To convert the Hadoop MapReduce code from Java to Python, we can use the `mrjob` library, which simplifies writing MapReduce jobs in Python for Hadoop. The following steps outline how to convert the code into Python and run it.

Python Code Implementation using `mrjob`

First, you'll need to install the `mrjob` library. If it's not installed yet, you can install it using pip:

```
pip install mrjob
```

Mapper and Reducer Implementation

`char_map.py` (Mapper and Reducer for Character Counting)

```
from mrjob.job import MRJob

class MRCharCount(MRJob):

    def mapper(self, _, line):
        # Mapper function: emit each character from the input text.
        for char in line:
            if char.isalnum():  # Only count alphanumeric characters.
                yield (char, 1)

    def reducer(self, char, counts):
        # Reducer function: sum up the counts for each character.
        yield (char, sum(counts))

if __name__ == '__main__':
    MRCharCount.run()
```

Steps to Run the Python MapReduce Job with Hadoop

1. **Install Hadoop (if not already done)**:
    - Follow the instructions to install Hadoop on your machine. You can download Hadoop from the official site.
2. **Set up `mrjob` Configuration**: The `mrjob` library can run on a local machine, on Hadoop, or on Amazon EMR. You can run it with Hadoop by configuring it in the `mrjob.conf` file.

   Create a configuration file `mrjob.conf` with the following content:

```
runner: hadoop
hadoop_home: /path/to/your/hadoop
```

Adjust the path of `hadoop_home` to your Hadoop installation path.

3. **Create a `sample.txt` File**: Create a sample text file in your home directory (or in a location of your choice). This will be the input for your MapReduce job.

   For example, the file `sample.txt` could contain:

   ```
   Hello Hadoop
   ```

4. **Run the Job Locally or on Hadoop**:

   To run the job locally or on a Hadoop cluster using the `mrjob` library, you would execute the following command in your terminal:

   ```
   python char_map.py sample.txt
   ```

   This will run the job locally and print the results to the terminal.

5. **Running on Hadoop Cluster**:

   - To run on your Hadoop cluster, first upload the `sample.txt` file to HDFS:

     ```
     ~/hadoop/bin/hdfs dfs -put ~/sample.txt /user/your_user_name/input_data
     ```

   - Then, run the job on Hadoop with the following command:

     ```
     python char_map.py hdfs:///user/your_user_name/input_data
     ```

     The output will be saved to a default output directory in HDFS. You can view the results by using the following command:

     ```
     ~/hadoop/bin/hdfs dfs -cat output/*
     ```

6. **Clean Up**: After the job has completed, clean up the input and output directories on HDFS:

```
~/hadoop/bin/hdfs dfs -rm -r /user/your_user_name/input_data /user/your_user_name/out
```

Expected Output

After running the job, the output will be printed in the following format (characters and their counts):

```
" "   2
H     1
e     1
l     2
o     2
a     1
d     1
p     1
```

This represents the character counts from the input text `"Hello Hadoop"`.

Additional Notes

- The above code uses `mrjob`, which is designed to make it easier to write MapReduce jobs in Python. It abstracts much of the complexity of Hadoop.
- For running the job on a Hadoop cluster, ensure you have the `mrjob.conf` file correctly set up and Hadoop is configured to run jobs from Python.