module11/MWattanasureepoch-Module_11_Assignment.py

```
import os
import json
from airflow import DAG
from airflow.operators.python_operator import PythonOperator, BranchPythonOperator
from airflow.operators.dummy_operator import DummyOperator
from datetime import datetime, timedelta

# define the default arguments
default_args = {
    'owner': 'data_engineer',
    'start_date': datetime(2023, 4, 6),
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}

# define the DAG
with DAG('process_student_data', default_args=default_args, schedule_interval='@daily')
as dag:

    def load_data():
        with open(os.path.join(os.path.dirname(__file__), 'data/input.json')) as f:
            data = json.load(f)
        f.close()
        return data

    def process_data():
        data = load_data()
        students = data['students']
        courses = data['courses']

        #breakpoint()

        # Nothing required here for submission - this function is complete


    def check_weekday(**context):
        execution_date = context['ds']

        # QUESTION #1 (10 points)
        # Extract the day (number) of the week
        weekday = datetime.strptime(execution_date, '%Y-%m-%d').weekday()
        #breakpoint()

        if weekday < 5:  # weekday is 0-based, with 0=Monday and 4=Friday
            return 'store_data_weekday'
        else:
            return 'store_data_weekend'

    def store_data_weekday():
        data = load_data()
        students = data['students']
        courses = data['courses']

        # QUESTION #2 (10 points)
        # Loop through the students and find each course and description
        # Save this data to the 'data/weekday_data.txt' file as follows:
        # '<LASTNAME, FIRSTNAME> took <COURSE> (<COURSE_DESCRIPTION>) on a weekday'
        # Example: 'Mosko, Scott took ENG101 (Data Engineering) on a weekday'
        #
        # Each entry should be on a new line
        # This function is only run on weekdays (due to the check_weekday function)
        lines=[]
        for student in students:
            for key, val in student.items():
                if key == 'name':
                    first_name = val.split(' ')[0]
                    last_name = val.split(' ')[1]
                if key == 'courses':
                    for stu_course in val:
                        for course in courses:
                            if course['name'] == stu_course:
                                course_desc = course['description']
                                line = last_name + ', ' + first_name + ' took ' + \
                                    stu_course + ' (' + course_desc + ') on a weekday'
                                lines.append(line)


        with open(os.path.join(os.path.dirname(__file__), 'data/weekday_data.txt'),'w')
as f:
            f.write('\n'.join(lines))


    def store_data_weekend():
        data = load_data()
        students = data['students']
        courses = data['courses']


        # QUESTION #3 (10 points)
        # Loop through the students and find each course and description
        # Save this data to the 'data/weekend_data.txt' file as follows:
        # '<LASTNAME, FIRSTNAME> took <COURSE> (<COURSE_DESCRIPTION>) on a weekend'
        # Example: 'Mosko, Scott took ENG101 (Data Engineering) on a weekend'
        #
        # Each entry should be on a new line
        # This function is only run on weekends (due to the check_weekdend function)
        lines=[]
        for student in students:
            for key, val in student.items():
                if key == 'name':
                    first_name = val.split(' ')[0]
                    last_name = val.split(' ')[1]
                if key == 'courses':
                    for stu_course in val:
                        for course in courses:
                            if course['name'] == stu_course:
                                course_desc = course['description']
                                line = last_name + ', ' + first_name + ' took ' + \
                                    stu_course + ' (' + course_desc + ') on a weekend'
                                lines.append(line)


        with open(os.path.join(os.path.dirname(__file__), 'data/weekend_data.txt'),'w')
as f:
            f.write('\n'.join(lines))

    # QUESTION #4
    load_data_task = PythonOperator(
            task_id="load_data",
            python_callable=load_data
        ) #(1 point)
    process_data_task = PythonOperator(
            task_id="process_data",
            python_callable=process_data
        ) #(1 point)
    check_weekday_task = BranchPythonOperator(
        task_id="check_weekday",
        provide_context=True,
        python_callable=check_weekday
        ) #(1 point)

    store_data_weekday_task =PythonOperator(
            task_id="store_data_weekday",
            python_callable=store_data_weekday
        ) #(1 point)
    store_data_weekend_task = PythonOperator(
            task_id="store_data_weekend",
            python_callable=store_data_weekend
        ) #(1 point)
    end_task = DummyOperator(
            task_id="end",
            trigger_rule='none_failed_or_skipped'
        ) #(4 points)
        # Note that end_task will be tricky. There is a way to make it complete even
though
        # only one of the branches finishes successfully and the other is skipped.
Normally
```

```
        # this would cause anything following the branch to be skipped.

    # QUESTION #5 (10 points)
    # Create the flow for the DAG to match the provided diagram.
    load_data_task >> process_data_task >> check_weekday_task >>
[store_data_weekday_task, store_data_weekend_task] >> end_task
```