Testdata and frameworks

13 August 2023 22:36

Working with Test Data

Handling Excel- openpyxl module

1. Read data from excel
2. Write data to excel
3. Data-driven test case

.xlsx file -->	Workbook->	sheet->	rows& coloumn	cells

Read data from excel

```
import openpyxl

#.xlsx file -->      Workbook->      sheet->  rows& coloumn---> cells

xlpath = r"C:\Users\user\Desktop\sample.xlsx"
wb=openpyxl.load_workbook(xlpath)
sheet=wb["Sheet1"]

n_row=sheet.max_row
n_col=sheet.max_column

print("number of rows", n_row)
print("number of column", n_col)

for r in range(1, n_row+1):
   for c in range(1, n_col + 1):
      print(sheet.cell(r,c).value, end="\t")
   print("\n")
```

Write data to excel

```
#########################################################################3
# writes same data
###################3#####################################################

import openpyxl

#.xlsx file -->      Workbook->      sheet->  rows& coloumn---> cells

xlpath = r"C:\Users\user\Desktop\sample.xlsx"
wb=openpyxl.load_workbook(xlpath)
sheet=wb["Sheet2"]#---more number of sheet
#print(wb.sheetnames)# return all sheet in workbook
#sheet=wb.active# return active sheet works only for wb having only one sheet

# 1. decide how many rows you want to write data
n_row=6
n_col=6

print("number of rows", n_row)
print("number of column", n_col)

for r in range(1, n_row+1):
   for c in range(1, n_col + 1):
      sheet.cell(r,c).value="sample"

wb.save(xlpath)

#########################################################################3
# writes different data
###################3#####################################################

import openpyxl

#.xlsx file -->      Workbook->      sheet->  rows& coloumn---> cells

xlpath = r"C:\Users\user\Desktop\sample.xlsx"
wb=openpyxl.load_workbook(xlpath)
sheet=wb["Sheet3"]

print("number of rows", n_row)
print("number of column", n_col)

sheet.cell(1,1).value="emp1"
sheet.cell(1,2).value="123"

sheet.cell(2,1).value="emp2"
sheet.cell(2,2).value="345"

sheet.cell(3,1).value="emp3"
sheet.cell(3,2).value="567"

sheet.cell(4,1).value="emp4"
```

```
sheet.cell(4,2).value="789"

wb.save(xlpath)
```

***Most of the time these data reading writing will present in utility files

- If we include the excel related automation in all test case.
- It will increase complexity of code
- To reduce that we have to write common logic for all excel related operation and include that in utility

Identify the reusable functionality:

1.

1	Get number of rows
2	Get number of columns
3	Read data from cell
4	Write data from cell
5	Fill colour pattern if wanted

Simple interest calculator

https://cleartax.in/s/simple-compound-interest-calculator

```
import openpyxl
from openpyxl.styles import PatternFill

def getRowCount(xlpath,sheetname):
    wb=openpyxl.load_workbook(xlpath)
    sheet=wb[sheetname]
    return sheet.max_row

def getColoumnCount(xlpath,sheetname):
    wb = openpyxl.load_workbook(xlpath)
    sheet = wb[sheetname]
    return sheet.max_column

def readData(xlpath,sheetname, r_no, c_no):
    wb = openpyxl.load_workbook(xlpath)
    sheet = wb[sheetname]
    return sheet.cell(r_no, c_no).value

def writeData(xlpath,sheetname, r_no, c_no, data):
    wb = openpyxl.load_workbook(xlpath)
    sheet = wb[sheetname]
    sheet.cell(r_no, c_no).value = data
    wb.save(xlpath)

def fillGreenColour(xlpath,sheetname, r_no, c_no):
    wb = openpyxl.load_workbook(xlpath)
    sheet = wb[sheetname]
    green=PatternFill(start_color="60b212",
                      end_color="60b212",
                      fill_type="solid")
    sheet.cell(r_no, c_no).fill=green
    wb.save(xlpath)

def fillRedColour(xlpath,sheetname, r_no, c_no):
    wb = openpyxl.load_workbook(xlpath)
    sheet = wb[sheetname]
    red=PatternFill(start_color="ff0000",
                      end_color="ff0000",
                      fill_type="solid")
    sheet.cell(r_no, c_no).fill=red
    wb.save(xlpath)
```

