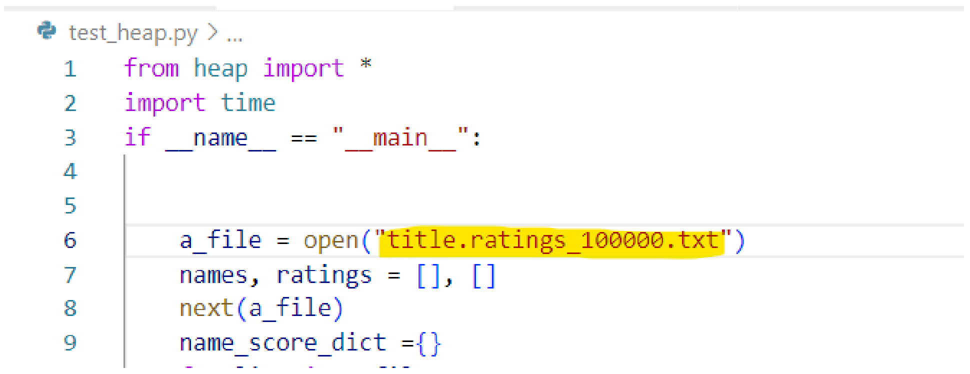
Heap Sort Performance Analysis

The implemented heap sort algorithm, both in batch and incremental versions, was used to sort datasets of 1000, 10,000, and 100,000 sizes, and their performances were measured. Additionally, the test data was compared with built-in Python methods. The following table shows the time taken to perform the sorting, measured in seconds:

Input Size	Heap Sort Batch	Heap Sort Increment	Sorted () function	Sort () function
1,000	0.015617132186889648	0.02404475212097168	0.008013010025024414	0.0
10,000	0.0790247917175293	0.14893722534179688	0.008918046951293945	0.005869150161743164
1,00,000	1.2886736392974854	1.2455964088439941	0.15207314491271973	0.025870084762573242

According to the table above, we can see a small difference for the 1,000-size dataset. As the dataset size increases, the built-in Python sorted () and sort () functions perform better when compared to the implemented heap sort, both in batch and incremental versions.

The datasets **title.ratings_1000.txt**, **title.ratings_10000.txt**, and **title.ratings_100000.txt** were tested by replacing the file name in the code, as shown below:



test_heap.py > ...

```
from heap import *
import time
if __name__ == "__main__":


    a_file = open("title.ratings_100000.txt")
    names, ratings = [], []
    next(a_file)
    name_score_dict ={}
```