Pytest:

- Using Pytest for testing the Python code.
- Install and use the Python IDE, then in the terminal, write command “pip install pytest” to install the package of Pytest.

```
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial> pip install pytest
```

```python
from typing import List

class ShoppingCart:
    def __init__(self, max_size: int) -> None:
        self.items: List[str] = []
        self.max_size = max_size

    def add(self, item: str):
        if self.size() == self.max_size:
            raise OverflowError("cannot add more items")
        self.items.append(item)

    def size(self) -> int:
        return len(self.items)

    def get_items(self) -> List[str]:
        return self.items

    def get_total_price(self, price_map):
        total_price = 0
        for item in self.items:
            total_price += price_map.get(item)
        return total_price
```

```python
class ItemDatabase:
    def __init__(self) -> None:
        pass

    def get(self, item: str) -> float:
        pass
```

- The file is called shopping_cart.py and another file called item_database.py are just examples of functions that are coded in Python for testing purposes for demonstrating Pytest, which check the shopping cart.

```python
from unittest.mock import Mock
from item_database import ItemDatabase
from shopping_cart import ShoppingCart
import pytest

@pytest.fixture
def cart():
    # All setup for the cart here...
    return ShoppingCart(5)

def test_can_add_item_to_cart(cart):
    cart.add("apple")
    print("I have an apple in the cart")
    assert cart.size() == 1

def test_when_item_added_then_cart_contains_item(cart):
    cart.add("apple")
    assert "apple" in cart.get_items()

def test_when_add_more_than_max_items_should_fail(cart):
    for _ in range(5):
        cart.add("apple")

    with pytest.raises(OverflowError):
        cart.add("apple")

def test_can_get_total_price(cart):
    cart.add("apple")
    cart.add("orange")
```

```
def test_can_get_total_price(cart):
    cart.add("apple")
    cart.add("orange")
    item_database = ItemDatabase()
    def mock_get_item(item: str):
        if item == "apple":
            return 1.0
        if item == "orange":
            return 2.0
    item_database.get = Mock(side_effect=mock_get_item)
    assert cart.get_total_price(item_database) == 3.0
```

- This file called test_shopping_cart.py, which has testing functions to test out the shopping cart functions, it imports the class, ShoppingCart from the file, shopping_cart.py and imports the class, ItemDatabse from the file, item_database.py. To test the function, you need to import Pytest, but there will be an error when importing it, and to fix it, by hovering the error to import all the necessary package for Pytest.

```
assert cart.size() == 1
```

- The "assert" for Pytest is used to check whether the cart.size() == 1 pass the test or fail the test, the "assert" checks whether it is true or false.
- To run the test, in the terminal, type "pytest".

```
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial> pytest
```

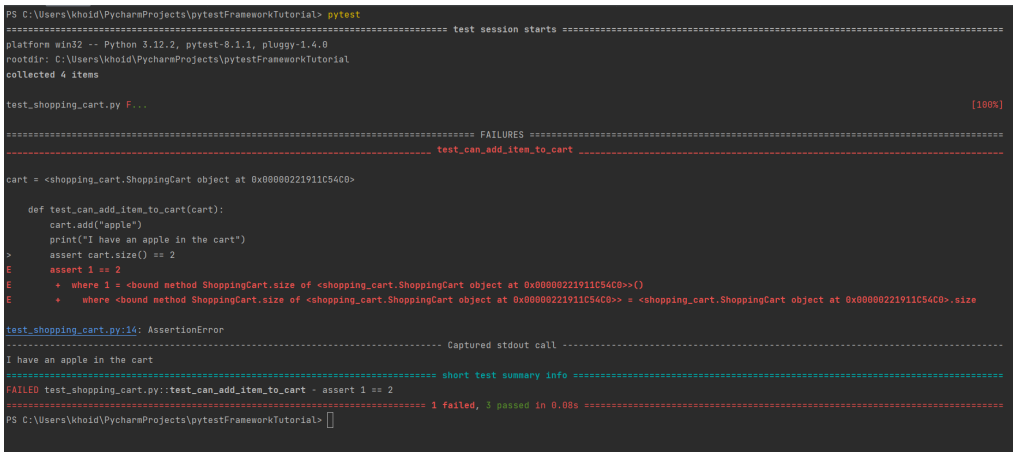
- This happens when the function passes the test.

```
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial> pytest
============================= test session starts =============================
platform win32 -- Python 3.12.2, pytest-8.1.1, pluggy-1.4.0
rootdir: C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial
collected 4 items

test_shopping_cart.py ....                                               [100%]

============================== 4 passed in 0.04s ==============================
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial>
```

- This happens when the function fails the test.

