# Running mapper and reducer locally

**Objective: Receive feedback quickly on your mapper and reducer code without running Hadoop overhead.**



### *Summary*


This tutorial shows how to test mapper and reducer code locally, on a Unix terminal, without launching a Hadoop job. By avoiding the Hadoop launch overhead, we quickly obtain useful feedback on the correctness of our mapper and reducer code. We also introduce some useful terminal utilities for working with large text files.


### *Getting Organized*

Let's say we are given the purchases.txt file and we want to compute how much money is paid with each of the payment methods (e.g. Amex, Visa, Cash, etc). The default setup in the udacity_training folder is as follows:

```
code/
    sample code
data/
    purchases.txt
    access_log
```

*Running code on Hadoop*

To keep things organized, let's create a new folder called *tutorial*.

```
[training@localhost udacity_training]$ mkdir tutorial
```

I will place my mapper.py and reducer.py files inside this folder. Now let's try running the Hadoop job directly, from tutorial folder, as follows.

Here I have already placed the purchases.txt file in HDFS in a directory called *input_1*.

```
[training@localhost tutorial]$ hs mapper.py reducer.py
input_1 tutorial_out
```

While Hadoop is wordy during the job progress, most of the logs refer to the map/reduce framework, and do not provide useful feedback on the correctness of our code. Alas, in this case, our job fails:

```
13/11/11 19:13:28 ERROR streaming.StreamJob: Job not
successful. Error: NA
13/11/11 19:13:28 INFO streaming.StreamJob: killJob...
Streaming Command Failed!
```

***There is Hope: the Alternative***

After an hour of staring into our code, we decide to test it on the terminal. First, we need to take a look at our data. The `head` command allows us to print the first few lines of a text file. It is especially helpful with big files, when we want to take a look at a few lines from the file, without printing all the content. It is used as follows (from inside the data folder):

```
[training@localhost data]$ head -n 5 purchases.txt
2012-01-01    09:00    San Jose    Men's Clothing
214.05    Amex
2012-01-01    09:00    Fort Worth    Women's Clothing
153.57    Visa
2012-01-01    09:00    San Diego    Music    66.08    Cash
2012-01-01    09:00    Pittsburgh    Pet Supplies
493.51    Discover
2012-01-01    09:00    Omaha    Children's Clothing
235.63    MasterCard
```

I am using `head` utility with an option `-n 5` which prints the first five lines of our file. We now can take a small sample from our big purchases.txt file and use it to run a local test. The command to do so is:

```
[training@localhost data]$ head -n 1000 purchases.txt
```

```
>> ../tutorial/test_in.txt
```

We are accessing the purchase.txt file in the data directory: the head utility takes the first 1000 lines from the file. The >> command will take the output from head and simultaneously create a file test_in.txt (if one does not already exist in tutorial) and write to it those 1000 lines. Note that we access the original file in the data directory, but we create the new file in tutorial, which is outside of data. In order for the commands to work properly, the udacity training folder should be organized as follows:

```
code/
    sample code
data/       (this is where we access the data file)
    purchases.txt
    access_log
tutorial/   (this is where we create the new file:
test_in.txt)
    mapper.py
    reducer.py
    test_in.txt
```

***Running the Code Locally***

We are ready to run our test locally. The command to do so from inside tutorial is:

```
[training@localhost tutorial]$ cat test_in.txt | python
mapper.py | sort | python reducer.py
```

Let's break it down into pieces. Surely a command which involves a cat and a python will be of some use!

We use `cat` to access the entire content of `test_in.txt`. The `|` command says to take the standard output of the command on the left (cat) and feed it as standard input to the command on the right. This way the mapper script receives as standard input the contents of our file and it can read it line by line.

We then pass the output of our mapper (lines of key-value pairs) to the sort utility. This will sort the output so that all rows with the same key are grouped together. Note that when running a Hadoop job, we don't

need to worry about sorting our results, because Hadoop does that for us (during the shuffle-sort phase). Here we need to include it since we are testing our mapper and reducer locally.

We then input the sorted key-value pairs into the reducer. It then prints (as standard output, on the terminal) the final reduced output. Alternatively, we can save it to a file by appending the `>> test_out.txt` command at the end.

The result of running the complete command on our mapper and reducer is:

```
  File "reducer.py", line 29
      salesTotal+ = float(thisSale)
                  ^
SyntaxError: invalid syntax
```

Finally some useful feedback! And it turns out that the problem is a simple syntax error. We can easily fix that and run the test locally once again.

```
[training@localhost tutorial]$ cat test_in.txt | python
mapper.py | sort | python reducer.py

Amex     43138.99
Cash     49065.39
Discover     52008.03
MasterCard     47020.69
Visa     52353.4
```

Great! We have obtained some reasonable results based on our small sample of 1000 records. We can now proceed to running the job on Hadoop with the complete data set with more confidence.

### *Conclusion*

This tutorial introduced a way of testing the mapper and reducer code locally, on Unix terminal. The error that we found in the local run is a simple, syntax error. However, by carefully studying the final output produced locally, we can uncover tricky mistakes in the logic of our code before even running a Hadoop job.