Alex Preston
CS 5343
Dr. Hakki Candan Cankaya
30 June 2021

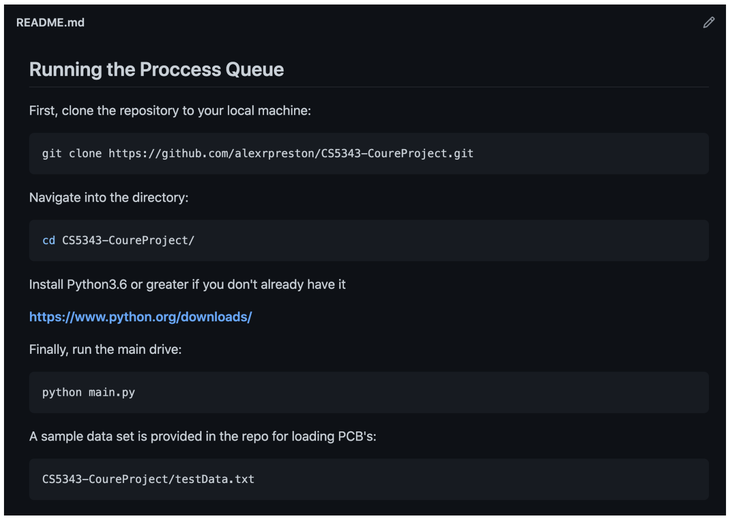
The data structure I used for my project was a singly linked list. I considered using a doubly linked list but since I didn’t see the need to iterate from the tail to the head and wanted to keep the implementation lightweight I decided to go with the singly linked list approach.

The algorithm for adding a PCB first checks if the head of the linked list is undefined, if it is then the PCB to add becomes the head. However if the head is defined then we want to add the PCB at the end (tail) of the linked list. This happens by iterating until we reach the end of the list and then the new node to add is set at the end with the old tail pointing to the new node.

The algorithm for deleting a PCB first checks to see if a PID was supplied as an argument, if the argument wasn’t supplied the function removes the head node and sets the new head as the previous heads next node. However if a PID was added as a function argument the algorithm iterates through the Linked List and checks if the current node’s PID is equal to the user supplied PID, if they are equal then we found the node to delete. In this case we set the previous node’s next pointer to the node after the one we want to delete. This algorithm also has exception handling; this occurs if we reach the end of the list without finding any PCB with the same PID. The program will throw the error “PCB with PID not in list.”

The program also lets the user print the list to visualize the current queue. This algorithm works by iterating through the list and printing out each node’s data until we reach the tail. The program itself was primarily written with Python3 on a Mac environment in VSCode. The program used one of Python's modules, the OS module. This was used in order to implement exception handling, the one of the module's functions can check if a file exists given a file name.

The following steps can be used to run the program locally.



README.md

Running the Proccess Queue

First, clone the repository to your local machine:

```
git clone https://github.com/alexrpreston/CS5343-CoureProject.git
```

