客實際需求,提供客製化的最佳路線方案。乘客難以權衡不同路線的車程時間、換乘次數和車資費用等因素,錯失了出行的最優化方案。

有鑑於此,本研究旨在開發一款智能的公車路線規劃系統,為乘客提供量身定製的最佳路線解決方案。系統首先透過程式建立公車路線網路資料,包括各站點座標和站間距離等,並存入CSV檔案中,以備後續路徑搜索之用。

接著,系統採用廣度優先搜尋(BFS)演算法,根據使用者輸入的起訖站點,計算出所有可行路線。同時,使用者可指定額外的優化條件,如轉乘次數最少、車程時間最短或總車資最低等,系統將依此條件對候選路線進行排序,呈現給使用者最佳方案。

我們希望透過直覺友好的網頁介面,讓一般大眾無需專業知識,也能輕鬆使用本規劃系統,充分滿足不同出行需求。本系統的推廣應用,有望提高公車的吸引力,鼓勵更多民眾使用大眾運輸,間接減少私車使用量,為都市環境的可持續發展貢獻一份力量。

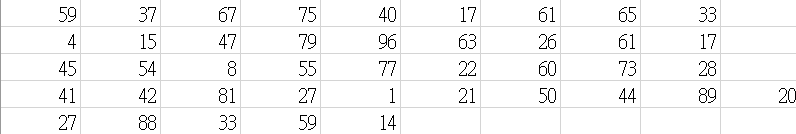
# (一)理論推演

1.1 Python

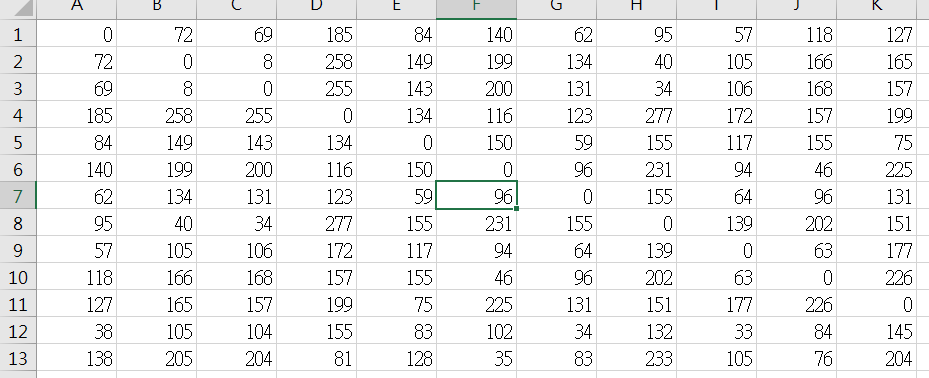
Python是一種高階語言，以其簡單易學的語法和跨平台特性而聞名。作為解釋型語言，Python在運行時逐行解釋並執行代碼，這提供了更大的靈活性和便利性。Python擁有豐富的標準庫和第三方庫（如NumPy、Pandas、Matplotlib等），可以極大地擴展其功能。它支持多種編程範式，包括面向對象和函數式編程，並且是動態類型語言，使代碼更加靈活。自動內存管理功能避免了內存泄漏問題，而強大的社區支持則提供了大量的資源和協作機會，這些特點使Python成為一種強大而靈活的編程語言，適合用於Web開發、數據分析、人工智能、科學計算、腳本編寫和自動化等各種應用。

**1.2 公車資料創建**

公車資料分為兩個部分，一個是公車路線(bus.csv)，一個是站牌與站牌之間通勤所需時間(stop.csv)，而路線有100條，站牌有100個，路線與站牌都是以1號-100號命名。