##############################data driven testing

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.select import Select
# 1. open excel
from framework import xlutils

servcie_obj = Service("E:\selenium\drivers\chromedriver.exe")
driver = webdriver.Chrome(service=servcie_obj)

driver.get("https://cleartax.in/s/simple-compound-interest-calculator")
driver.maximize_window()
driver.implicitly_wait(10)

# 2. get number of rows and column
file = r"C:\Users\user\Desktop\simple_interest.xlsx"
n_row = xlutils.getRowCount(file, "Sheet1")
n_col = xlutils.getColoumnCount(file, "Sheet1")

# skip the first row
for r in range(2, n_row + 1):
    # 3. get data from excel sheet
    principal_amt = xlutils.readData(file, "Sheet1", r, 1)
    interest = xlutils.readData(file, "Sheet1", r, 2)
    period_num = xlutils.readData(file, "Sheet1", r, 3)
    period_type = xlutils.readData(file, "Sheet1", r, 4)
```

```
int_type = xlutils.readData(file, "Sheet1", r, 5)
total_amt = xlutils.readData(file, "Sheet1", r, 6)

# give input in web page
int_type_elem = driver.find_element(By.XPATH, '//*[@id="a"]')
int_type_drp = Select(int_type_elem)
int_type_drp.select_by_visible_text(int_type)  # ----------> pass value from xl
driver.find_element(By.XPATH, '//*[@id="c"]').clear()
driver.find_element(By.XPATH, '//*[@id="c"]').send_keys(principal_amt)  # ----------> pass value from xl
driver.find_element(By.XPATH, '//*[@id="d"]').clear()
driver.find_element(By.XPATH, '//*[@id="d"]').send_keys(interest)  # ----------> pass value from xl

period_unit_elem = driver.find_element(By.XPATH, '//*[@id="f"]')
period_unit_drp = Select(period_unit_elem)
period_unit_drp.select_by_visible_text(period_type)

driver.find_element(By.XPATH, '//*[@id="e"]').clear()
driver.find_element(By.XPATH, '//*[@id="e"]').send_keys(period_num)  # ----------> pass value from xl

# validation part
get_total_amt = driver.find_element(By.XPATH, '//*[@id="calc"]/div/div[8]/div/span').text
xlutils.writeData(file, "Sheet1", r, 8, get_total_amt)
print(get_total_amt[1:].replace(",", "").replace(" ", ""))
print(total_amt)
if float(get_total_amt[1:].replace(",", "").replace(" ", "")) == float(total_amt):
    print("test passed...")
    xlutils.writeData(file, "Sheet1", r, 9, "Passed")
    xlutils.fillGreenColour(file, "Sheet1", r, 9)
else:
    print("test falied...")
    xlutils.writeData(file, "Sheet1", r, 9, "Failed")
    xlutils.fillRedColour(file, "Sheet1", r, 9)
```

➢ Logging module

```
import logging

logging.debug("this is a debug msg")
logging.info("this is a info msg")
logging.warning("this is a warning msg")
logging.error("this is a error msg")
logging.critical("this is a critical msg")
```

```
logger
C:\Users\user\AppData\Local\Programs\Python\Python38\p
WARNING:root:this is a warning msg
ERROR:root:this is a error msg
CRITICAL:root:this is a critical msg

Process finished with exit code 0
```

Debug and info not printed in console

Default logger settings

```
print(logging.getLogger()) #--><RootLogger root (WARNING)>
```

Level	Numeric Value
NOTSET	0
DEBUG	10
INFO	20
WARNING	30
ERROR	40
CRITICAL	50

```
logging.basicConfig()
logger = logging.getLogger()

logger.setLevel(logging.DEBUG)
logging.debug("this is a debug msg")
logging.info("this is a info msg")
logging.warning("this is a warning msg")
logging.error("this is a error msg")
logging.critical("this is a critical msg")
```

```
C:\Users\user\AppData\Local\Programs\Python\
DEBUG:root:this is a debug msg
INFO:root:this is a info msg
WARNING:root:this is a warning msg
ERROR:root:this is a error msg
CRITICAL:root:this is a critical msg

Process finished with exit code 0
```

```
logging.basicConfig(filename="test.log",
                    format="%(asctime)s %(levelname)s %(message)s",
                    filemode="w")
logger = logging.getLogger()

logger.setLevel(logging.DEBUG)
logging.debug("this is a debug msg")
logging.info("this is a info msg")
logging.warning("this is a warning msg")
logging.error("this is a error msg")
logging.critical("this is a critical msg")
```

logger.py | test.log

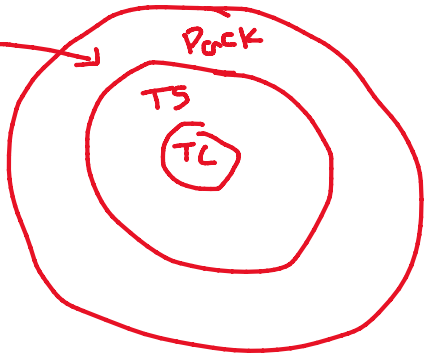
Plugins supporting *.log files found.

