OS\_HW2程式說明文件

1. 開發環境
2. 處理器：Intel(R) Core(TM) i7-10750H
3. 記憶體：8.00 GB
4. 系統類型：64 位元作業系統，x64 型處理器
5. Windows 規格：Windows 10 家用版
6. IDE：Visual Studio Code
7. 使用語言：Python
8. 實作方法和流程
9. 讀檔：

首先請使用者輸入檔案名稱，我會先把Method、Time Slice分別記錄起來，接者呼叫ReadProcess()，ReadProcess()會先讀取標題(ID, CPU Burst, Arrival Time, Priority)，之後再將process資訊依序儲存在processList中。

1. 寫檔：

我寫了兩個function(PrintResult(), PrintAllResult())，前者是給method 1-5使用，而後者專門處理method 6(print ALL)，PrintResult()會先寫入method的名稱，再寫入該方法的甘特圖，接著依序PID寫入waiting time, turnaround time，並做適當排版，PrintAllResult()其實大同小異，只是因為同時要處理5個method，所以我分別使用了5個class所產生的object，在處理時也比較方便。

1. method 1： # FCFS (First Come First Serve)

首先建立FCFS\_Process\_List，並複製processList的內容，代表input資訊，接者利用class FCFS建立FCFS\_Simulate，之後呼叫FCFS\_Simulate.Start()進行CPU排程，FCFS\_Simulate.Start()會先依Arrival Time排序所有process，如果Arrival Time相同，再利用PID排序，確認順序後執行CheckProcess()，CheckProcess()的目的是將抵達的process放進Waiting\_Queue裡面，並將放入Waiting\_Queue的process從FCFS\_Process\_List中pop出來，代表process已處裡過，接者執行RunProcess()，RunProcess()從Waiting\_Queue中pop process出來並執行，執行時將process的CPU\_Burst\_Remaining減一，並記錄PID(0-9, A-Z) 於甘特圖，如果process的CPU\_Burst\_Remaining等於0，代表程序執行完畢，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time，接者將完成的process加入至Done\_List中，完成FCFS\_Simulate.Start()後，最後呼叫PrintResult()將結果寫檔。

1. method 2： # RR (Round Robin)

首先建立RR\_Process\_List，並複製processList的內容，代表input資訊，接者利用class RR建立RR\_Simulate，之後呼叫RR\_Simulate.Start()進行CPU排程，RR\_Simulate.Start()會先依Arrival Time排序所有process，如果Arrival Time相同，再利用PID排序，確認順序後執行CheckProcess()，CheckProcess()的目的是將抵達的process放進Waiting\_Queue裡面，並將放入Waiting\_Queue的process從RR\_Process\_List中pop出來，代表process已處裡過，在這裡要特別處裡process尚未完成但timeout的情形，因為process的Time\_Slice等於0，所以要把該process再重新放入Waiting\_Queue中，接者執行RunProcess()，RunProcess()從Waiting\_Queue中pop process出來並執行，執行時將process的CPU\_Burst\_Remaining減1，也把Time\_Slice減1，並記錄PID(0-9, A-Z) 於甘特圖，如果process的Time\_Slice等於0，CPU\_Burst\_Remaining亦為0，代表程序執行完畢，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time(Turnaround\_Time需先算出)，接者將完成的process加入至Done\_List中，如果process的CPU\_Burst\_Remaining為0，但Time\_Slice不為0，代表程序提前結束，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time，接者將完成的process加入至Done\_List中，完成RR\_Simulate.Start()後，最後呼叫PrintResult()將結果寫檔。

1. method 3： # SRTF (Shortest Remaining Time First)

首先建立SRTF\_Process\_List，並複製processList的內容，代表input資訊，接者利用class SRTF建立SRTF\_Simulate，之後呼叫SRTF\_Simulate.Start()進行CPU排程，SRTF\_Simulate.Start()會先依Arrival Time排序所有process，如果Arrival Time相同，再利用CPU\_Burst排序，如果還是一樣，再利用PID排序，確認順序後執行CheckProcess()，CheckProcess()的目的是將抵達的process放進Waiting\_Queue裡面，並將放入Waiting\_Queue的process從SRTF\_Process\_List中pop出來，代表process已處裡過，這裡有個重點要注意，因為SRTF是Preemptive，所以如果有process剩餘的時間小於正在執行的process的剩餘時間，就進行搶奪，原本正在執行的process進入Waiting\_Queue，另外，如果剩餘時間相同，就比較Arrival Time，如果還是相同，再比較PID，如果成立，依然可以進行搶奪，接者執行RunProcess()，RunProcess()從Waiting\_Queue中pop process出來並執行，執行時將process的CPU\_Burst\_Remaining減一，並記錄PID(0-9, A-Z)於甘特圖，如果process的CPU\_Burst\_Remaining等於0，代表程序執行完畢，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time，接者將完成的process加入至Done\_List中，完成SRTF\_Simulate.Start()後，最後呼叫PrintResult()將結果寫檔。