公車路線設定為經過隨機5-10個站牌，每條路線經過站牌為隨機，然後將資料存進csv檔。

(**圖)部分公車路線資料**

站牌資料的部分，隨機先生成100個座標點，透過點跟點之間的距離計算出通勤時間，此資料為100\*100的對稱矩陣，，然後將資料存進csv檔。

**(圖)部分公站牌資料**

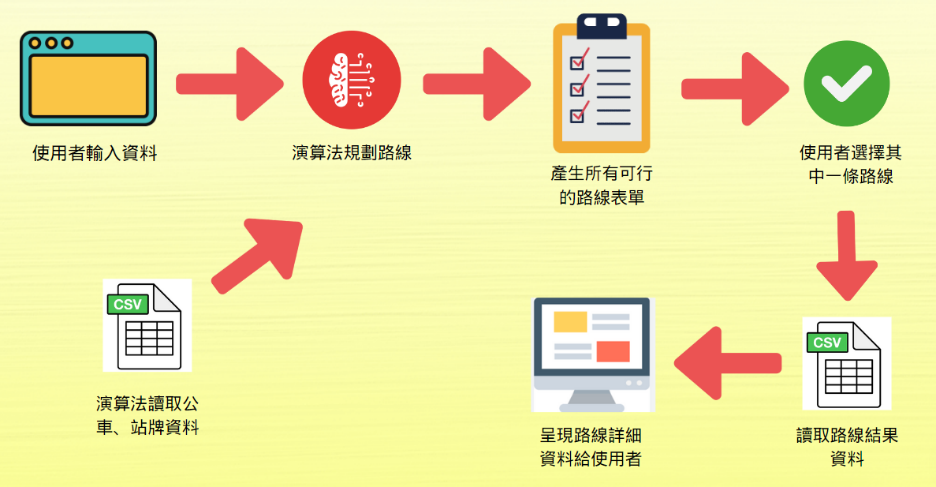
**1.3如何規劃路線**

本公車路線規劃選擇使用BFS算法，因為原因是BFS簡單高效、易於操作和擴展。

BFS能夠保證在無權無向圖(如公車路線網路)中找到任意兩點之間的最短路徑,同時其最壞情況時間複雜度O(|V|+|E|)對於一般規模的問題而言,執行效率是可以接受的。更重要的是,BFS非常簡單直觀,只需使用一個隊列來存儲待探索節點,實現和維護都相對容易。

這種簡單性和高效性意味著BFS非常適合在資源有限的嵌入式系統或移動端上運行, 夠輕鬆地找出所有可行路徑,再根據不同條件(換乘次數、行車時間、費用等)進行二次篩選和排序,完美滿足了本系統需要為用戶提供多條備選路線的需求。

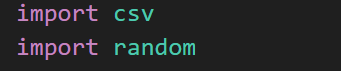
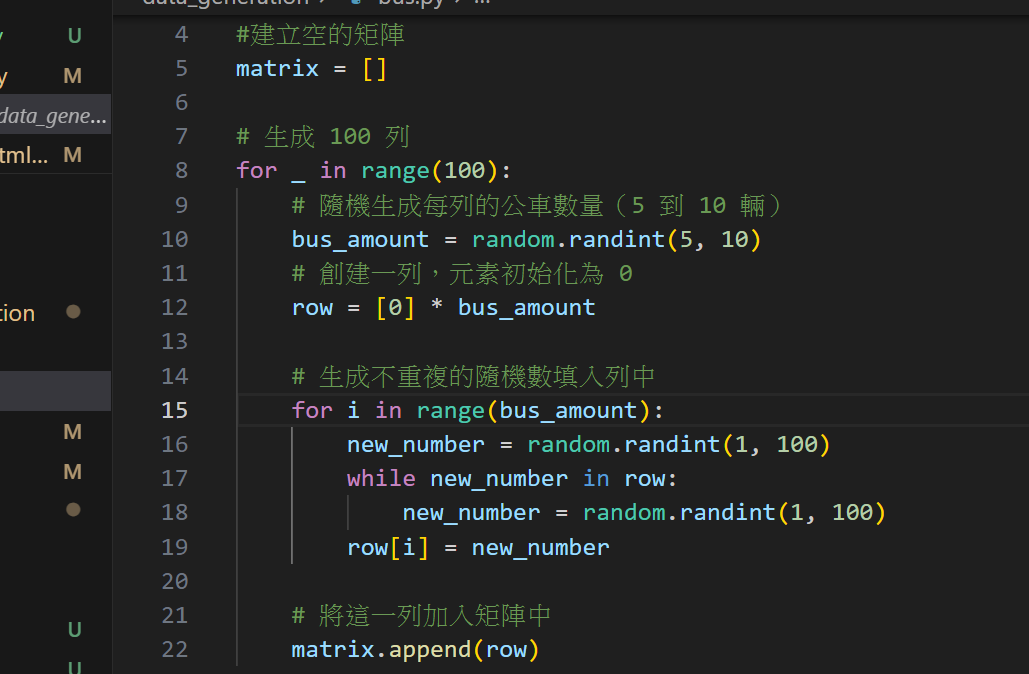
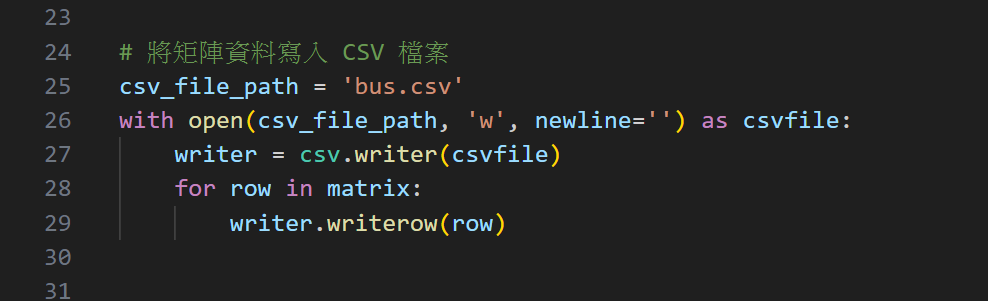
* 1. **利用python 的fastapi 框架去連接前端網頁**

