Unit 10 Activities.

Activity 1

Run the following code using pylint and identify the errors.

```
def factorial (x)
        if x == 1:
                return 1

        else:
                return (x * factorial(x-1))

num = 3
print("The factorial of", num, "is", factorial(num))
```

```
************* Module main
main.py:1:18: E0001: Parsing failed: 'expected ':' (main, line 1)' (syntax-error)

************* Module main
main.py:6:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
main.py:9:0: C0304: Final newline missing (missing-final-newline)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:1:0: C0116: Missing function or method docstring (missing-function-docstring)
main.py:2:4: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code
inside it (no-else-return)
main.py:8:0: C0103: Constant name "num" doesn't conform to UPPER_CASE naming style
(invalid-name)

------------------------------------------------------------------
Your code has been rated at 0.00/10 (previous run: 0.00/10, +0.00)

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 0.00/10, +10.00)
```

```
"""
This module provides a recursive implementation of the factorial function.

Functions:
     factorial(x): Recursively computes the factorial of a given positive integer.
"""

def factorial(x):
     """
     Calculate the factorial of a given positive integer recursively.

     The factorial of a number is defined as the product of all positive integers
     from 1 up to the number itself. For example, factorial(3) = 3 * 2 * 1 = 6.

     Parameters:
          x (int): The positive integer for which the factorial is to be calculated.

     Returns:
          int: The factorial of the input integer.

     Example:
          factorial(3)
          # Output: 6
     """
     if x == 1:
          return 1
     return x * factorial(x - 1)

NUM = 3
print("The factorial of", NUM, "is", factorial(NUM))
```

Activity 2

In ‘Packaging & Testing’ (unit 9), we examined the use of documentation to support code developments. Add appropriate commenting and documentation for the code below.

The comments on the first activity felt too long for simple functions, simplified for the calculator.

```python
"""
This module provides a simple calculator program for basic arithmetic operations.
"""

def add(x, y):
    """Return the sum of x and y."""
    return x + y

def subtract(x, y):
    """Return the difference of x and y."""
    return x - y

def multiply(x, y):
    """Return the product of x and y."""
    return x * y

def divide(x, y):
    """Return the result of dividing x by y. Raises ZeroDivisionError if y is 0."""
    if y == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return x / y

print("Select operation.")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

while True:
    choice = input("Enter choice (1/2/3/4): ")

    if choice in ('1', '2', '3', '4'):
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))
        elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))
        elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))
        elif choice == '4':
            try:
                print(num1, "/", num2, "=", divide(num1, num2))
            except ZeroDivisionError as e:
                print("Error:", e)

        next_calculation = input("Let's do the next calculation? (yes/no): ")
        if next_calculation.lower() == "no":
            break
    else:
        print("Invalid Input")
```

Activity 3

Read the article by Rani et al. (2021). What impact does this article have on the way in which you have commented the code in the task above?

Rani, P. *et al.* (2021) “Do Comments follow Commenting Conventions? A Case Study in Java and Python.” Available at: https://arxiv.org/pdf/2108.10766.

Rani et al suggest that:

- All comments should be content driven that explain the purpose, behaviour and expected inputs outputs.
- Be consistent in writing style, using proper grammar, capitalisation and punctuality.
- Follow structure related conventions with clear summarise.
- Avoid redundant comments.
- Include examples.

This is an example of how the above code comments could be adjusted to conform with the suggestions. However, this does add a lot of comments to a simple function.

```python
def add(x, y):
   """
   Return the sum of x and y.

   Parameters:
   x (float): The first number.
   y (float): The second number.

   Returns:
   float: The sum of x and y.

   Example:
   >>> add(2, 3)
   5
   """
   return x + y
```

Activity 4

Integrate unit tests into the code in Activity 2 to test operation of the methods.

```python
"""
This module provides a simple calculator program for performing basic arithmetic
operations such as addition, subtraction, multiplication, and division.
"""

def add(x, y):
   """Return the sum of x and y."""
   return x + y

def subtract(x, y):
   """Return the difference of x and y."""
   return x - y

def multiply(x, y):
   """Return the product of x and y."""
   return x * y

def divide(x, y):
   """Return the result of dividing x by y. Raises ZeroDivisionError if y is 0."""
   if y == 0:
      raise ZeroDivisionError("Cannot divide by zero.")
   return x / y

def calculator_program():
   print("Select operation.")
   print("1. Add")
   print("2. Subtract")
   print("3. Multiply")
   print("4. Divide")

   while True:
      choice = input("Enter choice (1/2/3/4): ")

      if choice in ('1', '2', '3', '4'):
         num1 = float(input("Enter first number: "))
         num2 = float(input("Enter second number: "))

         if choice == '1':
            print(num1, "+", num2, "=", add(num1, num2))
         elif choice == '2':
            print(num1, "-", num2, "=", subtract(num1, num2))
         elif choice == '3':
            print(num1, "*", num2, "=", multiply(num1, num2))
         elif choice == '4':
            try:
               print(num1, "/", num2, "=", divide(num1, num2))
            except ZeroDivisionError as e:
               print("Error:", e)

         next_calculation = input("Let's do the next calculation? (yes/no): ")
         if next_calculation.lower() == "no":
            break
      else:
         print("Invalid Input")
```

```python
import unittest
from calculator import add, subtract, multiply, divide

class TestCalculator(unittest.TestCase):

  def test_add(self):
    self.assertEqual(add(2, 3), 5)
    self.assertEqual(add(-1, 1), 0)
    self.assertEqual(add(0, 0), 0)

  def test_subtract(self):
    self.assertEqual(subtract(5, 3), 2)
    self.assertEqual(subtract(3, 5), -2)
    self.assertEqual(subtract(0, 0), 0)

  def test_multiply(self):
    self.assertEqual(multiply(2, 3), 6)
    self.assertEqual(multiply(-1, 1), -1)
    self.assertEqual(multiply(0, 5), 0)

  def test_divide(self):
    self.assertEqual(divide(6, 3), 2.0)
    self.assertEqual(divide(-6, 3), -2.0)
    self.assertRaises(ZeroDivisionError, divide, 1, 0)

if __name__ == "__main__":
  unittest.main()
```

```
~/workspace$ python calculator_test.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```