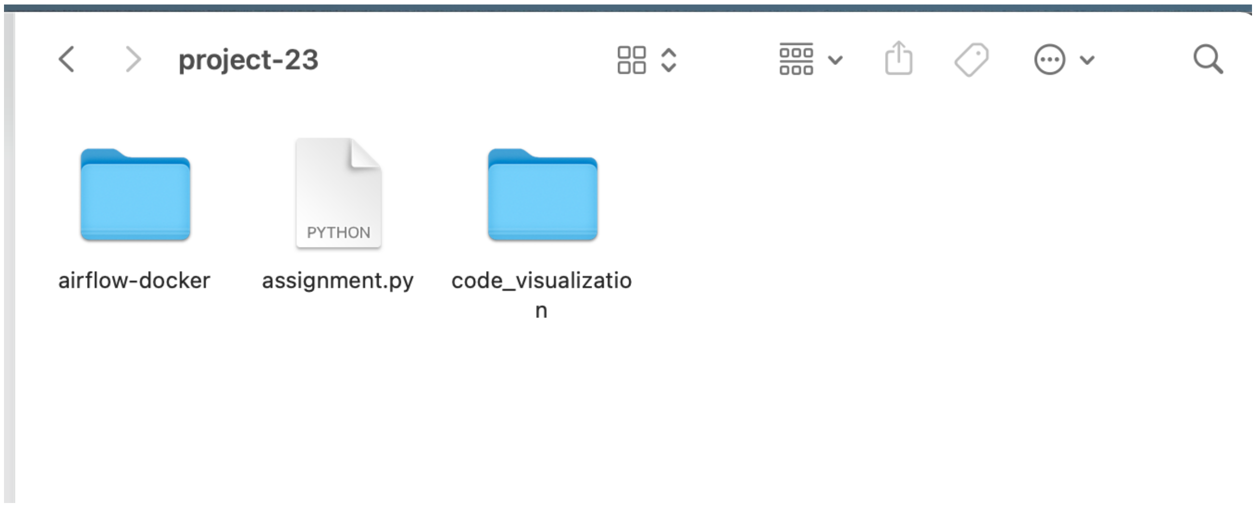
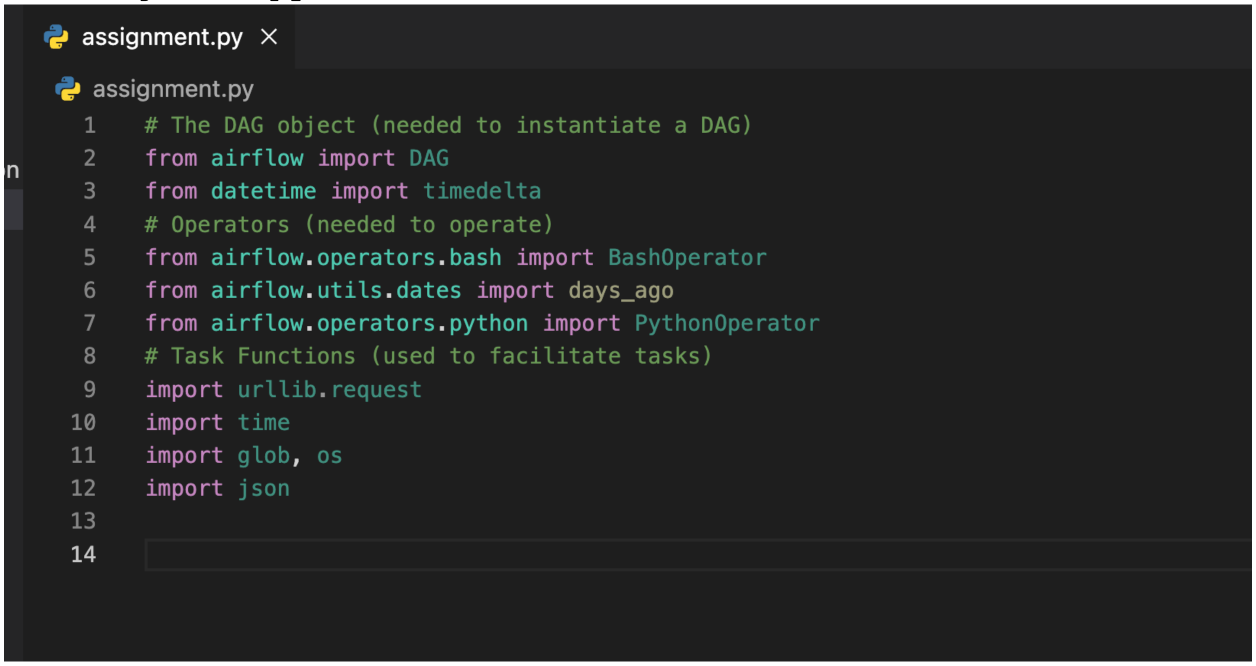
Part 1: Code Development

1. Provide a screenshot of the `project-23` folder with the `code visualization` folder, `airflow-docker` folder, and `assignment.py` file within it.

