# Word Count in Python

## 📁 STEP 2: Create Working Directory

### ✅ What?

Create a folder where you'll store all WordCount-related files.

### ❓ Why?

To organize the mapper/reducer code and input files.

### ⚙️ How?

```
mkdir ~/wordcount
cd ~/wordcount
```

- `mkdir` creates the folder.
- `cd` moves you into it.

## STEP 3: Create Mapper and Reducer Scripts

### ✅ What?

Write two Python files: `mapper.py` and `reducer.py`.

### ❓ Why?

Hadoop Streaming lets us write Mapper/Reducer in Python, which will be used to process the input data.

### ⚙️ How?

*Create `mapper.py`*

**nano mapper.py**

#!/usr/bin/env python3

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    for word in line.strip().split():
        print(f"{word}\t1")
```

Create
nano reducer.py

```
#!/usr/bin/env python3
import sys
current_word = None
current_count = 0
for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)
    if word == current_word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count
if current_word:
    print(f"{current_word}\t{current_count}")
```

## 🔓 STEP 4: Make Scripts Executable

### ✅ What?

Give the Python scripts permission to run.

### ❓ Why?

Hadoop needs to be able to execute them.

### ⚙️ How?

**chmod +x mapper.py reducer.py**

## 📝 STEP 5: Create Input Files

### ✅ What?

Create some sample input files.

### ❓ Why?

To provide data for the WordCount job.

### ⚙ How?

**mkdir input**
**echo "Hello world Bye world" > input/file1.txt**
**echo "Hello Hadoop Goodbye Hadoop" > input/file2.txt**

## STEP 6: Start Hadoop Services

### ✅ What?

Start HDFS and YARN daemons.

### ❓ Why?

HDFS stores data and YARN runs the job.

### ⚙ How?

**start-dfs.sh**
**start-yarn.sh**

## STEP 7: Verify Hadoop Daemons

### ✅ What?

Check if Hadoop services are running.

### ❓ Why?

To confirm NameNode, DataNode, ResourceManager, etc. are running properly.

⚙ How?

**jps**

## 🗀 STEP 8: Create Input Directory in HDFS

✅ What?

Create a directory in HDFS for storing input files.

❓ Why?

Because Hadoop only processes files from HDFS, not your local file system.

⚙ How?

**hadoop fs -mkdir -p /user/ubuntu/input**

## NOTE:

IF different path :
Find path:
**hadoop fs –pwd**
**/user/yourname**

So your command becomes:

**hadoop fs -mkdir -p /user/yourname/input**

STEP 9: Upload Files to HDFS

**hadoop fs -put ~/wordcount/input/* /user/ubuntu/input**

**or**
**hadoop fs -put ~/wordcount/input/* /user/yourname/input**

# STEP 10: Run the Hadoop Streaming Job

✅ What?

Execute the MapReduce job using Hadoop Streaming.

**hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar **
**-input /user/ubuntu/input **
**-output /user/ubuntu/output_wordcount **
**-mapper ~/wordcount/mapper.py **
**-reducer ~/wordcount/reducer.py **
**-file ~/wordcount/mapper.py **
**-file ~/wordcount/reducer.py**

# STEP 11: View Output

✅ What?

See the final word count result.

**hadoop fs -ls /user/ubuntu/output_wordcount**

**hadoop fs -cat /user/ubuntu/output_wordcount/part-00000**

**If inuput should be file**

**mkdir ~/wordcount**

**cd ~/wordcount**

**cp /home/yourname/Documents/mydata.txt ~/wordcount/**

**mkdir input**

**mv mydata.txt input/**

**start-dfs.sh**

**start-yarn.sh**

**hadoop fs -mkdir -p /user/yourname/input**

**hadoop fs -put ~/wordcount/input/mydata.txt /user/yourname/input**

To check if uploaded: hadoop fs -ls /user/yourname/input

mapper.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
   for word in line.strip().split():
      print(f"{word}\t1")
```

reducer.py

```
#!/usr/bin/env python3
import sys
current_word = None
current_count = 0
for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)
    if word == current_word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count
if current_word:
    print(f"{current_word}\t{current_count}")

chmod +x mapper.py reducer.py

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar \
-input /user/yourname/input \
-output /user/yourname/output_wordcount \
-mapper ~/wordcount/mapper.py \
-reducer ~/wordcount/reducer.py \
-file ~/wordcount/mapper.py \
-file ~/wordcount/reducer.py
```

**hadoop fs -ls /user/yourname/output_wordcount**

**hadoop fs -cat /user/yourname/output_wordcount/part-00000**

**Additional :**

# 📁 Basic HDFS Commands

🔢 **Task Command**

## ✅ 1. Make a Directory in HDFS

```
bash
CopyEdit
hadoop fs -mkdir /user/yourname/foldername
```

**Example:**

```
bash
CopyEdit
hadoop fs -mkdir /user/yourname/input
```

To make parent folders automatically (if not already present):

```
bash
CopyEdit
hadoop fs -mkdir -p /user/yourname/input
```

---

## ✅ 2. View All Directories and Files in HDFS

```
bash
CopyEdit
hadoop fs -ls /
```

To list contents of a specific directory:

```
bash
CopyEdit
hadoop fs -ls /user/yourname
```

---

## ✅ 3. Move a File to a Directory in HDFS

### Upload from local file system to HDFS:

```
bash
CopyEdit
hadoop fs -put /path/to/localfile.txt /user/yourname/input
```

### Example:

```
bash
CopyEdit
hadoop fs -put ~/wordcount/input/mydata.txt /user/yourname/input
```

### Move a file inside HDFS:

```
bash
CopyEdit
hadoop fs -mv /user/yourname/input/mydata.txt /user/yourname/backup/
```

---

## ✅ 4. View All Files in a Directory

```
bash
CopyEdit
hadoop fs -ls /user/yourname/input
```

---

## ✅ 5. Remove a Directory in HDFS

```
bash
CopyEdit
hadoop fs -rm -r /user/yourname/input
```

This will delete the directory **and all its contents**.

---

## ✅ 6. Remove a File in a Directory

```
bash
CopyEdit
hadoop fs -rm /user/yourname/input/mydata.txt
```

---

### ✅ 7. See Your HDFS Current Directory (HDFS Home Path)

```
bash
CopyEdit
hadoop fs -pwd
```

This shows your current working directory in HDFS (typically `/user/yourname`).

---

### ✅ 8. Find the Hadoop Streaming JAR on Your System

```
bash
CopyEdit
ls $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-*.jar
```

This shows the **full path** to your streaming JAR (needed to run Python-based MapReduce jobs).

---

## ✅ Quick Tip: If `$HADOOP_HOME` is not set

Try this:

```
bash
CopyEdit
find / -name "hadoop-streaming-*.jar" 2>/dev/null
```