Python test case generator with OAuth

* Objective:
  + To generate the standard test cases which has a lot of redundant or works and duplication of test case code and data, i needed a test case generator to save my time of implementing redundant codes and unit test cases.
* Staff:
  + Lead Programmer: Samil Chai
  + Junior Programmer: Nick Jang
    - Email: [nickjang114@gmail.com](mailto:nickjang114@gmail.com)
* Start date: 2017/10/02
* End date: 2017/10/03
* Reference:
  + [WY\_PYTHON\_ACCESS\_GOOGLE\_SPREADSHEET\_WITH\_OAUTH](https://docs.google.com/document/d/1tjjrUvoaPH0H2Ec1McGZulHH27z_gWw_h31cDG-orD8/edit)
  + [WY\_PYTHON\_ARGPARSE](https://docs.google.com/document/d/1RgNrHR_bV8a0ORTQa6f_nbQN_Ka4-jomJwtIbNsqTPM/edit#heading=h.8hui0ssfb7qj)
* OAuth test case generator project directory:
  + Client\_secret.json:
    - OAuth client secret file for authentication.
  + Diagrams:
    - Collection of important algorithm diagrams.
  + Documents:
    - Collection of important algorithm setup guide, framework and documents.
  + Makefile
    - Main makefile to compile and run the OAuth test case generator.
      * All the spreadsheet URL will be indicated here to manage them in one place.
  + Oauth\_test\_case\_generator\_executor.py
    - Sub Python script that executes the unit test case file generated from the generator, and also generates the HTML page for unit test case Pydoc inner document.
  + Oauth\_test\_case\_generator\_generator.py
    - Sub Python script that processes the Google spreadsheet data and generates the unit test cases.
  + Oauth\_test\_case\_generator\_get\_spreadsheet\_data.py
    - Sub Python script that authenticate with Google Developer API to retrieve the data from a Google spreadsheet with given URL
  + Oauth\_test\_case\_generator\_manager.py
    - Main Python script that manages all the subscripts in one place to manage the framework, after the execution of this script will be the followings:
      * Unit test case generated
      * Unit test case Pydoc HTML page generated
      * Unit test case executed
      * Unit test case result output file generated
  + Oauth\_test\_case\_genarator\_util.py
* Diagrams:
  + [Oauth\_test\_case\_generator\_manager\_use\_case.html](https://www.draw.io/?state=%7B%22ids%22:%5B%220B8vIEi2xiwB4cDB1empfdGhIbEU%22%5D,%22action%22:%22open%22,%22userId%22:%22112934744561083886759%22%7D#G0B8vIEi2xiwB4cDB1empfdGhIbEU)
  + [Oauth\_test\_case\_generator\_manager\_sequence\_diagram.html](https://www.draw.io/?state=%7B%22ids%22:%5B%220B8vIEi2xiwB4Ri1qeG01bmVzeWM%22%5D,%22action%22:%22open%22,%22userId%22:%22112934744561083886759%22%7D#G0B8vIEi2xiwB4Ri1qeG01bmVzeWM)
  + [Oauth\_test\_case\_generator\_generator\_sequence\_diagram.html](https://www.draw.io/?state=%7B%22ids%22:%5B%220B8vIEi2xiwB4LXhqZmxoQnJsek0%22%5D,%22action%22:%22open%22,%22userId%22:%22112934744561083886759%22%7D#G0B8vIEi2xiwB4LXhqZmxoQnJsek0)

# Framework:

* 1. A Python script requests the data from Google Spreadsheet using OAuth 2.0 library with Google Developer API credential setup complete.
     1. Detail on setup go to: [WY\_PYTHON\_ACCESS\_GOOGLE\_SPREADSHEET\_WITH\_OAUTH](https://docs.google.com/document/d/1tjjrUvoaPH0H2Ec1McGZulHH27z_gWw_h31cDG-orD8/edit)
     2. Let’s call this requesting Python script “oauth\_testcase\_generator.py”
  2. Execute the Python script with custom command line arguments set up using Python Argparse.
     1. Detail on setup go to: [WY\_PYTHON\_ARGPARSE](https://docs.google.com/document/d/1RgNrHR_bV8a0ORTQa6f_nbQN_Ka4-jomJwtIbNsqTPM/edit#heading=h.8hui0ssfb7qj)
        1. Let’s say i have the command line argument options setup using following code:

import argparse

parser = argparse.ArgumentParser(parents=[tools.argparser])

