How to execute the code:

In our program, we are using dataframes and calculating the memory. Therefore user has to install the pandas and psutil packages before importing our package.

1. **Frequent Pattern Mining Process:**
    1.1. Import our package and initialize the method called **'Apriori'** using the input file path and minimum support.
    1.2. Then call the method **'startMine'** using the following command

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm

2. To get the frequent patterns along with their support count:
    2.1. Complete the Frequent Pattern Mining Process mentioned in **(1)**
    2.2. Then call the method **'getFPs'** using the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
variable = apri.getFPs()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- All the Frequent itemsets will be stored in a dictionary, with itemsets as keys and support count as value and returned to the called function

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
FP = apri.getFPs()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- All the Frequent itemsets will be stored in a dictionary, with itemsets as keys and support count as value and assigned to the variable called 'FP'

3. To get the frequent patterns along with their support count in a file:
   3.1. Complete the Frequent Pattern Mining Process mentioned in **(1)**
   3.2. Then call the method **'getPatternsInFile'** using the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
apri.getPatternsInFile("outputfile")
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- All the Frequent itemsets will be stored in a file named as "outputfile"

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
apri.getPatternsInFile("sampleoutput")
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- All the Frequent itemsets will be stored in a file named as "sampleoutput"

4. To get the frequent patterns along with their support count in a DataFrame:
   4.1. Complete the Frequent Pattern Mining Process mentioned in **(1)**
   4.2. Then call the method **'getPatternsInDf'** using the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
variable = apri.getPatternsInDf()
```

   output is displayed as follows:

   - Frequent itemsets were generated successfully using AprioriHashtree algorithm
   - All the Frequent itemsets will be stored in a dataframe, their columns named as 'Patterns' and 'Support' and returned to the called function

   For example:
   If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
DF = apri.getPatternsInDf()
```

   output is displayed as follows:

   - Frequent itemsets were generated successfully using AprioriHashtree algorithm
   - All the Frequent itemsets will be stored in a dataframe, their columns named as 'Patterns' and 'Support' and stored in a variable called 'DF'

5. If we want to know the memory consumed by the apriori algorithm created by us:
   5.1. Complete the Frequent Pattern Mining Process mentioned in **(1)**
   5.2. Then call the method **'getMemory'** using the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
variable = apri.getMemory()
```

   output is displayed as follows:

   - Frequent itemsets were generated successfully using AprioriHashtree algorithm
   - Total memory consumed by the program in MB will be computed and returned to the called function

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
Mem = apri.getMemory()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- Total memory consumed by the program in MB will be computed and returned to the variable called 'Mem'

Note: In the above scenario, you may use any of the methods such as getFPs(), getPatternsInFile(output), or getPatternsInDf(). Just before calling getMemory() function and based on that memory details will be varied because of the extra storage requirements.

6. If we want to know the execution time taken by the apriori algorithm created by us:
   6.1. Complete the Frequent Pattern Mining Process mentioned in **(1)**
   6.2. Then call the method **'getRuntime'** using the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
variable = apri.getRuntime()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- Total execution time taken by the program in seconds will be computed and returned to the called function

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
Run = apri.getRuntime()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- Total execution time taken by the program in seconds will be computed and returned to the variable called 'Run'

Note: In the above scenario, you may use any of the methods such as getFPs(), getPatternsInFile(output), or getPatternsInDf(). Just before calling getRuntime() function and based on that execution time details will be varied because of the extra operations.

7. If we want to know memory details and execution time taken by the program then use both methods at a time:

```
import Apriori as Myap
apri = Myap.Apriori(r"filepath or filename", minimumsupport)
apri.startMine()
variable1 = apri.getMemory()
variable2 = apri.getRuntime()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- Total memory consumed by the program in MB will be computed and returned to the called function
- Total execution time taken by the program in seconds will be computed and returned to the called function

For example:
If we execute the following command:

```
import Apriori as Myap
apri = Myap.Apriori(r" transactional_T10I4D100K.csv", [0.01])
apri.startMine()
Mem = apri.getMemory()
Run = apri.getRuntime()
```

output is displayed as follows:

- Frequent itemsets were generated successfully using AprioriHashtree algorithm
- Total memory consumed by the program in MB will be computed and returned to the variable called 'Mem'
- Total execution time taken by the program in seconds will be computed and returned to the variable called 'Run'

Note: In the above scenario, you may use any of the methods such as getFPs(), getPatternsInFile(output), or getPatternsInDf(). Just before calling getMemory() and getRuntime() function and based on that memory consumption and execution time details will be varied because of the extra memory and operations.

8. If we want to check the statistics of the input file or database then call the method **'getStatsInFile'** using the following command:

   import Apriori as Myap
   apri = Myap.Apriori(r"filepath or filename", minimumsupport)
   apri.getStatsInFile(statsfilename)
   output is displayed as follows:
   - Frequent itemsets were generated successfully using AprioriHashtree algorithm
   - Complete statistics of the input file or input database such as Total number of transactions, items, minimum length of the transactions, maximum length of the transactions, and average length of the transactions will be stored in a file named as "statsfilename"

   For example:
   If we execute the following command:

   import Apriori as Myap
   apri = Myap.Apriori(r"filepath or filename", minimumsupport)
   apri.getStatsInFile(stats)

   output is displayed as follows:
   - Frequent itemsets were generated successfully using AprioriHashtree algorithm
   - All the statistics of the input file or input database is stored in a file named as "Stats"

   Note: You may use any of the above mentioned funtions along with getStatsInFile(statistics) function.