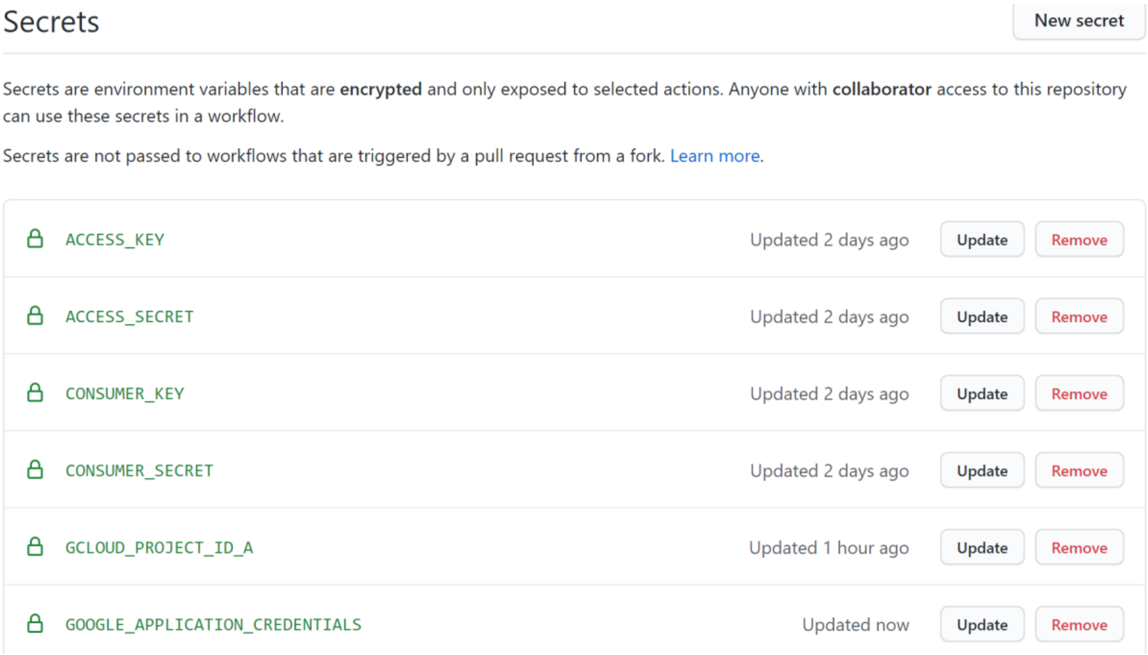
EC601 Project 4
Jiaming Yu

Phase 1: Unit Test for project 2: https://github.com/JimY233/EC601_project2
Learn how to use Actions
Learn how to use Unit Tests

(1)Use Github Secrets to separate Twitter API keys from the code:
Reference: https://github.com/h4x0rMadness/ec601-github-secret-tutorial
Go to Secret under Settings, add access_key, access_secret, consumer_key, consumer_secret as needed by Twitter API.



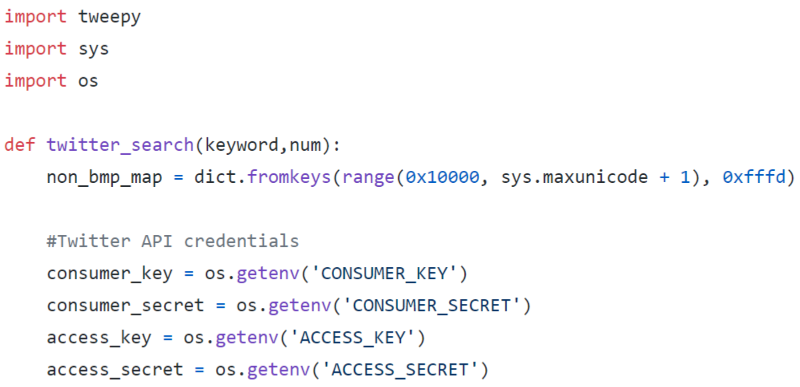
Secrets are environment variables that are **encrypted** and only exposed to selected actions. Anyone with **collaborator** access to this repository can use these secrets in a workflow.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

ACCESS_KEY	Updated 2 days ago	Update	Remove
ACCESS_SECRET	Updated 2 days ago	Update	Remove
CONSUMER_KEY	Updated 2 days ago	Update	Remove
CONSUMER_SECRET	Updated 2 days ago	Update	Remove
GCLOUD_PROJECT_ID_A	Updated 1 hour ago	Update	Remove
GOOGLE_APPLICATION_CREDENTIALS	Updated now	Update	Remove

Here we also added the base64 encoded Google Cloud Language json file. Reference: https://github.com/actions-hub/gcloud

(2)Use Twitter API keys in **TwitterAPI.py**
Change Twitter API file to use environment variable access_key, access_secret, consumer_key, consumer_secret saved in Github Secrets as authorization. Using os.getenv(). Remember to "import os".



```
import tweepy
import sys
import os

def twitter_search(keyword,num):
    non_bmp_map = dict.fromkeys(range(0x10000, sys.maxunicode + 1), 0xfffd)

    #Twitter API credentials
    consumer_key = os.getenv('CONSUMER_KEY')
    consumer_secret = os.getenv('CONSUMER_SECRET')
    access_key = os.getenv('ACCESS_KEY')
    access_secret = os.getenv('ACCESS_SECRET')
```

