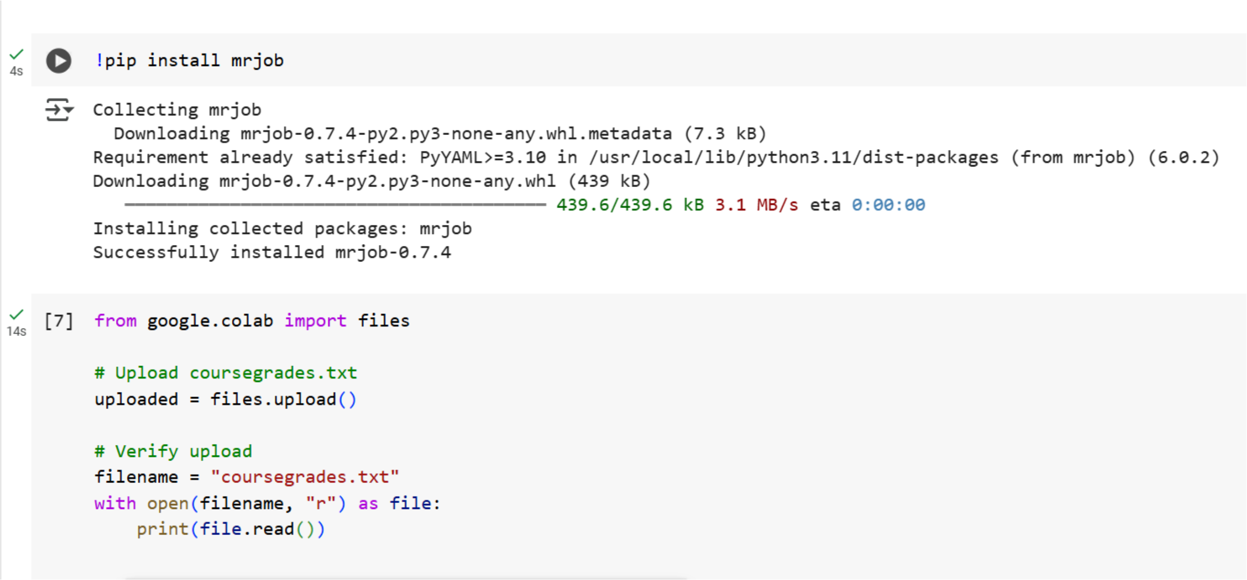
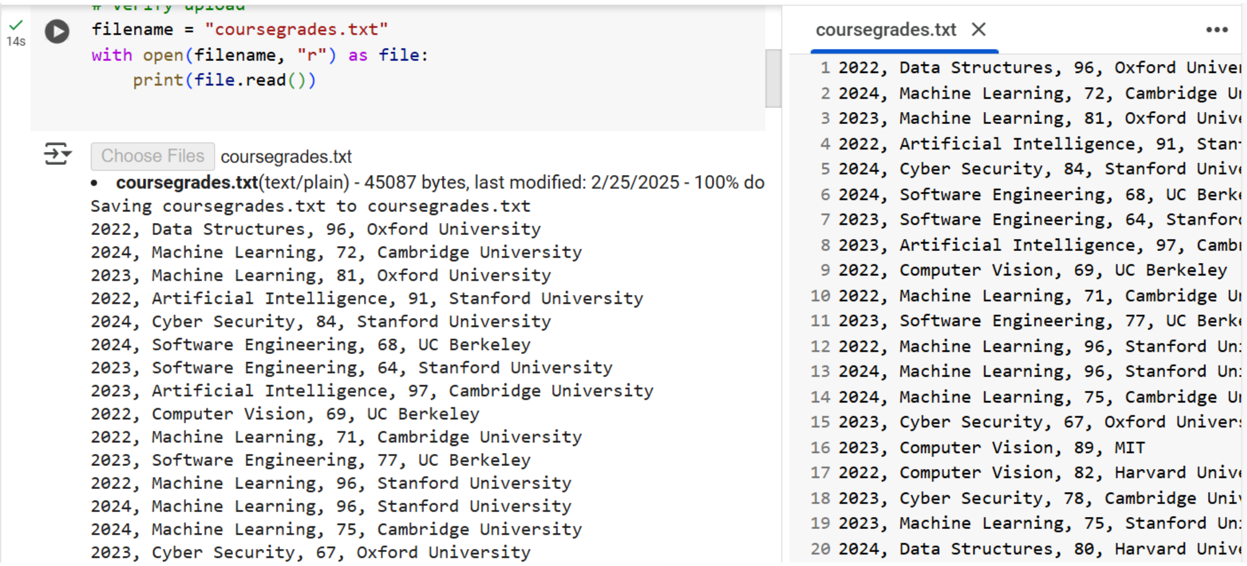
Screenshots of the code and the output files:



```
!pip install mrjob
```

```
Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.11/dist-packages (from mrjob) (6.0.2)
Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.6/439.6 kB 3.1 MB/s eta 0:00:00
Installing collected packages: mrjob
Successfully installed mrjob-0.7.4
```

```python
from google.colab import files

# Upload coursegrades.txt
uploaded = files.upload()

# Verify upload
filename = "coursegrades.txt"
with open(filename, "r") as file:
    print(file.read())
```



