Lab 3.5.7 Membuat Unit Test di Phyton

Tujuan :

Part 1 : Jalankan DEVASC VM

Part 2 : Eksplore di framework unitetest

Part 3 : Test function di phyton dengan unit test

Latar Belakang

Unit Testing :

Unit Testing adalah Pengujian fungsional yang terperinci pada dari potongan-potongan kecil kode (garis, blok, fungsi, kelas, dan komponen lain secara terpisah).

Framework pengujian ini adalah perangkat lunak yang memungkinkan Anda membuat pernyataan tentang kondisi yang dapat diuji dan menentukan apakah pernyataan ini valid pada suatu titik dalam eksekusi.

Contoh framework pengujian di phyton :

- phyTest
    - otomatis mengeksekusi skrip apa pun yang dimulai dengan “test_ “ atau diakhiri dengan “_test.py” dan di dalam skrip itu, secara otomatis menjalankan fungsi apa pun yang dimulai dengan 'test_' atau 'tests_'.
    - Kita dapat menguji unit sepotong kode dengan menyalinnya ke dalam sebuah file, mengimpor pytest, menambahkan fungsi pengujian dengan nama yang tepat, menyimpan file di bawah nama file yang juga dimulai dengan 'tests_,' dan menjalankannya dengan PyTest.

- unittest
  Kerangka kerja yang paling kecil menuntut sintaks yang berbeda dari PyTest.
  Untuk unittest, Anda perlu membuat subkelas dari kelas TestCase bawaan dan menguji dengan mengganti metode bawaannya atau menambahkan metode baru yang namanya dimulai dengan 'test_'.

Resource yang dibutuhkan :

- 1 PC/Laptop
- Virtual Box / VMWare
- DEVASC Virtual Machine

Langkah-langkah :

Part 1 : Jalankan DEVASC VM

Part 2 : Eksplore framework unittest

Phyton menyediakan framework untuk mengujian namanya unittest dengan menggunakan library standart. Kalian bisa mengeksplore framework ini di google/youtube dengan menuliskan referensi kata kunci : "Phyton unittetst Framework" karena DIbawah ini akan ada beberapa pertanyaan yang harus kalian jawab :

Class unittest apa yang Anda gunakan untuk membuat unit pengujian individual?

Modul unittest menyediakan class dasar, TestCase, yang dapat digunakan untuk membuat kasus pengujian baru.

Test runner bertanggung jawab untuk menjalankan pengujian dan memberi Anda hasil. Runner test bisa menjadi antarmuka grafis tetapi, di lab ini, Anda akan menggunakan baris perintah untuk menjalankan pengujian.

Bagaimana runner pengujian mengetahui metode mana yang merupakan pengujian?

Metode yang namanya dimulai dengan "test_" memberi tahu runner pengujian tentang method mana yang di test.

Perintah yang dapat digunakan di unittest dapat di tampilkan dengan perintah **python –m unitetst –h**

```
devasc@labvm:~/labs/devnet-src$ python3 -m unittest -h

<output omitted>

optional arguments:

 -h, --help show this help message and exit

 -v, --verbose Verbose output

 -q, --quiet Quiet output

 --locals Show local variables in tracebacks

 -f, --failfast Stop on first fail or error

 -c, --catch Catch Ctrl-C and display results so far

 -b, --buffer Buffer stdout and stderr during tests

 -k TESTNAMEPATTERNS Only run tests which match the given substring

Examples:

 python3 -m unittest test_module - run tests from test_module
```

```
 python3 -m unittest module.TestClass - run tests from module.TestClass

 python3 -m unittest module.Class.test_method - run specified test method

 python3 -m unittest path/to/test_file.py - run tests from test_file.py

<output omitted>

For test discovery all test modules must be importable from the top level

directory of the project.

devasc@labvm:~/labs/devnet-src$
```

Part 3 : pengujian function di python dengan unittest

Di bagian ini, Anda akan menggunakan unittest untuk menguji fungsi yang melakukan pencarian rekursif dari objek JSON. fungsi tersebut akan mengembalikan nilai berupa key tertentu.

File yang akan anda butuhkan adalah :

1. recursive_json_search.py : Skrip ini akan menyertakan fungsi json_search () yang ingin kita uji.
2. test_data.py : Ini adalah data yang dicari oleh fungsi json_search ()
3. test_json_search.py : Ini adalah file yang akan Anda buat untuk menguji fungsi json_search () di recursive_json_search.py