Navigate into the directory:

```
cd CS5343-CoureProject/
```

Install Python3.6 or greater if you don't already have it

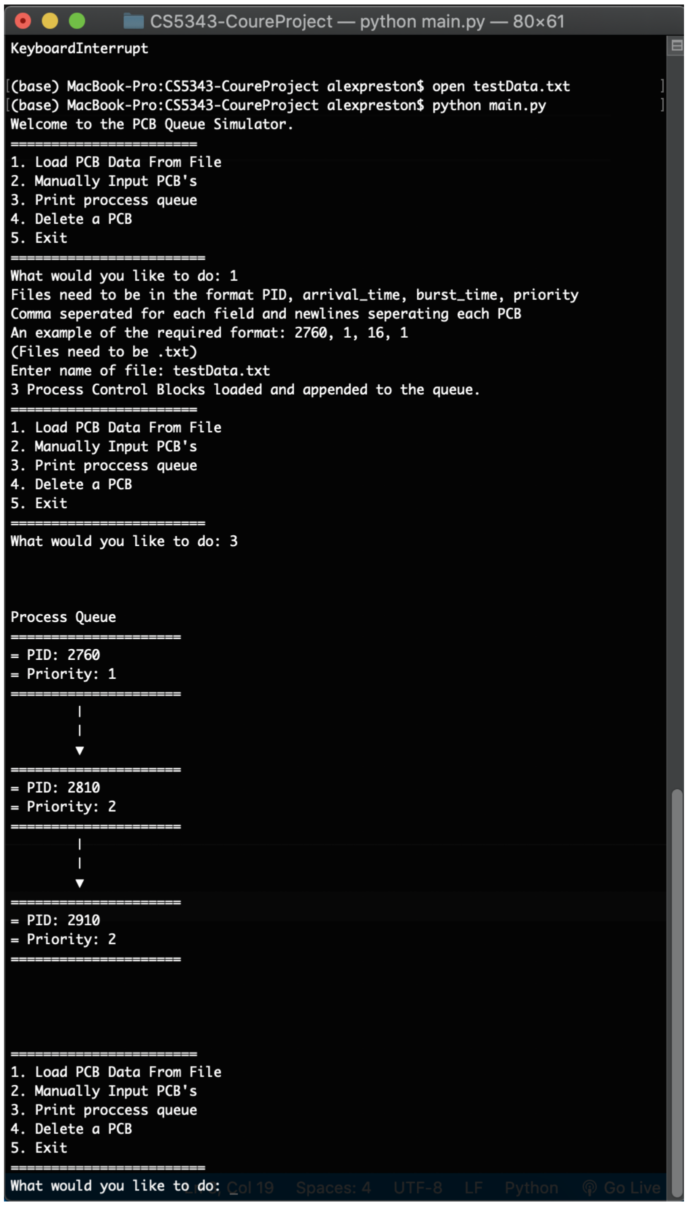
https://www.python.org/downloads/

Finally, run the main drive:

```
python main.py
```

A sample data set is provided in the repo for loading PCB's:

```
CS5343-CoureProject/testData.txt
```

