./

Learning Report – Test driven life cycle.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PS**  **Number** | **Name** | **Mail ID** | **Contribution** | **To be**  **Approved By** | **Remarks/Revision Details** |
| 99002672 | Amilineni sai vinutha | amilinei.saivinutha@ltts.com | What is Test Driven Development?  Where Test Driven Development is used?  Application of TDD on Mini Project | Pagala Prithvi Sekhar |  |
| 99002610 | Gagan N | gagan.n@ltts.com | how is Test Development Life cycle used?  How is Test Driven Development used? |  |
| 99002646 | Asha N | asha.n@ltts.com | What is Test Development Life cycle?  Why is Test Development Life cycle used?  Application of TDLC on Mini Project |  |
| 99002659 | Aditya b Kulkarni | aditya.bkulkarni@ltts.com | Where is Test Development Life cycle used?  How is Test Development Life cycle used?  Application of TDLC on Mini Project |  |
| 99002660 | Ankita D. Joil | ankita.dipakjoil@ltts.com | Why is Test Driven Development used?  Application of TDD on Mini Project  Debugging and useful plugin/extensions |  |
| 99002570 | Geethanjali B | Geethanjali.b@ltts.com | When is Test Driven Development used?  Application of TDD on Mini Project  Differentiating UT, IT and AT |  |
| 99002679 | Grace Juliet Jennifer | Sgrace.julietjennifer@ltts.com | When is Test Development Life cycle used?  Application of TDLC on Mini Project |  |
| 99002625 | Lakshmisha Gowda B | lakshmisha.gowdab@ltts.com | How is Test Driven Development used?  Application of TDD on Mini Project |  |
|  |  |  |  |  |  |

Table of Contents

[Team activity 1: TDLC 5](#_Toc55303438)

[Test Driven life cycle: 5](#_Toc55303439)

[Introduction: 5](#_Toc55303440)

[**1.What is TDLC?** 5](#_Toc55303441)

[**2.Why TDLC?** 6](#_Toc55303442)

[**3.Where to use TDLC?** 6](#_Toc55303443)

[**4.When to use TDLC?** 6](#_Toc55303444)

[**5. How do we use TDLC?** 7](#_Toc55303445)

[References: 8](#_Toc55303446)

[Team activity 2: TDD 9](#_Toc55303447)

[Test Driven Development: 9](#_Toc55303448)

[**1.What is TDD?** 9](#_Toc55303449)

[**2.Why TDD?** 10](#_Toc55303450)

[**3.Where to use TDD?** 10](#_Toc55303451)

[**4.When to do TDD?** 10](#_Toc55303452)

[**5.When not to do TDD?** 10](#_Toc55303453)

[**6.How to do TDD?** 11](#_Toc55303454)

[Team activity 3: Application of TDLC on mini project 12](#_Toc55303455)

[**1.Test cases(passed)** 12](#_Toc55303456)

[**2.Run Test cases** 13](#_Toc55303457)

[**3. New Test cases(Failed)** 13](#_Toc55303458)

[**4.Re-Run** 14](#_Toc55303459)

[TASK-4: Application of mini project on TDD 15](#_Toc55303460)

[**1.Test cases (Failed)** 15](#_Toc55303461)

[**2. Run Test cases** 16](#_Toc55303462)

[**3.Write code** 16](#_Toc55303463)

[**4.Refactor** 17](#_Toc55303464)

[Task5: C/C++ Mini Project 18](#_Toc55303465)

[**1. Debugging** 18](#_Toc55303466)

[**2. AUTO COMPLETE:** 20](#_Toc55303467)

[**3. SYMBOL VIEWER:** 20](#_Toc55303468)

[**4. BUILD** 21](#_Toc55303469)

[**5. CPP CHECK** 22](#_Toc55303470)

[**6. VALGRIND** 22](#_Toc55303471)

[REFERENCES:- 23](#_Toc55303472)

[PYTHON PROJECT 25](#_Toc55303473)

[**1.PYLINT** 25](#_Toc55303474)

[**2. UNIT TESTING IN PYTHON** 26](#_Toc55303475)

[REFERENCES:- 26](#_Toc55303476)

# Team activity 1: TDLC

## **Test Driven life cycle:**

## **Introduction:**

TDLC is a sequence of different activities performed during the software testing process. TDLC is mainly related to software testing. It focuses only on testing the software. TDLC involves only five phases or steps and less number of members (testers) are needed. Testing team makes the plans and designs. Goal of TDLC is to complete successful testing of software. It helps in making the software defects free. TDLC phases are performed after SDLC phases.

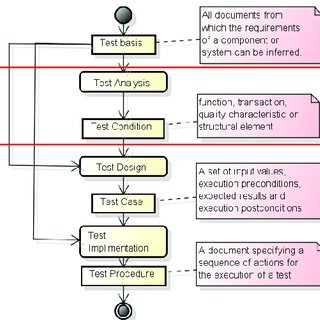


Figure1.Different Phases of TDLC

### **1.What is TDLC?**

* TDLC is life cycle of testing. If testing is not planned and not done in a particular process it may not only take time to be completed but also will increase the cost So TDLC is consisting of phases
* Test plan-plan the test and test cases properly.
* Identify test scenarios- when and where will you be testing; scenarios are the situations in which testing need to be done
* Preparing test cases- now we prepare test cases, those will contain all inputs and desired outputs and which input does not give desired output will also be documented in this phase.
* Executing test cases- we will execute test cases which will be made in the previous stages/
* Identify defects.
* Tracking defects.
* Close the test.

