

Assignment Date	18 November 2022
Student Name	K.Ananthavalli
Student Roll Number	815819104004
Maximum Marks	

```
#!/usr/bin/env python3
```

```
from testflows.core import  
Scenario
```

```
with Scenario("Hello World!"):
```

```
    pass
```

The same test can be defined using [TestScenario](#) decorated function. See [Decorated Tests](#).

```
#!/usr/bin/env python3
```

```
from testflows.core import TestScenario,  
Name
```

```
@TestScenario
```

```
@Name("Hello World!")
```

```
def hello_world(self):
```

```
    pass
```

```
# run `Hello World!` test
```

```
hello_world()
```

```
from testflows.core import
Scenario
```

```
with Scenario("Hello World!"):
```

```
    assert 1 == 0, "1 != 0"
```

The result will be as follows.

```
$ python3 hello_world.py
```

```
Nov 03,2021 17:09:17   □ Scenario Hello World!
```

```
      8ms   □ Exception: Traceback (most recent call last):
```

```
                File "hello_world.py", line 4, in <module>
```

```
                    assert 1 == 0, "1 != 0"
```

```
                AssertionError: 1 != 0
```

```
      8ms   □□ Fail Hello World!, /Hello World!, AssertionError
```

```
                Traceback (most recent call last):
```

```
                    File "hello_world.py", line 4, in <module>
```

```
                        assert 1 == 0, "1 != 0"
```

```
                    AssertionError: 1 != 0
```

Now raise let's raise some other exception like [RuntimeError](#) to see [Error](#) result.

```
from testflows.core import
Scenario
```

```
with Scenario("Hello World!"):
```

```
    raise RuntimeError("boom!")
```

```
$ python3 hello_world.py
```

```
Nov 03,2021 17:14:10   Scenario Hello World!
```

```
5ms   Exception: Traceback (most recent call last):
```

```
File "hello_world.py", line 4, in
```

```
<module>
```

```
raise RuntimeError("boom!")
```

```
RuntimeError: boom!
```

```
5ms   Error Hello World!, /Hello World!, RuntimeError
```

```
Traceback (most recent call last):
```

```
File "hello_world.py", line 4, in <module>
```

```
raise RuntimeError("boom!")
```

```
RuntimeError: boom!
```

Flexibility In Writing Tests



provides unmatched flexibility in how you can author your tests

and

this is what makes it adaptable to your testing projects at hand.

Let's see this using an example of how you could verify functionality of a simple `add(a, b)` function.

👉 Note that this is just a toy example used for demonstration purposes only.

```
from testflows.core import *
```

```
def add(a, b):
```

```
    return a + b
```

```
with Feature("check `add(a, b)`  
function"):
```

```
    with Scenario("check 2 + 2 == 4"):
```

```
        assert add(2, 2) == 4
```

```
    with Scenario("check -5 + 100 == -95"):
```

```
        assert add(-5, 100) == 95
```

```
    with Scenario("check -5 + -5 == -10"):
```

```
        assert add(-5, -5) == -10
```

Now you can put the code above anywhere you want. Let's move it into a function. For example,

```
from testflows.core import *
```

```
def add(a, b):
```

```
    return a + b
```

```
def regression():
```

```
    with Feature("check `add(a, b)` function"):
```

```
        with Scenario("check 2 + 2 == 4"):
```

```
            assert add(2, 2) == 4
```

```
        with Scenario("check -5 + 100 == -95"):
```

```
            assert add(-5, 100) == 95
```

```
with Scenario("check -5 + -5 == -10"):
```

```
    assert add(-5,-5) == -10
```

```
if main(): # short for `if __name__ == "__main__":` which is  
ugly
```

```
    regression()
```