Langkah 1 : Review file test_data.py

- buka file : test_data.py di derektory : **~/labs/devnet-src/unittest/**
  Data JSON ini tipikal data yang dikembalikan melalui panggilan keCisco DNA Center API. Data sampel cukup kompleks untuk dijadikan pengujian yang baik. Misalnya, ia memiliki jenis dictionary dan tipe list yang berselang-seling.

```
devasc@labvm:~/labs/devnet-src$ more unittest/test_data.py
key1 = "issueSummary"
key2 = "XY&^$#*@!1234%^&"
data = {
 "id": "AWcvsjx864kVeDHDi2gB",
 "instanceId": "E-NETWORK-EVENT-AWcvsjx864kVeDHDi2gB-1542693469197",
 "category": "Warn",
 "status": "NEW",
 "timestamp": 1542693469197,
 "severity": "P1",
 "domain": "Availability",
 "source": "DNAC",
 "priority": "P1",
 "type": "Network",
 "title": "Device unreachable",
 "description": "This network device leaf2.abc.inc is unreachable from controll
er. The device role is ACCESS.",
 "actualServiceId": "10.10.20.82",
 "assignedTo": "",
 "enrichmentInfo": {
 "issueDetails": {
```

```
        "issue": [
        {
       --More--(12%)
```

Langkah 2 : Buat function json_search() yang nantinya akan dilakukan pengujian

Fungsi ini akan membutuhkan parameter input berupa key dan object JSON dan akan mengembalikan list key yang cocok dengan pasangannya.

Tujuan dari fungsi ini adalah untuk mengimpor data pengujian terlebih dahulu. Kemudian mencari data yang cocok variabel kunci dalam file test_data.py. Jika menemukan kecocokan, itu akan menambahkan data yang cocok ke daftar. Itu print () fungsi di akhir mencetak konten untuk daftar untuk variabel pertama key1 = "issueSummary".

Agar lebih mudah anda bisa membuat function ini dengan bantuan editor VScode

```
from test_data import *
def json_search(key,input_object):
    ret_val=[]
    if isinstance(input_object, dict): # Iterate dictionary
        for k, v in input_object.items(): #searching key in dict
            if k == key:
                temp={k:v}
                ret_val.append(temp)
            if isinstance(v, dict): # the value is another dict so repeat
                json_search(key,v)
            elif isinstance(v, list): # it's a list
                for item in v:
                    if not isinstance(item, (str,int)): # if dict or list
repeat
                        json_search(key,item)
    else: # Iterate a list because some APIs return JSON object in a list
        for val in input_object:
            if not isinstance(val, (str,int)):
                json_search(key,val)
    return ret_val
print(json_search("issueSummary",data))
```

a. open file **recursive_json_search.py** yang ada di **~/labs/devnet-src/unittest/**
b. tuliskan script diatas dan simpan
c. Kemudian running dengan menjalankan perintah ini :
   devasc@labvm:~/labs/devnet-src/unittest$ **python3 recursive_json_search.py**

```
[]
devasc@labvm:~/labs/devnet-src/unittest$
```

Note : jika tidak ada error, maka function akan menampilkan output [], artinya list nya kosong.

d. Open **test_data.py**, kemudian cari dengan keyword : “issueSummary”,

```
<output omitted>
      "issue": [
        {
          "issueId": "AWcvsjx864kVeDHDi2gB",
          "issueSource": "Cisco DNA",
          "issueCategory": "Availability",
          "issueName": "snmp_device_down",
          "issueDescription": "This network device leaf2.abc.inc is unreachable from
controller. The device role is ACCESS.",
          "issueEntity": "network_device",
          "issueEntityValue": "10.10.20.82",
          "issueSeverity": "HIGH",
          "issuePriority": "",

          "issueSummary": "Network Device 10.10.20.82 Is Unreachable From Controller",
          "issueTimestamp": 1542693469197,
          "suggestedActions": [
            {
<output omitted>
```

Langkah 3 : buat beberapa unit test

a. Open ~ **labs/devnet-src/unittest/test_json_search.py file.**

b. Pada line pertama import library, dengan menuliskan :

```
import unittest
```

c. Tambahkan baris dibawahnya dan ketikkan import function yang akan di test

```
from recursive_json_search import *
from test_data import *
```

d. Lalu, tambahkan code untuk mmbuat class json_search_test ke dalam code test_json_search.py.

```
class json_search_test(unittest.TestCase):
    '''test module to test search fuction in `recursive_json_search.py`'''
    def test_search_found(self):
        '''key should be found, return list should not be empty'''
        self.assertTrue([]!=json_search(key1,data))
    def test_search_not_found(self)
        '''key should not be found, should return an empty list'''
        self.assertTrue([]==json_search(key2,data))
    def test_is_a_list(self):
        '''Should return a list'''
        self.assertIsInstance(json_search(key1,data),list)
```

Pada video, tolong jelaskan ketiga method yang digunakan untuk melakukan pengujian pada function search tersebut.

e. Tambahkan method unittest.main()

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

Langkah 4 : jalankan dan lihat hasilnya

a. Jalankan **test_json_search.py**

```
[]
.F.
======================================================================
FAIL: test_search_found (__main__.json_search_test)
key should be found, return list should not be empty
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_json_search.py", line 9, in test_search_found
    self.assertTrue([]!=json_search(key1,data))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
devasc@labvm:~/labs/devnet-src/unittest$
di video Jelaskan hasil dari runningnya.
```

setelah di run, outputnya ada :

- [] : list kosong
- .F. : (.)artinya lolos test, dan F menunjukkan Test failed.
- Method Pertama test passed/lulus , the second test failed, and the third test passed.

b. Jalankan unittest dengan perintah **verbose (-v).** , lalu perhatikan hasilnya.