### **2.Why TDLC?**

* STLC provides a step-by-step process to ensure quality software.
* In the early stage of STLC, while the software or the product is developing, the tester can analyse and define the scope of testing, entry and exit criteria and the Test Cases. It helps to reduce the test cycle time along with better quality.

### **3.Where to use TDLC?**

* TDLC is used for analysing the test plan, identify test scenarios preparing test cases executing test cases identify defects reporting defects tracking defects close.



Figure2. Test Driven Life Cycle Methodology

### **4.When to use TDLC?**

* TDLC is used in the testing phase of the product which includes unit testing, Integration testing and Acceptance testing which helps in detecting the bugs and fixing them and also analyze whether the designed product has met the customer requirement as expected.

### **5. How do we use TDLC?**

* In TDLC, the QA team analyze requirement from documents like functional and non-functional documents and create System Test Plan.
* In TDLC, the test analyst creates the Integration Test Plan.
* The testing team prepares the test environment and executes them.
* Testers execute regression suits, usually automation scripts to check maintenance code deployed.

## References:

1. <https://www.allinterview.com/showanswers/63095/1-what-is-sdlc-and-tdlc.html>
2. <https://www.guru99.com/sdlc-vs-stlc.html>
3. <https://www.tutorialspoint.com/stlc/stlc_sdlc_comparison.htm>
4. <https://www.learnpick.in/questions/details/9400/what-is-tdlc>

## **Team activity 2: TDD**

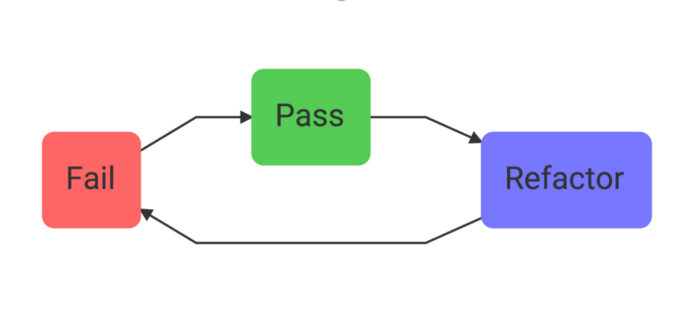
## **Test Driven Development:**

### **1.What is TDD?**

Test-driven development (TDD) is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfil that test and [refactoring](http://www.agiledata.org/essays/databaseRefactoring.html).

What is the primary goal of TDD?

One view is the goal of TDD is specification and not validation. In other words, it’s one way to think through your requirements or design before your write your functional code (implying that TDD is both an important [agile requirements](http://www.agilemodeling.com/essays/agileRequirements.htm) and [agile design](http://www.agilemodeling.com/essays/agileDesign.htm) technique). Another view is that TDD is a programming technique.



* Before you write implementation code, write some code that proves that the implementation works or fails. Watch the test fail before moving to the next step (this is how we know that a passing test is not a false positive — how we test our tests).
* Write the implementation code and watch the test pass.
* Refactor if needed. You should feel confident refactoring your code now that you have a test to tell you if you’ve broken something.

### **2.Why TDD?**

* A significant advantage of TDD is that it enables you to take small steps when writing software.
* For example, assume you add some new functional code, compile, and test it. Chances are pretty good that your tests will be broken by defects that exist in the new code. It is much easier to find, and then fix, those defects if you've written two new lines of code than two thousand.
* The implication is that the faster your compiler and regression test suite, the more attractive it is to proceed in smaller and smaller steps.
* TDD is for improved test coverage which leads fewerbugs  in production.

### **3.Where to use TDD?**

* Test-driven development (TDD) is a development technique where you must first write a test that fails before you write new functional code.
* TDD is being quickly adopted by agile software developers for development of application source code and is even being adopted by Agile DBAs for database development.

### **4.When to do TDD?**

* TDD lends itself really well to when you have a pure logic function that you need to write.
* When the work you need to do has a clearly defined set of expected inputs and outputs, it's a great signal that you should use TDD to build out your tests and code.
* Anytime you recognize that your code needs to make decisions or calculations, you should break it out into its own method and use TDD to write that method.
* When coordinating different modules or integrating with systems you don't control, TDD can help find the correct seams between layers. To do this, start with the entry point to the system and try to integrate with all of the dependencies.

### **5.When not to do TDD?**

* If you have an existing codebase that doesn't have any unit tests at all, it can be difficult to start doing TDD.
* It also might not be worth it to use TDD on something that's already working. In these situations, when you make changes, you should definitely start by creating some unit tests around the existing system.
* As you build a suite of tests, you can look for opportunities to start following TDD. Don't try to rewrite or start from scratch something that's already working just so you can use TDD.

### **6.How to do TDD?**

The following sequence shows the steps to do test driven development

1. Add a test. In test-driven development, each new feature begins with writing a test. Write a test that defines a function or improvements of a function, which should be very succinct. To write a test, the developer must clearly understand the feature's specification and requirements.
2. Run all tests and see if the new test fails. This validates that the test harness is working correctly, shows that the new test does not pass without requiring new code because the required behaviour already exists, and it rules out the possibility that the new test is flawed and will always pass. The new test should fail for the expected reason. This step increases the developer's confidence in the new test.
3. Write the code. This step is to write some code that causes the test to pass. The new code written at this stage is not perfect and may, for example, pass the test in an inelegant way. That is acceptable because it will be improved and honed in Step.
4. Run tests. If all test cases now pass, the programmer can be confident that the new code meets the test requirements and does not break or degrade any existing features. If they do not, the new code must be adjusted until they do.
5. Refactor code. The growing code base must be cleaned up regularly during test-driven development. New code can be moved from where it was convenient for passing a test to where it more logically belongs. Duplication must be removed. Object, class, module, variable and method names should clearly represent their current purpose and use, as extra functionality is added. By continually re-running the test cases throughout each refactoring phase, the developer can be confident that process is not altering any existing functionality.
6. Repeat Starting with another new test, the cycle is then repeated to push forward the functionality. If new code does not rapidly satisfy a new test, or other tests fail unexpectedly, the programmer should undo or revert in preference to excessive debugging. Continuous integration helps by providing revertible checkpoints.