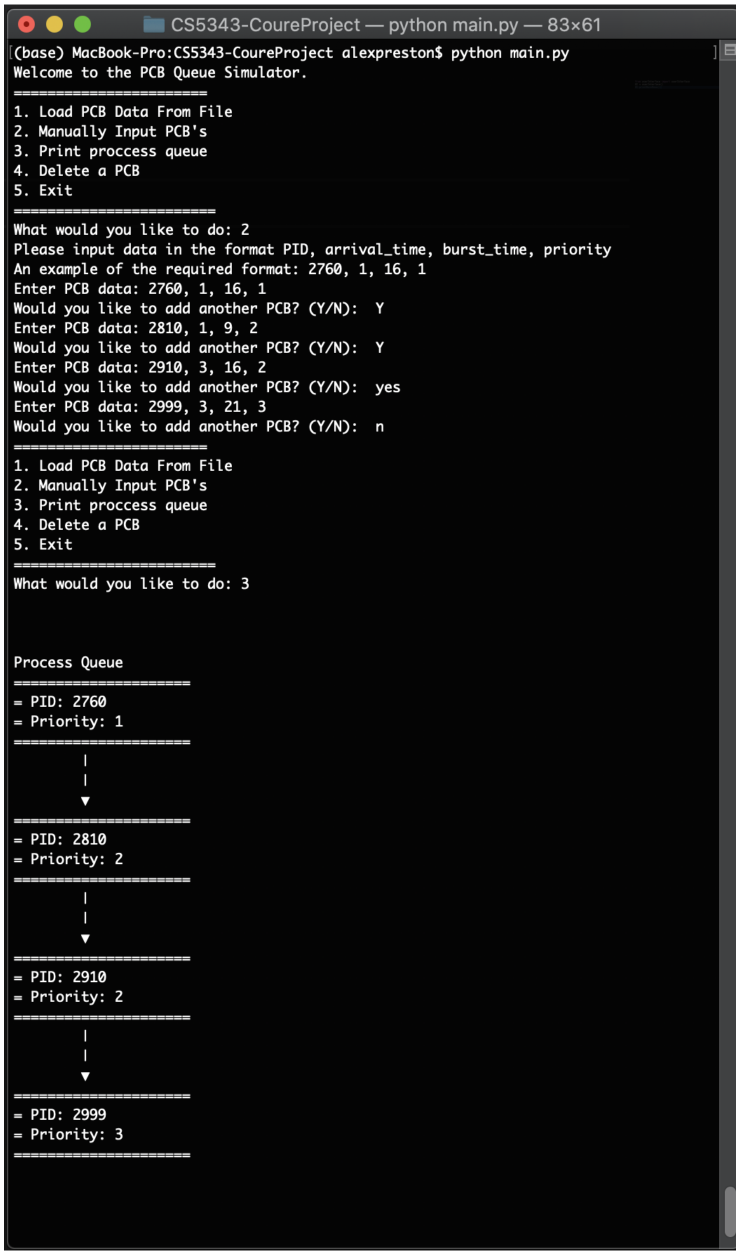


CS5343-CoureProject — python main.py — 80×61

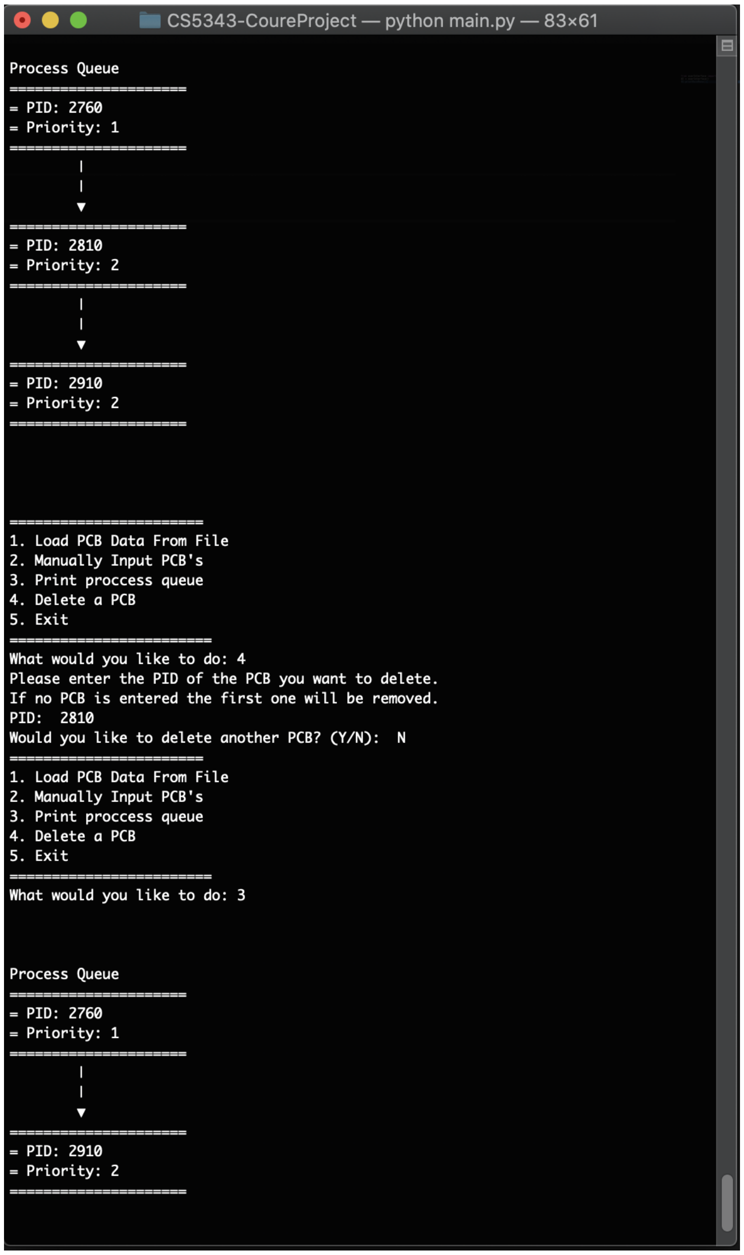
```
KeyboardInterrupt

(base) MacBook-Pro:CS5343-CoureProject alexpreston$ open testData.txt
(base) MacBook-Pro:CS5343-CoureProject alexpreston$ python main.py
Welcome to the PCB Queue Simulator.
=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: 1
Files need to be in the format PID, arrival_time, burst_time, priority
Comma seperated for each field and newlines seperating each PCB
An example of the required format: 2760, 1, 16, 1
(Files need to be .txt)
Enter name of file: testData.txt
3 Process Control Blocks loaded and appended to the queue.
=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: 3



Process Queue
=====================
= PID: 2760
= Priority: 1
=====================
        |
        |
        ▼
=====================
= PID: 2810
= Priority: 2
=====================
        |
        |
        ▼
=====================
= PID: 2910
= Priority: 2
=====================




=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: _
```

The image on the left shows the process of running the program, entering in an input file, and seeing the PCB's added to the Queue.