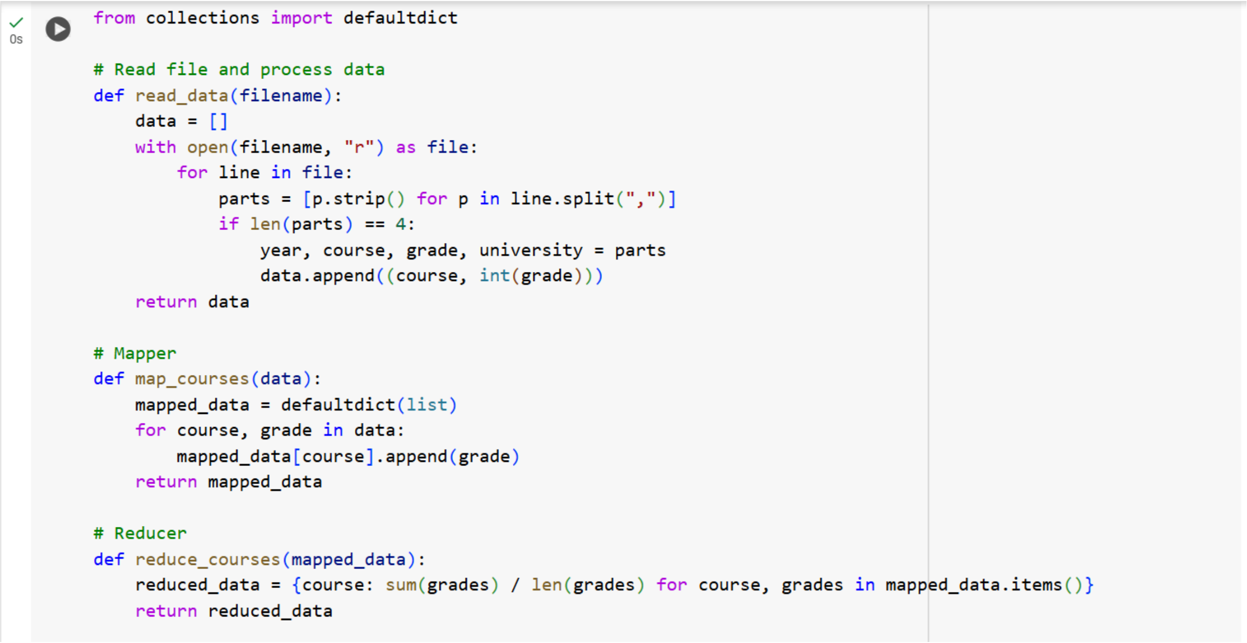
```python
# Verify upload
filename = "coursegrades.txt"
with open(filename, "r") as file:
    print(file.read())
```

