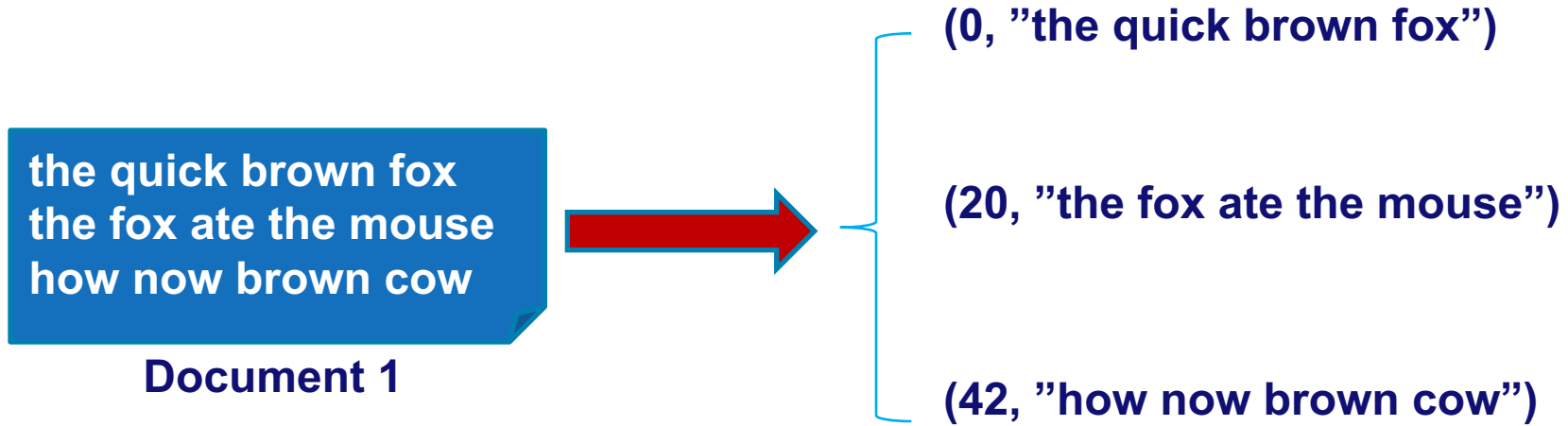


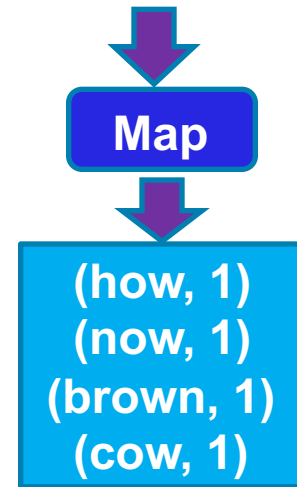
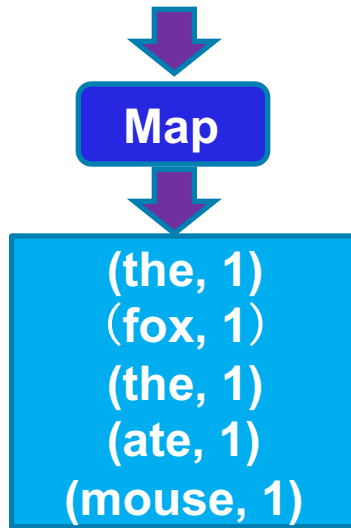
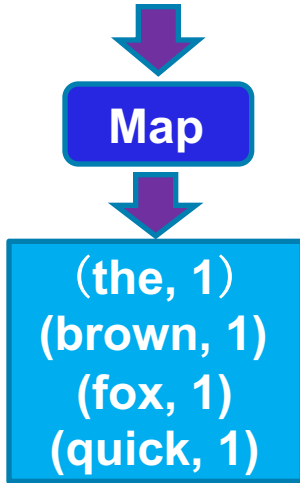
# MapReduce Programming Model