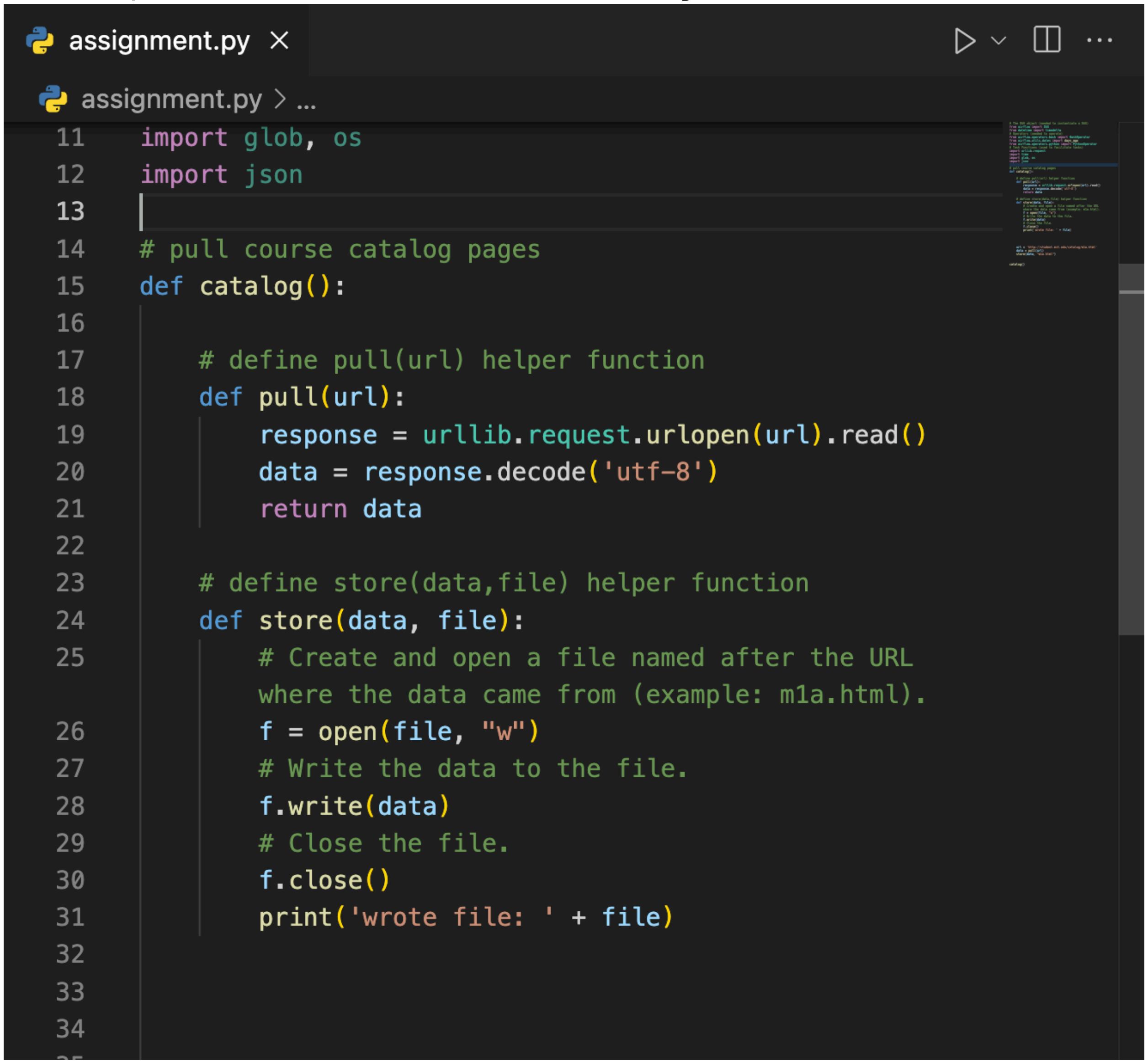


2. Provide a screenshot to show that you have imported the DAG object, the *operators*, and all of the necessary *task functions* into the `assignment.py` file.



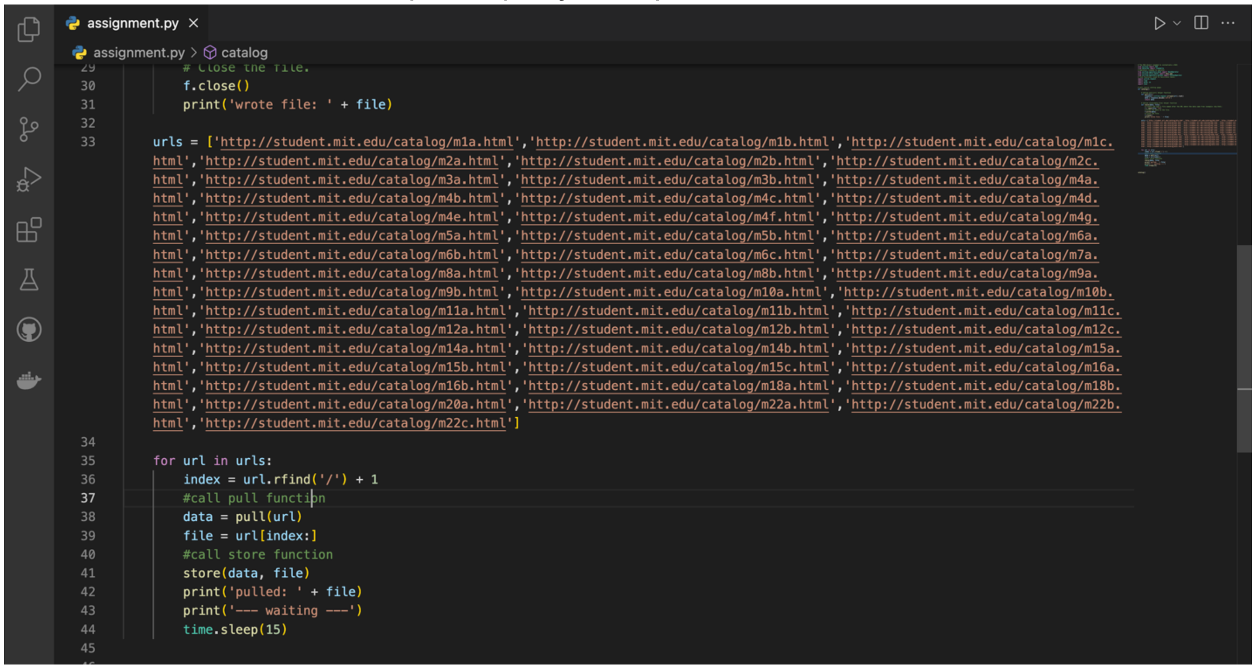
```python
# The DAG object (needed to instantiate a DAG)
from airflow import DAG
from datetime import timedelta
# Operators (needed to operate)
from airflow.operators.bash import BashOperator
from airflow.utils.dates import days_ago
from airflow.operators.python import PythonOperator
# Task Functions (used to facilitate tasks)
import urllib.request
import time
import glob, os
import json

```

3. Provide a screenshot of the `pull(url)` and `store(data, file)` helper *functions* defined inside of the `catalog()` *task*.



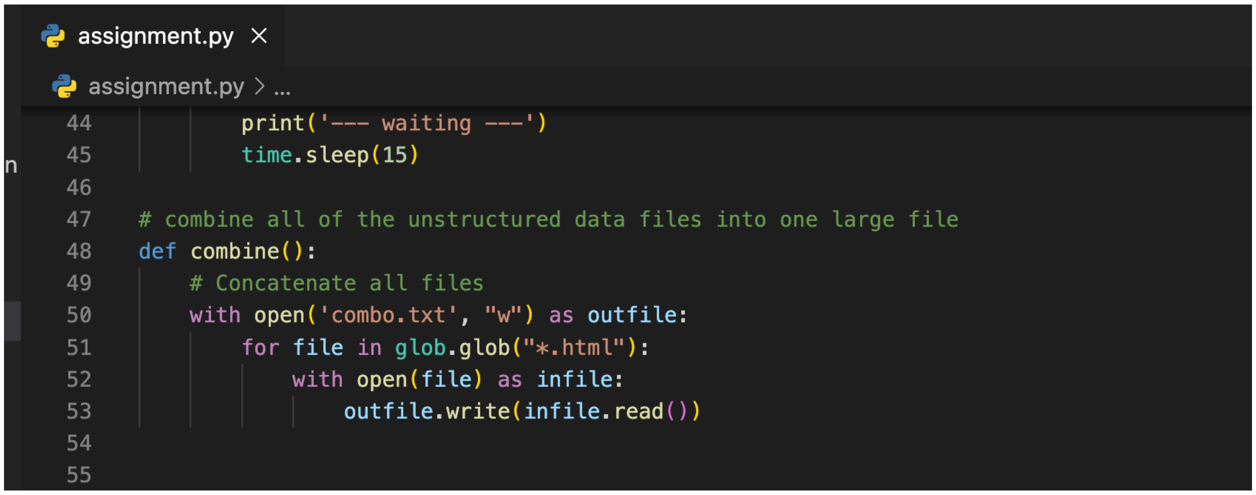
```python
import glob, os
import json

# pull course catalog pages
def catalog():

    # define pull(url) helper function
    def pull(url):
        response = urllib.request.urlopen(url).read()
        data = response.decode('utf-8')
        return data

    # define store(data,file) helper function
    def store(data, file):
        # Create and open a file named after the URL
        where the data came from (example: m1a.html).
        f = open(file, "w")
        # Write the data to the file.
        f.write(data)
        # Close the file.
        f.close()
        print('wrote file: ' + file)
```

4. Provide a screenshot of the entire `catalog()` *function*, including the `urls` *list* and the `for` *loop* that you just implemented.