# Team activity 3: Application of TDLC on mini project

### **1.Test cases(passed)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DESCRIPTION | EXPECTED INPUT | EXPECTED  OUTPUT | ACTUAL OUTPUT |
| HL\_01 | Check the minimum version for python | Version | Version=3.5 | Version=3.5 |

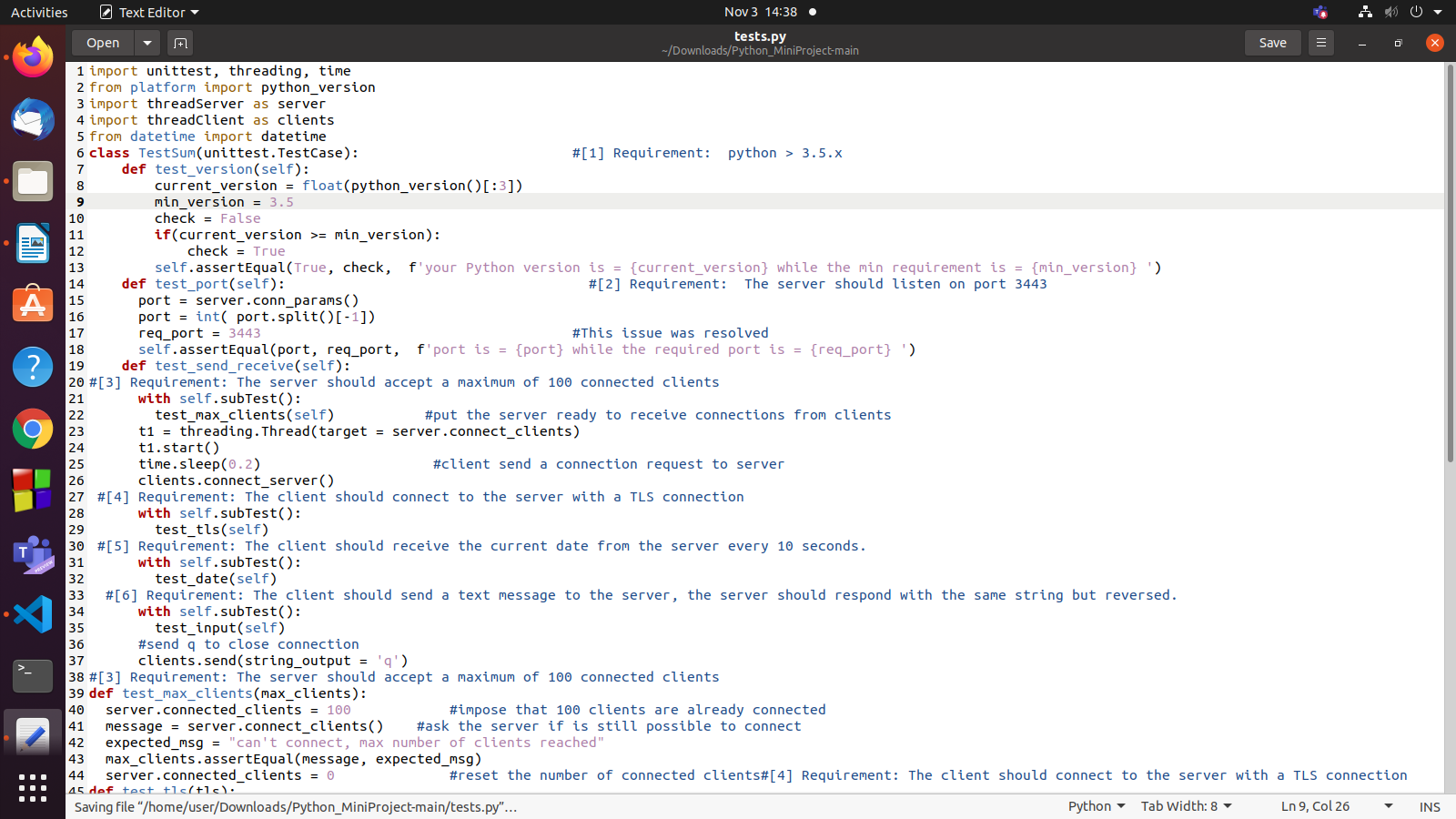


Figure3.Test cases are written.