The utility functions can be seen in the below images for adding a PCB and deleting a PCB given a Process ID. The program also provides the user with a formatting guideline for how to enter PCB data. Input exception handling can also be shown here on the left image: the user can enter “yes,” or “y” in any capitalization format, for example “Y” instead of “y” and the program will accept either one. The user can navigate through the main menu so they can add PCB’s, print the queue, and delete a PCB all by navigating through the user interface.

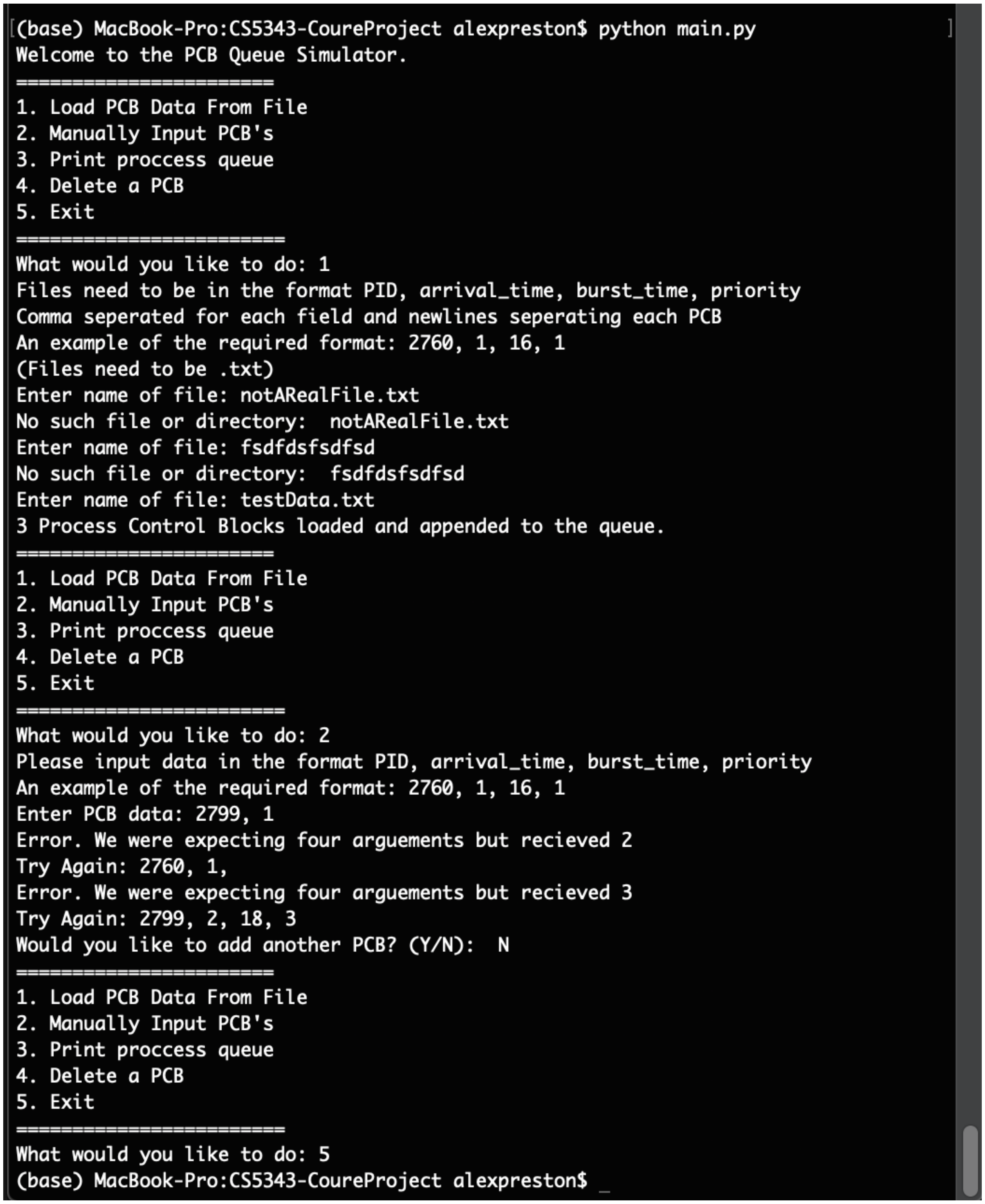


```
(base) MacBook-Pro:CS5343-CoureProject alexpreston$ python main.py
Welcome to the PCB Queue Simulator.
======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
======================
What would you like to do: 2
Please input data in the format PID, arrival_time, burst_time, priority
An example of the required format: 2760, 1, 16, 1
Enter PCB data: 2760, 1, 16, 1
Would you like to add another PCB? (Y/N):  Y
Enter PCB data: 2810, 1, 9, 2
Would you like to add another PCB? (Y/N):  Y
Enter PCB data: 2910, 3, 16, 2
Would you like to add another PCB? (Y/N):  yes
Enter PCB data: 2999, 3, 21, 3
Would you like to add another PCB? (Y/N):  n
======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
======================
What would you like to do: 3



Process Queue
=====================
= PID: 2760
= Priority: 1
=====================
        |
        |
        ▼
=====================
= PID: 2810
= Priority: 2
=====================
        |
        |
        ▼
=====================
= PID: 2910
= Priority: 2
=====================
        |
        |
        ▼
=====================
= PID: 2999
= Priority: 3
=====================
```



```
Process Queue
=====================
= PID: 2760
= Priority: 1
=====================
        |
        |
        ▼
=====================
= PID: 2810
= Priority: 2
=====================
        |
        |
        ▼
=====================
= PID: 2910
= Priority: 2
=====================




======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
======================
What would you like to do: 4
Please enter the PID of the PCB you want to delete.
If no PCB is entered the first one will be removed.
PID:  2810
Would you like to delete another PCB? (Y/N):  N
======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
======================
What would you like to do: 3



Process Queue
=====================
= PID: 2760
= Priority: 1
=====================
        |
        |
        ▼
=====================
= PID: 2910
= Priority: 2
=====================
```

