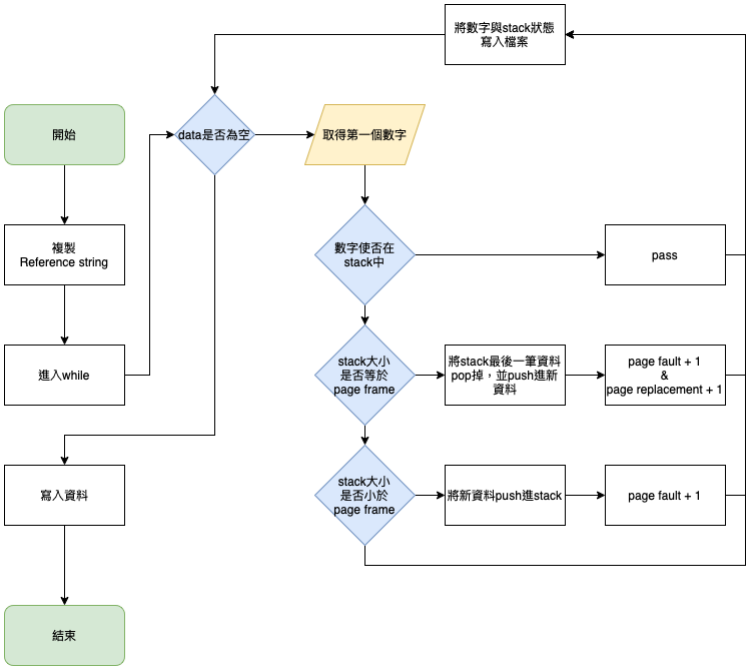
作業系統—作業三

開發環境：VS code　　開發語言：Python　　學號：10827117 姓名：陳柏宇

實作方法與流程：

FIFO：

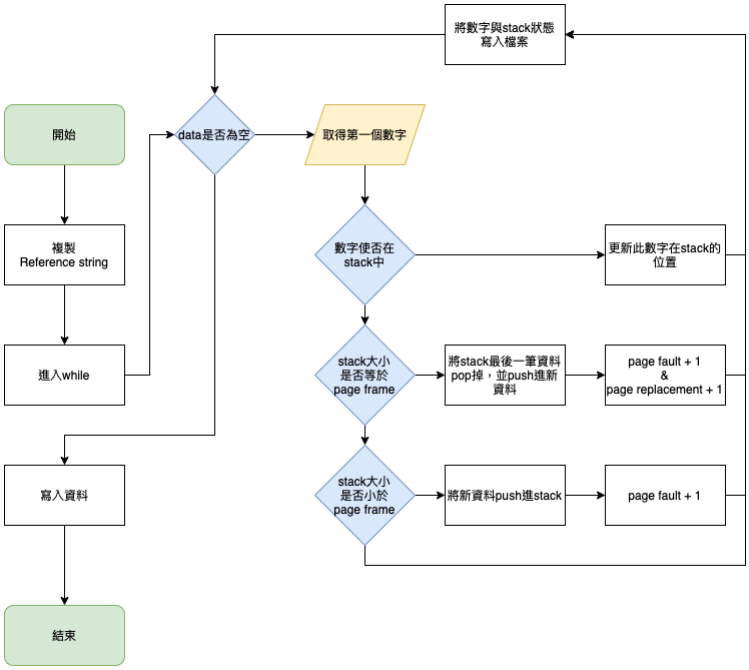
在FIFO這一個Page Replacement中，會先將輸入的Reference string做複製到，接著再進入到while迴圈，在迴圈裡會取出Reference string的第一個數字，取出數字後進入判斷，最先判斷此數字是否已經存在stack中，如果已經存在，就不做任何事情，如果不存在，則做接下來的判斷，接著判斷目前stack中的資料是否已經等於輸入的最大Page frame，如果相等，則將stack中最先push進來的，也就是在最下方的資料，pop出stack，並將新的資料push進stack中，且會將Page fault和Page replacment各加一，而如果判斷不相等，則是做最後一個判斷，如果目前stack中的資料數小於最大的Page frame，則直接將新的資料push進stack中，並將Page fault加一，還有把此次取得的數字和stack最後的狀態寫入檔案。結束回圈後，在檔案中寫入Page fault、Page replacement和Page frame。