### **2.Run Test cases**

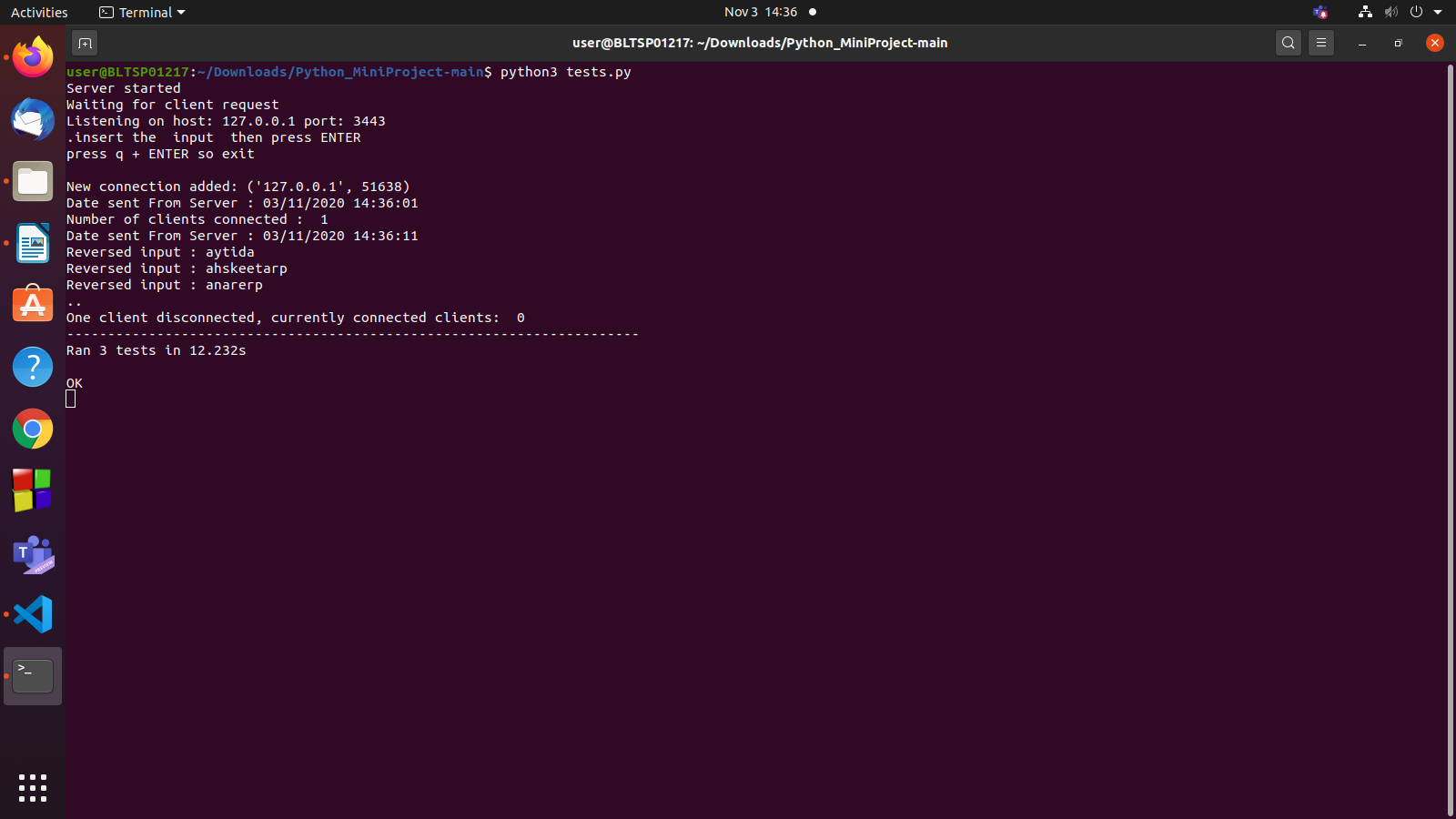


Figure 4.All test cases are passed.