# Command line argument options:

**parser.add\_argument('-app','--application\_name', help='OAuth 2.0 client ID name')**

**parser.add\_argument('-ssurl','--google\_spreadsheet\_url', help='Google Spreadsheet URL')**

**parser.add\_argument('-feed','--feed\_name', help='Name of the spreadsheet for processing')**

# Parse the command line arguments.

flags = parser.parse\_args(sys.argv[1:])

print("flags.application\_name: ", flags.application\_name)

print("flags.google\_spreadsheet\_url: ", flags.google\_spreadsheet\_url)

print("flags.feed\_name: ", flags.feed\_name)

* + 1. So now i have command line argument options:
       1. “-app” option “--application\_name”
          1. Print description “APPLICATION\_NAME” when not prompted.
          2. Which i can provide the Application Name of the OAuth client ID name.
       2. “-ssurl” option “--google\_spreadsheet\_url”
          1. Which i can provide the Google Spreadsheet URL to process with.
          2. Print description “GOOGLE\_SPREADSHEET\_URL” when not prompted.
       3. “-feed” option “--feed\_name”
          1. Which i can provide the Google Spreadsheet feed name to process with.
          2. Print description “FEED\_NAME” when not prompted.
    2. Execution:
       1. $ python oauth\_test\_case\_generator.py -app WY\_PROJ\_BLENDER\_EDITOR\_OAUTH\_CLIENT -ssurl https://docs.google.com/spreadsheets/d/1FAibaRk-by5\_1UbVad9Nhmu\_QPOuBZDEeZidoQqIxFA/edit#gid=0 -feed Sheet1
  1. Manager:
     1. Using the command line arguments passed in with argparser, i can process them to proceed on retrieving data from the provided Google API credential OAuth client application name, Google Spreadsheet and feed name:
     2. Setup and working code is provided here:
        1. [WY\_PYTHON\_ACCESS\_GOOGLE\_SPREADSHEET\_WITH\_OAUTH](https://docs.google.com/document/d/1tjjrUvoaPH0H2Ec1McGZulHH27z_gWw_h31cDG-orD8/edit)
     3. Assume the data is read and loaded in the script properly, now i can process them however i want, but before that i create a log file to double check the data retrieval.
  2. Generator:
     1. Playing around with strings to generate the hard coded test case and pydoc docstrings with multiple data values retrieved from Google Spreadsheet.
     2. This part is modularized into multiple parts:
        1. Generate test case file string:
           1. Loop start.
           2. Generate test case function string.

Generate test case function Pydoc inner comment string.

Generate test case function with Pydoc inner comments inserted.

* + - * 1. Append to final test case functions string.
        2. Go to step (a) to repeat until the end of spreadsheet data.
        3. Loop end.
        4. Generate test case file header which contains import modules code, class definition and class Pydoc inner comment codes.
        5. Generate final test case file string: Header code + class definition + class Pydoc + test case functions code + unit test case main function.
      1. Write the unit test case code string to text file.
      2. Return the list contains the final generated unit test case file path at index 0 and unit test case execution result output file path at index 1 to store the unit test case execution result output.
  1. Execution:
     1. Final generated unit test case file path → Run unit test cases → Unit test case execution result output file path → Write resulting output into a text file.
     2. → Generate Pydoc with command “python -m pydoc -w [Python file without extension]” → HTML API page created.

# Framework summary:

* 1. Implement test case data Google Spreadsheet --> Project directory --> Adjust makefile providing spreadsheet URLs --> $ make
  2. Parse command line arguments get

- OAuth client application name.

- OAuth client secret JSON file path.

- Google Spreadsheet URL.

- Google Spreadhseet feed name.

- Google Spreadhseet feed range.

* 1. Authenticate with Google using OAuth to access private Google Spreadsheets data.
  2. Get Google Spreadsheet data and process it into Python data structure list.
     1. (Also get rid of empty lists inside the lists)
  3. Process spreadsheet data list into unit test case file string and generate the unit test case file onto class's test cases folder.
  4. Execute the generated unit test case file and store the result output onto a text file path also given in the Google Spreadsheet.
     1. Also generates the HTML Pydoc API documentation for each classes and the unit test case files just generated.