```
2023-08-15 23:41:26,094 DEBUG this is a debug msg
2023-08-15 23:41:26,094 INFO this is a info msg
2023-08-15 23:41:26,094 WARNING this is a warning msg
2023-08-15 23:41:26,095 ERROR this is a error msg
2023-08-15 23:41:26,095 CRITICAL this is a critical msg
```

- Unittest framework
  Inbuilt in python
  Good log structure

```
######################################################################33

import unittest


class Test(unittest.TestCase):
    def test_firstTest(self):
        print("this is my first unit test case")


if __name__ == "__main__":
    unittest.main()
######################################################################33
import unittest
from selenium import webdriver

class SearchEnginesTest(unittest.TestCase):
    def test_google(self):
        self.driver=webdriver.Chrome()
        self.driver.get("https://www.google.com/")
        print("Title of the page is ", self.driver.title)
        self.driver.close()

    def test_bing(self):
        self.driver=webdriver.Chrome()
        self.driver.get("https://www.bing.com/")
        print("Title of the page is ", self.driver.title)
        self.driver.close()

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

- Keywords available in unittest framework
  - setUp
  - tearDown
  - setUpClass
  - tearDownClass
  - setUpModule
  - tearDownModule




```
import unittest
from selenium import webdriver

class AppTesting(unittest.TestCase):

    @classmethod
    def setUp(self):# executed before all the class method
        print("this is login test")
    @classmethod
    def tearDown(self):# executed after all the class method
        print("this is logout test")

    @classmethod
    def setUpClass(cls):# executed once when the class method started
        print("opening the application")
```

```
    @classmethod
    def tearDownClass(cls):# executed once when the class method completed
        print("Closing the application")

    def test_search(self):
        print("this is the search test")

    def test_advancedserach(self):
        print("this is the advanced search test")

    def test_prepaidrecharge(self):
        print("this is the prepaid recharge test")

    def test_postpaidrecharge(self):
        print("this is the postpaid recharge test")

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

```
C:\Users\user\PycharmProjects\pythonProject_django\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm C
Testing started at 10:38 ...
Launching unittests with arguments python -m unittest TestCase3.AppTesting in C:\Users\user\PycharmProjects\pyth

opening the application
this is login test
this is the advanced search test
this is logout test
this is login test
this is the postpaid recharge test
this is logout test
this is login test
this is the prepaid recharge test
this is logout test
this is login test
this is the search test
this is logout test
Closing the application


Ran 4 tests in 0.009s

OK
```



```
import unittest
from selenium import webdriver

def setUpModule():# will be executed before executing any class or any method present in the test class
    print("setup module")

def tearDownModule():# will be executed after completing everything present in the python module
    print("TearDown module")

class AppTesting(unittest.TestCase):

    @classmethod
    def setUp(self):# executed before all the class method
        print("this is login test")
    @classmethod
    def tearDown(self):# executed after all the class method
        print("this is logout test")

    @classmethod
    def setUpClass(cls):# executed once when the class method started
        print("opening the application")

    @classmethod
    def tearDownClass(cls):# executed once when the class method completed
        print("Closing the application")

    def test_search(self):
        print("this is the search test")

    def test_advancedserach(self):
        print("this is the advanced search test")

    def test_prepaidrecharge(self):
        print("this is the prepaid recharge test")

    def test_postpaidrecharge(self):
        print("this is the postpaid recharge test")

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

```
C:\Users\user\PycharmProjects\pythonProject_django\ven
Testing started at 10:46 ...
Launching unittests with arguments python -m unittest

setup module
opening the application
this is login test -
this is the advanced search test
this is logout test -
this is login test -
this is the postpaid recharge test
this is logout test -
this is login test -
this is the prepaid recharge test
this is logout test -
this is login test -
this is the search test
this is logout test -
Closing the application
TearDown module


Ran 4 tests in 0.008s
```