### **3. New Test cases(Failed)**

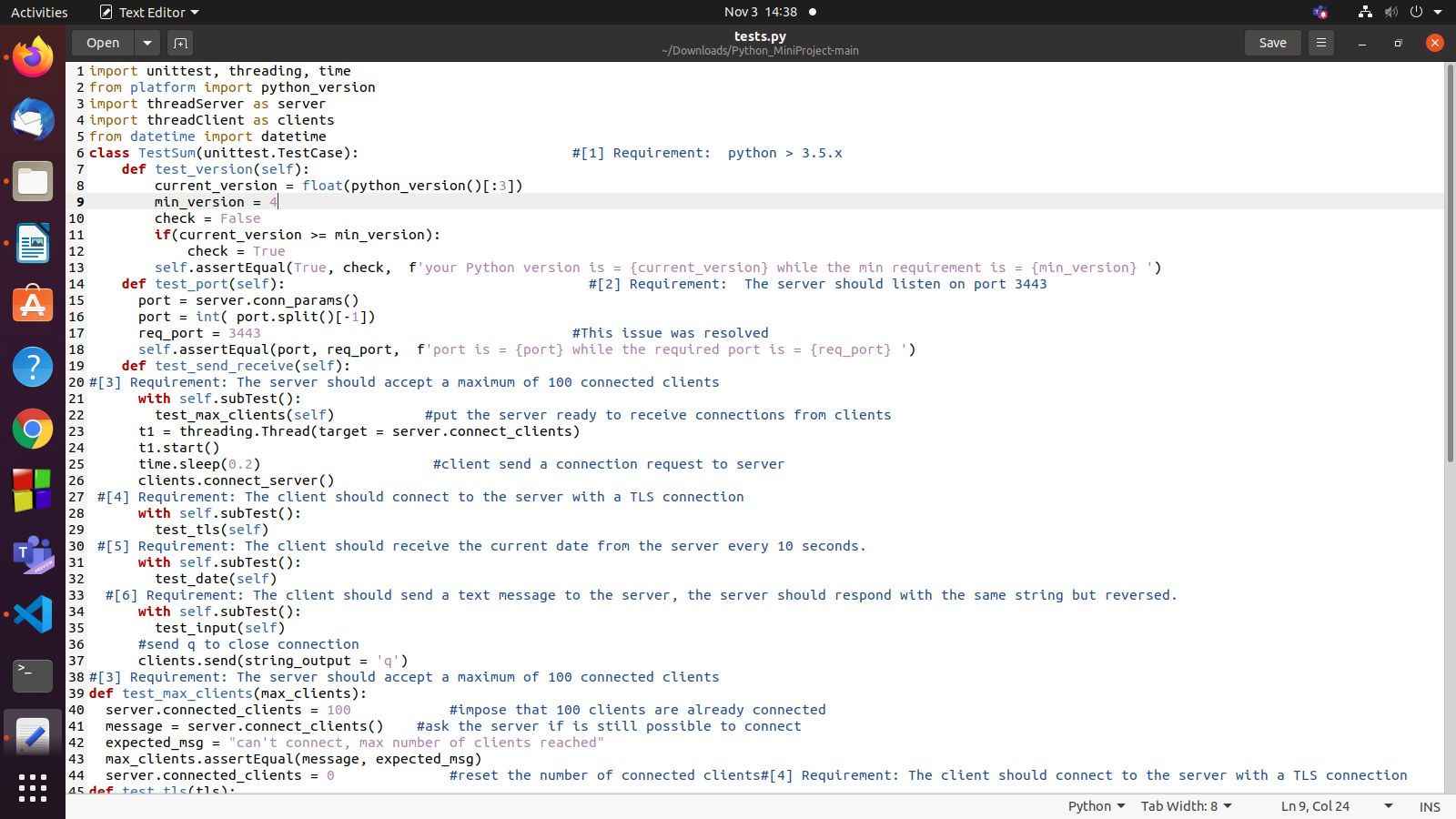


Figure 3.Added failed test case.

### **4.Re-Run**

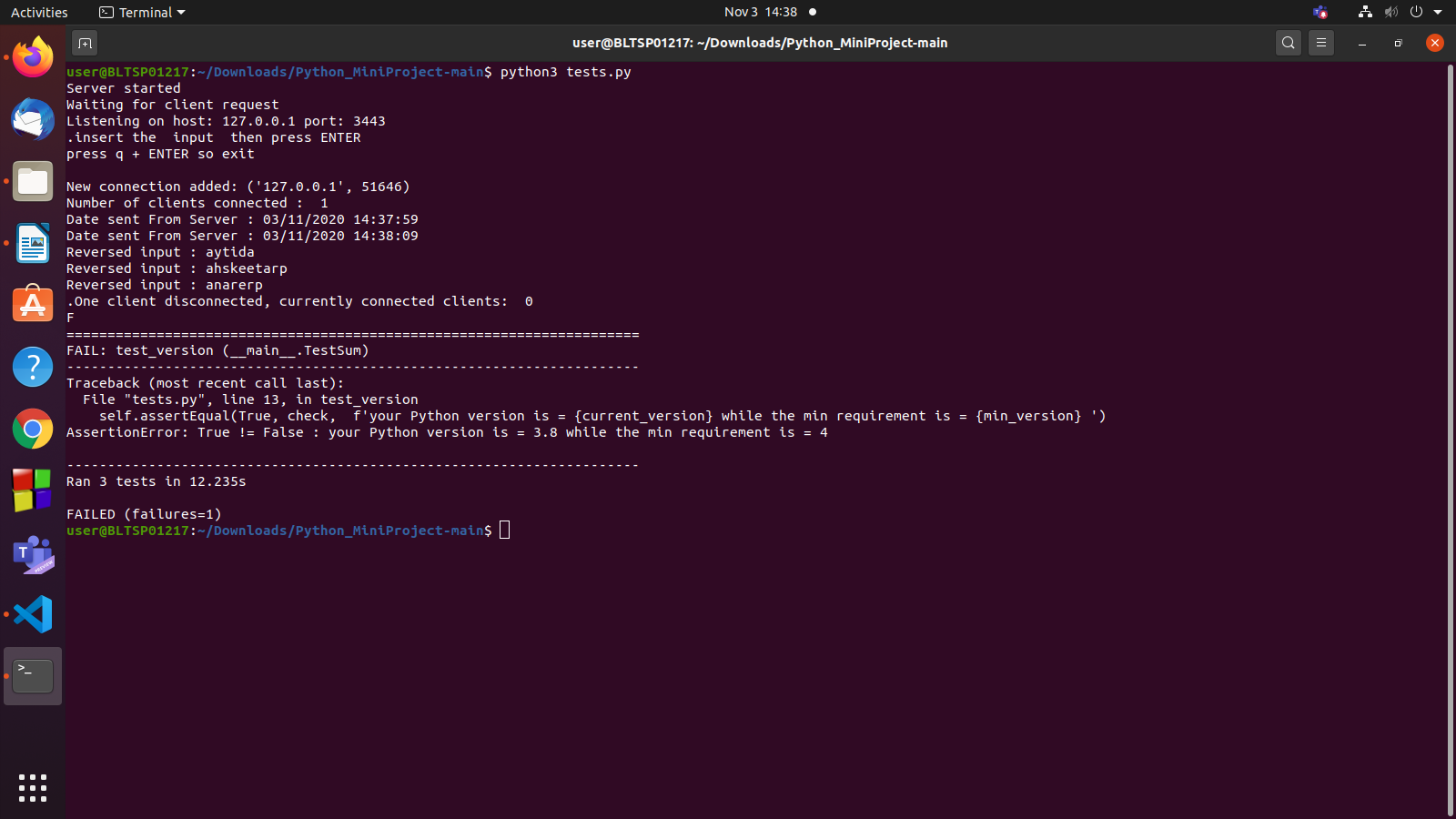


Figure 4.Re-Run.