1. method 4： # PPRR (Preemptive Priority Round Robin)

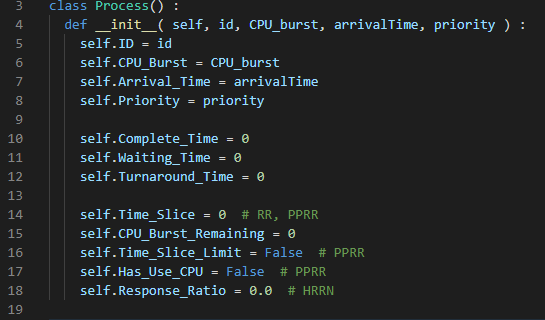
首先建立PPRR\_Process\_List，並複製processList的內容，代表input資訊，接者利用class PPRR建立PPRR\_Simulate，之後呼叫PPRR\_Simulate.Start()進行CPU排程，PPRR\_Simulate.Start()會先依Arrival Time排序所有process，如果Arrival Time相同，再利用Priority排序，還是一樣再利用PID排序，確認順序後執行CheckProcess()，CheckProcess()的目的是將抵達的process放進Waiting\_Queue裡面，並將放入Waiting\_Queue的process從PPRR\_Process\_List中pop出來，代表process已處裡過，在這裡我會先重新排序一次，依據process的Priority, Has\_Use\_CPU(是否曾經執行過), Arrival Time, process ID排序，接者我會把Waiting\_Queue中與正在執行的process的Priority相同者放入Same\_Priority\_Queue，代表等一下會優先處理，因為PPRR是以Priority優先處理為原則，這裡有個重點要注意，因為PPRR是Preemptive，所以如果有process的Priority小於正在執行的process的Priority，就進行搶奪，原本正在執行的process進入Waiting\_Queue，另外，如果Priority相同，就比較Arrival Time，如果還是相同，再比較PID，如果成立，依然可以進行搶奪，接者執行RunProcess()，RunProcess()優先從Same\_Priority\_Queue pop process出來執行，其次才是Waiting\_Queue，執行時將process的CPU\_Burst\_Remaining減1，也把Time\_Slice減1，還有Has\_Use\_CPU設為True，並記錄PID(0-9, A-Z) 於甘特圖，如果process的CPU\_Burst\_Remaining等於0，代表程序執行完畢，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time，接者將完成的process加入至Done\_List中，完成PPRR\_Simulate.Start()後，最後呼叫PrintResult()將結果寫檔。

1. method 5： # HRRN (High Response Ratio Next)

首先建立HRRN\_Process\_List，並複製processList的內容，代表input資訊，接者利用class HRRN建立HRRN\_Simulate，之後呼叫HRRN\_Simulate.Start()進行CPU排程，HRRN\_Simulate.Start()會先依Arrival Time排序所有process，如果Arrival Time相同，再利用PID排序，確認順序後執行CheckProcess()，CheckProcess()的目的是將抵達的process放進Waiting\_Queue裡面，並將放入Waiting\_Queue的process從HRRN\_Process\_List中pop出來，代表process已處裡過，另外，也要計算Waiting\_Queue中各個process的Response\_Ratio，利用Current\_Time減去process的Arrival\_Time得到目前等待時間，再加上CPU\_Burst，最後再同除CPU\_Burst，所得結果即為Response\_Ratio，之後重新排序一次，依據Response\_Ratio(越大者越前)，Arrival\_Time, PID，接者執行RunProcess()，RunProcess()從Waiting\_Queue中pop process出來並執行，執行時將process的CPU\_Burst\_Remaining減1，並記錄PID(0-9, A-Z) 於甘特圖，如果process的CPU\_Burst\_Remaining等於0，代表程序執行完畢，記錄完成時間，並計算出Turnaround\_Time, Waiting\_Time，接者將完成的process加入至Done\_List中，完成HRRN\_Simulate.Start()後，最後呼叫PrintResult()將結果寫檔。

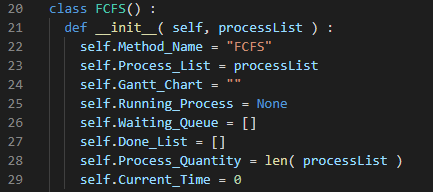
1. method 6： # ALL

其實就是method 1-5的整合，依序執行上述的程式碼，最後呼叫PrintAllResult()將結果寫檔。

1. 使用的Data Structure
2. Process：

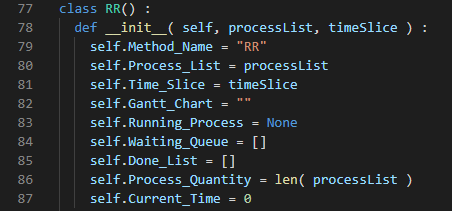
前4項為讀檔而來的資料，型別是integer，中間3項為記錄結果之用途，型別是integer，後面數項是特別處理各方法時所需的資料，例如：在RR, PPRR時會需要用到Time\_Slice，型別是integer，CPU\_Burst\_Remaining在各方法皆需要用到，目的是在記錄該process還剩下多少作業時間，特別是在SRTF時，會依此作為優先排序依據，型別是integer，在PPRR中我會利用Time\_Slice\_Limit代表是否受到Time\_Slice的限制，因為如果Priority較小時，可優先處理並且不受RR限制，型別是boolean，Response\_Ratio就是HRRN排序時的依據，比率越高可優先處理，型別是float。