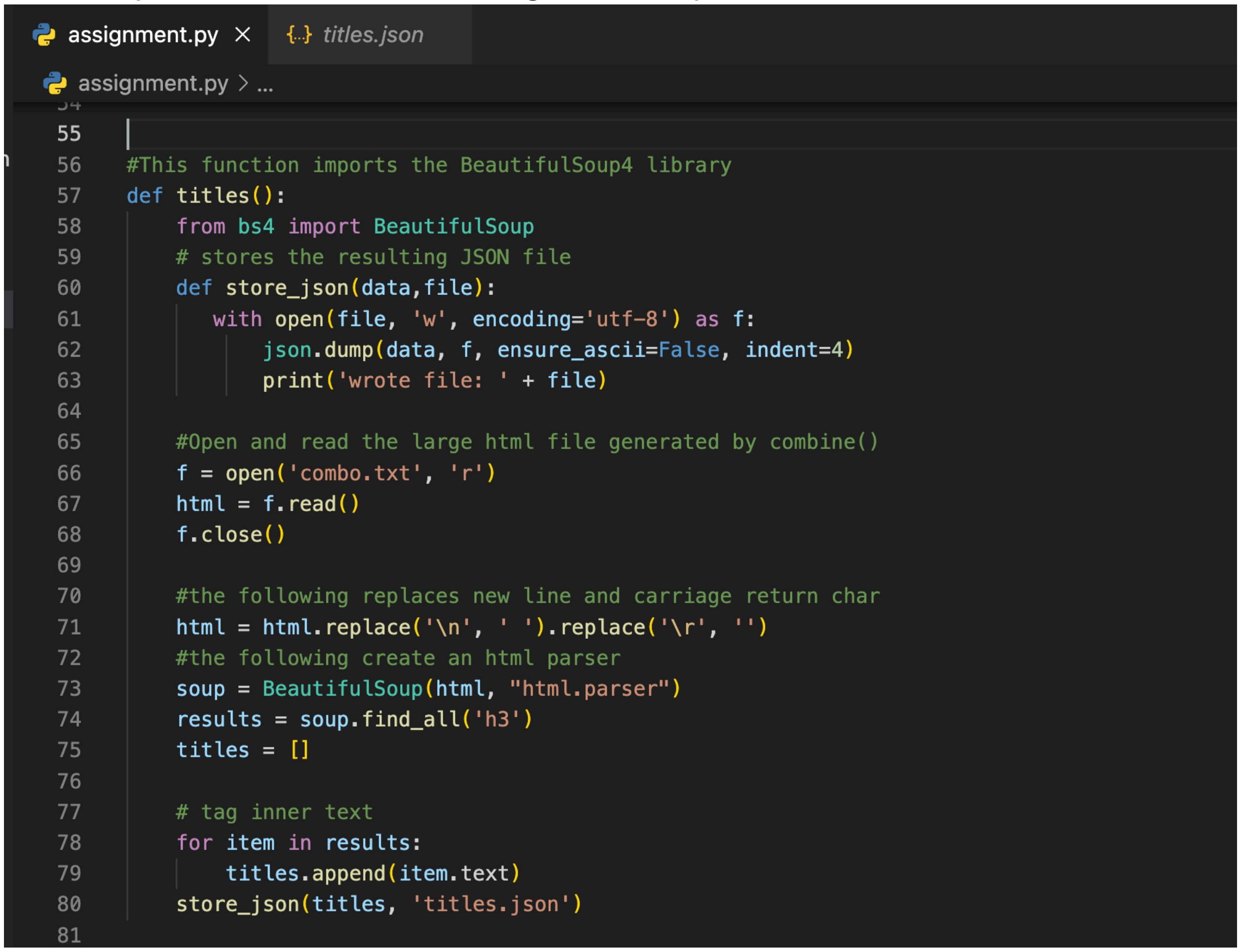
```
        # Close the file.
        f.close()
        print('wrote file: ' + file)

    urls = ['http://student.mit.edu/catalog/m1a.html','http://student.mit.edu/catalog/m1b.html','http://student.mit.edu/catalog/m1c.
    html','http://student.mit.edu/catalog/m2a.html','http://student.mit.edu/catalog/m2b.html','http://student.mit.edu/catalog/m2c.
    html','http://student.mit.edu/catalog/m3a.html','http://student.mit.edu/catalog/m3b.html','http://student.mit.edu/catalog/m4a.
    html','http://student.mit.edu/catalog/m4b.html','http://student.mit.edu/catalog/m4c.html','http://student.mit.edu/catalog/m4d.
    html','http://student.mit.edu/catalog/m4e.html','http://student.mit.edu/catalog/m4f.html','http://student.mit.edu/catalog/m4g.
    html','http://student.mit.edu/catalog/m5a.html','http://student.mit.edu/catalog/m5b.html','http://student.mit.edu/catalog/m6a.
    html','http://student.mit.edu/catalog/m6b.html','http://student.mit.edu/catalog/m6c.html','http://student.mit.edu/catalog/m7a.
    html','http://student.mit.edu/catalog/m8a.html','http://student.mit.edu/catalog/m8b.html','http://student.mit.edu/catalog/m9a.
    html','http://student.mit.edu/catalog/m9b.html','http://student.mit.edu/catalog/m10a.html','http://student.mit.edu/catalog/m10b.
    html','http://student.mit.edu/catalog/m11a.html','http://student.mit.edu/catalog/m11b.html','http://student.mit.edu/catalog/m11c.
    html','http://student.mit.edu/catalog/m12a.html','http://student.mit.edu/catalog/m12b.html','http://student.mit.edu/catalog/m12c.
    html','http://student.mit.edu/catalog/m14a.html','http://student.mit.edu/catalog/m14b.html','http://student.mit.edu/catalog/m15a.
    html','http://student.mit.edu/catalog/m15b.html','http://student.mit.edu/catalog/m15c.html','http://student.mit.edu/catalog/m16a.
    html','http://student.mit.edu/catalog/m16b.html','http://student.mit.edu/catalog/m18a.html','http://student.mit.edu/catalog/m18b.
    html','http://student.mit.edu/catalog/m20a.html','http://student.mit.edu/catalog/m22a.html','http://student.mit.edu/catalog/m22b.
    html','http://student.mit.edu/catalog/m22c.html']

    for url in urls:
        index = url.rfind('/') + 1
        #call pull function
        data = pull(url)
        file = url[index:]
        #call store function
        store(data, file)
        print('pulled: ' + file)
        print('--- waiting ---')
        time.sleep(15)