****FastAPI作為前端框架的選擇,主要基於其簡單高效的特點。它提供極快的響應速度、自動化API文檔、友好的Python化開發體驗,並遵循標準化API設計規範,使得前端開發工作變得簡單直觀,極大提升了開發效率。

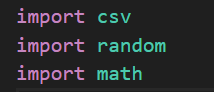
**(圖)整體執行步驟**

# (二)實作與模擬過程

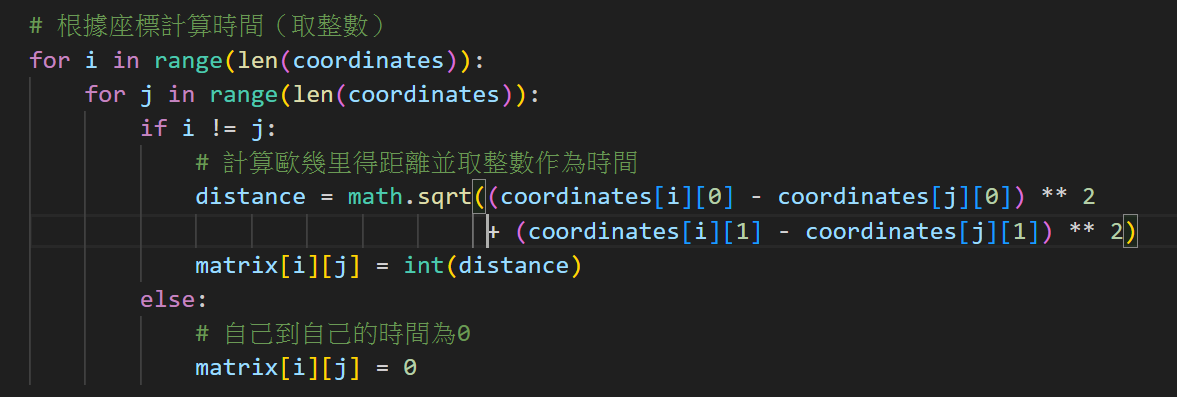
2.1公車路線創建

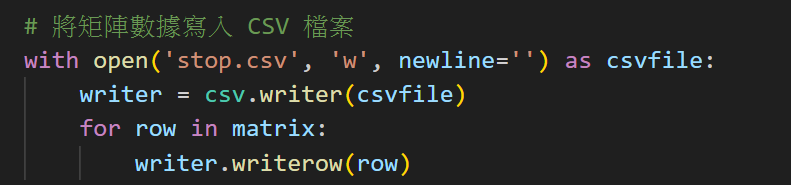
* + 1. 導入用於讀寫csv檔案的模組(csv)，以及生成隨機亂數的模組(random)
    2. 先建立一個空矩陣matrix，接著使用 Random函數，生成一條條路線，存進matrix
    3. 接著將matrix的資料存進bus.csv中

