Simeon Patton

In-class activity Week 7

CS362 OSU Spring 2021

*All Code can be found at: https://github.com/pattons-OSU/CS362_InClassActivities/tree/master/Week_7

*To run pytest programs: in a powershell window type “pytest (program title)

Program 1;

Create program that takes in user input and checks to see if input is a palindrome. Create Unittest and Pytest programs and catalog output here.

Program Code: palindrome.py

```
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021

Palindrome -    1. Ask the user for a string and determine whether it is a palindrome or not.
                2. Write tests for the above specification using unittest and pytest.
                3. Document the outputs in a pdf - (screenshots of the output from unittest and pytest)

    Some code inspiration from www.shorturl.at/eglJ6
"""

def user_input():
    ## Taking in user input and returning string value
    usr_input = input("\nPlease enter a word or string to check:\n")
    return usr_input


def pal_check(string):
    ## Taking in "usr_input" as a parameter and checking a boolean return
    ## if the original string is the same as the string in reverse
    return string == string[::-1]



if __name__ == "__main__":

    ## Driver code
    string = user_input()
    ans = pal_check(string)

    if ans:
        print(f"\nYour input of '{string}' is a palendrome.\n")
    else:
        print(f"\nYour input of '{string}' is NOT a palendrome.\n")
```

Unit-test Code: test_palindrome.py

```
#! python3
"""
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021

Palindrome -    1. Ask the user for a string and determine whether it is a palindrome or not.
                2. Write tests for the above specification using unittest and pytest.
                3. Document the outputs in a pdf - (screenshots of the output from unittest and pytest)

    Some code inspiration from www.shorturl.at/eglJ6

"""

import unittest
import palindrome

## UnitTest Module
class testCaseVolume(unittest.TestCase):
##Unit Test
##Testing pass and fail of the input type
    ## Testing to see if the input is a type value of "string"
    def test_type(self):
      self.assertTrue(type(palindrome.user_input) is str)
    ## This test should fail as the input type IS string
    def test_type_fail(self):
      self.assertTrue(type(palindrome.user_input) is not str)


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

Unit-test powershell output:

One of the tests is designed to fail and one to pass. This was checking to make sure that the user input is indeed a data type of string.

```
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7> python .\test_palindrome.py
F.
======================================================================
FAIL: test_type (__main__.testCaseVolume)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7\test_palindrome.py", line 28, in test_type
    self.assertTrue(type(palindrome.user_input) is str)
AssertionError: False is not true

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7>
```

Pytest Code: pytest_palindrome.py

```python
#! python3
"""
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021

Palindrome -    1. Ask the user for a string and determine whether it is a palindrome or not.
                2. Write tests for the above specification using unittest and pytest.
                3. Document the outputs in a pdf - (screenshots of the output from unittest and pytest)

    Some code inspiration from www.shorturl.at/eglJ6

"""
import palindrome
import pytest

## Had to hard code a user input here because the "fixture"
## was not found otherwise. Hardcode suggested for final draft
## by TA Raghuram
@pytest.fixture
def user_input():
    return "hello world"

##PyTest Module
##user_input = palindrome.user_input()

## Pytest, testing to see if user input is a string and not another data type
def test_type(user_input):
    user_input = "hello world"
    assert type(user_input) is str

if __name__ == '__main__':
    user_input = "hello world"

    test_type(user_input)
```

Pytest powershell output: Run command “pytest ./pytest_palindrome.py”

```
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7> pytest .\pytest_palindrome.py
============================= test session starts =============================
platform win32 -- Python 3.7.3, pytest-4.4.1, py-1.8.0, pluggy-0.9.0
rootdir: C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7
collected 1 item

pytest_palindrome.py .                                                   [100%]

========================== 1 passed in 0.03 seconds ===========================
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7>
```

Program 2;

Create program that takes in user input and counts the number of words found within the input string. Create Unittest and Pytest programs and catalog output here.

Program Code: word_count.py

```
#! python3
"""
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021

Word Count-
        1. Ask the user for a sentence and determine the number of
        words in that sentence.
        Example - “This is an activity”, Output - 4.
        2. Write tests for the above specification using unittest and
        pytest.
        3. Document the outputs in a pdf - (screenshots of the output
        from unittest and pytest)
"""

def user_input():
    ## Taking in user input and returning it for use
    usr_input = input("\nPlease enter a sentence to count:\n")
    return usr_input

def string_split(string):
    ## Breaking user input into separate items and then counting them
    separate = string.split()
    count = len(separate)
    print(f"\nOutput - {count}\n")
    return count

if __name__ == "__main__":
    string = user_input()
    string_split(string)
```

Unit-test code: test_word_count.py

```
#! python3
"""
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021

Word Count-
        1. Ask the user for a sentence and determine the number of
        words in that sentence.
        Example - “This is an activity”, Output - 4.
        2. Write tests for the above specification using unittest and
        pytest.
        3. Document the outputs in a pdf - (screenshots of the output
        from unittest and pytest)
"""
import unittest
import word_count


class testCaseVolume(unittest.TestCase):
##Unit Test
##Testing pass and fail of the input type
    ## Testing to see if the input is a type value of "string"
    def test_type(self):
      self.assertTrue(type(word_count.user_input) is str)
    ## This test should fail as the input type IS string
    def test_type_fail(self):
      self.assertTrue(type(word_count.user_input) is not str)

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

Unit-test powershell output: Once again, one of the tests is design to fail as a failsafe to make sure that the user is inputting the correct data type.

```
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7> python .\test_word_count.py
F.
======================================================================
FAIL: test_type (__main__.testCaseVolume)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7\test_word_count.py", line 25, in test_type
    self.assertTrue(type(word_count.user_input) is str)
AssertionError: False is not true

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (failures=1)
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7>
```

Pytest code: pytest_word_count.py

```python
#! python3
"""
Simeon Patton
Inclass activity Week 7
CS362 - OSU Spring 2021
Word Count-
        1. Ask the user for a sentence and determine the number of
        words in that sentence.
        Example - “This is an activity”, Output - 4.
        2. Write tests for the above specification using unittest and
        pytest.
        3. Document the outputs in a pdf - (screenshots of the output
        from unittest and pytest)
"""
import word_count
import pytest

## I was forced to hardcode the input on this one too as the pytest
## module would not allow the user to input into the program eventhough
## it was called as seen below
@pytest.fixture
def user_input():
    ##user_input = word_count.user_input()
    user_input = "Hello I had to hardcode this too"
    return user_input
## Pytest, testing to see if user input is a string and not another data type
def test_type(user_input):
    assert type(user_input) is str

if __name__ == '__main__':
    test_type(user_input)
```

Pytest Powershell output: Run command “pytest ./pytest_word_count.py”

```
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7> pytest .\pytest_word_count.py
============================= test session starts =============================
platform win32 -- Python 3.7.3, pytest-4.4.1, py-1.8.0, pluggy-0.9.0
rootdir: C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7
collected 1 item

pytest_word_count.py .                                                   [100%]

========================== 1 passed in 0.02 seconds ===========================
PS C:\Users\sup_u\Documents\College\CS362\On Git\CS362_InClassActivities\Week_7>
```

Conclusion:

I did not go very deep into creating a bunch of tests for this assignment as the brief did not mention how many tests were required and also, I believe that the idea behind this assignment was to learn and figure out how to implement a different testing framework (PyTest). Not only to learn the beginnings of the other framework, but also figure out the differences between UnitTest and PyTest.