# Google Spreadsheet format template:

* 1. Sample spreadsheet template:
     1. <https://drive.google.com/drive/folders/0B8vIEi2xiwB4aFE2dzJVTHVRZWc>
  2. This spreadsheet contains the testing data, input parameters, expected outputs, testing file name, descriptions and many other elements which i needed to generate the test cases with. So i must follow the format of this spreadsheet on any other testing files.
  3. **MUST FILL IN ALL THE SECTION EXACTLY THE SAME AS SHOWN IN THE TEMPLATE!!!**
  4. Google Spreadsheet elements:
     1. Project title description element cells
     2. [Project Name ]
        1. Name of the project which i am creating test cases for.
     3. [Created by ]
        1. Name of the project creator.
     4. [Reference documents ]
        1. Reference document files and folders.
     5. [Date of creation ]
        1. Date of this test case data spreadsheet created.
     6. [Last update ]
        1. Date of last update on this test case data spreadsheet.
     7. [Test case file name ]
        1. Name of the test case file which test cases will be generated onto.
           1. **Caution: This will be appended to test case generator working directory so i must use relative path.**
     8. [Test case data file name ]
        1. Custom data text file name which is to double check the successful data retrieval from Google Spreadsheet using OAuth 2.0.
           1. Relative to OAuth test case generator to export with (Will be appended to test case generator working directory). (eg. ../WYMesh/WYMeshTestCases/WYMeshTestCaseData\_\_\_init\_\_.txt)
     9. [Testing class file name ]
        1. Destination file path of the main class file which current test case data is belong to.
           1. Relative to OAuth test case generator to export with (Will be appended to test case generator working directory). (eg. ../WYMesh/WYMesh.py)
     10. [Testing class function name ]
         1. Main testing function definition name with parameter names. (eg. \_\_init\_\_(XU, M\_XU, ….))
     11. [Unit test case class name ]
         1. Name of the class of the Python unit test case which current test case data spreadsheet is belong to.

(eg. WYMesh\_testfunction())

* + 1. [Unit test case class description ]
       1. Class description of the Python unit test case which current test case data spreadsheet is belong to.
    2. [Unit test case class header imports ]
       1. Import code that will be prepended to the generating unit test case file.
    3. [Unit test case result file name ]
       1. Destination file path of the file path which has the unit test case result output.
          1. Relative to OAuth test case generator to export with (Will be appended to test case generator working directory). (eg. ../WYMesh/WYMeshTestCases/WYMeshTestCaseResult\_\_\_init\_\_.txt)
    4. Test case input parameters element cells
    5. [Param Name ]
       1. Name of the input parameters in current test case data spreadsheet to generate the Pydoc @param element.
    6. [Param Type ]
       1. Type of the input parameters in current test case data spreadsheet to generate the Pydoc @type element.
    7. [Param Description ]
       1. Description of the input parameters in current test case data spreadsheet to generate the Pydoc @param element.
    8. Test case column element description title element cells
    9. [NC - Test Case ID ]
       1. Current column is the list of normal test case ID.
    10. [Test case title ]
        1. Current column is the list of test case title which will be prompted to very top of the function Pydoc docstring.
    11. [Test case description ]
        1. Current column is the list of test case title which will be prompted after the function test case title Pydoc docstring.
    12. [Testing class name ]
        1. Current column is the list of testing class name which will be used in the testing codes (eg. Allocation, calling member functions etc ...).
    13. [Testing function name ]
        1. Current column is the list of testing function name which will be used in the testing codes (eg. Class.testfunc() → This element should be “testfunc”).
    14. [Testing function return type ]
        1. Current column is the list of testing function return type value which will be used in the Pydoc inner comments.
    15. [Testing variable name ]
        1. Current column is the list of main testing variable which is the name of the object allocated with testing class name to be used in the testing codes to call the member functions (eg. testvar.testfunc() → This element should be “testvar”).
    16. Parameters
    17. [Param name ]
        1. Name of the parameter indicated in current column.
    18. [Param type ]
        1. Type of the parameter indicated in current column.
    19. [Param description ]
        1. Description of the parameter indicated in current column.
    20. [Alloc param 01 Alloc param 02 Alloc param 03 ... ]
        1. Current column is the list of testing class object allocation parameter data values where i can have many parameters passed into the constructor allocation call depending on constructors.
    21. [Func param 01 Func param 02 Func param 03 ... ]
        1. Current column is the list of testing function input parameter data values where i can have many parameters passed into the testing function..

**Note: This can be none when testing constructors.**

* + 1. [Expected Result 01 Expected Result 02 ... ]
       1. Current column is the list of testing function expected result tested with given input parameters data values where i can have many expected results which will be used in the assertion code (eg. testfunc(int a,int b,int c) → This section could be [Expected result 01 a==b][Expected result 02 b==a][Param 03 c==10]).