coursegrades.txt

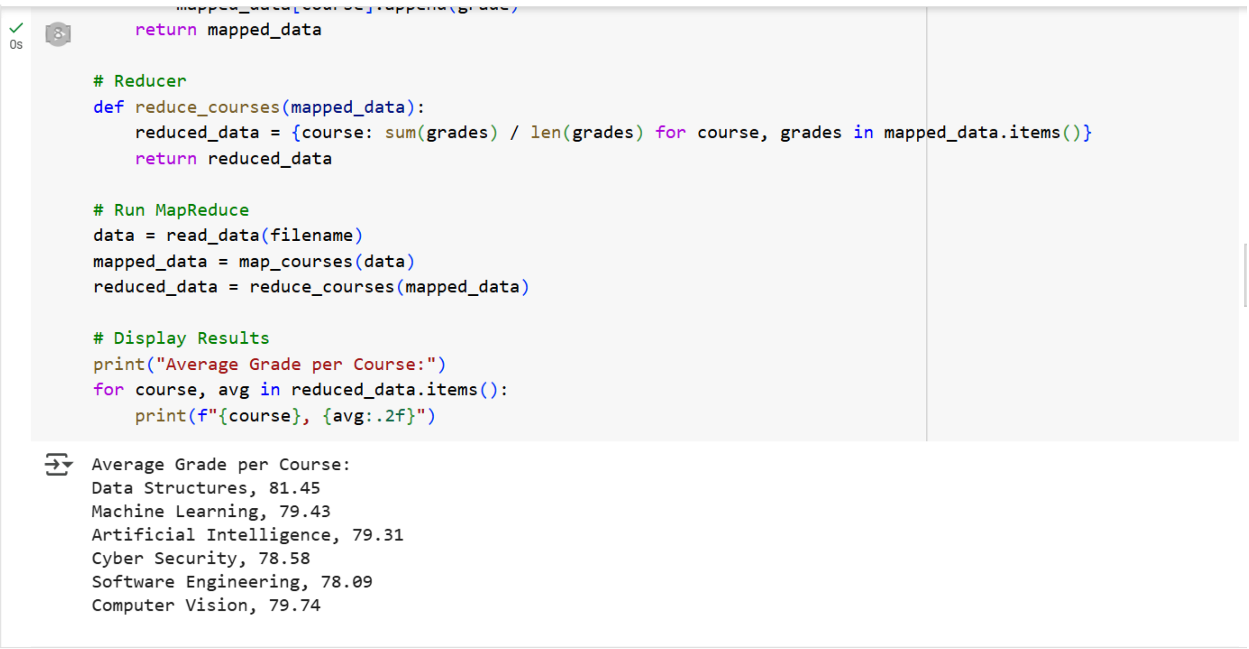
- **coursegrades.txt**(text/plain) - 45087 bytes, last modified: 2/25/2025 - 100% do

```
Saving coursegrades.txt to coursegrades.txt
2022, Data Structures, 96, Oxford University
2024, Machine Learning, 72, Cambridge University
2023, Machine Learning, 81, Oxford University
2022, Artificial Intelligence, 91, Stanford University
2024, Cyber Security, 84, Stanford University
2024, Software Engineering, 68, UC Berkeley
2023, Software Engineering, 64, Stanford University
2023, Artificial Intelligence, 97, Cambridge University
2022, Computer Vision, 69, UC Berkeley
2022, Machine Learning, 71, Cambridge University
2023, Software Engineering, 77, UC Berkeley
2022, Machine Learning, 96, Stanford University
2024, Machine Learning, 96, Stanford University
2024, Machine Learning, 75, Cambridge University
2023, Cyber Security, 67, Oxford University
```

```
2022, Data Structures, 96, Oxford Unive
2024, Machine Learning, 72, Cambridge U
2023, Machine Learning, 81, Oxford Univ
2022, Artificial Intelligence, 91, Stan
2024, Cyber Security, 84, Stanford Univ
2024, Software Engineering, 68, UC Berk
2023, Software Engineering, 64, Stanfor
2023, Artificial Intelligence, 97, Camb
2022, Computer Vision, 69, UC Berkeley
2022, Machine Learning, 71, Cambridge U
2023, Software Engineering, 77, UC Berk
2022, Machine Learning, 96, Stanford Un
2024, Machine Learning, 96, Stanford Un
2024, Machine Learning, 75, Cambridge U
2023, Cyber Security, 67, Oxford Univer
2023, Computer Vision, 89, MIT
2022, Computer Vision, 82, Harvard Univ
2023, Cyber Security, 78, Cambridge Uni
2023, Machine Learning, 75, Stanford Un
2024, Data Structures, 80, Harvard Univ
```

