Software Design and Development Coursework

Part 1: Module coupling and cohesion

- ⇨ The purpose of the program you provided is to define a unit test for the mean function from the snakestats module and run the test. The program uses the unittest module from the Python standard library to define and run the unit test.

  The unittest module is a unit testing framework that is part of the Python standard library. It provides tools for defining and running tests, as well as for reporting the results of the tests. The unittest module is commonly used to test the functionality and correctness of Python code, and it is a useful tool for improving the quality and reliability of software.

- ⇨ In the provided code, there are two types of module coupling happening:

1) The first type of coupling is between the TestsForSnakeStats test class and the snakestats module. This coupling is illustrated in the following extract of code:

```
import snakestats


class TestsForSnakeStats(unittest.TestCase):
    def test_mean(self):
        res= snakestats.mean([1,2,3])
```

The TestsForSnakeStats class imports the snakestats module and calls the mean function from that module inside the test_mean test method. This means that the TestsForSnakeStats class is coupled to the snakestats module, and any changes to the snakestats module may affect the behavior of the TestsForSnakeStats class.

This type of coupling is happening because the TestsForSnakeStats class depends on the mean function from the snakestats module to perform its testing duties. Without access to the mean function, the TestsForSnakeStats class would not be able to run the test and check the correctness of the mean function.

2) The second type of coupling is between the unittest.main function and the TestsForSnakeStats test class. This coupling is illustrated in the following extract of code:

unittest.main(argv=['ignored', '-v'], exit=False)

The unittest.main function takes the TestsForSnakeStats class as an argument and runs the tests defined in that class. This means that the unittest.main function is coupled to the TestsForSnakeStats class, and any changes to the TestsForSnakeStats class may affect the behavior of the unittest.main function.

This type of coupling is happening because the unittest.main function needs to know which tests to run, and it obtains this information from the TestsForSnakeStats class. Without access to the tests defined in the TestsForSnakeStats class, the unittest.main function would not be able to run any tests and would not be able to produce any output.

- ⇨ In the provided code, there are two types of module cohesion happening:

1) The first type of cohesion is within the TestsForSnakeStats test class. This cohesion is illustrated in the following extract of code:

```
class TestsForSnakeStats(unittest.TestCase):
    def test_mean(self):
        res= snakestats.mean([1,2,3])
```

The TestsForSnakeStats class defines a single test method called test_mean that calls the mean function from the snakestats module and stores the result in the res variable. All of the elements within the TestsForSnakeStats class (the import statement, the class definition, and the test method) work together towards the common goal of testing the mean function from the snakestats module.

This type of cohesion is happening because the TestsForSnakeStats class is focused on a single, well-defined purpose: testing the mean function. All of the elements within the class contribute to this purpose by defining a test method that calls the mean function and performs any necessary checks or assertions.

2) The second type of cohesion is within the unittest.main function. This cohesion is illustrated in the following extract of code:

unittest.main(argv=['ignored', '-v'], exit=False)

The unittest.main function takes two optional arguments: argv and exit. These arguments control the behavior of the test runner and allow the user to specify command-line arguments and configure the way the tests are run. All of the elements within the unittest.main function (the function definition and the two arguments) work together towards the common goal of running the tests and producing the output.

This type of cohesion is happening because the unittest.main function is focused on a single, well-defined purpose: running the tests and producing the output. All of the elements within the function contribute to this purpose by providing the necessary configuration and control over the test runner.