- Skipping tests
  - Skip test
  - Skip test with reason
  - Skip test based on condition

```
import unittest
from selenium import webdriver

class AppTesting(unittest.TestCase):

    @unittest.SkipTest
    def test_search(self):
        print("this is the search test")

    @unittest.skip("skipping this test method because it is underdeveloped")
    def test_advancedserach(self):
        print("this is the advanced search test")

    @unittest.skipIf(1==1, "condition is True")# True
    def test_prepaidrecharge(self):
        print("this is the prepaid recharge test")

    def test_postpaidrecharge(self):
        print("this is the postpaid recharge test")

    def test_loginbygmail(self):
        print("this is the login by gmail test")\

    def test_loginbyfb(self):
        print("this is the login by facebook test")


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

```
Tests passed: 3, ignored: 2 of 5 tests – 5 ms
C:\Users\user\PycharmProjects\pythonProject_django\venv\Scripts\python.exe "C:\Program Files\Je
Testing started at 10:59 ...
Launching unittests with arguments python -m unittest skiptest.AppTesting in C:\Users\user\Pycha


Skipped: skipping this test method because it is underdeveloped
this is the login by facebook test
this is the login by gmail test
this is the postpaid recharge test

Skipped: condition is True


Ran 5 tests in 0.010s

OK (skipped=2)

Process finished with exit code 0
```

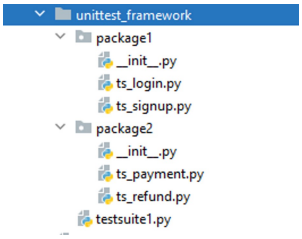
Creating Test Suite using unittest

unittest framework

-package1
- login(testsuite)
  - login using gmail
  - login using fb
  - login using twitter
- signup(testsuite)
  - signup using gmail
  - signup using fb
  - signup using twitter

-package2
- payment(testsuite)
  - in dollars(tc)
  - in rupee(tc)
- refund(testsuite)
  - refund in bank(tc)
  - refund in wallet(tc)

testsuite.py
- sanity
- functional
- master



ts_login.py

```
import unittest

class Testlogin(unittest.TestCase):

    def test_loginusinggmail(self):
        print("Login using gmail finished")

    def test_loginusingfb(self):
        print("Login using facebook finished")

    def test_loginusingtwitter(self):
        print("Login using twitter finished")


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

ts_signup.py

```
import unittest

class TestSignup(unittest.TestCase):

    def test_signupusinggmail(self):
        print("Signup using gmail finished")

    def test_signupusingfb(self):
        print("Signup using facebook finished")

    def test_signupusingtwitter(self):
        print("Signup using twitter finished")


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

ts_payment.py

```
import unittest

class TestPayment(unittest.TestCase):

    def test_paymentindollars(self):
        print("Payment in dollar finished")

    def test_paymentinrupee(self):
        print("Payment in rupee finished")
```

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

ts_refund.py

```
import unittest

class TestRefund(unittest.TestCase):

    def test_refundinbank(self):
        print("Refund in bank finished")

    def test_refundinwallet(self):
        print("Refund in wallet finished")


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

Testsuite1.py

```
importunittest

frompackage1.ts_loginimportTestlogin
frompackage1.ts_signupimportTestSignup

frompackage2.ts_refundimportTestRefund
frompackage2.ts_paymentimportTestPayment

#getalltestcasesfromtestlogin,testsignup,testrefund,testpayment
tc1=unittest.TestLoader().loadTestsFromTestCase(Testlogin)
tc2=unittest.TestLoader().loadTestsFromTestCase(TestSignup)
tc3=unittest.TestLoader().loadTestsFromTestCase(TestPayment)
tc4=unittest.TestLoader().loadTestsFromTestCase(TestRefund)

#createthetestsuite
sanity=unittest.TestSuite([tc1,tc2])
functionality=unittest.TestSuite([tc3,tc4])
master=unittest.TestSuite([tc1,tc2,tc3,tc4])

unittest.TextTestRunner(verbosity=2).run(master)
```

```
C:\Users\user\PycharmProjects\pythonProject_django\venv\Scripts\python.exe C:/U
Login using facebook finished
Login using gmail finished
Login using twitter finished
Signup using facebook finished
Signup using gmail finished
Signup using twitter finished
Payment in dollar finished
Payment in rupee finished
Refund in bank finished
Refund in wallet finished
test_loginusingfb (package1.ts_login.Testlogin) ... ok
test_loginusinggmail (package1.ts_login.Testlogin) ... ok
test_loginusingtwitter (package1.ts_login.Testlogin) ... ok
test_signupusingfb (package1.ts_signup.TestSignup) ... ok
test_signupusinggmail (package1.ts_signup.TestSignup) ... ok
test_signupusingtwitter (package1.ts_signup.TestSignup) ... ok
test_paymentindollars (package2.ts_payment.TestPayment) ... ok
test_paymentinrupee (package2.ts_payment.TestPayment) ... ok
test_refundinbank (package2.ts_refund.TestRefund) ... ok
test_refundinwallet (package2.ts_refund.TestRefund) ... ok

----------------------------------------------------------------------
Ran 10 tests in 0.001s

OK
```