```

5. Provide a screenshot of the `combine()` *method* with the correct code to combine the files.



```
        print('--- waiting ---')
        time.sleep(15)

# combine all of the unstructured data files into one large file
def combine():
    # Concatenate all files
    with open('combo.txt', "w") as outfile:
        for file in glob.glob("*.html"):
            with open(file) as infile:
                outfile.write(infile.read())


```

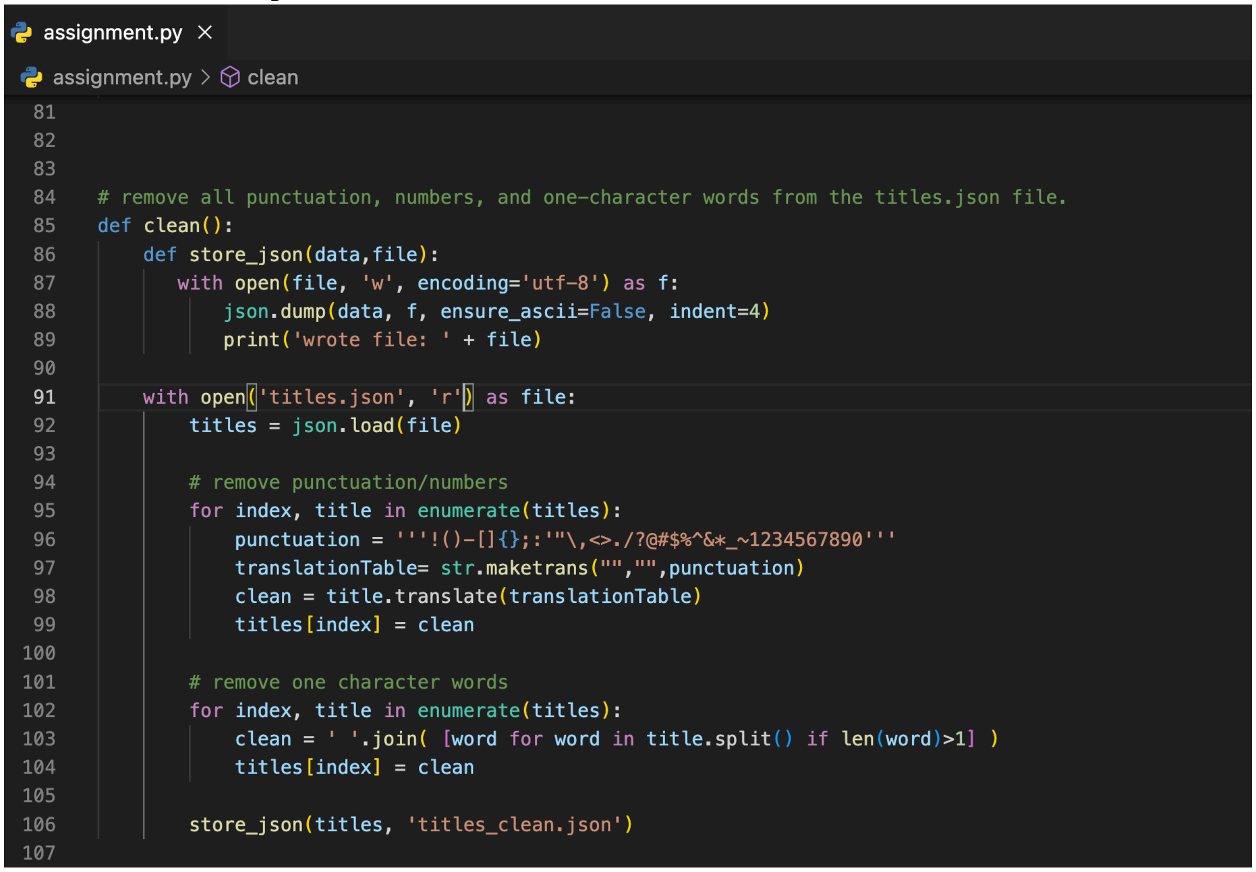
6. Provide a screenshot of the completed `titles()` *method* with the correct code to open and read the HTML file generated by the `combine()` *function*.



```python

#This function imports the BeautifulSoup4 library
def titles():
    from bs4 import BeautifulSoup
    # stores the resulting JSON file
    def store_json(data,file):
       with open(file, 'w', encoding='utf-8') as f:
           json.dump(data, f, ensure_ascii=False, indent=4)
           print('wrote file: ' + file)

    #Open and read the large html file generated by combine()
    f = open('combo.txt', 'r')
    html = f.read()
    f.close()

    #the following replaces new line and carriage return char
    html = html.replace('\n', ' ').replace('\r', '')
    #the following create an html parser
    soup = BeautifulSoup(html, "html.parser")
    results = soup.find_all('h3')
    titles = []

    # tag inner text
    for item in results:
        titles.append(item.text)
    store_json(titles, 'titles.json')

```

7. Provide a screenshot of the fully implemented `clean()` *method* with the correct code to remove all punctuation, numbers, and one-character words

from the `titles.json` file.