2.2站牌資訊創建

導入模組

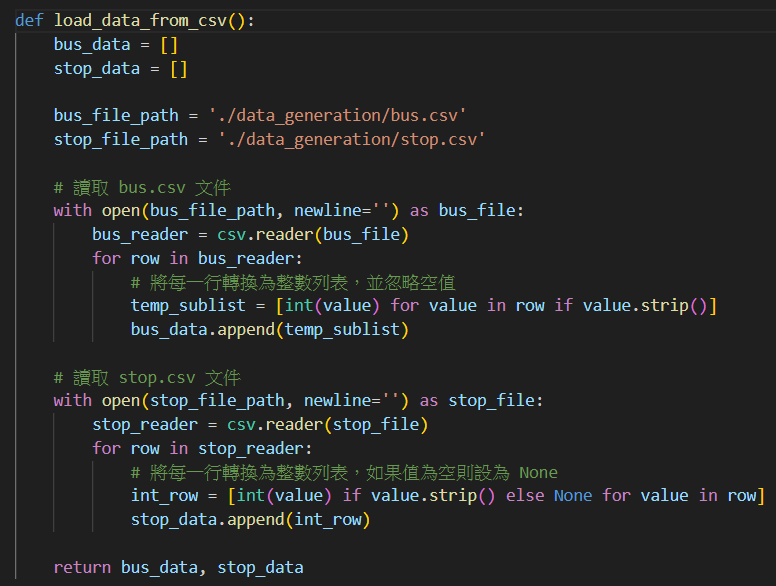
隨機生成100個點，範圍在(0,0)到(250,250)。

然後計算點與點之間的距離，存進100\*100矩陣matrix裡面。

將資料存進csv檔案裡。

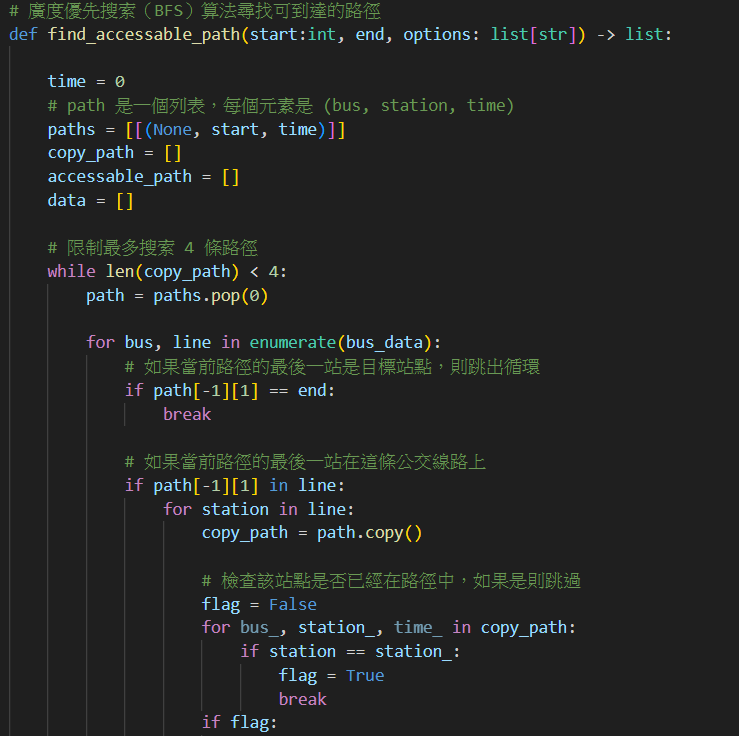
2.3公車路線建立

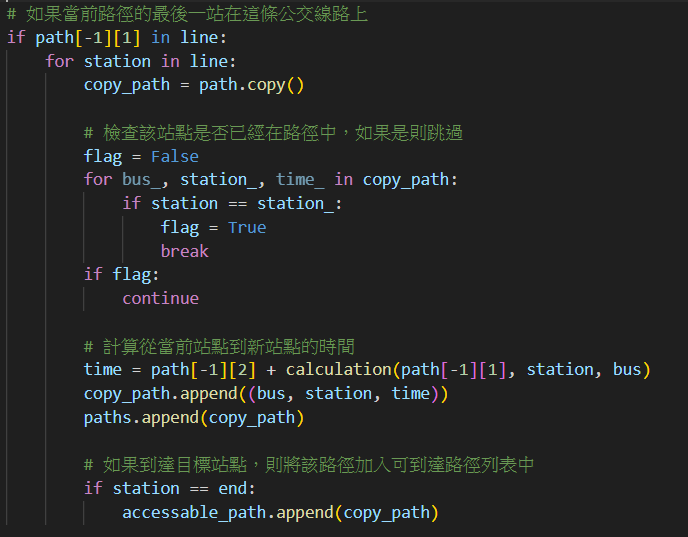
先讀取剛才建立的資料，透過load\_data\_from\_scv函數，將資料存進bus\_data、stop\_data。

