```python
"""
Code that goes along with the Airflow tutorial located at:
https://github.com/apache/incubator-airflow/blob/master/airflow/example_dags/tutorial.py
"""
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta

#
default_args = {
    'owner': 'airflow', # O dono da Dag.
    'depends_on_past': False, # Impede que uma task seja rodada se a anterior a ela falhou.
    'start_date': datetime(2015, 6, 1), # Define o dia em que a Task vai começar a rodar.
    'email': ['airflow@example.com'], # Define o email que irá receber mensagens da situação da DAG.
    'email_on_failure': False, # True: Se houver uma falha, ele mandará um email. False: Não manda email em caso de falha.
    'email_on_retry': False, # True: Se houver uma retry, ele mandará um email. False: Não manda email em caso de retry.
    'retries': 1, # O número de tentaivas que devem ser feitas antes de falhar a task.
    'retry_delay': timedelta(minutes=5), # De quanto em quanto tempo ele vai tentar de novo.
    # 'queue': 'bash_queue', Não sei o que é isso.
    # 'pool': 'backfill', Limita a execução paralela em DAGS.
    # 'priority_weight': 10, Define a prioridade na fila de execução
    # 'end_date': datetime(2016, 1, 1), O dia em que essa DAG vai parar de rodar.
}

dag = DAG('tutorial', default_args=default_args, # O defualt args é colocada aqui na DAG.
          schedule_interval=timedelta(days=1)) # De quando em quanto tempo ela vai rodar.

# t1, t2 and t3 are examples of tasks created by instantiating operators
t1 = BashOperator( # O BashOperator é utilizado para excutar comandos no Bash shell
    task_id='print_date', # Esse aqui é o nome da task.
    bash_command='date', # define o comando que será realizado.
    dag=dag) # define sua dag

t2 = BashOperator(
    task_id='sleep',
    bash_command='sleep 5',
    retries=3, # define quantas tentivas
    dag=dag)
```

```
# Jinja é um motor de templates que nos permite escrever um código parecido
# com a sintaxe do Python. Depois o template recebe os dados que serão
# renderizados no documento final.
templated_command = """
    {% for i in range(5) %}
        echo "{{ ds }}"
        echo "{{ macros.ds_add(ds, 7)}}"
        echo "{{ params.my_param }}"
    {% endfor %}
"""

t3 = BashOperator(
    task_id='templated',
    bash_command=templated_command,
    params={'my_param': 'Parameter I passed in'}, #Permite passar um dicionário de parâmetros ou/e aos templates.
    dag=dag)

t2.set_upstream(t1) # É o mesmo que:
t3.set_upstream(t1) # t2 >> [t2, t3]
```