|  |
| --- |
| LEARNING OBJECTIVES FOR THIS LESSON |
| * Analyze the role, characteristics and benefits of unit testing in software quality assurance * Apply the "Arrange, Act, Assert" pattern to effectively structure unit tests * Discuss the use of test coverage metrics to assess the comprehensiveness of unit testing * Configure and use Pytest for writing and automating unit tests |
|  |
| Listen attentively and take notes |
|  |

|  |
| --- |
| Introduction to pytest |
|  |
| Go through the guides in the guides folder |
| * 01 Python3, Visual Studio Code and pytest |
|  |
| In the P06\_pytest\_project folder, create a folder called **calculator** |
| In the calculator folder, create an empty file called **\_\_init.py\_\_** |
| In the calculator folder, create a file called **calculator.py** |
|  |
| Paste the following code in **calculator.py** |
| class Calculator:  def add(self, a, b):  return a + b  def subtract(self, a, b):  return a - b  def multiply(self, a, b):  return a \* b  def divide(self, a, b):  if b == 0:  raise ZeroDivisionError("Division by zero error")  return a / b |
|  |
| In the tests folder, create a file called **test\_calculator.py** |
|  |
| Paste the following code in **test\_calculator.py** |
| from calculator.calculator import Calculator  class TestCalculator:  def test\_add(self):  # arrange  a = 4321  b = 1234  cal = Calculator()  # act  result = cal.add(a, b)  # assert  expected = 5555  assert result == expected |
|  |
| Run pytest and ensure that all tests pass |
|  |

|  |
| --- |
| In the **TestCalculator** class in **test\_calculator.py** : |
| * Create at least 1 test case for each method in the Calculator class |
| * Apply the "Arrange, Act, Assert" pattern to effectively structure your unit tests |
| * Use the assert method to compare the expected result with the actual result |
|  |
| Paste the methods from the Calculator class and corresponding pytest methods in the table below |
|  |
| |  |  |  | | --- | --- | --- | | **function** | **Calculator code** | **pytest code** | | add | def add(self, a, b):  return a + b | def test\_add(self):  # arrange  a = 4321  b = 1234  cal = Calculator()  # act  result = cal.add(a, b)  # assert  expected = 5555  assert result == expected | | subtract | def subtract(self, a, b):          return a - b | def test\_subtract(self):          # arrange          a = 4          b = 1          cal = Calculator()          # act          result = cal.subtract(a, b)          # assert          expected = 2          assert result == expected | | multiply | def multiply(self, a, b):          return a \* b | def test\_multiply(self):          # arrange          a = 3          b = 3          cal = Calculator()          # act          result = cal.multiply(a, b)          # assert          expected = 9          assert result == expected | | divide | def divide(self, a, b):          if b == 0:              raise ZeroDivisionError("Division by zero error")          return a / b | def test\_divide(self):          # arrange          a = 4          b = 2          cal = Calculator()          # act          result = cal.divide(a, b)          # assert          expected = 2          assert result == expected | |
|  |
|  |
| Install and run coverage.py |
|  |
| To install coverage, run the following command |
| pip install coverage |
|  |
| To run coverage, run the following command |
| ͏coverage run --branch -m pytest |
|  |
| To generate the report in HTML format, run the following 2 commands |
| ͏coverage report -m |
| ͏coverage html |
|  |

|  |
| --- |
| L06a Deliverables |
|  |
| * Show your lecturer your pytest code. Ensure that it can execute without error and all tests pass. |
| * Generate the coverage report to show that you have 100% coverage |
| * Paste the code from test\_calculator.py into the box below |
| from calculator.calculator import Calculator  class TestCalculator:  def test\_add(self):  # arrange  a = 4321  b = 1234  cal = Calculator()  # act  result = cal.add(a, b)  # assert  expected = 5555  assert result == expected  def test\_subtract(self):  # arrange  a = 4  b = 1  cal = Calculator()  # act  result = cal.subtract(a, b)  # assert  expected = 3  assert result == expected    def test\_multiply(self):  # arrange  a = 3  b = 3  cal = Calculator()  # act  result = cal.multiply(a, b)  # assert  expected = 9  assert result == expected  def test\_divide(self):  # arrange  a = 4  b = 2  cal = Calculator()  # act  result = cal.divide(a, b)  # assert  expected = 2  assert result == expected |
|  |
| * Paste a screenshot of the coverage report into the box below |
| A screenshot of a computer  Description automatically generated |
|  |