# TASK-4: Application of mini project on TDD

### **1.Test cases (Failed)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | DESCRIPTION | EXPECTED INPUT | EXPECTED  OUTPUT | ACTUAL OUTPUT |
| HL\_01 | Check the minimum version for python | Version | Version=3.5 | Version=4 (Failed) |

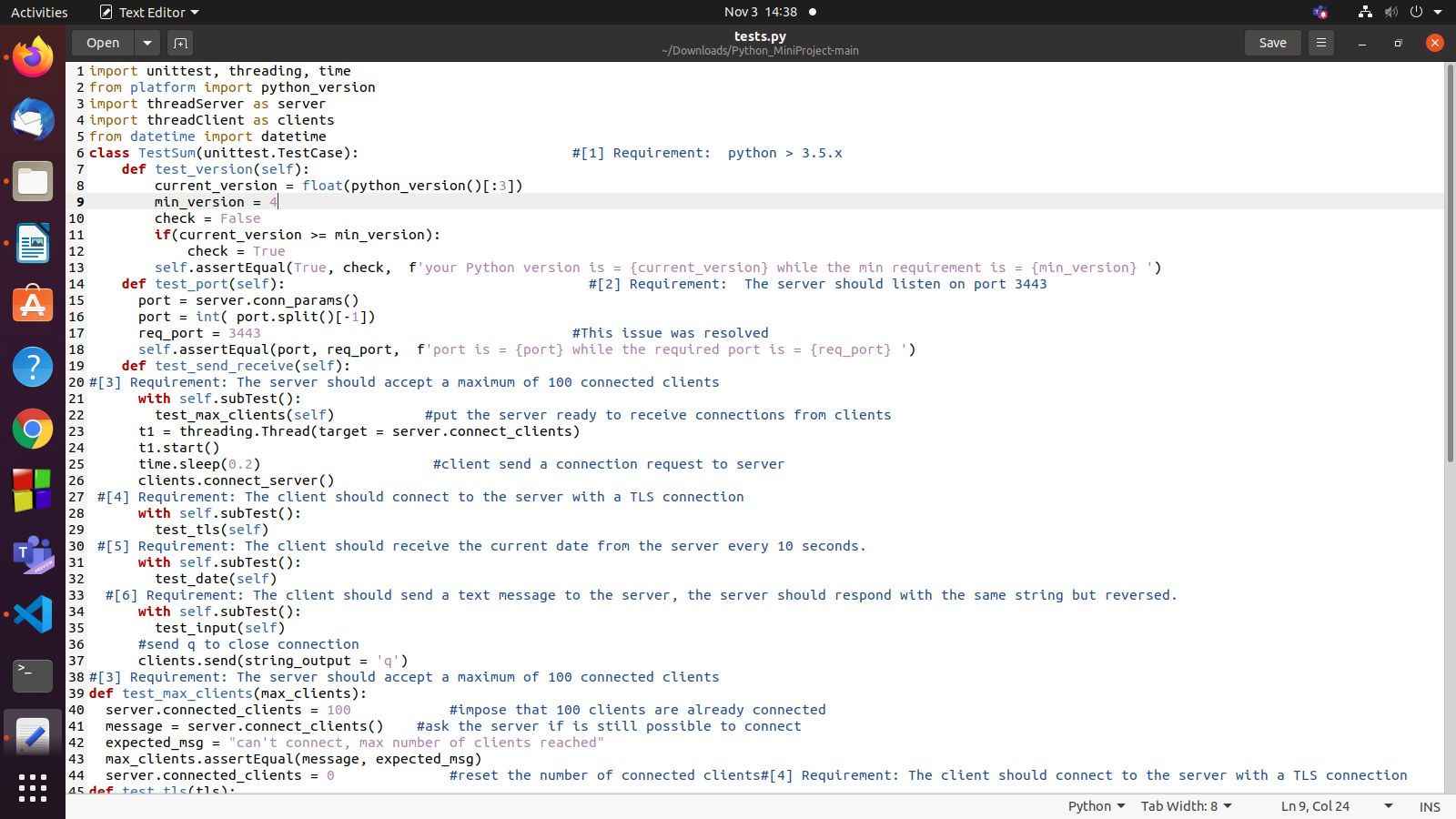


Figure 5. Test case failed for mini version.