1. FCFS：



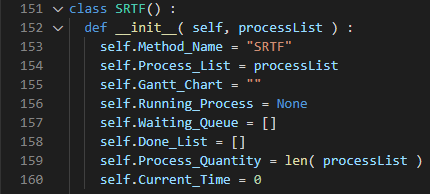
Method\_Name用來記錄方法的名稱，型別是string，Process\_List是讀入的內容，型別是list，Gantt\_Chart用來記錄甘特圖，型別是string，Running\_Process代表正在執行的process，型別是Process，Waiting\_Queue代表等待的佇列，裡面存的是process，型別是list，Done\_List代表完成的佇列，裡面存的是process，型別是list，Process\_Quantity用來記錄input的數量，型別是integer，Current\_Time代表此時程式運行的時間，型別是integer。

1. RR：



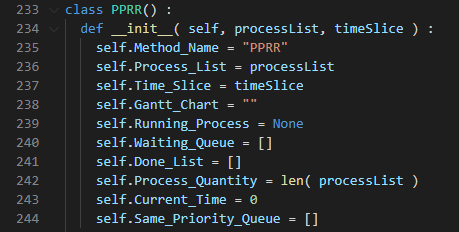
Method\_Name用來記錄方法的名稱，型別是string，Process\_List是讀入的內容，型別是list，Time\_Slice型別是integer，Gantt\_Chart用來記錄甘特圖，型別是string，Running\_Process代表正在執行的process，型別是Process，Waiting\_Queue代表等待的佇列，裡面存的是process，型別是list，Done\_List代表完成的佇列，裡面存的是process，型別是list，Process\_Quantity用來記錄input的數量，型別是integer，Current\_Time代表此時程式運行的時間，型別是integer。

1. SRTF：



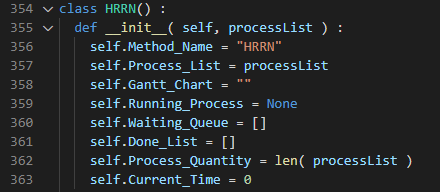
Method\_Name用來記錄方法的名稱，型別是string，Process\_List是讀入的內容，型別是list，Gantt\_Chart用來記錄甘特圖，型別是string，Running\_Process代表正在執行的process，型別是Process，Waiting\_Queue代表等待的佇列，裡面存的是process，型別是list，Done\_List代表完成的佇列，裡面存的是process，型別是list，Process\_Quantity用來記錄input的數量，型別是integer，Current\_Time代表此時程式運行的時間，型別是integer。

1. PPRR：



Method\_Name用來記錄方法的名稱，型別是string，Process\_List是讀入的內容，型別是list，Time\_Slice型別是integer，Gantt\_Chart用來記錄甘特圖，型別是string，Running\_Process代表正在執行的process，型別是Process，Waiting\_Queue代表等待的佇列，裡面存的是process，型別是list，Done\_List代表完成的佇列，裡面存的是process，型別是list，Process\_Quantity用來記錄input的數量，型別是integer，Current\_Time代表此時程式運行的時間，型別是integer，Same\_Priority\_Queue代表有相同priority的process的佇列，型別是list。

1. HRRN：



Method\_Name用來記錄方法的名稱，型別是string，Process\_List是讀入的內容，型別是list，Gantt\_Chart用來記錄甘特圖，型別是string，Running\_Process代表正在執行的process，型別是Process，Waiting\_Queue代表等待的佇列，裡面存的是process，型別是list，Done\_List代表完成的佇列，裡面存的是process，型別是list，Process\_Quantity用來記錄input的數量，型別是integer，Current\_Time代表此時程式運行的時間，型別是integer。

1. 不同排程法的比較
2. 平均等待時間：
3. 結果與討論:

上圖為3筆測資的結果，整體而言SRTF的表現最好，其次是HRRN及FCFS，最後是RR及PPRR，可以觀察到上述2種方法浮動性較大，因為Time Slice的設定很重要，在input 1中，Time Slice為1，在執行RR及PPRR時效果不佳，執行起來像是FCFS一樣，以穩定性來說，我覺得HRRN最好，透過Response Ratio的計算，也可以平衡process的等待時間，最不推薦PPRR，雖然有RR做限制，但是只要priority較高者，還是有機會霸占整個程序執行空間。

1. 未完成的功能

無。