➢ PYTEST

Install pytest

Pip install pytest

Pytest --version

It is build on top of unittest module. It has some additional features

```
Import pytest

def testMethod1():
print("Thisistestmethod1")

def testMethod2():
print("Thisistestmethod2")
```

In console, Execute pytest to run all test cases or run pytest -v -s for more details in console

```
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest> pytest
============================= test session starts =============================
platform win32 -- Python 3.8.5, pytest-7.1.3, pluggy-1.0.0
rootdir: C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest
collected 2 items

test_pytestdemo1.py ..                                                   [100%]

============================== 2 passed in 0.03s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest>
```

```
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest> pytest -v -s
============================= test session starts =============================
platform win32 -- Python 3.8.5, pytest-7.1.3, pluggy-1.0.0 -- c:\users\user\appdata\local\programs\python\python38\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest
collected 2 items

test_pytestdemo1.py::testMethod1 This is test method1
PASSED
test_pytestdemo1.py::testMethod2 This is test method2
PASSED

============================== 2 passed in 0.02s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest>
```


Pytest fixtures:
@pytest.fixture()---executes before every test methods
@pytest.yeild_fixture()----executes before and after test methods [depricated]

```
import pytest

@pytest.fixture()
def setup():
   print("this is setup method")

def testMethod1(setup):
   print("This is test method1")

def testMethod2(setup):
   print("This is test method2")
```

```
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest> pytest -v -s
============================= test session starts =============================
collected 2 items

test_pytestdemo1.py::testMethod1 this is setup method
This is test method1
PASSED
test_pytestdemo1.py::testMethod2 this is setup method
This is test method2
PASSED

============================== 2 passed in 0.02s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest>
```

```
importpytest

#@pytest.fixture()
#defsetup():
#print("thisissetupmethod")
@pytest.yield_fixture()
defsetup():
print("thisisexecutedbeforetestmethod")
yield
print("thisisexecutedaftertestmethod")



deftestMethod1(setup):
print("Thisistestmethod1")

deftestMethod2(setup):
print("Thisistestmethod2")
```

Ways to run pytest :

To run all the test methods in a module

Pytest -v -s <filename>.py

To run all modules in a path

Pytest -v -s path

To run specific test method from a module

Pytest -v -s <filename>.py::<modulename>

```
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest> pytest -v -s test_pytestdemo1.py
============================= test session starts =============================
platform win32 -- Python 3.8.5, pytest-7.1.3, pluggy-1.0.0 -- c:\users\user\appdata\local\programs\python\python38\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest
collected 2 items

test_pytestdemo1.py::testMethod1 this is setup method
This is test method1
PASSED
test_pytestdemo1.py::testMethod2 this is setup method
This is test method2
PASSED

============================== 2 passed in 0.02s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest>
```



```
PS C:\Users\user\PycharmProjects\pythonProject_django> pytest -v -s framework\pytest\
============================= test session starts =============================
platform win32 -- Python 3.8.5, pytest-7.1.3, pluggy-1.0.0 -- c:\users\user\appdata\local\programs\python\python38\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\user\PycharmProjects\pythonProject_django
collected 2 items

framework/pytest/test_pytestdemo1.py::testMethod1 this is setup method
This is test method1
PASSED
framework/pytest/test_pytestdemo1.py::testMethod2 this is setup method
This is test method2
PASSED

============================== 2 passed in 0.04s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django>
```



```
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest> pytest -v -s test_pytestdemo1.py::testMethod2
============================= test session starts =============================
platform win32 -- Python 3.8.5, pytest-7.1.3, pluggy-1.0.0 -- c:\users\user\appdata\local\programs\python\python38\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest
collected 1 item

test_pytestdemo1.py::testMethod2 this is setup method
This is test method2
PASSED

============================== 1 passed in 0.01s ==============================
PS C:\Users\user\PycharmProjects\pythonProject_django\framework\pytest>
```