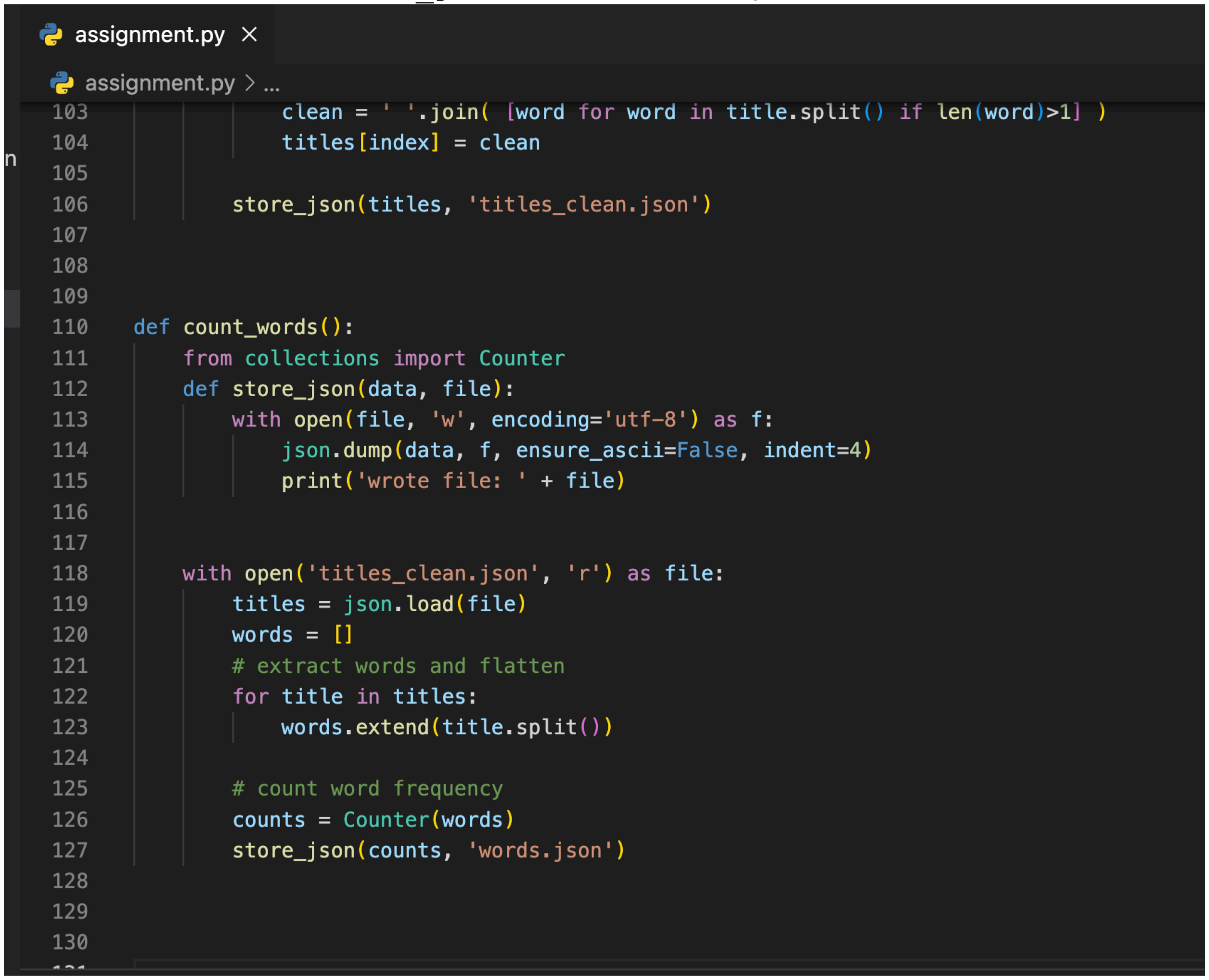


assignment.py ×

assignment.py > clean

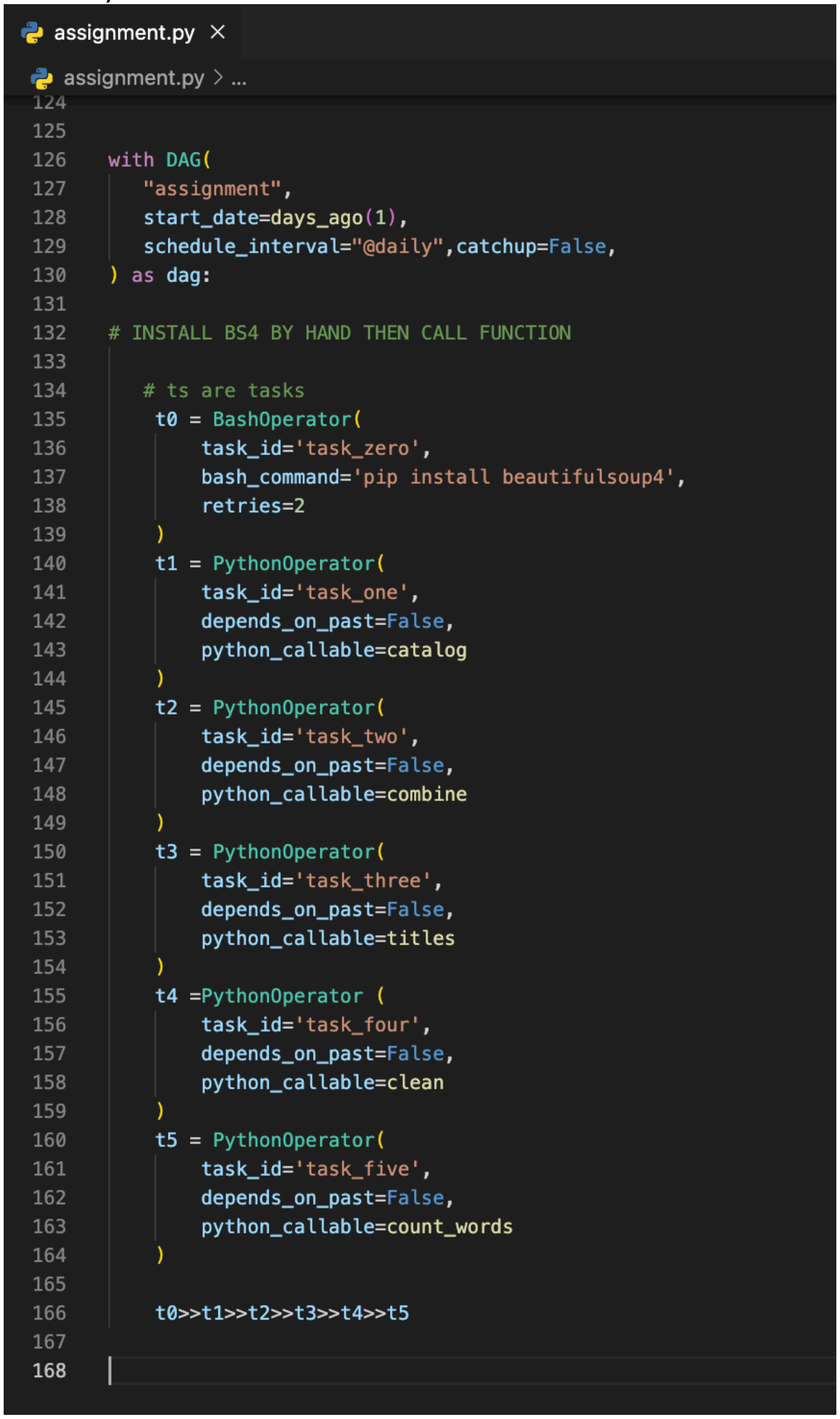
```python



# remove all punctuation, numbers, and one-character words from the titles.json file.
def clean():
    def store_json(data,file):
        with open(file, 'w', encoding='utf-8') as f:
            json.dump(data, f, ensure_ascii=False, indent=4)
            print('wrote file: ' + file)

    with open('titles.json', 'r') as file:
        titles = json.load(file)

        # remove punctuation/numbers
        for index, title in enumerate(titles):
            punctuation = '''!()-[]{};:'"\,<>./?@#$%^&*_~1234567890'''
            translationTable= str.maketrans("","",punctuation)
            clean = title.translate(translationTable)
            titles[index] = clean

        # remove one character words
        for index, title in enumerate(titles):
            clean = ' '.join( [word for word in title.split() if len(word)>1] )
            titles[index] = clean

        store_json(titles, 'titles_clean.json')

