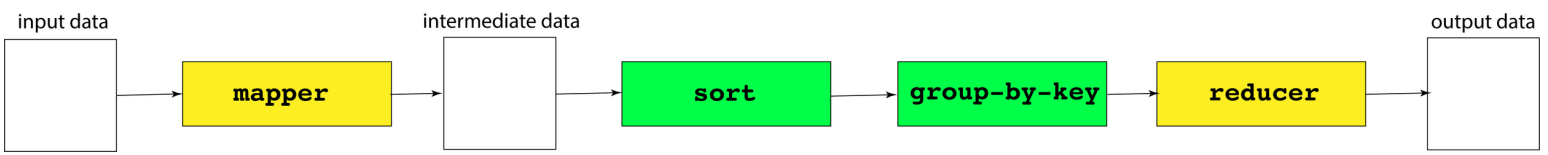
Lecture 16: Overview of MapReduce

- MapReduce is a parallel, distributed programming model and implementation used to process and generate large data sets.
  - The **map** component of a MapReduce job typically parses input data and distills it down to some intermediate result.
  - The **reduce** component of a MapReduce job collates these intermediate results and distills them down even further to the desired output.
  - The pipeline of processes involved in a MapReduce job is captured by the below illustration:



  - The processes shaded in yellow are programs specific to the data set being processed, whereas the processes shaded in green are present in all MapReduce pipelines.
  - We'll invest some energy over the next several slides explaining what a mapper, a reducer, and the group-by-key processes look like.

Lecture 16: Overview of MapReduce

- Here is an example of a map executable—written in Python—that reads an input file and outputs a line of the form **<word> 1** for every alphabetic token in that file.

```
import sys
import re

pattern = re.compile("^[a-z]+$") # matches purely alphabetic words
for line in sys.stdin:
    line = line.strip()
    tokens = line.split()
    for token in tokens:
        lowercaseword = token.lower()
        if pattern.match(lowercaseword):
            print '%s 1' % lowercaseword
```

- The above script can be invoked as follows to generate the stream of words in Anna Karenina:

```
myth61:$ cat anna-karenina.txt | ./word-count-mapper.py
happy 1
families 1
are 1
... // some 340000 words omitted for brevity
to 1
put 1
into 1
```

Lecture 16: Overview of MapReduce

- **`group-by-key`** contributes to all MapReduce pipelines, not just this one. Our **`group-by-key.py`** executable—presented on the next slide—assumes the mapper's output has been sorted so multiple instances of the same key are more easily grouped together, as with:

```
myth61:$ cat anna-karenina.txt | ./word-count-mapper.py | sort
a 1
a 1
a 1
a 1
a 1 // plus 6064 additional copies of this same line
...
zigzag 1
zoological 1
zoological 1
zoology 1
zu 1
myth61:$ cat anna-karenina.txt | ./word-count-mapper.py | sort | ./group-by-key.py
a 1 1 1 1 1 // plus 6064 more 1's on this same line
...
zeal 1 1 1
zealously 1
zest 1
zhivahov 1
zigzag 1
zoological 1 1
zoology 1
zu 1
```

Lecture 16: Overview of MapReduce

- Presented below is a short (but dense) Python script that reads from an incoming stream of key-value pairs, sorted by key, and outputs the same content, save for the fact that all lines with the same key have been merged into a single line, where all values themselves have been collapsed to a single vector-of-values presentation.
  - The implementation relies on some nontrivial features of Python that don't exist in C or C++. Don't worry about the implementation too much, as it's really just here for completeness.
  - Since you know what the overall script does, you can intuit what each line of it must do.

```
from itertools import groupby
from operator import itemgetter
import sys

def read_mapper_output(file):
    for line in file:
        yield line.strip().split(' ')

data = read_mapper_output(sys.stdin)
for key, keygroup in groupby(data, itemgetter(0)):
    values = ' '.join(sorted(v for k, v in keygroup))
    print "%s %s" % (key, values)
```

Lecture 16: Overview of MapReduce

- A reducer is a problem-specific program that expects a sorted input file, where each line is a key/vector-of-values pair as might be produced by our **`./group-by-key.py`** script.

```
import sys

def read_mapper_output(file):
    for line in file:
        yield line.strip().split(' ')

for vec in read_mapper_output(sys.stdin):
    word = vec[0]
    count = sum(int(number) for number in vec[1:])
    print "%s %d" % (word, count)
```

- The above reducer could be fed the sorted, key-grouped output of the previously supplied mapper if this chain of piped executables is supplied on the command line:

```
myth61:$ cat anna-karenina.txt | ./word-count-mapper.py | sort \
                               | ./group-by-key.py | ./word-count-reducer.py
a 6069
abandon 6
abandoned 9
abandonment 1
...
zoological 2
zoology 1
zu 1
```