### **2. Run Test cases**

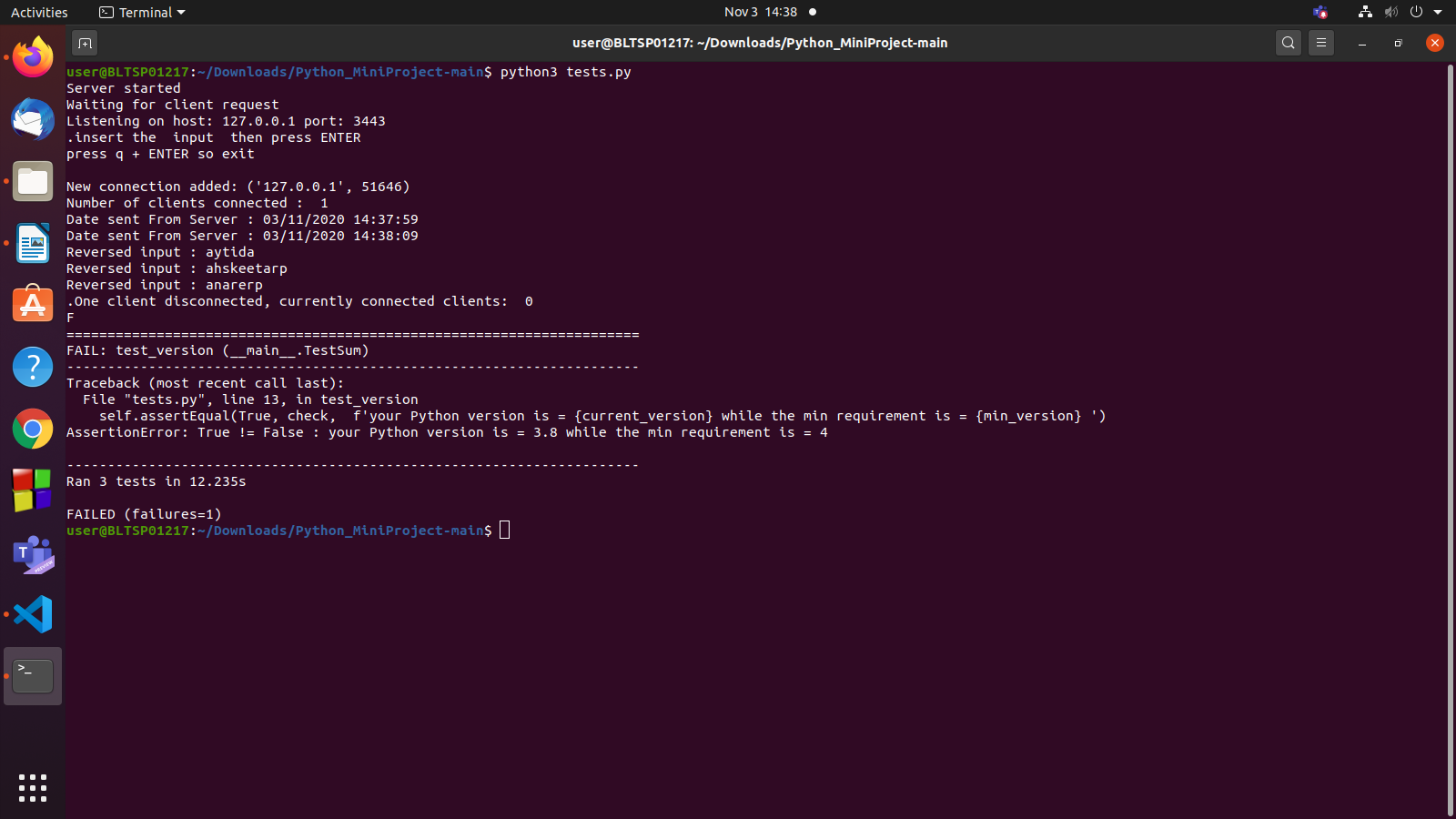


Figure 6. Doesn’t satisfy test cases of mini version.