- Review of Map function, shuffle & sort, Reduce function

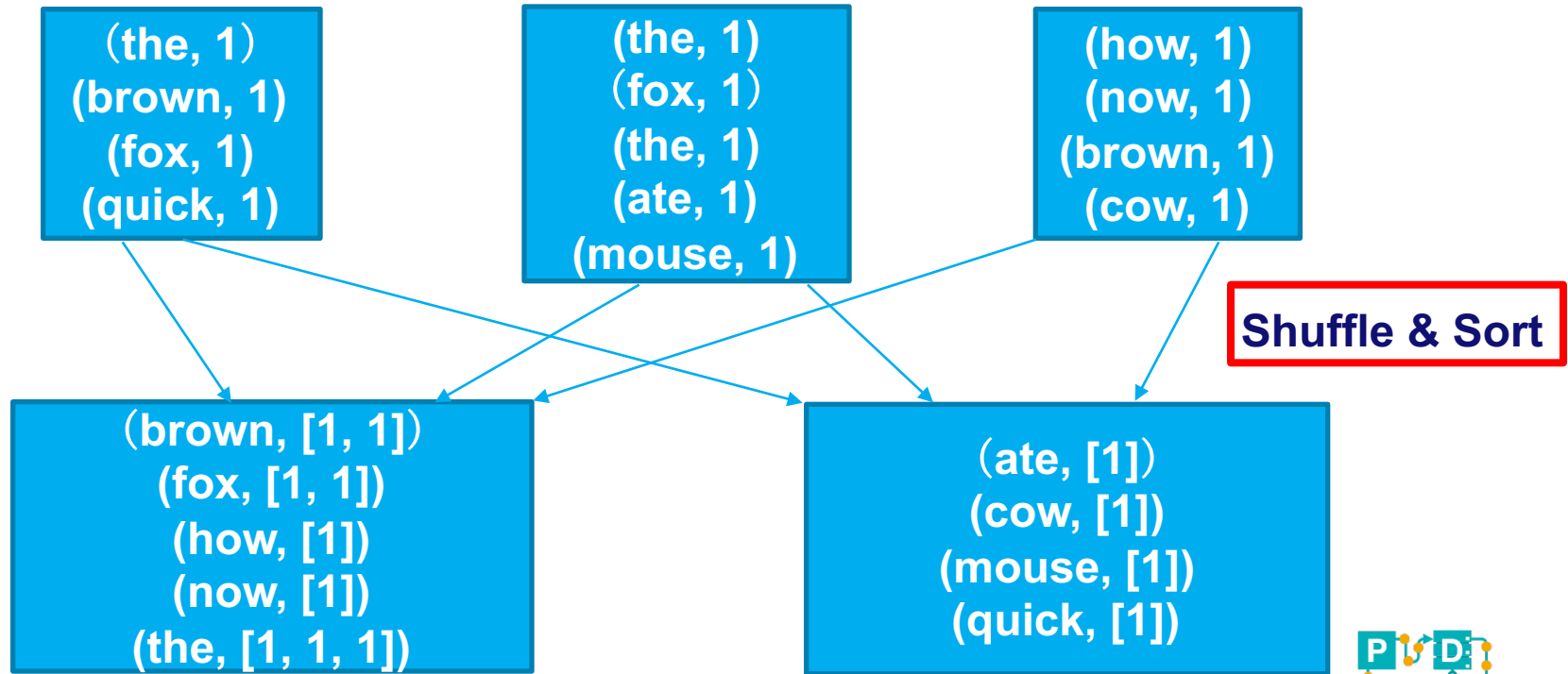


# MapReduce Programming Model

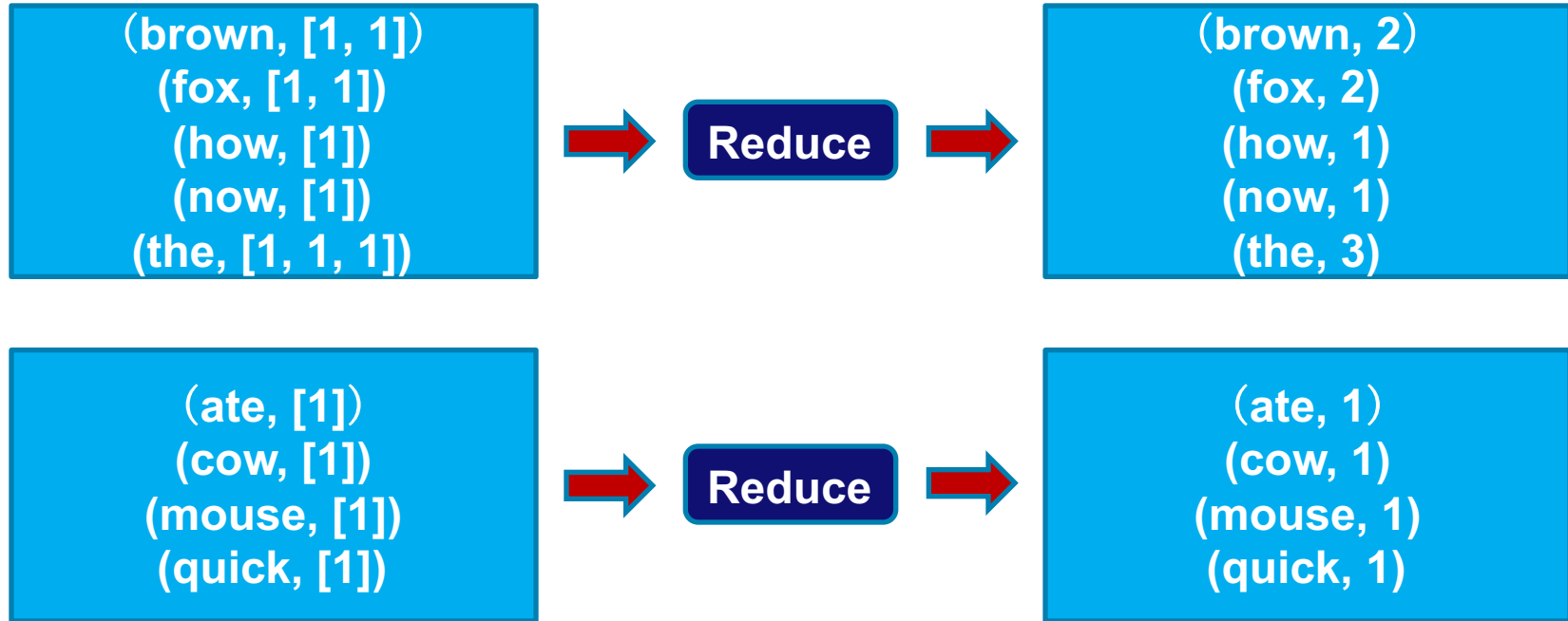
(0, "the quick brown fox") (20, "the fox ate the mouse") (42, "how now brown cow")