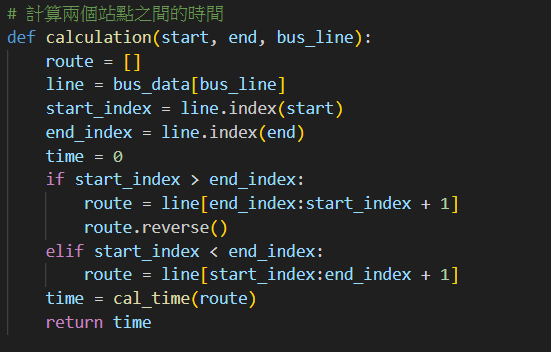


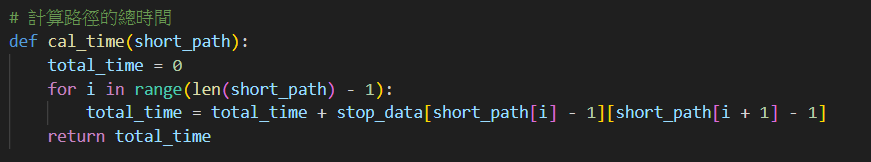
* 1. BFS公車路線規劃演算法

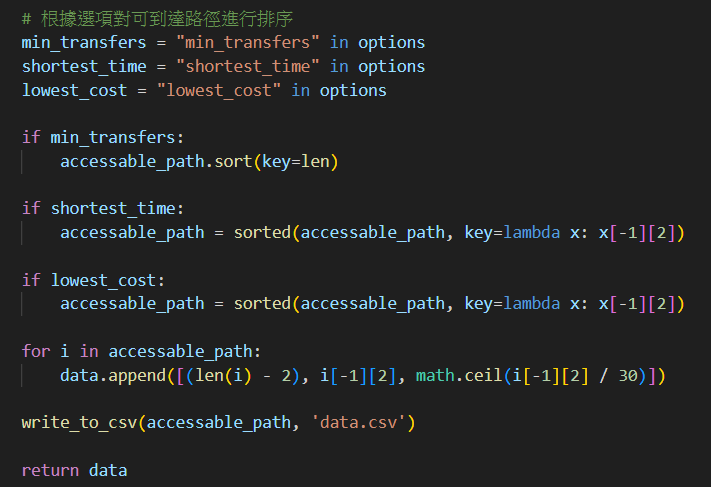
這個find\_accessable\_path函數是接收前端使用者輸入的起、終點還有，使用while循環在找到最多4條路徑前持續搜索，之所以是最多4條是因為如果轉成次數超過3次那這個路線就不適合被選擇，迴圈執行過程是從paths中取出一條路徑，遍歷所有公交線路。如果當前路徑的最後一站在某條公交線路上，則檢查新站點是否已在路徑中，若未重複則使用calculation 函數計算時間並生成新路徑加入paths。若新站點為終點站則將這條包含該站點的路徑加入到可到達路徑列表 accessable\_path 中。

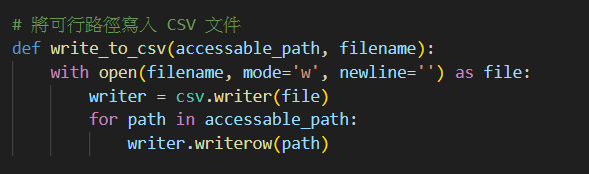




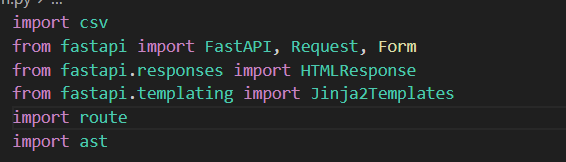
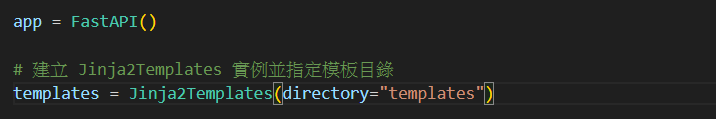
calculation 函數根據start和end的索引確定路徑方向，並透過bus\_line生成路徑列表 route。接著，利用 cal\_time 函數計算該路徑的總時間 time。

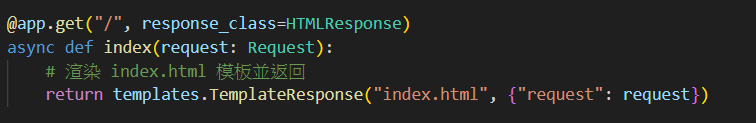
cal\_time 函數通過遍歷路徑中的每對相鄰站點，累加 stop\_data 中的時間值，返回總時間。