```

8. Provide a screenshot of the completed `count_words()` *method* with the correct code to call the `store_json(data,file)` helper *function*.



```python
            clean = ' '.join( [word for word in title.split() if len(word)>1] )
            titles[index] = clean

        store_json(titles, 'titles_clean.json')



def count_words():
    from collections import Counter
    def store_json(data, file):
        with open(file, 'w', encoding='utf-8') as f:
            json.dump(data, f, ensure_ascii=False, indent=4)
            print('wrote file: ' + file)


    with open('titles_clean.json', 'r') as file:
        titles = json.load(file)
        words = []
        # extract words and flatten
        for title in titles:
            words.extend(title.split())

        # count word frequency
        counts = Counter(words)
        store_json(counts, 'words.json')
```

9. Provide a screenshot of the DAG declaration with all six *tasks*, from `t0` to `t5`, correctly defined.

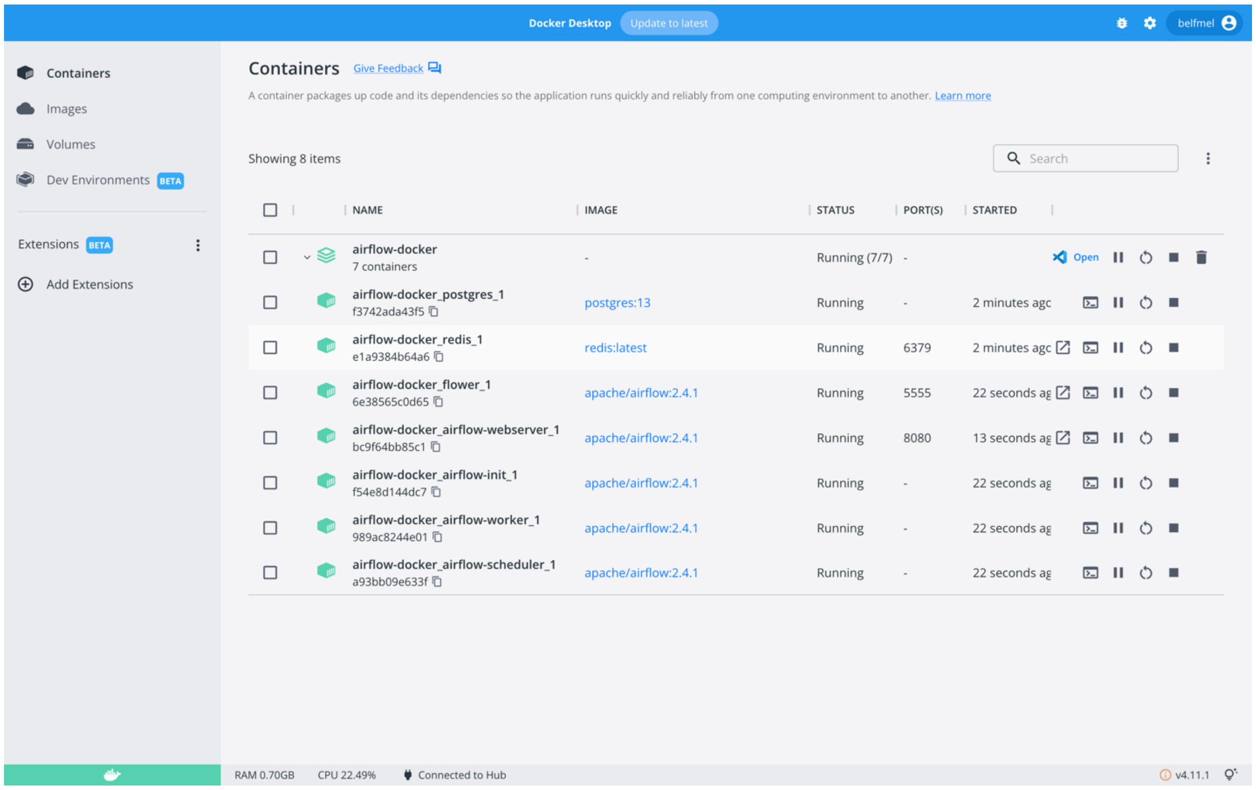

assignment.py

assignment.py > ...

```python


with DAG(
    "assignment",
    start_date=days_ago(1),
    schedule_interval="@daily",catchup=False,
) as dag:

# INSTALL BS4 BY HAND THEN CALL FUNCTION

    # ts are tasks
      t0 = BashOperator(
          task_id='task_zero',
          bash_command='pip install beautifulsoup4',
          retries=2
      )
      t1 = PythonOperator(
          task_id='task_one',
          depends_on_past=False,
          python_callable=catalog
      )
      t2 = PythonOperator(
          task_id='task_two',
          depends_on_past=False,
          python_callable=combine
      )
      t3 = PythonOperator(
          task_id='task_three',
          depends_on_past=False,
          python_callable=titles
      )
      t4 =PythonOperator (
          task_id='task_four',
          depends_on_past=False,
          python_callable=clean
      )
      t5 = PythonOperator(
          task_id='task_five',
          depends_on_past=False,
          python_callable=count_words
      )

      t0>>t1>>t2>>t3>>t4>>t5

```

Part 2: Code Execution

1. Provide a screenshot of your Docker application that shows that your Airflow Docker *container* has initiated.


Docker Desktop — Update to latest — belfmel

Containers — Give Feedback

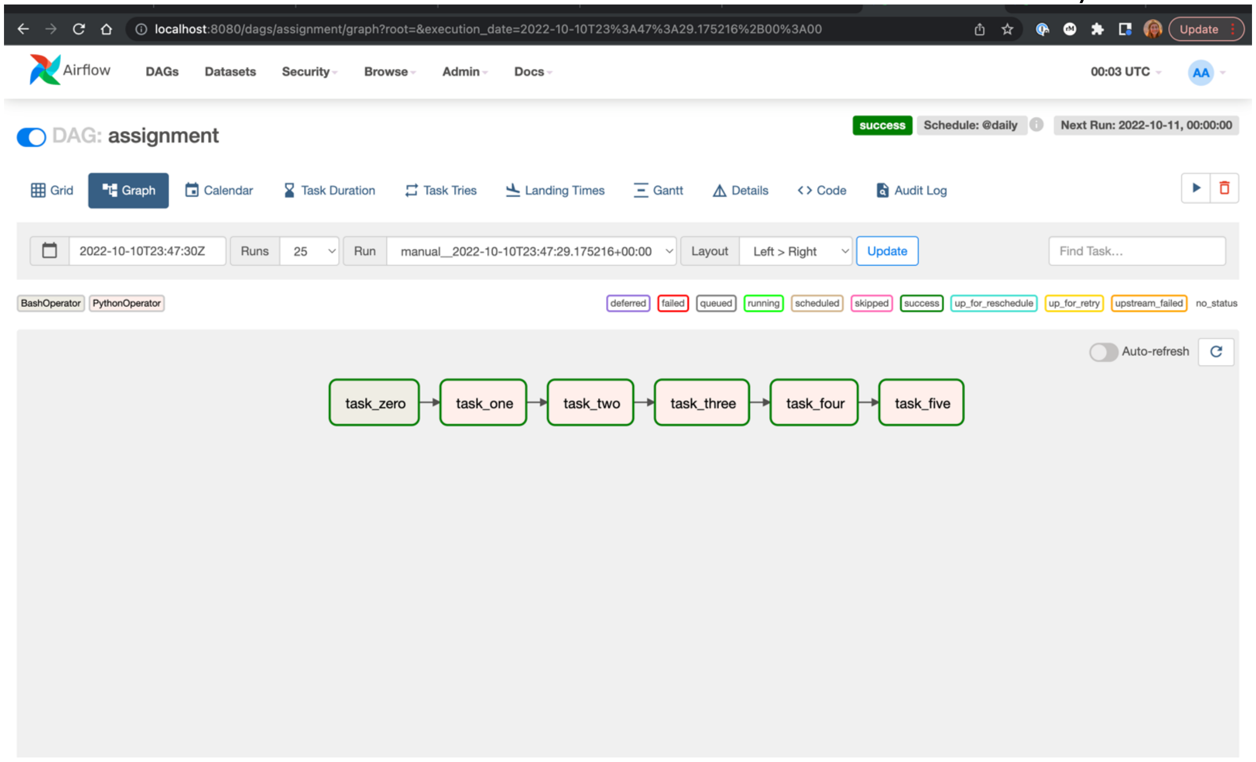
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Showing 8 items