Untuk membuat daftar setiap pengujian dan hasilnya, jalankan skrip lagi di bawah unittest dengan opsi verbose (-v). Anda tidak memerlukan ekstensi .py untuk skrip test_json_search.py. Anda dapat melihat bahwa ujian Anda metode test_search_found gagal/failed

```
devasc@labvm:~/labs/devnet-src/unittest$ python3 -m unittest -v
test_json_search
[]
test_is_a_list (test_json_search.json_search_test)
Should return a list ... ok
```

```
test_search_found (test_json_search.json_search_test)
key should be found, return list should not be empty ... FAIL
test_search_not_found (test_json_search.json_search_test)
key should not be found, should return an empty list ... ok

======================================================================
FAIL: test_search_found (test_json_search.json_search_test)
key should be found, return list should not be empty
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/devasc/labs/devnet-
src/unittest/test_json_search.py", line 9, in test_search_found
    self.assertTrue([]!=json_search(key1,data))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
devasc@labvm:~/labs/devnet-src/unittest$
```

Langkah 5 : Selidiki dan perbaiki kesalahan pertama dalam skrip recursive_json_search.py.

a. Perbaiki skrip dengan memindahkan ret_val=[] menjadi skrip berikut ini :

```
ret_val=[]
def json_search(key,input_object):
```

b. Simpan dan jalankan :

```
devasc@labvm:~/labs/devnet-src/unittest$ python3
recursive_json_search.py
[{'issueSummary': 'Network Device 10.10.20.82 Is Unreachable From
Controller'}]
devasc@labvm:~/labs/devnet-src/unittest$
```

Langkah 6 : Run the test kembali dan lihat jika masih ada yang error.

a. Jalankan perintah **python3 -m unittest test_json_search**

```
devasc@labvm:~/labs/devnet-src/unittest$ python3 -m unittest test_json_search
[{'issueSummary': 'Network Device 10.10.20.82 Is Unreachable From Controller'}]
..F
======================================================================
FAIL: test_search_not_found (test_json_search.json_search_test)
key should not be found, should return an empty list
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/devasc/labs/devnet-src/unittest/test_json_search.py", line 12, in
test_search_not_found
    self.assertTrue([]==json_search(key2,data))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (failures=1)
devasc@labvm:~/labs/devnet-src/unittest$
```

b. Buka file **test_data.py** dan cari issueSummary, yang merupakan nilai untuk key1. Anda harus menemukannya dua kali, tetapi hanya sekali dalam objek data JSON.

Tetapi jika Anda mencari nilai key2, yaitu XY&^$#*@!1234%^&, Anda hanya akan menemukannya di bagian atas yang ditentukan karena tidak ada dalam data JSON

obyek. Tes ketiga adalah memeriksa untuk memastikan tidak ada. Komentar tes ketiga menyatakan kunci harus tidak ditemukan, harus mengembalikan daftar kosong. Namun, fungsi tersebut mengembalikan daftar yang tidak kosong.

Langkah 7 : Selidiki dan perbaiki kesalahan kedua dalam skrip recursive_json_search.py

a. Tinjau kembali kode **recursive_json_search.py**. Perhatikan bahwa ret_val sekarang menjadi variabel global setelah anda memperbaikinya di langkah sebelumnya. Ini berarti nilainya dipertahankan di beberapa pemanggilan fungsi json_search fungsi.

b. Untuk mengatasi masalah ini, bungkus/wrap fungsi json_search () dengan fungsi luar. Lalu Hapus fungsi json_search () yang sudah ada dan ganti dengan fungsi refactored di bawah ini :

```
from test_data import *
def json_search(key,input_object):
    ret_val=[]
    def inner_function(key,input_object):
        if isinstance(input_object, dict): # Iterate dictionary
```

```
            for k, v in input_object.items(): #searching key in dict
                if k == key:
                    temp={k:v}
                    ret_val.append(temp)
                if isinstance(v, dict): # the value is another dict so repeat
                    inner_function(key,v)
                elif isinstance(v, list): # it's a list
                    for item in v:
                        if not isinstance(item, (str,int)): # if dict or list repeat
                            inner_function(key,item)
        else: # Iterate a list because some APIs return JSON object in a list
            for val in input_object:
                if not isinstance(val, (str,int)):
                    inner_function(key,val)
    inner_function(key,input_object)
    return ret_val
print(json_search("issueSummary",data))
```

c. Simpan dan run.

```
devasc@labvm:~/labs/devnet-src/unittest$ python3 -m unittest
[{'issueSummary': 'Network Device 10.10.20.82 Is Unreachable From Controller'}]
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
devasc@labvm:~/labs/devnet-src/unittest$
```