# How to run:

* 1. Implement test case data on Google Spreadsheet following the sample test case data spreadsheet format
     1. <https://docs.google.com/spreadsheets/d/1FAibaRk-by5_1UbVad9Nhmu_QPOuBZDEeZidoQqIxFA/edit?usp=drive_web>
     2. **CAUTION: IT IS MANDATORY TO FOLLOW THE SPREADSHEET TEMPLATE FORMAT!!!**
  2. Setup Google Developer Console for authentication and retrieve credential to get the access to the private spreadsheet following the instruction on guide:
     1. [WY\_PYTHON\_ACCESS\_GOOGLE\_SPREADSHEET\_WITH\_OAUTH](https://docs.google.com/document/d/1tjjrUvoaPH0H2Ec1McGZulHH27z_gWw_h31cDG-orD8/edit)
        1. Make sure the Google Developer Console OAuth client secret JSON file is downloaded and valid in the project directory.
  3. Navigate to project directory → Adjust makefile providing spreadsheet URLs
     1. Sample command:

python oauth\_test\_case\_generator\_manager.py \

-title "[ WYCoordsys \_\_init\_\_ function ]" \

-ssurl https://docs.google.com/spreadsheets/d/1FAibaRk-by5\_1UbVad9Nhmu\_QPOuBZDEeZidoQqIxFA/edit#gid=0 \

-feed Sheet1 \

-app WY\_PROJ\_BLENDER\_EDITOR\_OAUTH\_CLIENT \

> $(WYCOORDSYS\_TESTCASE\_FINAL\_RESULT\_FILE\_PATH)

* + 1. Can append as many of these just make sure the spreadsheet format is correct and the commands are correct. (Except for the first command, i will have to use “>>” instead of “>” for the last line of the command to append the text to file)
       1. Sample command:

python oauth\_test\_case\_generator\_manager.py \

title "[ WYCoordsys \_\_str\_\_ function ]" \

-ssurl https://docs.google.com/spreadsheets/d/1qaKGTW1wKdC7k4qASe1gIVy-WVNbQVIGq50hbtE3SJk/edit#gid=0 \

-feed Sheet1 \

-app WY\_PROJ\_BLENDER\_EDITOR\_OAUTH\_CLIENT \

>> $(WYCOORDSYS\_TESTCASE\_FINAL\_RESULT\_FILE\_PATH)

* 1. There are some important notes for above commands:
     1. Make sure the indentation is correct and no unnecessary “\t” tab characters.
     2. Make sure “>” is used for very first execution command, because this overwrites the file and then use “>>” for commands after to append the text to file.
     3. Command line arguments:
        1. “-ssurl” is provided with the proper Google Spreadsheet URL.
        2. “-app” is provided with valid Google Developer Console OAuth client name.
  2. $ make

# How to check:

* 1. The main testing class file name is "WYCoordsys.py" which is in "WYCoordsys" folder
  2. Generating unit test cases for \_\_init\_\_ function for WYCoordsys class with test case data from Google Spreadsheet:
     1. <https://docs.google.com/spreadsheets/d/1FAibaRk-by5_1UbVad9Nhmu_QPOuBZDEeZidoQqIxFA/edit?usp=drive_web>
  3. The unit test case related files will be generated under "WYCoordsys/WYCoordsysTestCases" folder

**- Unit test case data file will be named "WYCoordsysTestCaseData\_testingfuncname.txt"**

**- Unit test case file will be named "WYCoordsysTestCase\_testingfuncname.py"**

**- Unit test case Pydoc document HTML file will be named "WYCoordsysTestCase\_testingfuncname.html"**

**- Unit test case execution result output log file will be named "WYCoordsysTestCaseResult\_testingfuncname.txt"**

* 1. Directory:
     1. WYCoordsys/
        1. WYCoordsys.py
        2. WYCoordsysTestCases/
           1. WYCoordsysTestCaseData\_\_\_init\_\_.txt
           2. WYCoordsysTestCaseResult\_\_\_init\_\_.txt
           3. WYCoordsysTestCases\_\_\_init\_\_.html
           4. WYCoordsysTestCases\_\_\_init\_\_.py
     2. WYMain/
     3. WYMaterial/
     4. WYMesh/
     5. WYModel/
     6. WYModelManager/
     7. WYUtil/

# Test case generated format:

* 1. Sample test case generated with C:\Users\Nickj\Desktop\WY\_PROJ\_BLENDER\_DIRECTX\WY\_PROJ\_BLENDER\_EDITOR\mesh\_mat\_exporter\OAuthTestGenerator\oauth\_test\_case\_generator.py:

"""