LRU：

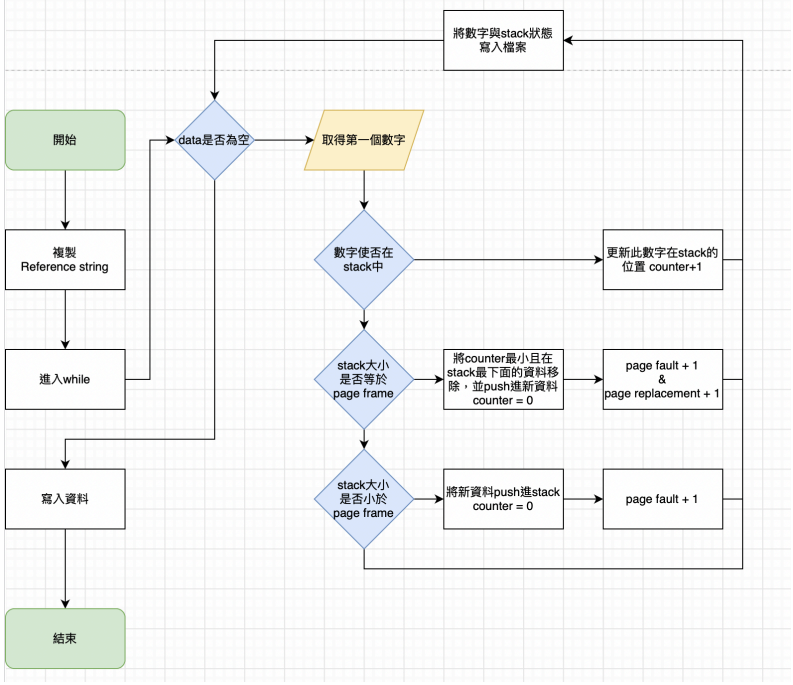
在LRU這一個Page Replacement中，會先將輸入的Reference string做複製到，接著再進入到while迴圈，在迴圈裡會取出Reference string的第一個數字，取出數字後進入判斷，最先判斷此數字是否已經存在stack中，如果已經存在，則更新此數字，將其從

stack中取出後重新push進stack，如果不存在，就判斷目前stack中的資料是否已經等於輸入的最大Page frame，如果相等，則將stack中最先push進來的，也就是在最下方的資料，pop出stack，並將新的資料push進stack中，且會將Page fault和Page replacment各加一，而如果判斷不相等，則是做最後一個判斷，如果目前stack中的資料數小於最大的Page frame，則直接將新的資料push進stack中，並將Page fault加一，還有把此次取得的數字和stack最後的狀態寫入檔案。結束回圈後，在檔案中寫入Page fault、Page replacement和Page frame。



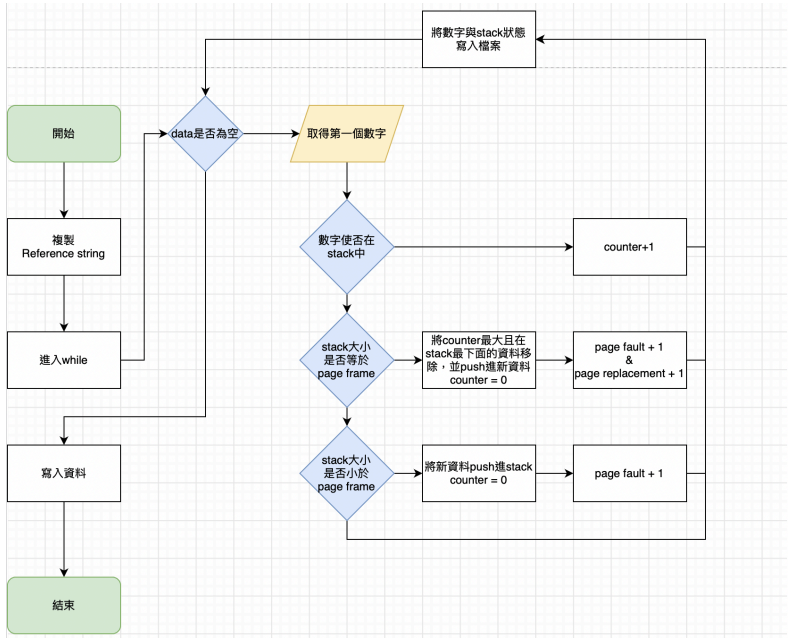
LEAST FREQUENTLY USED LRU PAGE REPLACEMENT :

在LFU+LRU中，會新增一個counter的概念，所以實作的資料結構會從用list實作stack換成dict實作stack，如此一來，就可以放置counter。而同樣也會將輸入的Reference string做複製到，接著再進入到while迴圈，在迴圈裡會取出Reference string的第一個數字，取出數字後進入判斷，最先判斷此數字是否已經存在stack中，如果已經存在，則更新此數字在stack中的位置，並將其counter加一，如果不存在，就判斷目前stack中的資料是否已經等於輸入的最大Page frame，如果相等，就可以從stack找出counter最小且最久沒有被用到的作為犧牲者，並將新的資料放進去stack，counter設定為0，且會將Page fault和Page replacment各加一，而如果判斷不相等，則是做最後一個判斷，如果目前stack中的資料數小於最大的Page frame，則直接將新資料放入stack，並將其counter設定為0，Page fault加一。接者把此次取得的數字和stack最後的狀態寫入檔案。結束回圈後，在檔案中寫入Page fault、Page replacement和Page frame。