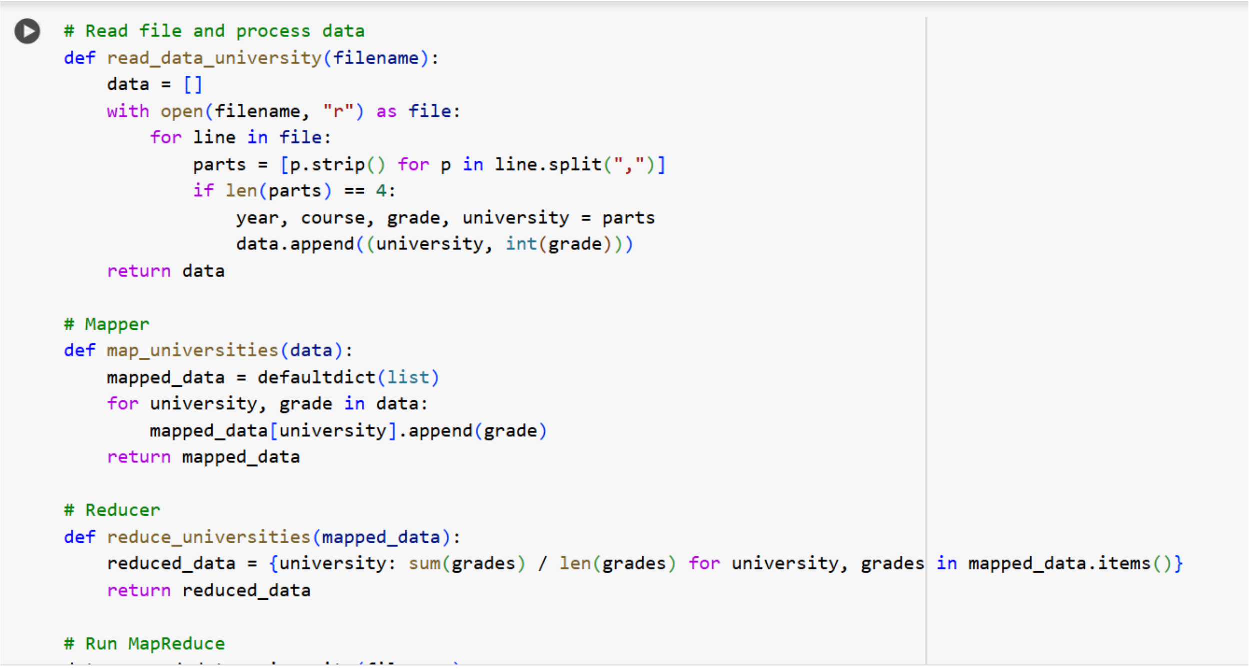


```python
from collections import defaultdict

# Read file and process data
def read_data(filename):
    data = []
    with open(filename, "r") as file:
        for line in file:
            parts = [p.strip() for p in line.split(",")]
            if len(parts) == 4:
                year, course, grade, university = parts
                data.append((course, int(grade)))
    return data

# Mapper
def map_courses(data):
    mapped_data = defaultdict(list)
    for course, grade in data:
        mapped_data[course].append(grade)
    return mapped_data

# Reducer
def reduce_courses(mapped_data):
    reduced_data = {course: sum(grades) / len(grades) for course, grades in mapped_data.items()}
    return reduced_data
```

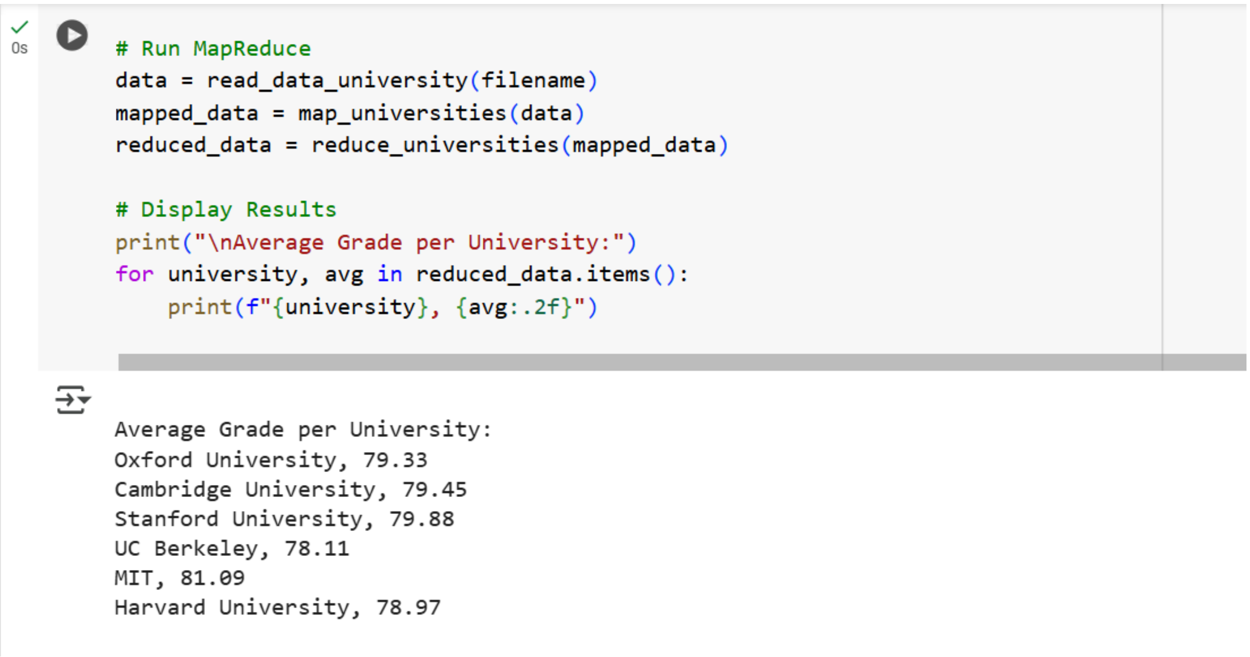


```python
        mapped_data[course].append(grade)
    return mapped_data

# Reducer
def reduce_courses(mapped_data):
    reduced_data = {course: sum(grades) / len(grades) for course, grades in mapped_data.items()}
    return reduced_data

# Run MapReduce
data = read_data(filename)
mapped_data = map_courses(data)
reduced_data = reduce_courses(mapped_data)

# Display Results
print("Average Grade per Course:")
for course, avg in reduced_data.items():
    print(f"{course}, {avg:.2f}")
```

```
Average Grade per Course:
Data Structures, 81.45
Machine Learning, 79.43
Artificial Intelligence, 79.31
Cyber Security, 78.58
Software Engineering, 78.09
Computer Vision, 79.74
```