# MapReduce Programming Model



# MapReduce Programming Model



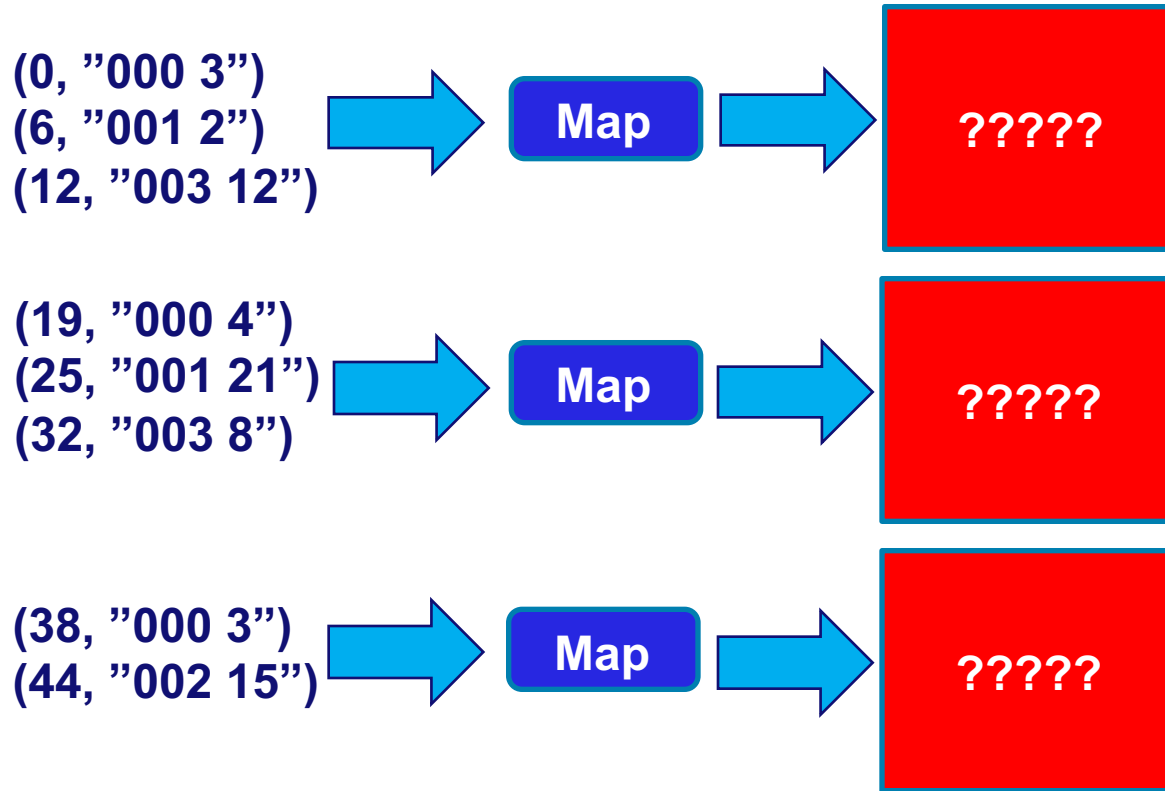
# MapReduce Programming Model

- Exercise 1: For the input document, **calculate the total price for each invoice ID**. Presume you use **MapReduce** to do this, please write down the **output** of each **Map function**, the **output** after **shuffle & sort**, the **output** of **Reduce function**.