NAME	IMAGE	STATUS	PORT(S)	STARTED
airflow-docker 7 containers	-	Running (7/7)	-	
airflow-docker_postgres_1 f3742ada43f5	postgres:13	Running	-	2 minutes agc
airflow-docker_redis_1 e1a9384b64a6	redis:latest	Running	6379	2 minutes agc
airflow-docker_flower_1 6e38565c0d65	apache/airflow:2.4.1	Running	5555	22 seconds ag
airflow-docker_airflow-webserver_1 bc9f64bb85c1	apache/airflow:2.4.1	Running	8080	13 seconds ag
airflow-docker_airflow-init_1 f54e8d144dc7	apache/airflow:2.4.1	Running	-	22 seconds ag
airflow-docker_airflow-worker_1 989ac8244e01	apache/airflow:2.4.1	Running	-	22 seconds ag
airflow-docker_airflow-scheduler_1 a93bb09e633f	apache/airflow:2.4.1	Running	-	22 seconds ag

RAM 0.70GB CPU 22.49% Connected to Hub v4.11.1

2. Provide a screenshot of the *task* boxes to show that the DAG ran successfully.


localhost:8080/dags/assignment/graph?root=&execution_date=2022-10-10T23%3A47%3A29.175216%2B00%3A00

Airflow — DAGs Datasets Security Browse Admin Docs — 00:03 UTC AA

DAG: assignment — success — Schedule: @daily — Next Run: 2022-10-11, 00:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

2022-10-10T23:47:30Z Runs 25 Run manual__2022-10-10T23:47:29.175216+00:00 Layout Left > Right Update Find Task...