The majority of the exception handling can be seen when the user is entering file names or manually adding the data for a PCB. The program will check to see if the file exists in the projects directory before attempting to open the file and will prompt the user if it can't find the file. This can be seen below, when the user enters the file "notARealFile.txt" instead of crashing, the program prompts the user to enter another file name. The second area exception handling is shown, is when the user enters data for a PCB. The program requires four arguments (PID, arrival_time, burst_time, priority), however if the user enters more or less than four arguments the program prompts the user to fix the data input.



```
(base) MacBook-Pro:CS5343-CoureProject alexpreston$ python main.py
Welcome to the PCB Queue Simulator.
=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: 1
Files need to be in the format PID, arrival_time, burst_time, priority
Comma seperated for each field and newlines seperating each PCB
An example of the required format: 2760, 1, 16, 1
(Files need to be .txt)
Enter name of file: notARealFile.txt
No such file or directory:  notARealFile.txt
Enter name of file: fsdfdsfsdfsd
No such file or directory:  fsdfdsfsdfsd
Enter name of file: testData.txt
3 Process Control Blocks loaded and appended to the queue.
=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: 2
Please input data in the format PID, arrival_time, burst_time, priority
An example of the required format: 2760, 1, 16, 1
Enter PCB data: 2799, 1
Error. We were expecting four arguements but recieved 2
Try Again: 2760, 1,
Error. We were expecting four arguements but recieved 3
Try Again: 2799, 2, 18, 3
Would you like to add another PCB? (Y/N):  N
=======================
1. Load PCB Data From File
2. Manually Input PCB's
3. Print proccess queue
4. Delete a PCB
5. Exit
========================
What would you like to do: 5
(base) MacBook-Pro:CS5343-CoureProject alexpreston$ _
```