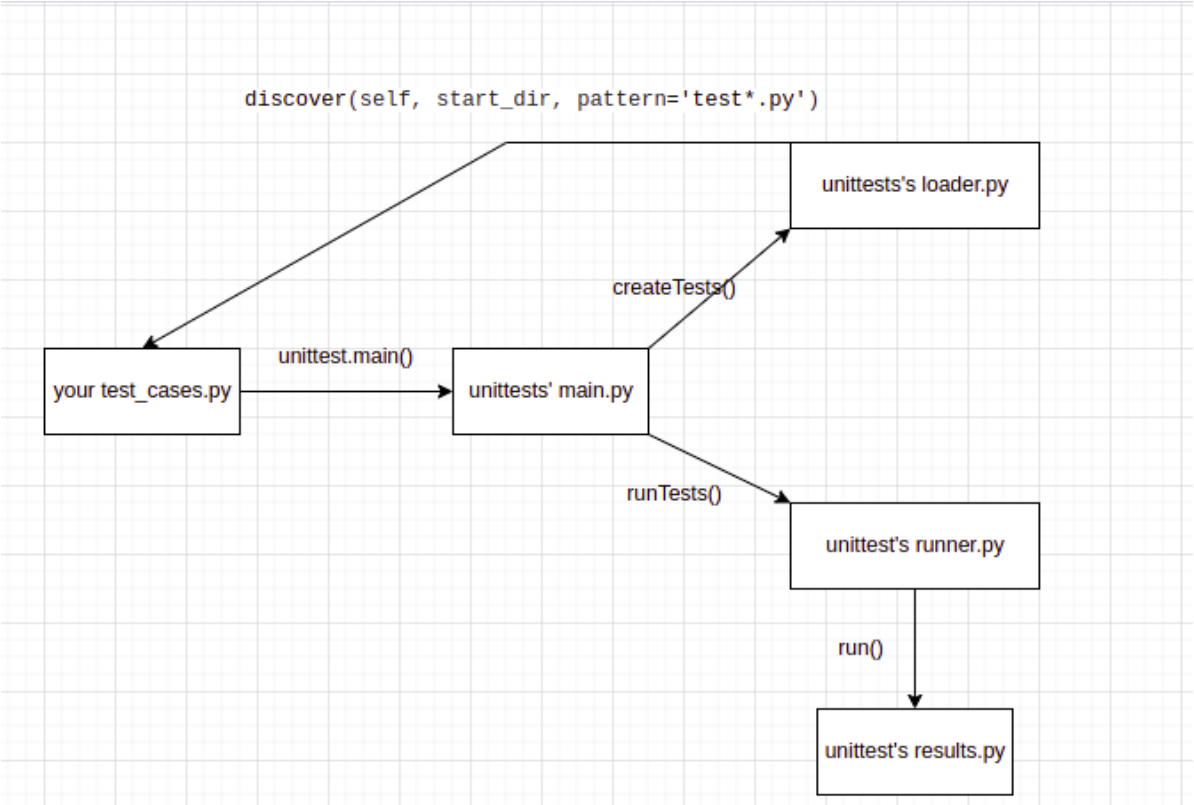
Writeup on the working of unittest

```python
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

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

test_cases.py



Flow of control in the unittest framework

Brief description of the key files

1. test_cases.py: this contains the test functions you want to run using the unittest framework. It imports the unittest module and calls its main function to transfer control to the framework.

2. Files of the unittest framework:

    a. loader.py - this contains functions for discovering the tests based on the arguments supplied to the framework
    b. runner.py - this contains functions for running the tests discovered by the loader module.
    c. results.py - This module compiles the results of all the tests that were run by the runner module.
    d. main.py - this file contains the main object which is invoked by the test_cases.py file, and it is calls the methods from the loader and runner modules.

The flow of control

Your test_cases.py file imports the unittest module. This runs the __init__.py code in the unittest framework. The __init__.py in turn contains the statement

```
from .main import TestProgram, main
```

which runs the code in the main.py file. The following statement is run as a result:

```
main = TestProgram
```

i.e. our main object is initialized to an object of the class TestProgram. This calls the __init__ function of the TestProgram class, which makes the following initialisations (and some others too)

```
self.testRunner = testRunner
self.testLoader = testLoader
```

Then it calls the self.parseArgs() function, which in turn calls the self.createTests() function, which calls functions from the loader.py module. Primarily, the discover() function of the loader.py module does the majority of the work, taking in our input directory and finding all the test methods in that directory.

After loading all our tests into the self.test object, we call the self.runTests() method, which in turn calls methods from the runner.py module to run all of the tests in the self.tests object. The run() function in the runner.py inturn calls methods from the results.py module to output the results after the run.