Invoice ID	Price
000	3
001	2
003	12
000	4
001	21
003	8
000	3
002	15

Input document

# MapReduce Programming Model

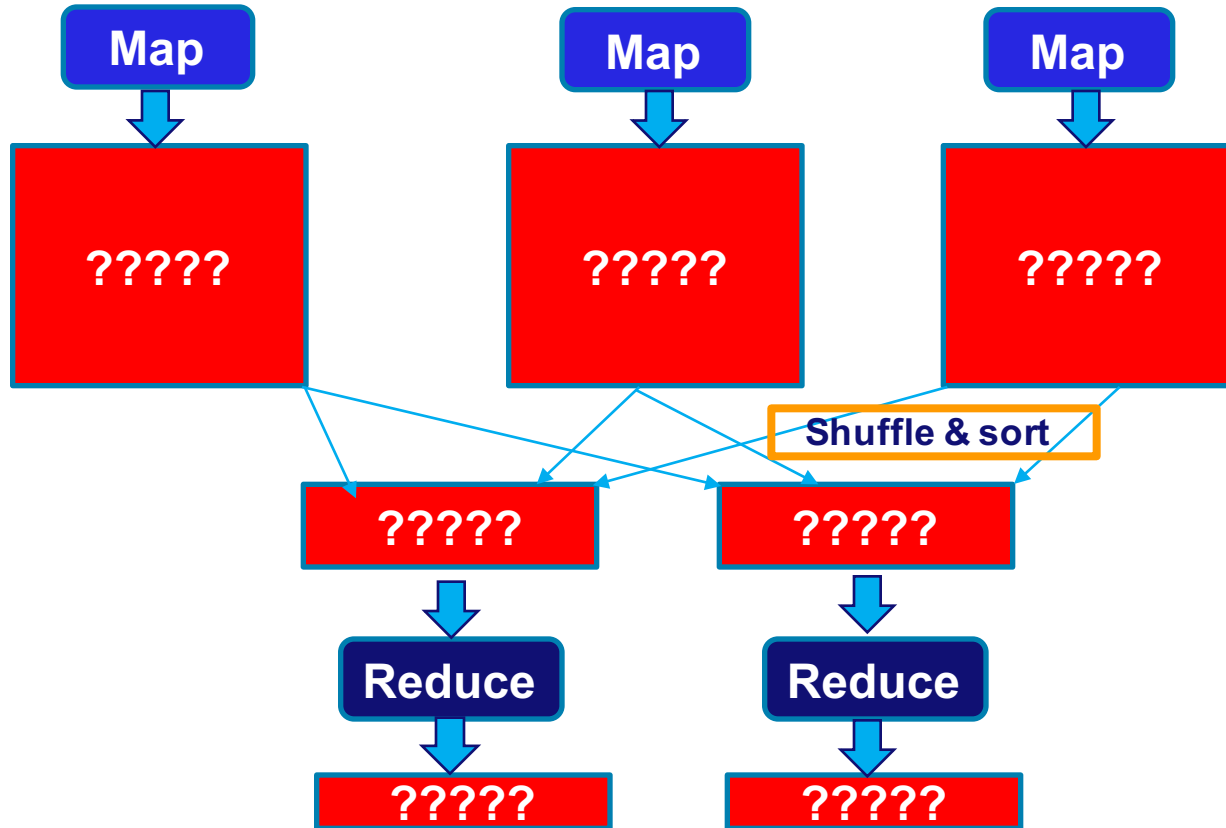


Invoice ID      Price

000	3
001	2
003	12
000	4
001	21
003	8
000	3
002	15

Input document

# MapReduce Programming Model



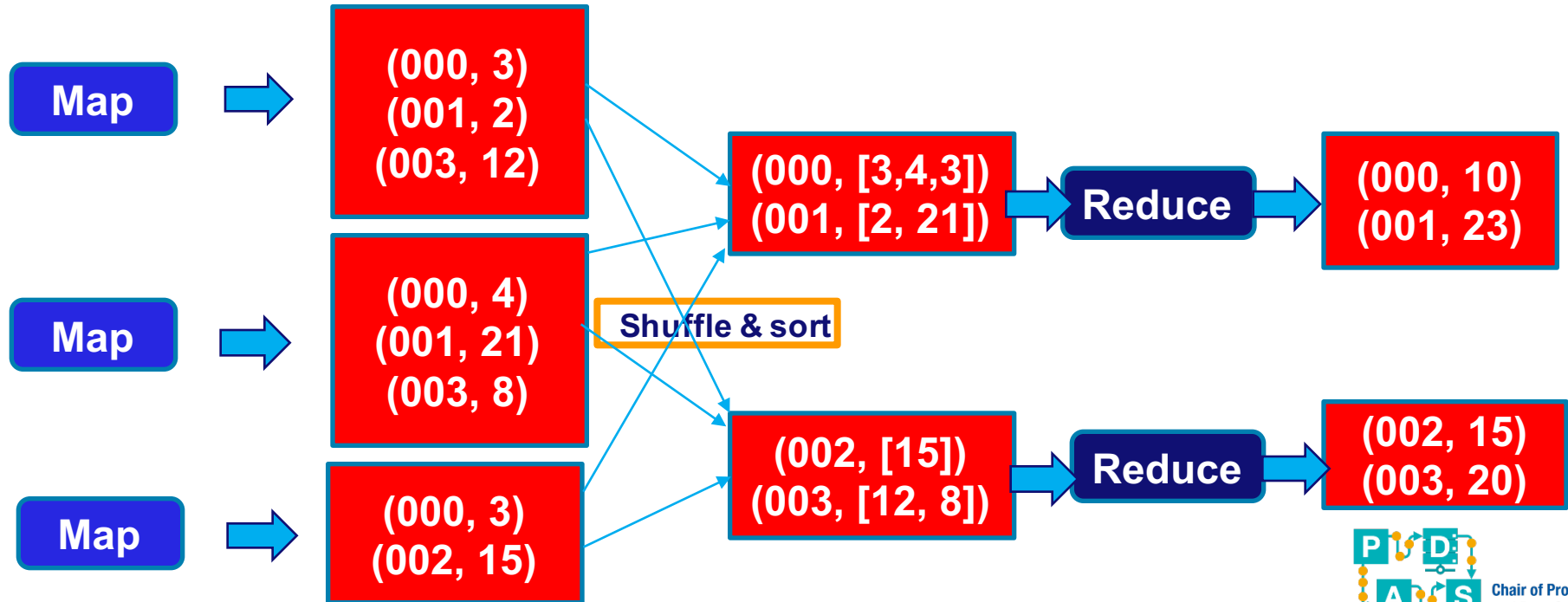
Invoice ID      Price

000	3
001	2
003	12
000	4
001	21
003	8
000	3
002	15

Input document

# MapReduce Programming Model

- Solution 1

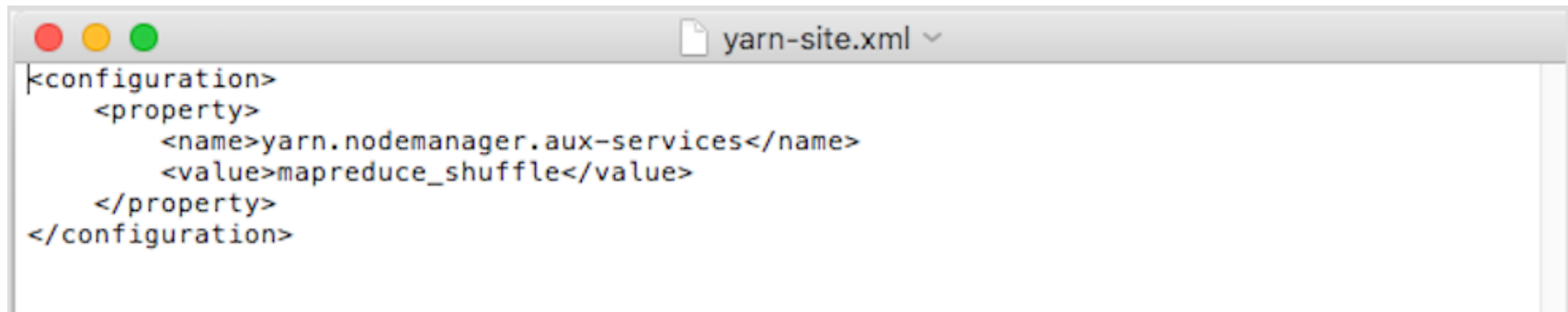




# MapReduce Programming Model

- **Before running Hadoop:**

**For Windows users, make sure you open your command terminal with administrator role, for both starting Hadoop and latter running MapReduce program**