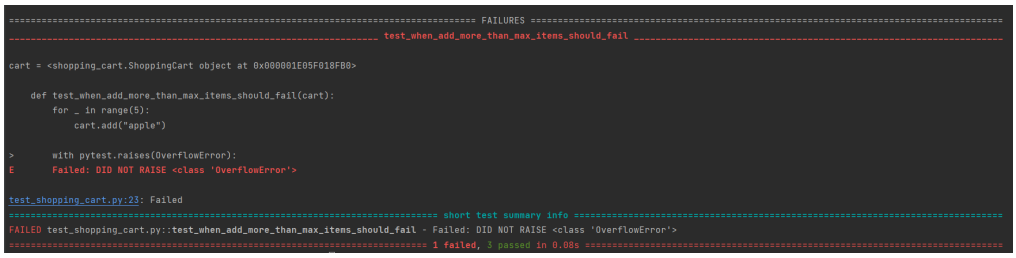



```
with pytest.raises(OverflowError):
    cart.add("apple")
```

- This line of code, “with pytest.raises(OverflowError):cart.add(“apple”)”, this is a test, where it raises the “OverflowError”, which checks the error, “OverflowError”, if it is true or false. This generally raises any error to test out whether the error is true or false.

```
raise OverflowError("cannot add more items")
```

- In the file, “shopping_cart.py”, you see “raise OverflowError(“cannot add more items”)” in the if statement, “if self.size() == self.max_size:”, which checks there is size is more than the max size, this is needed because without, it will fail the test, so you need to raise the OverflowError first in the file, “shopping_cart.py” to check the error, “OverflowError”.



- First import Mock from unittest.mock.

```
from unittest.mock import Mock
```

```
def mock_get_item(item: str):
    if item == "apple":
        return 1.0
    if item == "orange":
        return 2.0
item_database.get = Mock(side_effect=mock_get_item)
assert cart.get_total_price(item_database) == 3.0
```

- These lines of code above, shows the function “def mock_get_item(item: str)”, which mocks to get the item. Mocking is a technique in which you replace certain parts of your code with mock objects during testing. The function has 2 if conditions are if item == “apple”: return 1.0 and if item == “orange”: return 2.0. Then you have “item_database.get = Mock(side_effect=mock_get_item), which calls the function, mock_get_item whenever the Mock is called. Then “assert cart.get_total_price(item_database) == 3.0, which checks whether it passes the test or fails the test.

```
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial>  pytest test_shopping_cart.py::test_can_add_item_to_cart
```

- This above shows the command, “pytest test_shopping_cart.py::test_can_add_item_to_cart”, which tests the file, “test_shopping_cart.py” specific function, such as “test_can_add_item_to_cart”.

```
def test_can_add_item_to_cart(cart):
    cart.add("apple")
    print("I have an apple in the cart")
    assert cart.size() == 1
```

```
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial> pytest test_shopping_cart.py::test_can_add_item_to_cart -s
============================= test session starts =============================
platform win32 -- Python 3.12.2, pytest-8.1.1, pluggy-1.4.0
rootdir: C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial
collected 1 item

test_shopping_cart.py I have an apple in the cart
.

============================== 1 passed in 0.06s ==============================
PS C:\Users\khoid\PycharmProjects\pytestFrameworkTutorial>
```

- This above shows the command with the code above, “pytest test_shopping_cart.py::test_can_add_item_to_cart -s”, which tests the file, “test_shopping_cart.py” specific function, such as “test_can_add_item_to_cart” and forces to print out “print(“I have an apple in the cart”)”.