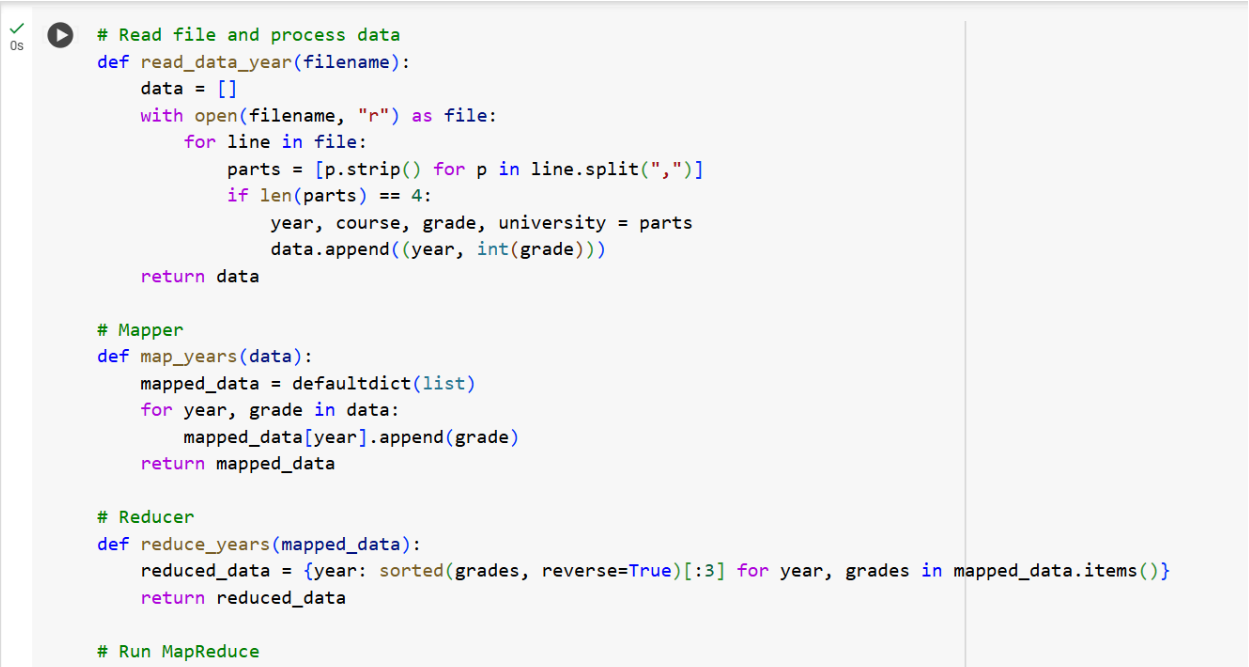


```python
# Read file and process data
def read_data_university(filename):
    data = []
    with open(filename, "r") as file:
        for line in file:
            parts = [p.strip() for p in line.split(",")]
            if len(parts) == 4:
                year, course, grade, university = parts
                data.append((university, int(grade)))
    return data

# Mapper
def map_universities(data):
    mapped_data = defaultdict(list)
    for university, grade in data:
        mapped_data[university].append(grade)
    return mapped_data

# Reducer
def reduce_universities(mapped_data):
    reduced_data = {university: sum(grades) / len(grades) for university, grades in mapped_data.items()}
    return reduced_data

# Run MapReduce
```