Project Name : WY\_PROJ\_BLENDER\_EDITOR

Date of creation : 2017-09-17

Last update : 2017-10-02

Created by : NICK JANG

Testing file name : ..\WYCoordsys\WYCoordsys.py

Test case file name : WYCoordsysTestCases\_\_\_init\_\_.py

Test case data file name : WYCoordsysTestCaseData\_\_\_init\_\_.txt

Testing function : \_\_init\_\_(XU, M\_XU, YU, M\_YU, ZU, M\_ZU, XF, M\_XF, YF, M\_YF, ZF, M\_ZF)

Unit test class name : WYCoordsysTestCases\_\_\_init\_\_

"""

import unittest

precompile\_filename = "..\\WYCoordsys\\WYCoordsys.py"

exec(compile(open(precompile\_filename ).read(), precompile\_filename , 'exec'))

class WYCoordsysTestCases\_\_\_init\_\_(unittest.TestCase):

"""

Unit test case for class WYCoordsys

----------------------------------------------------------------------

Summary

----------------------------------------------------------------------

Number of normal case test :24

Number of boundary case test :0

Number of error case test :26

Number of white box test :0

Number of black box test :0

"""

def WYCoordsysTestCase\_\_\_init\_\_\_TC\_NC\_0001():

"""

----------------------------------------------------------------------

Title:

----------------------------------------------------------------------

WYCoordsys \_\_init\_\_ general constructor testing normal case 0001

----------------------------------------------------------------------

Description:

----------------------------------------------------------------------

Testing if the member variables are properly intialized with correct values passed into the allocation call where "XU" and "YF" parameter values are set to true and others are set to false.

----------------------------------------------------------------------

Parameters:

----------------------------------------------------------------------

:param XU: CoordsnatesystemexportoptionUPvectoraxispointstoXaxis.

:type XU: boolean.

:test XU: True.

:param M\_XU: CoordsnatesystemexportoptionUPvectoraxispointsto-Xaxis.

:type M\_XU: boolean.

:test M\_XU: False.

:param YU: CoordsnatesystemexportoptionUPvectoraxispointstoYaxis.

:type YU: boolean.

:test YU: False.

:param M\_YU: CoordsnatesystemexportoptionUPvectoraxispointsto-Yaxis.

:type M\_YU: boolean.

:test M\_YU: False.

:param ZU: CoordsnatesystemexportoptionUPvectoraxispointstoZaxis.

:type ZU: boolean.

:test ZU: False.

:param M\_ZU: CoordsnatesystemexportoptionUPvectoraxispointsto-Zaxis.

:type M\_ZU: boolean.

:test M\_ZU: False.

:param XF: CoordsnatesystemexportoptionFORWARDvectoraxispointstoXaxis.

:type XF: boolean.

:test XF: False.

:param M\_XF: CoordsnatesystemexportoptionFORWARDvectoraxispointsto-Xaxis.

:type M\_XF: boolean.

:test M\_XF: False.

:param YF: CoordsnatesystemexportoptionFORWARDvectoraxispointstoYaxis.

:type YF: boolean.

:test YF: True.

:param M\_YF: CoordsnatesystemexportoptionFORWARDvectoraxispointsto-Yaxis.

:type M\_YF: boolean.

:test M\_YF: False.

:param ZF: CoordsnatesystemexportoptionFORWARDvectoraxispointstoZaxis.

:type ZF: boolean.

:test ZF: False.

:param M\_ZF: CoordsnatesystemexportoptionFORWARDvectoraxispointsto-Zaxis.

:type M\_ZF: boolean.

:test M\_ZF: False.

----------------------------------------------------------------------

Returns:

----------------------------------------------------------------------

WYCoordsys

Return initialized WYCoordsys object with member variables intialized properly

----------------------------------------------------------------------

Expected Result

----------------------------------------------------------------------

wycoordsys\_nc0001 object created.

wycoordsys\_nc0001.XU == True

wycoordsys\_nc0001.YF == True

wycoordsys\_nc0001.coordoption\_up\_str == "X"

wycoordsys\_nc0001.coordoption\_forward\_str == "Y"

"""

wycoordsys\_nc0001 = WYCoordsys(True,False,False,False,False,False,False,False,True,False,False,False)

assertTrue(wycoordsys\_nc0001.XU == True)

assertTrue(wycoordsys\_nc0001.YF == True)

assertTrue(wycoordsys\_nc0001.coordoption\_up\_str == "X")

assertTrue(wycoordsys\_nc0001.coordoption\_forward\_str == "Y")