**Lab Exercise 14- Complex Flow test using PyTest in Metaflow**

This will give you a more in-depth view of how to handle more complex data processing and test it effectively.

**1. Complex Metaflow Flow**

We will create a flow that:

1. Starts with a list of numbers.
2. Squares each number.
3. Calculates the sum of the squared numbers.
4. Finds the maximum squared number.

**Code for complex\_flow.py**

from metaflow import FlowSpec, step

class ComplexFlow(FlowSpec):

@step

def start(self):

self.numbers = [1, 2, 3, 4, 5]

self.next(self.square\_numbers)

@step

def square\_numbers(self):

self.squared\_numbers = [x \*\* 2 for x in self.numbers]

self.next(self.sum\_squares)

@step

def sum\_squares(self):

self.total\_sum = sum(self.squared\_numbers)

self.next(self.find\_max)

@step

def find\_max(self):

self.max\_value = max(self.squared\_numbers)

self.next(self.end)

@step

def end(self):

print(f"Numbers: {self.numbers}")

print(f"Squared Numbers: {self.squared\_numbers}")

print(f"Total Sum of Squares: {self.total\_sum}")

print(f"Maximum Squared Value: {self.max\_value}")

if \_\_name\_\_ == '\_\_main\_\_':

ComplexFlow()

**2. Write Unit Tests Using PyTest**

Now we’ll create a test file to validate each step of this flow, checking the intermediate results and the final outcomes.

**Code for test\_complex\_flow.py**

import os

from metaflow import Flow

import subprocess

def test\_flow():

# Command to run the flow

cmd = ['python', 'complex\_flow.py', 'run', '--run-id-file', 'test\_id']

subprocess.check\_call(cmd)

# Read the run ID from the file

with open('test\_id') as f:

run\_id = f.read().strip() # Use .strip() to remove any extra whitespace

# Load the flow run and check the outputs

run = Flow('ComplexFlow')[run\_id]

# Assertions to validate the outputs

assert run.numbers == [1, 2, 3, 4, 5]

assert run.squared\_numbers == [1, 4, 9, 16, 25]

assert run.total\_sum == 55

assert run.max\_value == 25

**3. Running the Tests**

Ensure that both complex\_flow.py and test\_complex\_flow.py are in the same directory. To run the tests, execute:

pytest test\_complex\_flow.py

**Explanation of the Flow and Tests**

1. **Flow Steps**:
   * **start**: Initializes a list of numbers.
   * **square\_numbers**: Squares each number in the list.
   * **sum\_squares**: Computes the total sum of the squared numbers.
   * **find\_max**: Finds the maximum value among the squared numbers.
   * **end**: Prints the results.
2. **Testing Logic**:
   * The test uses subprocess to run the complex\_flow.py file.
   * It reads the generated run ID from test\_id, which is created when the flow is executed.
   * The test verifies:
     + The original numbers are correct.
     + The squared numbers are calculated accurately.
     + The total sum of the squares matches the expected value.
     + The maximum squared value is as expected.