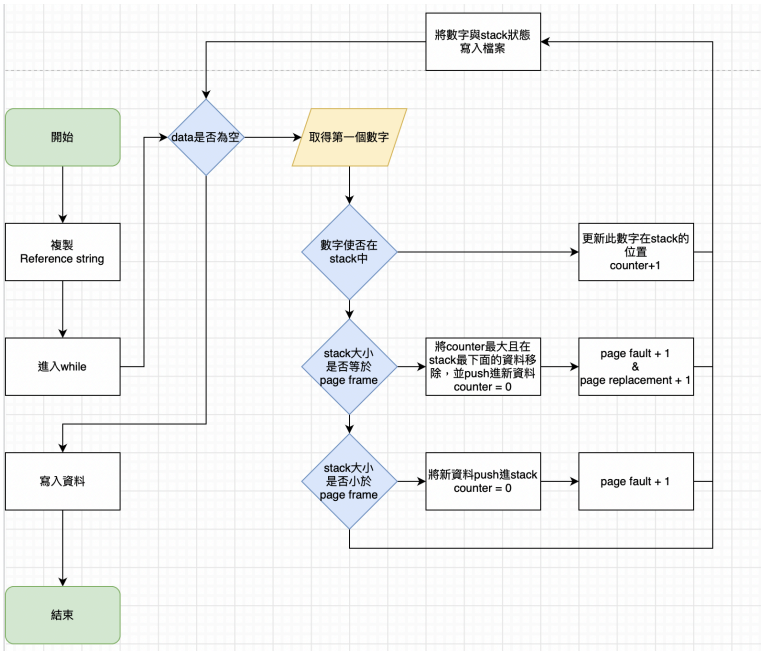
MOST FREQUENTLY USED PAGE REPLACEMENT :

在MFU+FIFO中，將輸入的Reference string做複製到，接著再進入到while迴圈，在迴圈裡會取出Reference string的第一個數字，取出數字後進入判斷，最先判斷此數字是否已經存在stack中，如果已經存在，則將其counter加一，如果不存在，就判斷目前stack中的資料是否已經等於輸入的最大Page frame，如果相等，就可以從stack找出counter最大且最先進到stack的犧牲者，並將新的資料放進去stack，counter設定為0，且會將Page fault和Page replacment各加一，而如果判斷不相等，則是做最後一個判斷，如果目前stack中的資料數小於最大的Page frame，則直接將新資料放入stack，並將其counter設定為0，Page fault加一。接者把此次取得的數字和stack最後的狀態寫入檔案。結束回圈後，在檔案中寫入Page fault、Page replacement和Page frame。



MOST FREQUENTLY USED LRU PAGE REPLACEMENT：

在MFU+LRU中，將輸入的Reference string做複製到，接著再進入到while迴圈，在迴圈裡會取出Reference string的第一個數字，取出數字後進入判斷，最先判斷此數字是否已經存在stack中，如果已經存在，則更新此數字在stack中的位置，並將其counter加一，如果不存在，就判斷目前stack中的資料是否已經等於輸入的最大Page frame，如果相等，就可以從stack找出counter最大且最久沒有被用到的作為犧牲者，並將新的資料放進去stack，counter設定為0，且會將Page fault和Page replacment各加一，而如果判斷不相等，則是做最後一個判斷，如果目前stack中的資料數小於最大的Page frame，則直接將新資料放入stack，並將其counter設定為0，Page fault加一。接者把此次取得的數字和stack最後的狀態寫入檔案。結束回圈後，在檔案中寫入Page fault、Page replacement和Page frame。



不同方法的比較：

	FIFO	LRU	LFU+LRU	MFU+FIFO	MFU+LRU
Input1 Page fault:	9	10	10	9	9
replacement:	6	7	7	6	6
frame:	3	3	3	3	3
Input2 Page fault:	15	12	11	15	12
replacement:	12	9	8	12	9
frame:	3	3	3	3	3

結果與討論：

在畢雷笛反例中，在少數的特例中，會因為增加了Page frame，讓頁錯誤和頁置換不減反增，這個案例通常只會發生在FIFO中，其他的排程法像是Optical（此次沒有實作）和LRU實際上，仍然是隨著Page frame的提升，而讓頁錯誤和頁置換下降。