The screenshot shows a text editor window titled 'yarn-site.xml'. The content is an XML configuration snippet for Hadoop Yarn. It defines a property for the auxiliary services of the Yarn NodeManager, setting the value to 'mapreduce\_shuffle'.

```
<configuration>  
  <property>  
    <name>yarn.nodemanager.aux-services</name>  
    <value>mapreduce_shuffle</value>  
  </property>  
</configuration>
```

# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -cat**
    - **hadoop fs -cat /file1**
    - **hadoop fs -cat /file1 /file2**

# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -copyFromLocal**
    - **hadoop fs -copyFromLocal file:///file1 /folder1**
    - **hadoop fs -copyFromLocal file:///file1 /folder1/file2**
    - **hadoop fs -copyFromLocal file:///folder1 /**
    - **hadoop fs -copyFromLocal file:///folder1 /folder2**
  - **hadoop fs -copyToLocal**



# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -cp**
    - **hadoop fs -cp /file1 /folder1**
    - **hadoop fs -cp /file1 /file2 /folder1**
    - **hadoop fs -cp /folder1 /folder2**

# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -ls**
    - **hadoop fs -ls /folder1**
    - **hadoop fs -ls /file1**

# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -mkdir**
    - **hadoop fs -mkdir /folder1**
    - **hadoop fs -mkdir -p /folder1/folder2/folder3**

# Hadoop Shell Commands

- **Basic Hadoop shell commands**
  - **hadoop fs -rm**
    - **hadoop fs -rm -r /folder1**
    - **hadoop fs -rm /file1**
    - **hadoop fs -rm -r /folder1**

# Hadoop Shell Commands

- **Exercise 2:**
  - **Build the folders /test/input in your HDFS**
  - **Build the folders /test/output in your HDFS**
  - **Copy the file PriceSum1.txt into folder /test/input**
  - **Show the contents of PriceSum1.txt in your terminal**
  - **Copy the file /test/input/PriceSum1.txt and generate two new files PriceSum2.txt and PriceSum3.txt which are put in folder /test/output**
  - **Delete the file /test/output/PriceSum3.txt**



# MapReduce Coding: MRJob

- Python code for WordCount: MRJob version

- 1. **from** mrjob.job **import** MRJob
- 2. **class** MRWordCount(**MRJob**):
- 3.     **def** mapper(self, \_, line):
- 4.         **for** word **in** line.split():
- 5.             **yield** word, 1
- 6.     **def** reducer(self, key, values):
- 7.         **yield** key, sum(values)
- 8. **if** \_\_name\_\_ == '\_\_main\_\_':
- 9.     MRWordCount.run()

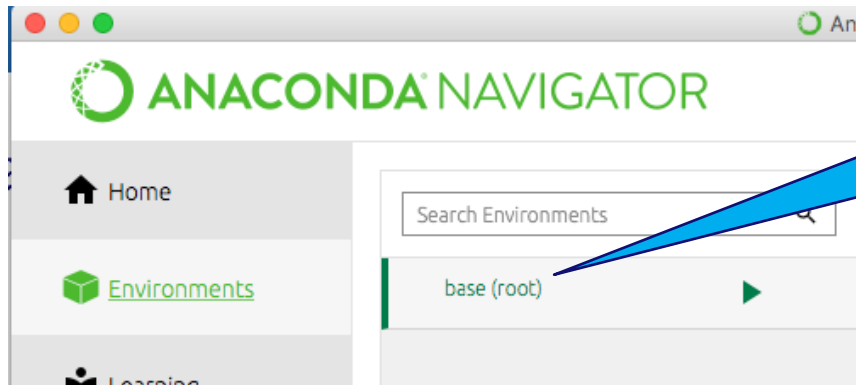
# MapReduce Coding: MRJob

- **Command to run WordCount with MRJob**
  - `python /Users/yaguangsun/PriceSum.py /Users/yaguangsun/PriceSum1.txt`

**If you use Windows operation system,  
pay attention to address format**

# MapReduce Coding: MRJob

- **Command to run WordCount with MRJob**
  - `python /Users/yaguangsun/PriceSum.py /Users/yaguangsun/PriceSum1.txt`



**If you installed MRJob with conda, then you need to open the terminal in ANACONDA**

# MapReduce Coding: MRJob

- **Exercise 3:** write down the **MapReduce code** to solve the problem from **exercise 1: calculate the total price for each invoice id** for a given document with the same format as shown in exercise 1. **Run your code** over the file **PriceSum1.txt** stored in your local file system.