此處是根據使用者的選項對所有可以從起點到終點的路線「accessable\_path」進行排序。「min\_transfers」使路徑按換乘次數排序，「shortest\_time」按總時間(秒)排序，「lowest\_cost」按時間所得出的成本排序。排序後，生成結果列表 data，包含換乘次數、總時間和成本（每30秒為一元）。最後將結果寫入data.csv檔案並返回 data 列表。

write\_to\_csv函數是將「accessable\_path」的每一筆資料存進data.csv裡。

2.5 FastApi框架

首先導入所需的模組，「csv」 用於讀取 CSV 文件，「FastAPI 」用於建立網頁應用，「HTMLResponse」 和 「Jinja2Templates」 用於渲染 HTML 頁面，「ast」 用解析csv檔案的資料將字串轉為list，創建一個 FastAPI 應用實例 「app」以及「Jinja2Templates 」實例，並指定模板檔案的目錄為 "templates"。

下圖為處理網站首頁 ("/") 請求的函數 index。當用戶訪問網站首頁時，這個函數會顯示 index.html 頁面，把請求信息傳遞給模板來生成頁面內容，然後把生成的網頁返回給用戶。

index.html是讓使用者用下拉式選單選取起點以及終點還有額外選項（最少轉車次數、最短時間或最短花費）來推薦最適合的路線，點擊“查詢”按鈕將表單數據提交到指定的/submit/路徑。

index.html(1)

當使用者打開界面時，起點和終點的下拉選單會初始化，並且提交按鈕處於禁用狀態。只有當用戶選擇了起點、終點，以及其中一個額外選項（如轉車次數最少、最短時間或最短花費）時，提交按鈕才會被啟用，而如果系統偵測到起點和終點是一樣的，會給予相應的提示。

以下程式碼讓使用者在使用系統時的選擇是有效和完整的，從而提升了系統的可靠性和用戶體驗。

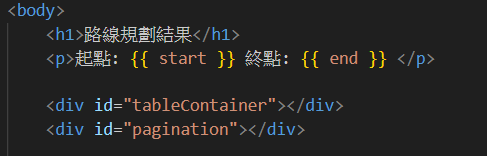
index.html(2)

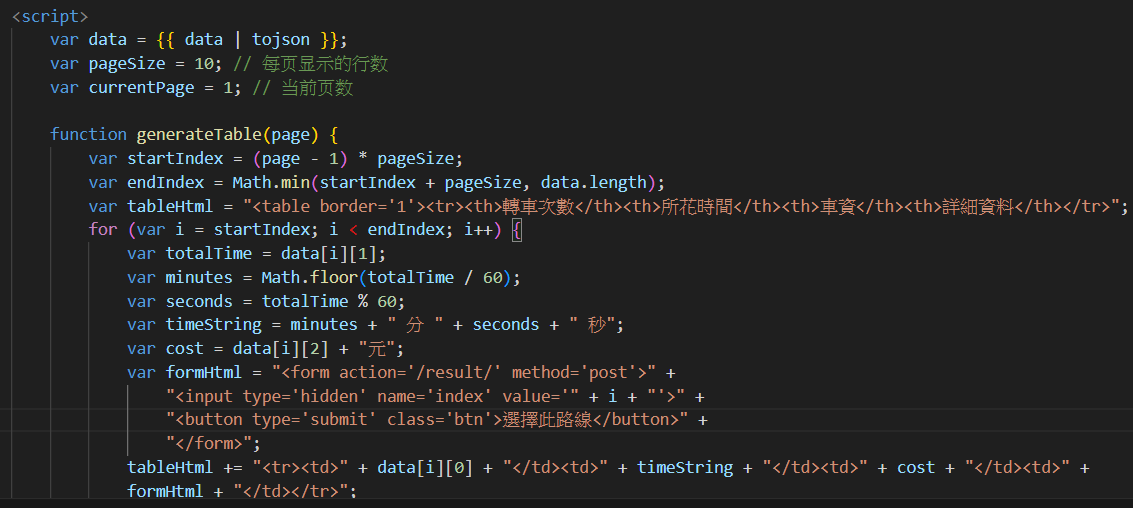
index.html(3)

Submit\_form用於處理公車路線推薦表單的提交。當使用者在index.html按下「查詢」，提交表單後，伺服器會接收起點、終點和選擇的額外選項，並使用撰寫著公車路線演算法 的「route.py」檔案中的find\_accessable\_path函數計算所有能抵達終點的路徑資料存入data，與此同時「find\_accessable\_path」函數會將可行路線放進「data.csv」檔案裡。然後，系統會將這些數據和計算結果傳遞給submit.html模板，生成包含路線推薦結果的HTML頁面，並返回給用戶，從而確保用戶獲得完整和準確的路線資訊