(3)Use Google Cloud Language API in **NLPAPI.py**

```
import os
os.getenv('GOOGLE_APPLICATION_CREDENTIALS')
```

Met problem so far, described latter

(3)Use pytest to do the unit test

First start with one simple example like:

test_case.py

```
import pytest
from search import search
def test_num1(): #test num of search
    keyword = 'COVID-19'
    num = 5
    res = search(keyword,num)
    assert len(res) == 5

if __name__ == '__main__':
    pytest.main()
```

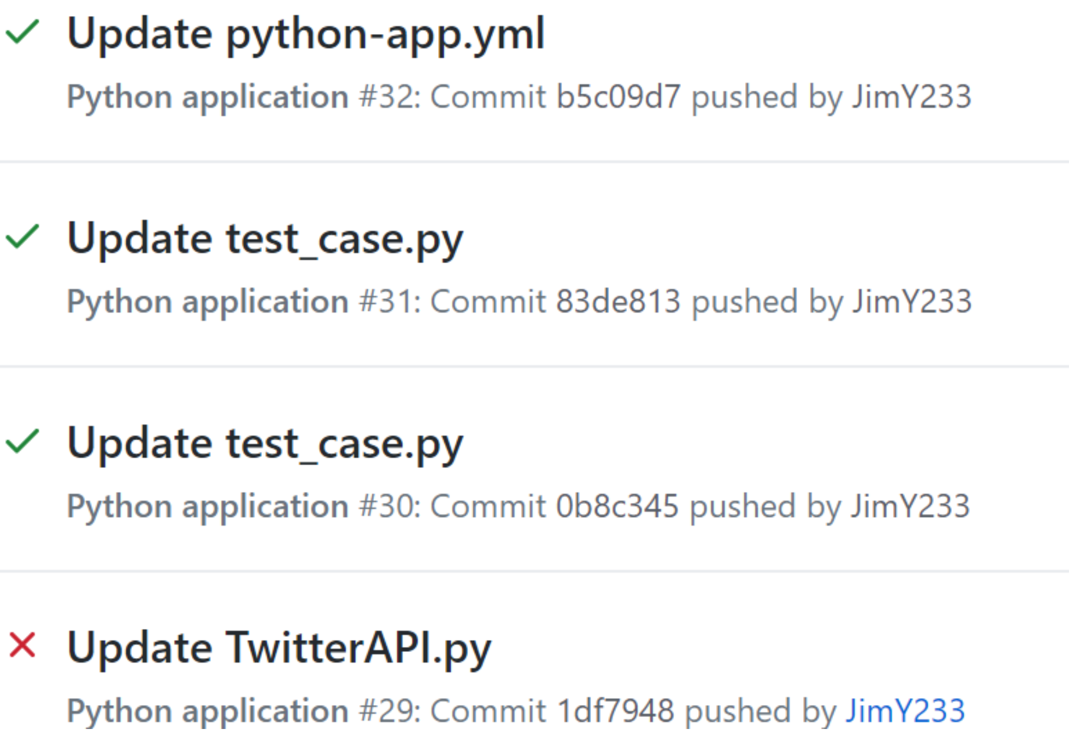
Here we just have one unit test function. But we can combine with Github Actions to find the errors in all the code. And then adding the unit test functions, we can test our code.

(4)Use Github Actions

First, new a workflow, in this project, we can use Python application and make some changes to .yml file

```
 - name: Install dependencies
   run: |
     python -m pip install --upgrade pip
     pip install flake8 pytest
     if [ -f requirements.txt ]; then pip install -r requirements.txt; fi
 - name: Lint with flake8
   run: |
     # stop the build if there are Python syntax errors or undefined names
     flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics
     # exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide
     flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics
 - name: Test with pytest
   env:
       CONSUMER_KEY: ${{ secrets.CONSUMER_KEY }}
       CONSUMER_SECRET:  ${{ secrets.CONSUMER_SECRET }}
       ACCESS_KEY: ${{ secrets.ACCESS_KEY }}
       ACCESS_SECRET: ${{ secrets.ACCESS_SECRET }}
   run: |
     pytest
```

Added the part: "env:" to add access_key, access_secret, consumer_key, consumer_secret as environment variables.
Thus we can use Twitter API and thus use pytest to do unit test.
Run the workflow. Sometimes we may get error. And we can use the error to make our code better.