Invoice ID	Price
000	3
001	2
003	12
000	4
001	21
003	8
000	3
002	15


# MapReduce Coding

- **Solution 3:**

- 1. **from** mrjob.job **import** MRJob
- 2. **class** PriceSum(**MRJob**):
- 3. **def** mapper(self, \_, line):
- 4.     **m** = line.split()
- 5.     **yield** m[0], float(m[1])
- 6. **def** reducer(self, key, values):
- 7.     **yield** key, sum(values)
- 8. **if** \_\_name\_\_ == '\_\_main\_\_':
- 9.     PriceSum.run()

# MapReduce Coding: Pure Python

- **Python code for WordCount: pure python**

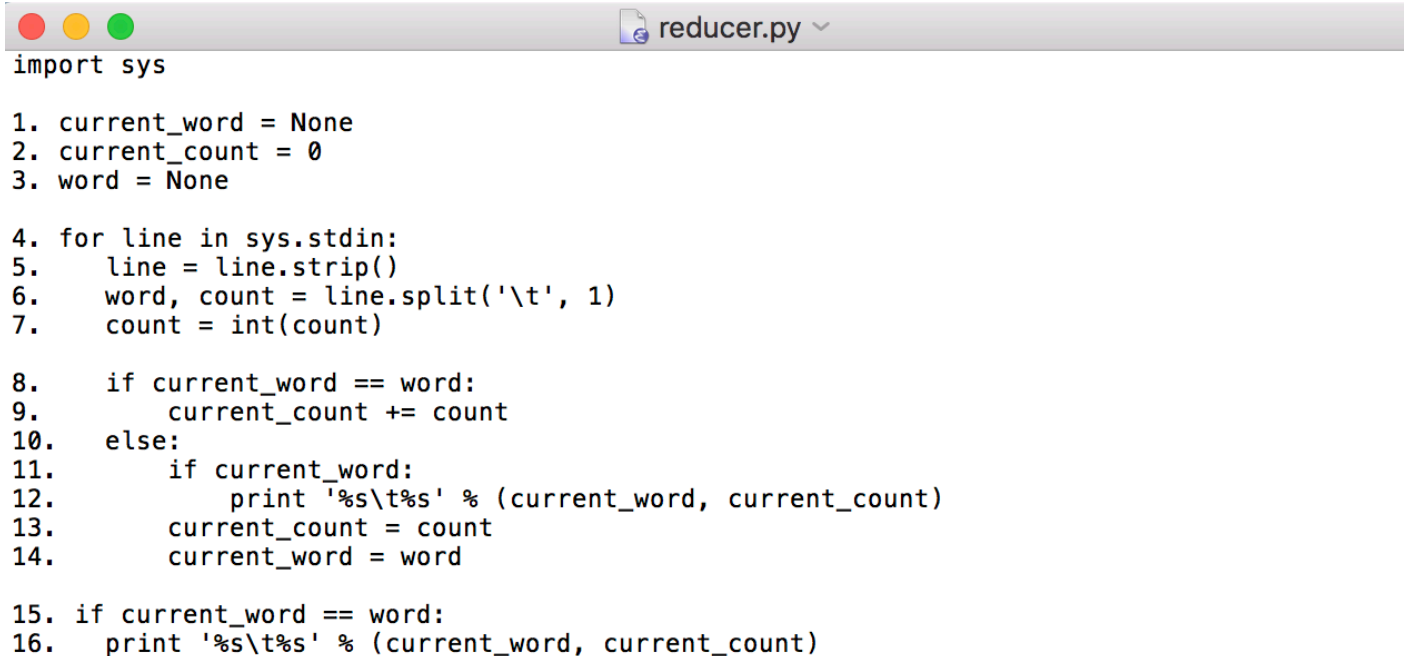
 mapper.py ▾

```
import sys
```

```
1. for line in sys.stdin:
2.     n = line.strip()
3.     m = n.split()
4.     for word in m:
5.         print('%s\t%s' % (word, 1))
```

# MapReduce Coding: Pure Python

- Python code for WordCount: pure python

A screenshot of a code editor window titled 'reducer.py'. The code is a Python script for a WordCount reducer. It imports the sys module and initializes variables for the current word, count, and word. It then iterates over lines from sys.stdin, stripping each line and splitting it by a tab character. If the current word matches the word in the line, it increments the count. If not, it prints the current word and count, resets the count to the new value, and updates the current word. Finally, it prints the last word and count.

```
import sys

1. current_word = None
2. current_count = 0
3. word = None


4. for line in sys.stdin:
5.     line = line.strip()
6.     word, count = line.split('\t', 1)
7.     count = int(count)

8.     if current_word == word:
9.         current_count += count
10.    else:
11.        if current_word:
12.            print '%s\t%s' % (current_word, current_count)
13.            current_count = count
14.            current_word = word

15. if current_word == word:
16.     print '%s\t%s' % (current_word, current_count)
```

# MapReduce Coding: Pure Python

- Python code for WordCount: pure python

 reducer.py ▾

```
import sys

1. current_word = None
2. current_count = 0
3. word = None

4. for line in sys.stdin:
5.     line = line.strip()
6.     word, count = line.split('\t', 1)
7.     count = int(count)

8.     if current_word == word:
9.         current_count += count
10.    else:
11.        if current_word:
12.            print '%s\t%s' % (current_word, current_count)
13.            current_count = count
14.            current_word = word

15. if current_word == word:
16.     print '%s\t%s' % (current_word, current_count)
```

