Laboratorio 3a

Crear una prueba unitaria de Python

Sergio Sebastian Pezo Jimenez - 20224087G

Parte 1: Inicializamos la VM de DEVASC.

Parte 2: Exploremos las opciones en el unittest

```
File  Edit  View  Search  Terminal  Help
devasc@labvm:~/labs/devnet-src$
devasc@labvm:~/labs/devnet-src$ python3 -m unittest -h
usage: python3 -m unittest [-h] [-v] [-q] [--locals] [-f] [-c] [-b] [-k TESTNAMEPATTERNS] [tests [tests ...]]

positional arguments:
  tests                a list of any number of test modules, classes and test methods.

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
```

Parte 3: Realizamos nuestro primer test.

Revisemos la dara con l aque vamos a trabajar

```
devasc@labvm:~/labs/devnet-src/unittest$ more 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",
```

Copiamos el código y lo ejecutamos

recursive_json_search.py > json_search

```python
# Fill the Python code in this file
from test_data import *
ret_val=[]
def json_search(key,input_object):
    if isinstance(input_object, dict): # Iterate dictionary
        for k, v in input_object.items(): # searching key in the 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 re
                        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))
```

```
devasc@labvm:~/labs/devnet-src/unittest$ python3 recursive_json_search.py
[]
```

Creamos un par de pruebas

test_json_search.py > ...

```python
# Fill the Python code in this file
import unittest
from recursive_json_search import *
from test_data import *

class json_search_test(unittest.TestCase):
    '''test module to test search function 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)

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

Las ejecutamos y vemos como falla nuestro test que verifica que se encuentre algo

```
devasc@labvm: ~/labs/devnet-src/unittest
File  Edit  View  Search  Terminal  Help

devasc@labvm:~/labs/devnet-src/unittest$ python3 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 10, in test_search_found
    self.assertTrue([]!=json_search(key1,data))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 3 tests in 0.000s

FAILED (failures=1)
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
----------------------------------------------------------------------
```

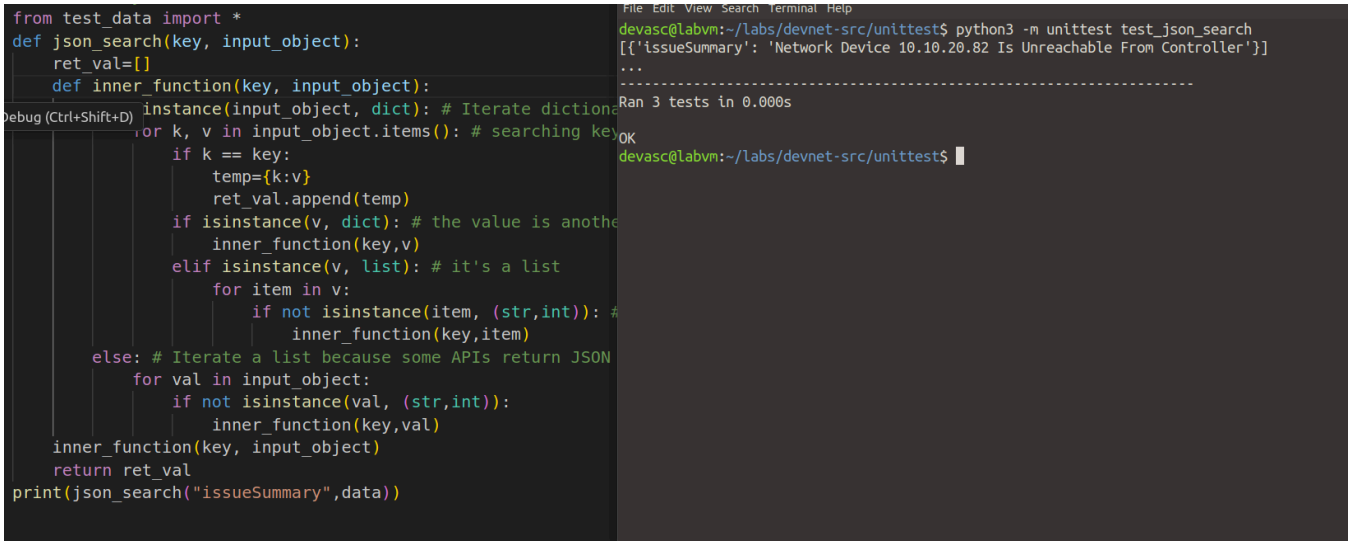
Así que procedemos a solucionar el problema, y ahora sí, se imprime lo esperado

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

Volvemos a correr nuestras pruebas y notamos que ahora la tercera prueba está fallando, la tercera, pues se está buscando con la key2=XY&^ $#*@!1234%^&, la cual no existe, pero como ret_val[] es un variable global, en el primer test dejó de se runa lista vacía, y por tal el test falla.

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

Así que agregamos una función interna, para que ret_vals[] se incialice en cada ejecución de las pruebas, finalizando correctamente este apartado.


from test_data import *
def json_search(key, input_object):
ret_val=[]
def inner_function(key, input_object):
Debug (Ctrl+Shift+D)
instance(input_object, dict): # Iterate dictiona
for k, v in input_object.items(): # searching key
if k == key:
temp={k:v}
ret_val.append(temp)
if isinstance(v, dict): # the value is anothe
inner_function(key,v)
elif isinstance(v, list): # it's a list
for item in v:
if not isinstance(item, (str,int)): #
inner_function(key,item)
else: # Iterate a list because some APIs return JSON
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))
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/unittest$ python3 -m unittest test_json_search
[{'issueSummary': 'Network Device 10.10.20.82 Is Unreachable From Controller'}]
...
Ran 3 tests in 0.000s
OK
devasc@labvm:~/labs/devnet-src/unittest$