Python Project Tutorial 3

Pytest

Pytest is the most popular package for testing in Python and offers a range of plugins, has intuitive syntax and can be used to run unit tests that are essential in strengthening the delivery of your production of code. Moreover, Pytest provides informative failure prompts without the need of a debugger (more on this later), and the tests that are applied to the code run in parallel.

As with any Python project, create a directory where you will store your python files. You can do this from the command line by writing:

In [ ]:
```
cd path_name mkdir project_name # change path_name and project name accordingly
```

In this new directory, create a virtual environment and install the relevant packages you will be using throughout your project, e.g.,

In [ ]:
```
venv ve
ve\Scripts\activate
python -m pip install numpy pandas matplotlib pytest
pip list # shows all packages stored globally
pip freeze # shows all packages stored in ve
```

To use Pytest, you can create a Python file, say factorial.py, that calculates the factorial of some input, for example:

In [2]:
```
def factorial(num):
    if type(num)!=int or num<0: # order matters here!
        return 'Please try again and make sure your input is a positive integer!'
    elif num==0 or num==1:
        return 1
    else:
        return num*factorial(num-1)
```

This is a simple function that requires the input to be a positive integer to perform the task of calculating the corresponding factorial. To apply Pytest to this, we can create a separate .py file that does the following:

In [3]:
```
from factorial import factorial

def test_factorial_true():
    assert factorial(0)==1
    assert factorial(1)==1
    assert factorial(2)==2
    assert factorial(3)==6
    assert factorial(4)==24
    assert factorial(5)==120
    assert factorial(6)==720

def test_factorial_false():
```

```
    assert factorial(-1)=='Please try again and make '
        +'sure your input is a positive integer!'
    assert factorial('hello')=='Please try again and make '
        +'sure your input is a positive integer!'
    assert factorial(1.5)=='Please try again and make '
        +'sure your input is a positive integer!'

def test_factorial_linear_time():
    inputs=[10**x for x in range(10)]
    durations=[]
    for input in inputs:
        start=time.time()
        is_factorial=factorial(input)
        end=time.time()
        duration=end-start
        durations.append(duration)

    tmp=durations[1]/durations[0]
    for i in range(1,len(durations)-1):
        if durations[i+1]/durations[i]!=tmp:
            return False
    return True
```

You can run the test file via the command line using:

In [ ]:
```
pytest test_factorial.py # assuming the test file is called test_factorial.py
py.test -v # gives you a more detailed output for each test case
```

The output will tell you what tests have passed or failed and the time take for the tests to finish.

We can make this testing file neater using parametrisation:

In [ ]:
```
from factorial import factorial

@pytest.mark.parametrize('test_input','expected_output',[(0,1),(1,1),(2,2),(3,6),
                                                         (4,24),(5,120),(6,720)])
    def test_factorial_true(test_input,expected_output):
        assert factorial(input)==expected_output
```