**Difference**

```
6. def reducer(self, key, values):
7.     yield key, sum(values)
```



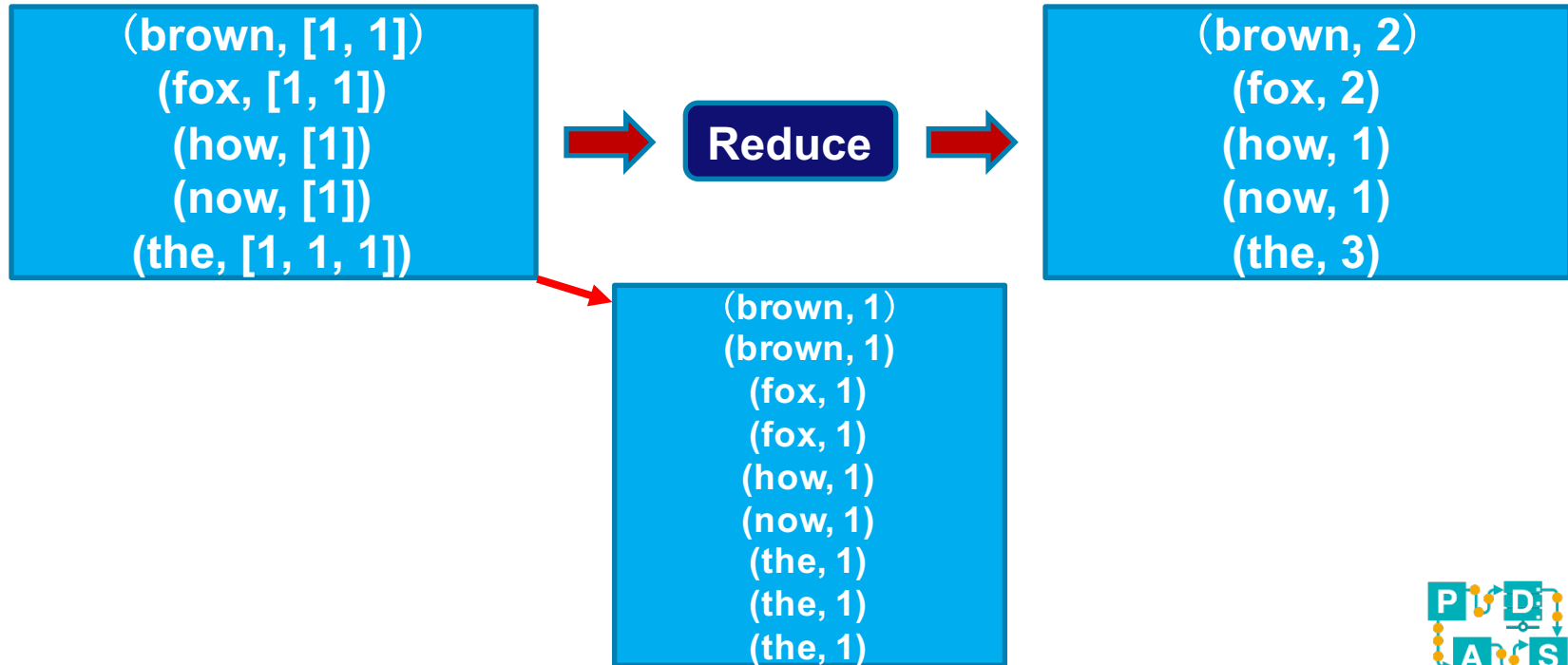
# MapReduce Coding: Pure Python

- Python code for WordCount: pure python




# MapReduce Coding: Pure Python

- Python code for WordCount: pure python



# MapReduce Coding: Pure Python

- Python code for WordCount: pure python

 reducer.py ▾


```
import sys

1. current_word = None
2. current_count = 0
3. word = None

4. for line in sys.stdin:
5.     line = line.strip()
6.     word, count = line.split('\t', 1)
7.     count = int(count)

8.     if current_word == word:
9.         current_count += count
10.    else:
11.        if current_word:
12.            print '%s\t%s' % (current_word, current_count)
13.            current_count = count
14.            current_word = word

15. if current_word == word:
16.     print '%s\t%s' % (current_word, current_count)
```



Difference

6. **def reducer(self, key, values):**  
7. **yield key, sum(values)**

# MapReduce Coding: Pure Python

- **Python code for WordCount: pure python**
  - **Command for running WordCount with Hadoop**

```
hadoop jar /Users/yaguangsun/file/mix/hadoop/hadoop-2.8.4/share/hadoop/tools/lib/hadoop-streaming-2.8.4.jar  
-D mapred.map.tasks=2  
-input hdfs:///test/input  
-output hdfs:///test/output/WordCountOutput  
-mapper "python /Users/yaguangsun/mapper.py"  
-reducer "python /Users/yaguangsun/reducer.py"  
-file /Users/yaguangsun/mapper.py  
-file /Users/yaguangsun/reducer.py
```



# MapReduce Coding: Pure Python

- Python code for WordCount: pure python
  - Command for running WordCount with Hadoop

```
hadoop jar /Users/yaguangsun/file/mix/hadoop/hadoop-2.8.4/share/hadoop/tools/lib/hadoop-streaming-2.8.4.jar  
-D mapred.map.tasks=2  
-input hdfs:///test/input  
-output hdfs:///test/output/WordCountOutput  
-mapper "python /Users/yaguangsun/mapper.py"  
-reducer "python /Users/yaguangsun/reducer.py"  
-file /Users/yaguangsun/mapper.py  
-file /Users/yaguangsun/reducer.py
```

-D should appear  
before the other  
parameters



# MapReduce Coding: Pure Python

- Python code for WordCount: pure python
  - Command for running WordCount with Hadoop

```
hadoop jar /Users/yaguangsun/file/mix/hadoop/hadoop-2.8.4/share/hadoop/tools/lib/hadoop-streaming-2.8.4.jar -D mapred.map.tasks=2 -input hdfs:///test/input -output hdfs:///test/output/WordCountOutput -mapper "python /Users/yaguangsun/mapper.py" -reducer "python /Users/yaguangsun/reducer.py" -file /Users/yaguangsun/mapper.py -file /Users/yaguangsun/reducer.py
```

When writing the command in terminal, keep a space between different parameter settings.