✓ Update python-app.yml

Python application #32: Commit b5c09d7 pushed by JimY233

✓ Update test_case.py

Python application #31: Commit 83de813 pushed by JimY233

✓ Update test_case.py

Python application #30: Commit 0b8c345 pushed by JimY233

✗ Update TwitterAPI.py

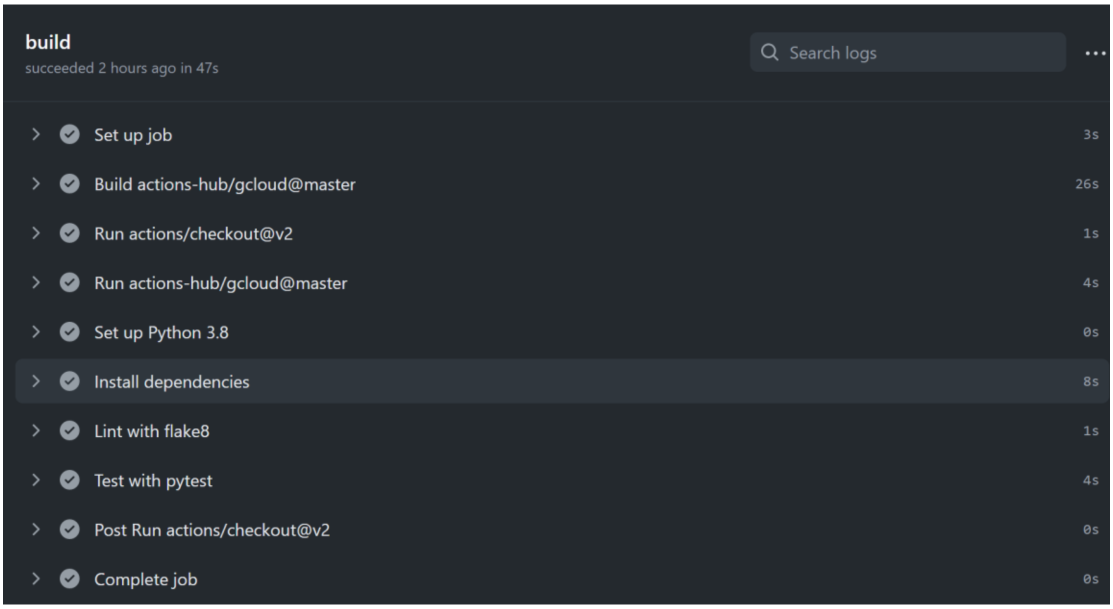
Python application #29: Commit 1df7948 pushed by JimY233

Final **test_case.py**

```
import pytest
import tweepy
from search import search
from TwitterAPI import twitter_search, twitter_timeline
#from NLPAPI import NLP_analyze

def test_num1(): #test num of search
    #consumer_key = os.getenv('CONSUMER_KEY')
    #consumer_secret = os.getenv('CONSUMER_SECRET')
    #access_key = os.getenv('ACCESS_KEY')
    #access_secret = os.getenv('ACCESS_SECRET')
    keyword = 'COVID-19'
    num = 5
    res = search(keyword,num)
    assert len(res) == 5

def test_num2(): #test num of search using different keyword and different num
    keyword = 'realDonaldTrump'
    num = 100
    res = search(keyword,num)
    assert len(res) == 100

def test_num3(): #test num of timeline
    userid = 'LeoDiCaprio'
    num = 10
    res = twitter_timeline(userid,num)
    assert len(res) == 10

def test_timeline():
    userid = 'LeoDiCaprio'
    num = 10
    res = twitter_timeline(userid,num)
    assert type(res[0]) is tweepy.models.Status

def test_type():
    keyword = 'COVID-19'
    num = 10
    res = twitter_search(keyword,num)
    assert type(res) is str

def test_type():
    keyword = 'COVID-19'
    num = 10
    res = twitter_search(keyword,num)
    assert 'COVID-19' in res

#def test_NLP1():
#       text = 'Hello world!'
#       sentiment = NLP_analyze(text)
#       assert hasattr(sentiment,'score') == True

#def test_NLP2():
#       text = 'Hello world!'
#       sentiment = NLP_analyze(text)
#       assert hasattr(sentiment,'magnitude') == True

if __name__ == '__main__':
    pytest.main()
```

However, since I have not solve the problem of using Google Cloud Language API with the json file using Github Secrets. The unit test functions for it are # like test_NLP()

Final Github Actions



build
succeeded 2 hours ago in 47s

```
Set up job                              3s
Build actions-hub/gcloud@master        26s
Run actions/checkout@v2                 1s
Run actions-hub/gcloud@master           4s
Set up Python 3.8                       0s
Install dependencies                    8s
Lint with flake8                        1s
Test with pytest                        4s
Post Run actions/checkout@v2            0s
Complete job                            0s
```

All successful

Issues with using Github Secret to separate Google Cloud Language API

Used

```
- uses: actions-hub/gcloud@master
  env:
    PROJECT_ID: ${{secrets.GCLOUD_PROJECT_ID_A}}
    APPLICATION_CREDENTIALS: ${{ secrets.GOOGLE_APPLICATION_CREDENTIALS }}
  with:
    args: info
```

According to https://github.com/actions-hub/gcloud

But still can not import google.cloud.language

Try pip install google-cloud-language, seems not working.