Exercise 1

Local WordCount

We first modified both the mapper and the reducer to be compatible with python 3
and then used a regex pattern in `mapper.py` matcher to better discriminate words from the lines produced by `cat ./data/<myfile>.txt`
for this exercise we consider capital words the same as lowercase, e.g. `Harry` = `harry` :

```
#!/usr/bin/env python
"""mapper.py"""

import sys
import re

# input comes from STDIN (standard input)
for line in sys.stdin:
    # split the line into words using regex
    words = re.split(r"[^A-Za-z]", line.strip().lower())
    # increase counters
    for word in words:
         if len(word) > 0 :
            # write the results to STDOUT (standard output);
            # what we output here will be the input for the
            # Reduce step, i.e. the input for reducer.py
            #
            # tab-delimited; the trivial word count is 1
            print (f'{word}\t1')
```

in this way we were able to get a list of words `<word> 1`

we then sort with `sort -k1,1` and pass the output to the reducer function
which we modified:

```python
"""reducer.py"""
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_word == word:
        current_count += count
    else:
        if current_word:
            # write result to STDOUT
            print (f"{current_word}\t{current_count}")
        current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print (f"{current_word}\t{current_count}")
```

To get the output we then ran the command

```
cat ./data/hp1.txt | python3 mapper.py | sort -k1,1 | python3 reducer.py | sort -k2,2 -nr >> out
```

The top 10 wordcount results we obtained in the head of the `output.txt` file:

```
the     3630
and     1924
to      1861
he      1758
a       1691
harry   1327
of      1267
was     1186
it      1185
you     1035
```

the complete output is in the `output.txt` file in the EX1 folder.