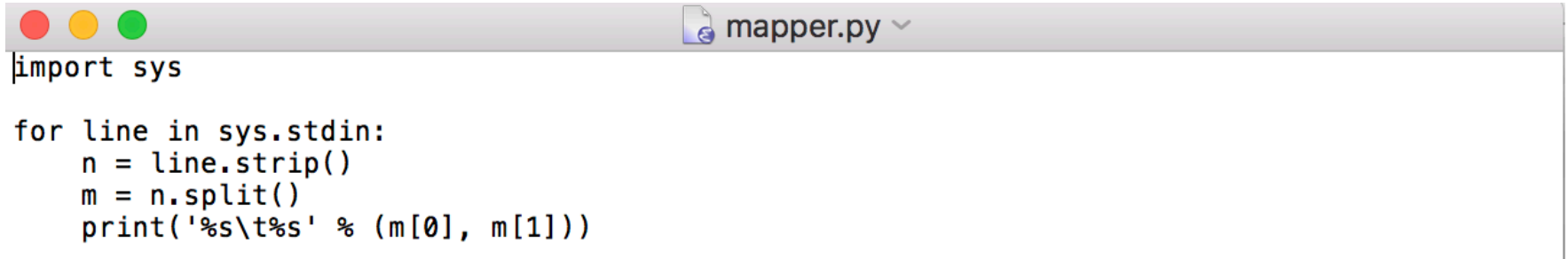
# MapReduce Coding: Pure Python

- Exercise 4: write down the **MapReduce code** to solve the problem from **exercise 1: calculate the total price for each invoice id** for a given document with the same format as shown in exercise 1. **Run your code** over the file **PriceSum1.txt** and **PriceSum2.txt** stored in **/test/input**

Invoice ID	Price
000	3
001	2
003	12
000	4
001	21
003	8
000	3
002	15

# MapReduce Coding: Pure Python

- **Solution 4: mapper.py**



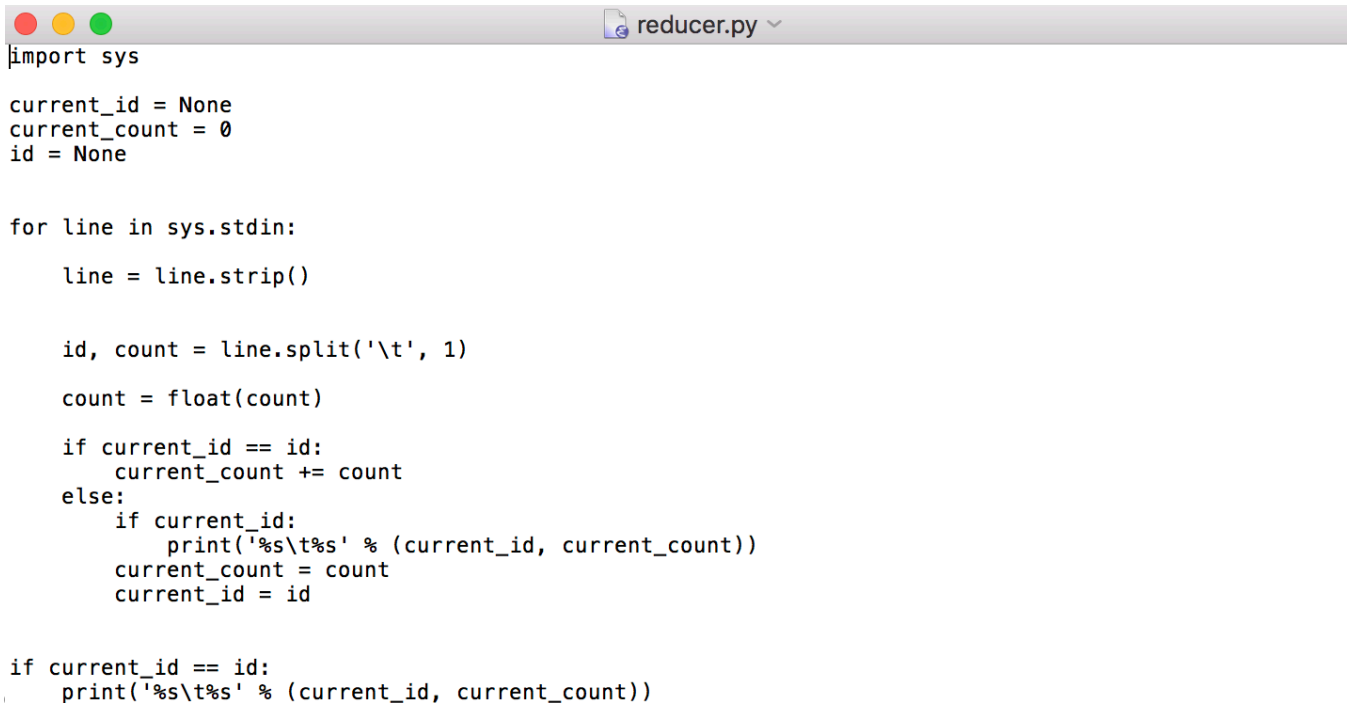
```
import sys

for line in sys.stdin:
    n = line.strip()
    m = n.split()
    print('%s\t%s' % (m[0], m[1]))
```



# MapReduce Coding: Pure Python

- **Solution 4: reducer.py**



```
import sys

current_id = None
current_count = 0
id = None

for line in sys.stdin:
    line = line.strip()

    id, count = line.split('\t', 1)
    count = float(count)

    if current_id == id:
        current_count += count
    else:
        if current_id:
            print('%s\t%s' % (current_id, current_count))
            current_count = count
            current_id = id

if current_id == id:
    print('%s\t%s' % (current_id, current_count))
```