Correction TD2

1 Solutions TD2

Exercice 1.1

```
[ ]: %%file wordlengthcount.py
from mrjob.step import MRStep
from mrjob.job import MRJob
class WordLengthCount(MRJob):
    def steps(self):
        return [MRStep(mapper=self.mapper_get_lengths, reducer=self.
 ↪reducer_count_lengths)]
    def mapper_get_lengths(self, _, line):
        for word in line.split():
            yield str(len(word)).zfill(2), 1
    def reducer_count_lengths(self, key, values):
        yield key, sum(values)
if __name__ == '__main__':
    WordLengthCount.run()
```

```
[ ]: ! python3 wordlengthcount.py -r hadoop hdfs://localhost:9000/user/iset/
 ↪shakespeare.txt
```

Exercice 1.2

```
[ ]: %%file wordlengthcount2.py
from mrjob.step import MRStep
from mrjob.job import MRJob
class WordLengthCount(MRJob):
    def steps(self):
        return [MRStep(mapper=self.mapper_get_lengths, reducer=self.
 ↪reducer_count_lengths), MRStep(reducer=self.reducer_flatten_lengths)]
    def mapper_get_lengths(self, _, line):
        for word in line.split():
            yield str(len(word)).zfill(2), 1
    def reducer_count_lengths(self, key, values):
        yield str(sum(values)).zfill(6),key
    def reducer_flatten_lengths(self, key, values):
        for value in values:
            yield key, value
if __name__ == '__main__':
    WordLengthCount.run()
```

[ ]:
```
! python3 wordlengthcount2.py -r hadoop hdfs://localhost:9000/user/iset/
↪shakespeare.txt
```

Exercice 1.3

[ ]:
```
%%file wordlengthcount3.py
from mrjob.step import MRStep
from mrjob.job import MRJob
class WordLengthCount(MRJob):
    def steps(self):
        return [MRStep(mapper=self.mapper_get_lengths, reducer=self.
↪reducer_count_lengths)]
    def mapper_get_lengths(self, _, line):
        for word in line.split():
            intervalle = "[>10]"
            if(len(word) in range(1,6)):
                intervalle="[1-5]"
            elif(len(word) in range(6,11)):
                intervalle = "[6-10]"
            yield intervalle, 1
    def reducer_count_lengths(self, key, values):
        yield key, sum(values)
if __name__ == '__main__':
    WordLengthCount.run()
```

[ ]:
```
! python3 wordlengthcount3.py -r hadoop hdfs://localhost:9000/user/iset/
↪shakespeare.txt
```

Exercice 1.3 v2

[ ]:
```
%%file wordlengthcount32.py
from mrjob.step import MRStep
from mrjob.job import MRJob
class WordLengthCount(MRJob):
    def steps(self):
        return [MRStep(mapper=self.mapper_get_lengths, reducer=self.
↪reducer_count_lengths), MRStep(mapper=self.mapper_length_interval,␣
↪reducer=self.reducer_count_lengths)]
    def mapper_get_lengths(self, _, line):
        for word in line.split():
            yield str(len(word)).zfill(2), 1
    def reducer_count_lengths(self, key, values):
        yield key, sum(values)
    def mapper_length_interval(self, key, value):
        intervalle = "[>10]"
        if(int(key) in range(1,6)):
            intervalle="[1-5]"
        elif(int(key) in range(6,11)):
            intervalle = "[6-10]"
        yield intervalle, value
if __name__ == '__main__':
    WordLengthCount.run()
```

[ ]:
```
! python3 wordlengthcount32.py -r hadoop hdfs://localhost:9000/user/iset/
↪shakespeare.txt
```

Exercice 2

[ ]:
```
%%file amis.py
from mrjob.step import MRStep
from mrjob.job import MRJob
class AmisCommuns(MRJob):
    def steps(self):
        return [MRStep(mapper=self.mapper_creer_couples, reducer=self.
↪reducer_intersection)]
    def mapper_creer_couples(self, _, line):
        utilisateur = line.split(":")[0]
        amis = line.split(":")[1]
        for ami in amis.split(","):
            yield␣
↪"["+str(min(int(utilisateur),int(ami)))+"-"+str(max(int(utilisateur),int(ami)))+"]",amis
    def reducer_intersection(self, key, values):
        lst = [set(v.split(",")) for v in values]
        yield key, list(lst[0].intersection(*lst))
if __name__ == '__main__':
    AmisCommuns.run()
```

[ ]:
```
! python3 amis.py -r hadoop hdfs://localhost:9000/user/iset/ex2.txt
```