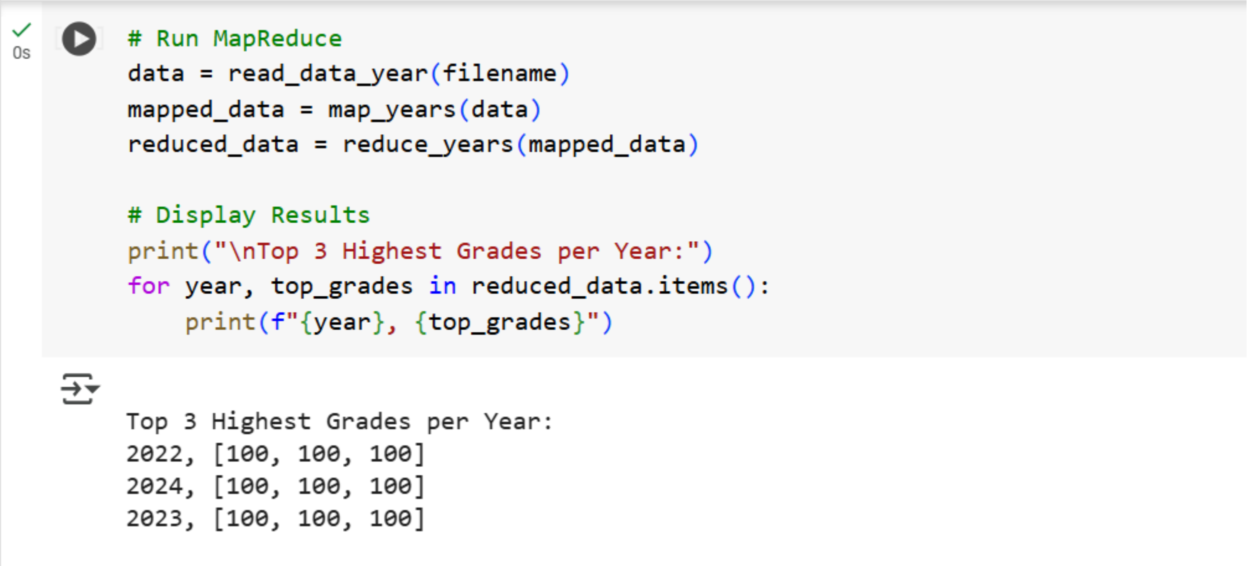


```python
# Run MapReduce
data = read_data_university(filename)
mapped_data = map_universities(data)
reduced_data = reduce_universities(mapped_data)

# Display Results
print("\nAverage Grade per University:")
for university, avg in reduced_data.items():
    print(f"{university}, {avg:.2f}")
```

```
Average Grade per University:
Oxford University, 79.33
Cambridge University, 79.45
Stanford University, 79.88
UC Berkeley, 78.11
MIT, 81.09
Harvard University, 78.97
```

