# Demonstrates defining a function with a return value

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

main()
# Demonstrates defining a function with a return value

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator1 import square

def main():
    test_square()

def test_square():
    if square(2) != 4:
        print("2 squared was not 4")
    if square(3) != 9:
        print("3 squared was not 9")

if __name__ == "__main__":
    main()
# Demonstrates defining a function with a return value

def main():
    x = int(input("What's x? \n"))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator2 import square

def main():
    test_square()

def test_square():
    assert square(2) == 4
    assert square(3) == 9

if __name__ == '__main__':
    main()
# Demonstrates defining a function with a return value

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator3 import square

def main():
    test_square()

def test_square():
    try:
        assert square(2) == 4
    except AssertionError:
        print("2 squared was not 4")
    try:
        assert square(3) == 9
    except AssertionError:
        print("3 squared was not 9")

if __name__ == "__main__":
    main()
# Demonstrates defining a function with a return value

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator4 import square

def main():
    test_square()

def test_square():
    try:
        assert square(2) == 4
    except AssertionError:
        print("2 squared was not 4")
    try:
        assert square(3) == 9
    except AssertionError:
        print("3 squared was not 9")
    try:
        assert square(-2) == 4
    except AssertionError:
        print("-2 squared was not 4")
    try:
        assert square(-3) == 9
    except AssertionError:
        print("-3 squared was not 9")
    try:
        assert square(0) == 0
    except AssertionError:
        print("0 squared was not 0")

if __name__ == "__main__":
    main()
# Tests a function with one function via pytest

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator5 import square

def test_square():
    assert square(2) == 4
    assert square(3) == 9
    assert square(-2) == 4
    assert square(-3) == 9
    assert square(0) == 0
# Tests a function with multiple functions via pytest

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
from calculator6 import square

def test_positive():
    assert square(1) == 1
    assert square(2) == 4
    assert square(3) == 9

def test_negative():
    assert square(-1) == 1
    assert square(-2) == 4
    assert square(-3) == 9

def test_zero():
    assert square(0) == 0
# Tests a function with multiple functions via pytest

def main():
    x = int(input("What's x? "))
    print("x squared is", square(x))

def square(n):
    return n * n

if __name__ == "__main__":
    main()
import pytest

from calculator import square

def test_positive():
    assert square(2) == 4
    assert square(3) == 9

def test_negative():
    assert square(-2) == 4
    assert square(-3) == 9

def test_zero():
    assert square(0) == 0

def test_str():
    with pytest.raises(TypeError):
        square("cat")
# Function to be tested

def main():
    name = input("What's your name? ")
    hello(name)

def hello(to="world"):
    print("hello,", to)

if __name__ == "__main__":
    main()
# Has function return a str instead

def main():
    name = input("What's your name? ")
    print(hello(name))

def hello(to="world"):
    return f"hello, {to}"

if __name__ == "__main__":
    main()
from hello1 import hello

def test_default():
    assert hello() == "hello, world"

def test_argument():
    assert hello("David") == "hello, David"
from hello1 import hello

def test_default():
    assert hello() == "hello, world"

def test_argument():
    for name in ["Hermione", "Harry", "Ron"]:  
        assert hello(name) == f"hello, {name}"
from hello1 import hello

def test_default():
    assert hello() == "hello, world"

def test_argument():
    assert hello("David") == "hello, David"