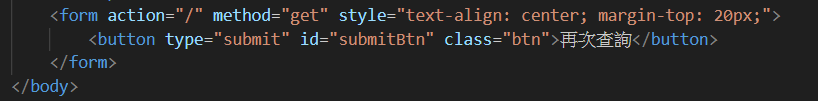
「submit.html」使用javascript結合html顯示起點和終點資訊。資料以表格形式呈現，每頁顯示10行，包含轉車次數、所花時間、車資和詳細資料按鈕。時間格式轉換為“分”和“秒”，車資以“元”顯示。使用generateTable函數根據當前頁數生成資料表格，使用generatePagination函數生成分頁導航，顯示當前頁數和總頁數，並提供“上一頁”和“下一頁”的連結。changePage函數根據用戶點擊更新當前頁數，重新生成表格和分頁導航。每個結果行包含一個表單，點擊“選擇此路線”按鈕提交該路線的索引至伺服器進一步處理。頁面底部有一個返回首頁的按鈕，允許用戶重新查詢。



submit.html (1)

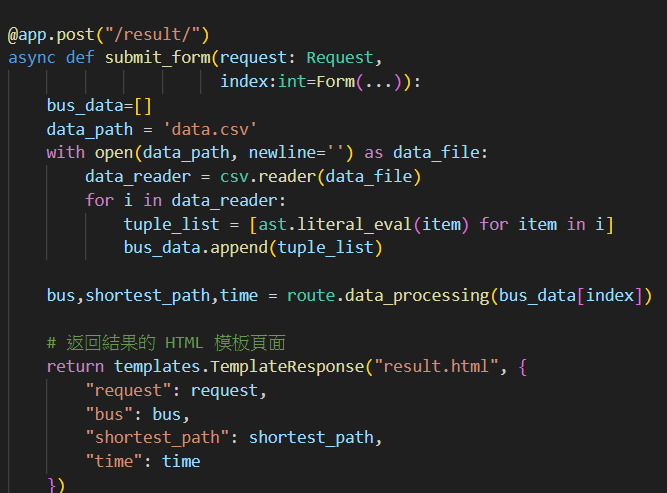
submit.html(2)

submit.html(3)

submit.html (4)

接收選擇路線的編號，讀取data.csv中使用者選擇的路線並將資料的型態轉換成list，

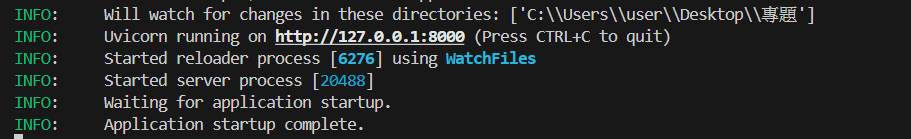
使用「route.py」檔案中的data\_processing函數來取得路線的詳細資料。比如在哪個站牌上車，在哪個站牌下車後轉乘幾號公車…，將資料傳送到result.html。