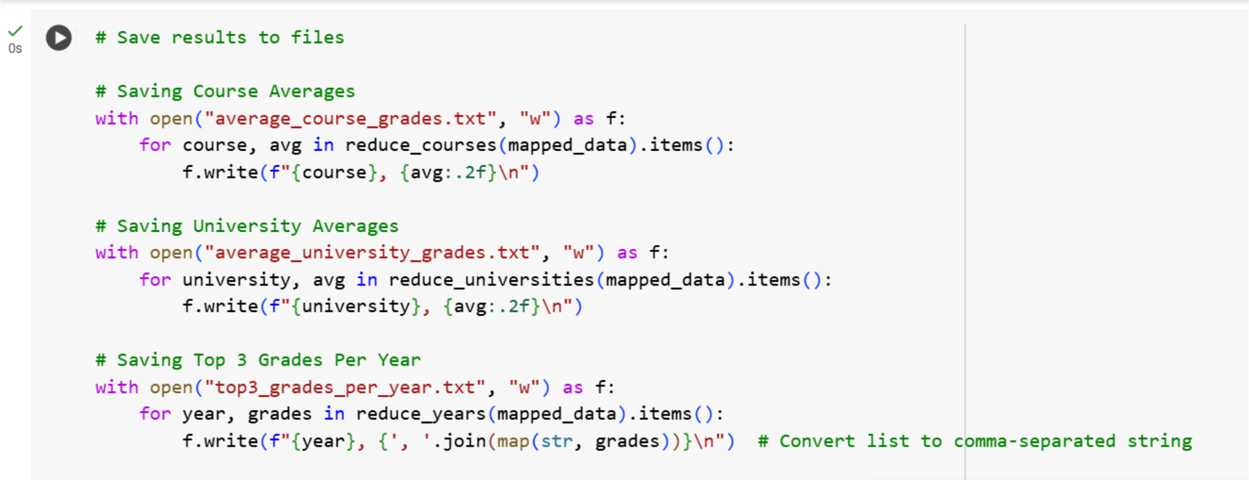


```
# Read file and process data
def read_data_year(filename):
    data = []
    with open(filename, "r") as file:
        for line in file:
            parts = [p.strip() for p in line.split(",")]
            if len(parts) == 4:
                year, course, grade, university = parts
                data.append((year, int(grade)))
    return data

# Mapper
def map_years(data):
    mapped_data = defaultdict(list)
    for year, grade in data:
        mapped_data[year].append(grade)
    return mapped_data

# Reducer
def reduce_years(mapped_data):
    reduced_data = {year: sorted(grades, reverse=True)[:3] for year, grades in mapped_data.items()}
    return reduced_data

# Run MapReduce
```

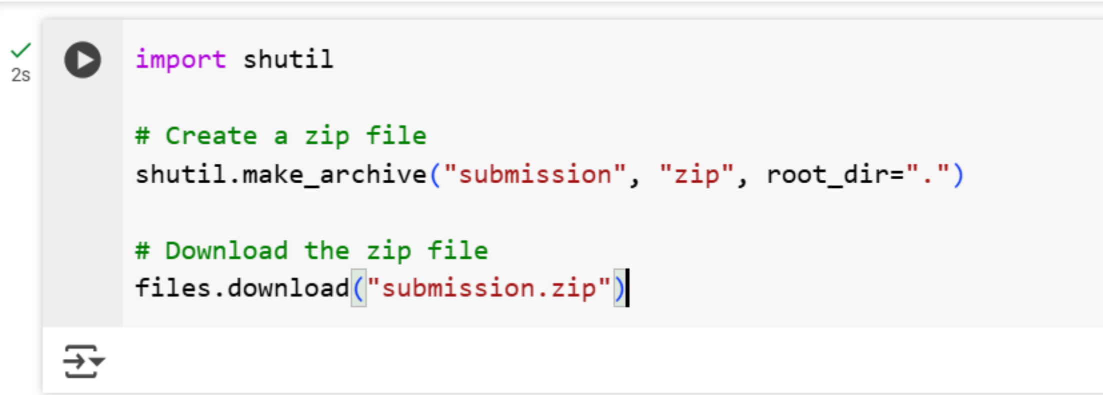


```
# Run MapReduce
data = read_data_year(filename)
mapped_data = map_years(data)
reduced_data = reduce_years(mapped_data)

# Display Results
print("\nTop 3 Highest Grades per Year:")
for year, top_grades in reduced_data.items():
    print(f"{year}, {top_grades}")
```

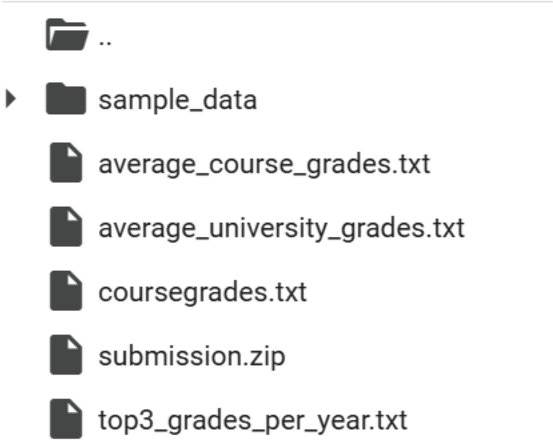
```

Top 3 Highest Grades per Year:
2022, [100, 100, 100]
2024, [100, 100, 100]
2023, [100, 100, 100]
```