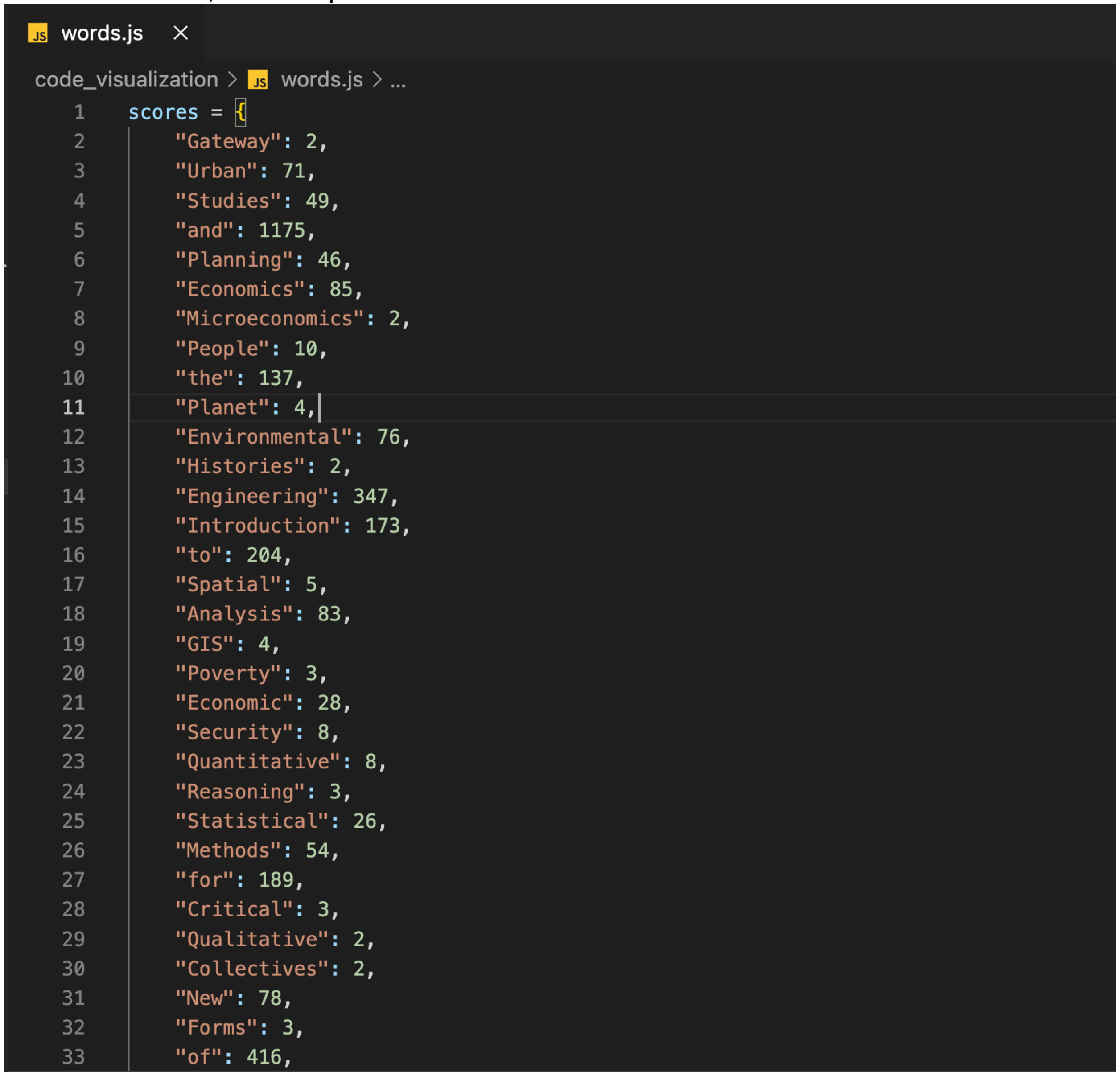
BashOperator PythonOperator

deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

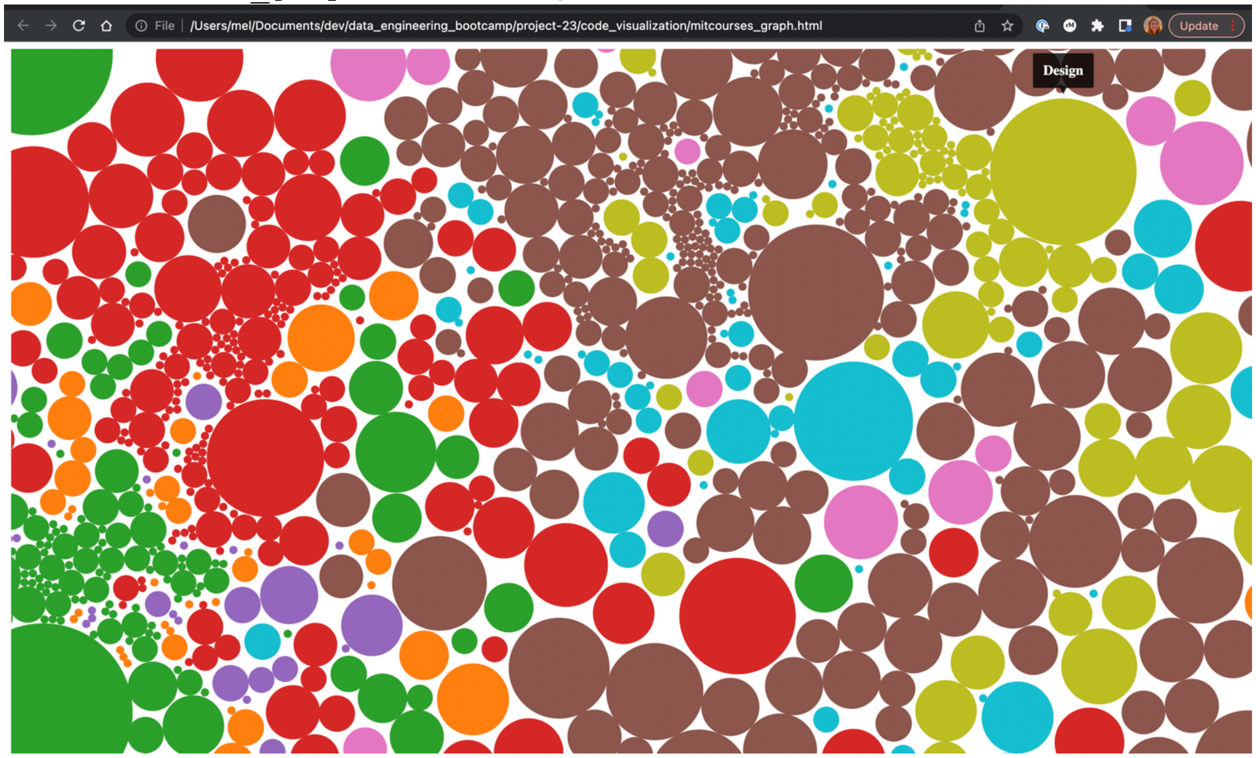
task_zero → task_one → task_two → task_three → task_four → task_five

3. Provide a screenshot of the `words.js` file with the data from the `words.json` file. You may not be able to fit all the data in one screenshot. If so, the first part of the file is sufficient.

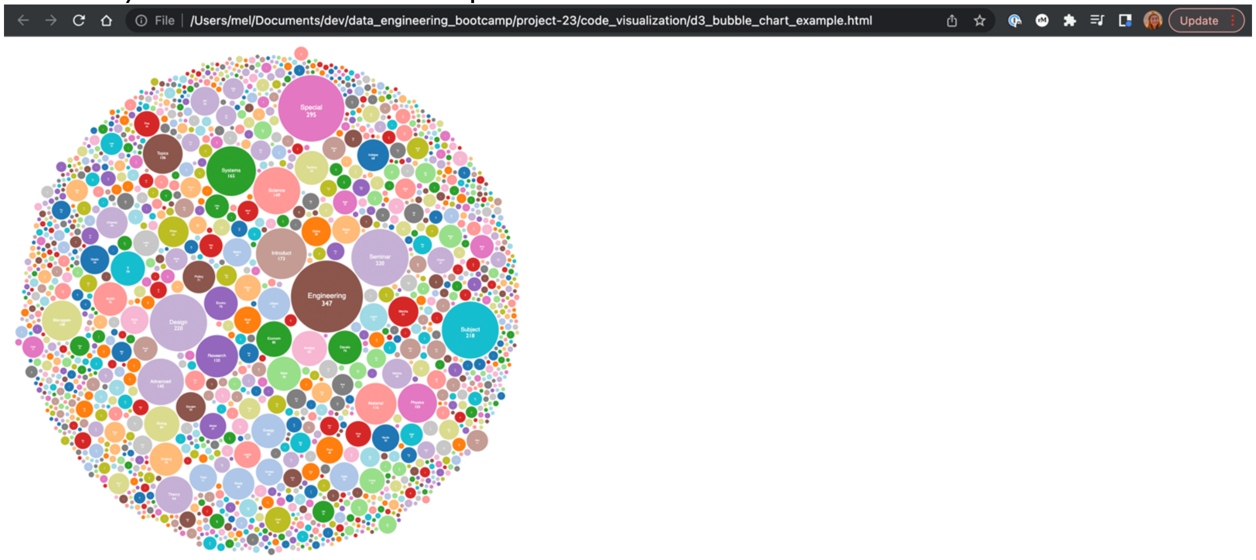


```
scores = {
    "Gateway": 2,
    "Urban": 71,
    "Studies": 49,
    "and": 1175,
    "Planning": 46,
    "Economics": 85,
    "Microeconomics": 2,
    "People": 10,
    "the": 137,
    "Planet": 4,
    "Environmental": 76,
    "Histories": 2,
    "Engineering": 347,
    "Introduction": 173,
    "to": 204,
    "Spatial": 5,
    "Analysis": 83,
    "GIS": 4,
    "Poverty": 3,
    "Economic": 28,
    "Security": 8,
    "Quantitative": 8,
    "Reasoning": 3,
    "Statistical": 26,
    "Methods": 54,
    "for": 189,
    "Critical": 3,
    "Qualitative": 2,
    "Collectives": 2,
    "New": 78,
    "Forms": 3,
    "of": 416,
```

4. Provide a screenshot of the visualization produced with the `mitcourses_graph.html` file in your web browser.


File
/Users/mel/Documents/dev/data_engineering_bootcamp/project-23/code_visualization/mitcourses_graph.html
Update
Design

5. Provide a screenshot of your enhanced visualization created with the D3 *library* and the modified example code.


File
/Users/mel/Documents/dev/data_engineering_bootcamp/project-23/code_visualization/d3_bubble_chart_example.html
Update
Special
295
Topics
Systems
165
Science
149
Introduct
173
Seminar
220
Engineering
347
Design
220
Subject
218
Research
120
Advanced
145
Material
Physics
Theory