result.html顯示公車路線詳細結果，呈現詳細路線資訊。

# (三)測試與模擬分析

首先在終端機上執行uvicorn main:app –reload，並在瀏覽器連結輸入http://127.0.0.1:8000/

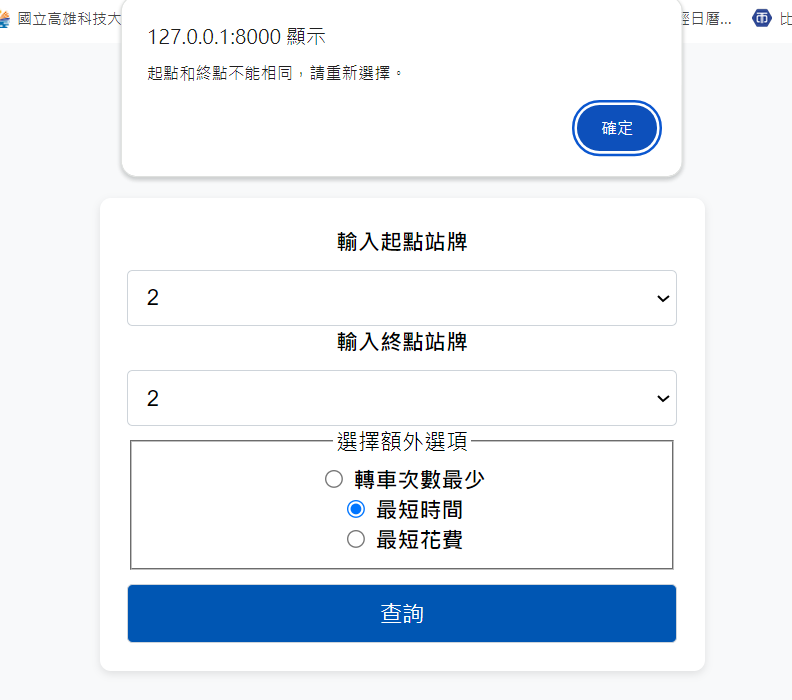


發送get請求成功。

此為index.html的畫面，讓使用者輸入起點、終點，以及額外的選項條件，如果沒有

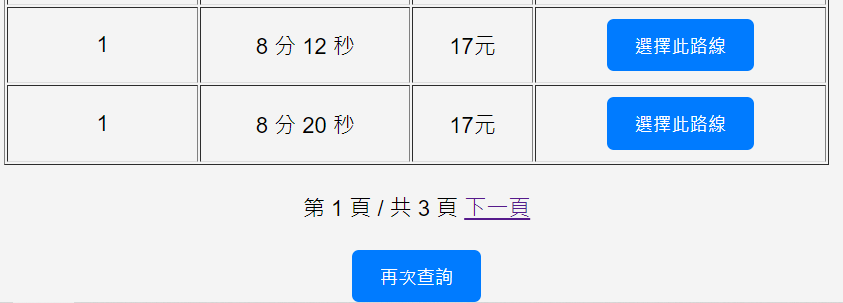
都輸入完畢那按鈕會呈現灰色，使用者點擊”查詢”也沒有反應，

如果起點與終點是同一個站牌，會被系統提示起重新選擇。



按下查詢後，會以表單方式呈現所有結果，並以使用者選擇的額外選項進行排序

由於結果可能會有20個以上，因此用了分頁功能，10個為一頁，按下再次查詢可回

到首頁。

再任意個表格按下”選擇此路線”，就可以到我們陸系詳細資料畫面，顯示搭乘幾號公車在幾號站牌上車、下車，按下”再次規劃路線”可回到首頁。

# (四)結論

本專題通過實現基於廣度優先搜索（BFS）算法的公車路線規劃系統，並將其透過 FastAPI 與使用html、javascript以及css的前端網頁進行連接。

廣度優先搜索（BFS）是一種遍歷或搜索樹或圖的算法。其特點是從根節點開始，沿著樹的深度遍歷節點，直到找到目標節點。BFS 可以有效處理較大規模的路線網絡，因此非常適合應用於公車路線規劃這類需要找到最短路徑的問題。

而FastAPI 是一個現代的、快速（高性能）的 Web 框架，用於基於 Python 標準類型提示構建 API。其性能可與 Node.js 和 Go 相媲美，並且比許多其他 Python 框架快得多，並且以其高性能的特性確保了系統能夠快速響應用戶請求，並且其自帶的數據驗證功能提高了系統的可靠性和安全性。

在此基礎上，我們可以進一步優化和擴展系統，例如：接入交通數據，模擬真實環境，動態調整路線建議；使用地圖 API 將路線可視化，提供直觀的導航體驗；並增加用戶賬戶系統，讓用戶能夠保存常用路線和歷史搜索，根據歷史數據提供個性化建議。

# (五)參考資料

[1]

[快速上手，簡單易懂：Python新手的FastAPI之旅](https://medium.com/seaniap/%E5%BF%AB%E9%80%9F%E4%B8%8A%E6%89%8B-%E7%B0%A1%E5%96%AE%E6%98%93%E6%87%82-python%E6%96%B0%E6%89%8B%E7%9A%84fastapi%E4%B9%8B%E6%97%85-ebd09dc0167b)

[2]

[【Day33】[演算法]-深度優先搜尋DFS與廣度優先搜尋BFS](https://ithelp.ithome.com.tw/articles/10281404)

[3]

[Jinja2模板引擎的用法](https://www.osgeo.cn/python-tutorial/webpub-jinja2.html)

[4]

[ChatGPT](https://chat.openai.com/)

[5]

[JavaScript 教程](https://www.runoob.com/js/js-tutorial.html)  
[6]

[學習 HTML ：指南與教學](https://developer.mozilla.org/zh-TW/docs/Learn/HTML)