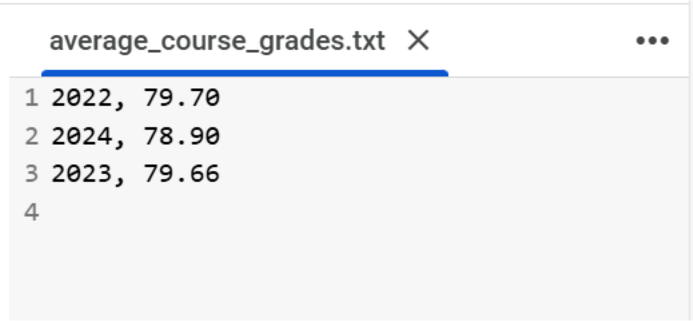
```python
# Save results to files

# Saving Course Averages
with open("average_course_grades.txt", "w") as f:
    for course, avg in reduce_courses(mapped_data).items():
        f.write(f"{course}, {avg:.2f}\n")

# Saving University Averages
with open("average_university_grades.txt", "w") as f:
    for university, avg in reduce_universities(mapped_data).items():
        f.write(f"{university}, {avg:.2f}\n")

# Saving Top 3 Grades Per Year
with open("top3_grades_per_year.txt", "w") as f:
    for year, grades in reduce_years(mapped_data).items():
        f.write(f"{year}, {', '.join(map(str, grades))}\n")  # Convert list to comma-separated string
```



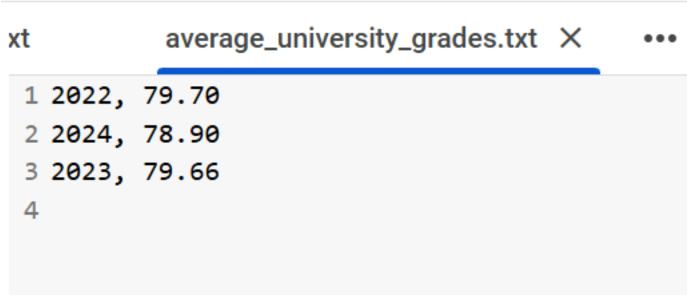
```python
import shutil

# Create a zip file
shutil.make_archive("submission", "zip", root_dir=".")

# Download the zip file
files.download("submission.zip")
```



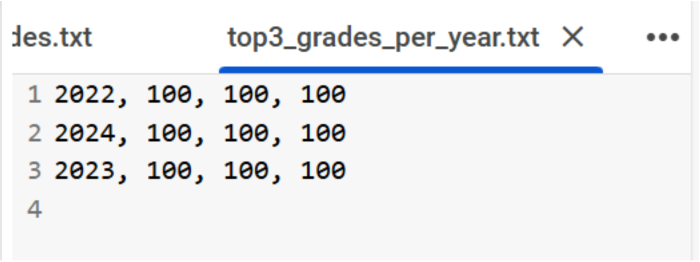
```
..
sample_data
average_course_grades.txt
average_university_grades.txt
coursegrades.txt
submission.zip
top3_grades_per_year.txt
```



```
2022, 79.70
2024, 78.90
2023, 79.66
```



```
2022, 79.70
2024, 78.90
2023, 79.66
```



```
2022, 100, 100, 100
2024, 100, 100, 100
2023, 100, 100, 100
```

Brief Description of the tasks:

Task 1: Average Grade per Course

Approach:

1. **Read the dataset** (coursegrades.txt) and extract relevant columns: Course Name and Grade.
2. **Map Step:** Create key-value pairs where the key is the **Course Name** and the value is the **Grade**.
3. **Reduce Step:** Aggregate all grades for each course, compute the **average**, and store the result.
4. **Output:** A list of courses with their average grades.

Task 2: Average Grade per University

Approach:

1. **Extract** the University Name and Grade from the dataset.
2. **Map Step:** Create key-value pairs where the key is the **University Name** and the value is the **Grade**.
3. **Reduce Step:** Aggregate all grades for each university, compute the **average**, and store the result.
4. **Output:** A list of universities with their average grades.

Bonus Task: Top 3 Highest Grades per Year

Approach:

1. **Extract** the Year and Grade from the dataset.
2. **Map Step:** Create key-value pairs where the key is the **Year** and the value is the **Grade**.
3. **Reduce Step:** Sort all grades for each year in **descending order** and select the **top 3 highest grades**.
4. **Output:** The top 3 grades for each year.