### **3.Write code**

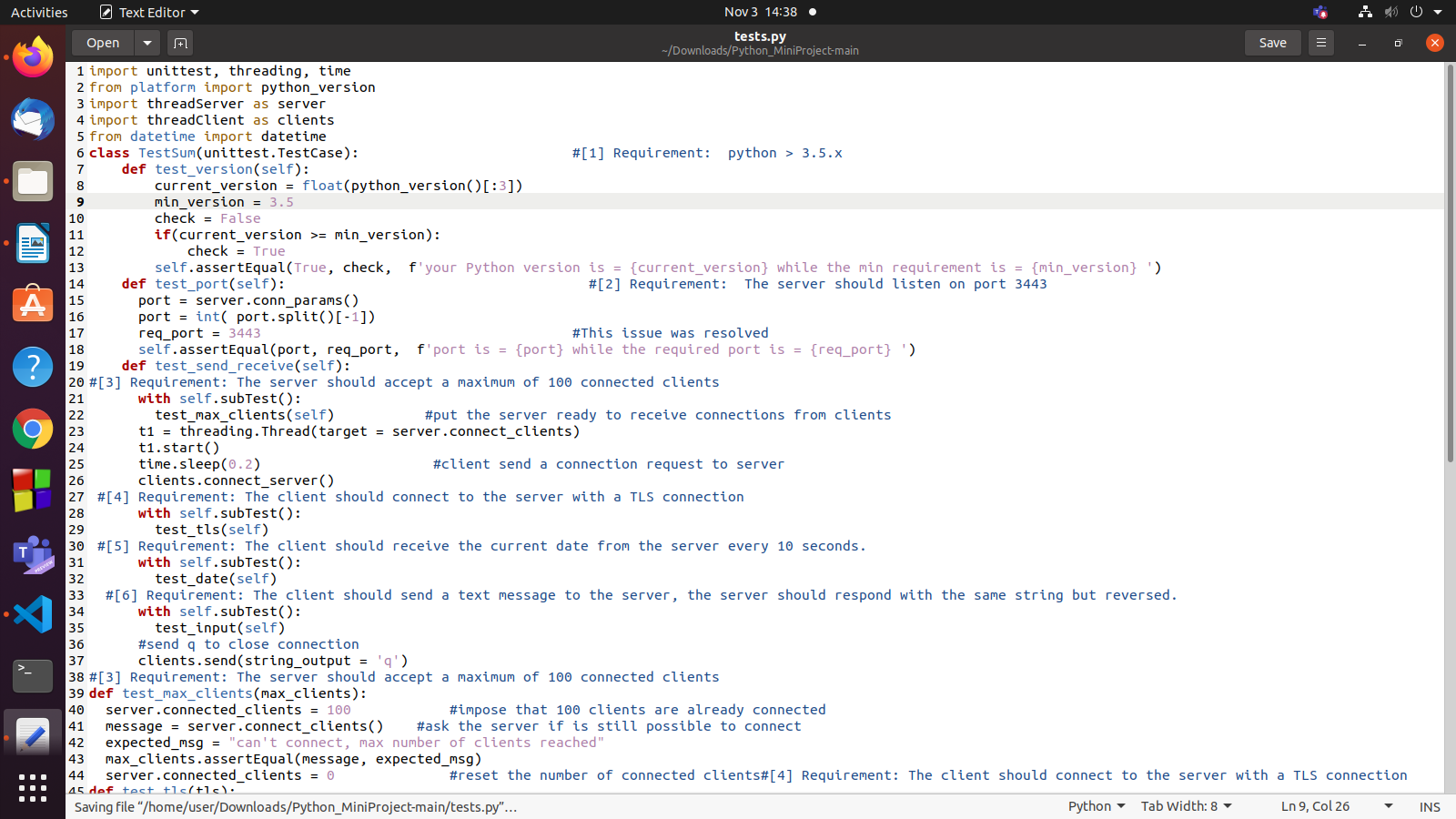


Figure 7. Code rewritten to satisfy the mini version.

### **4.Refactor**

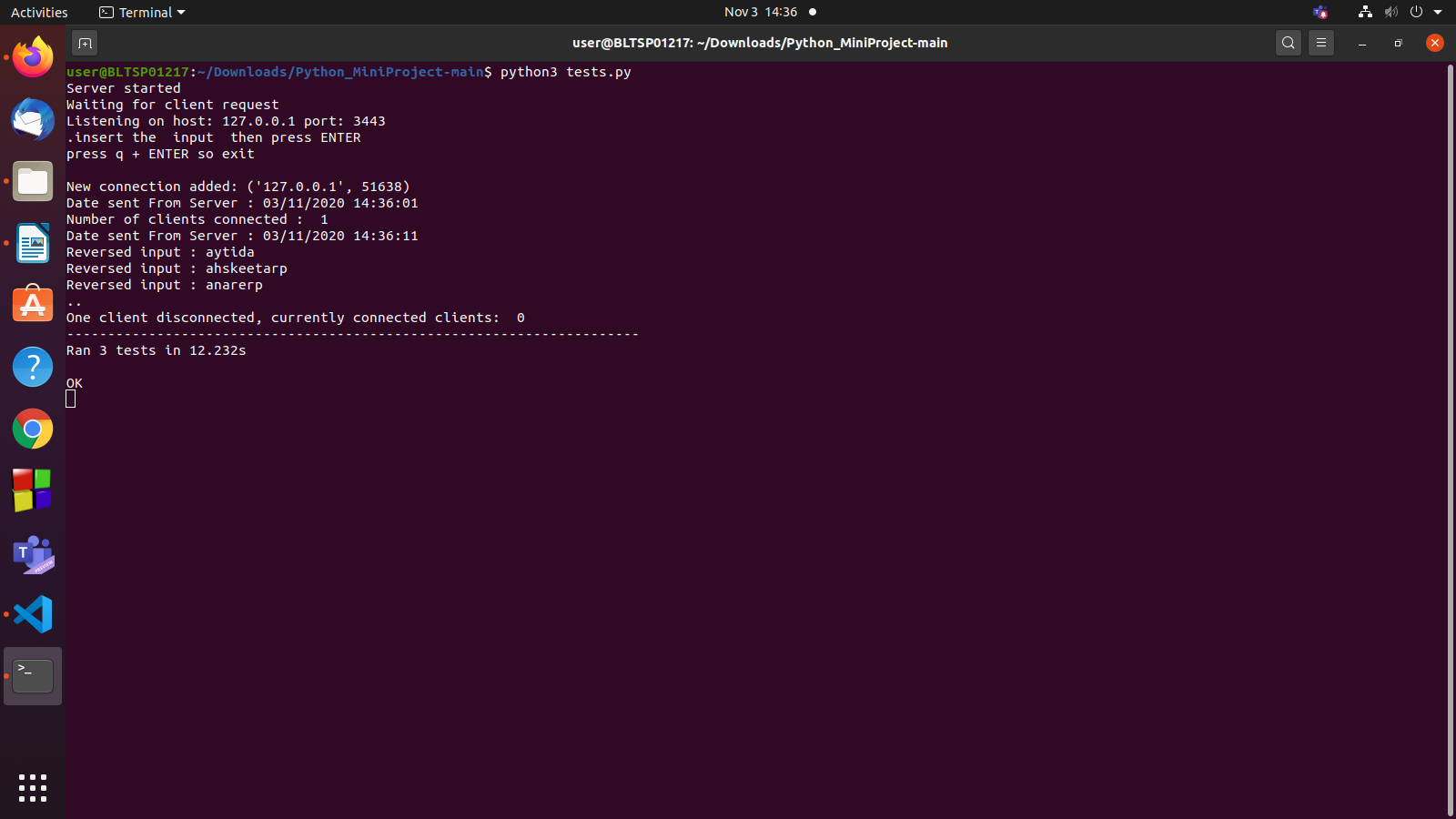


Figure 8. Test cases are passed for mini version.

# Task5: C/C++ Mini Project

### **1. Debugging**

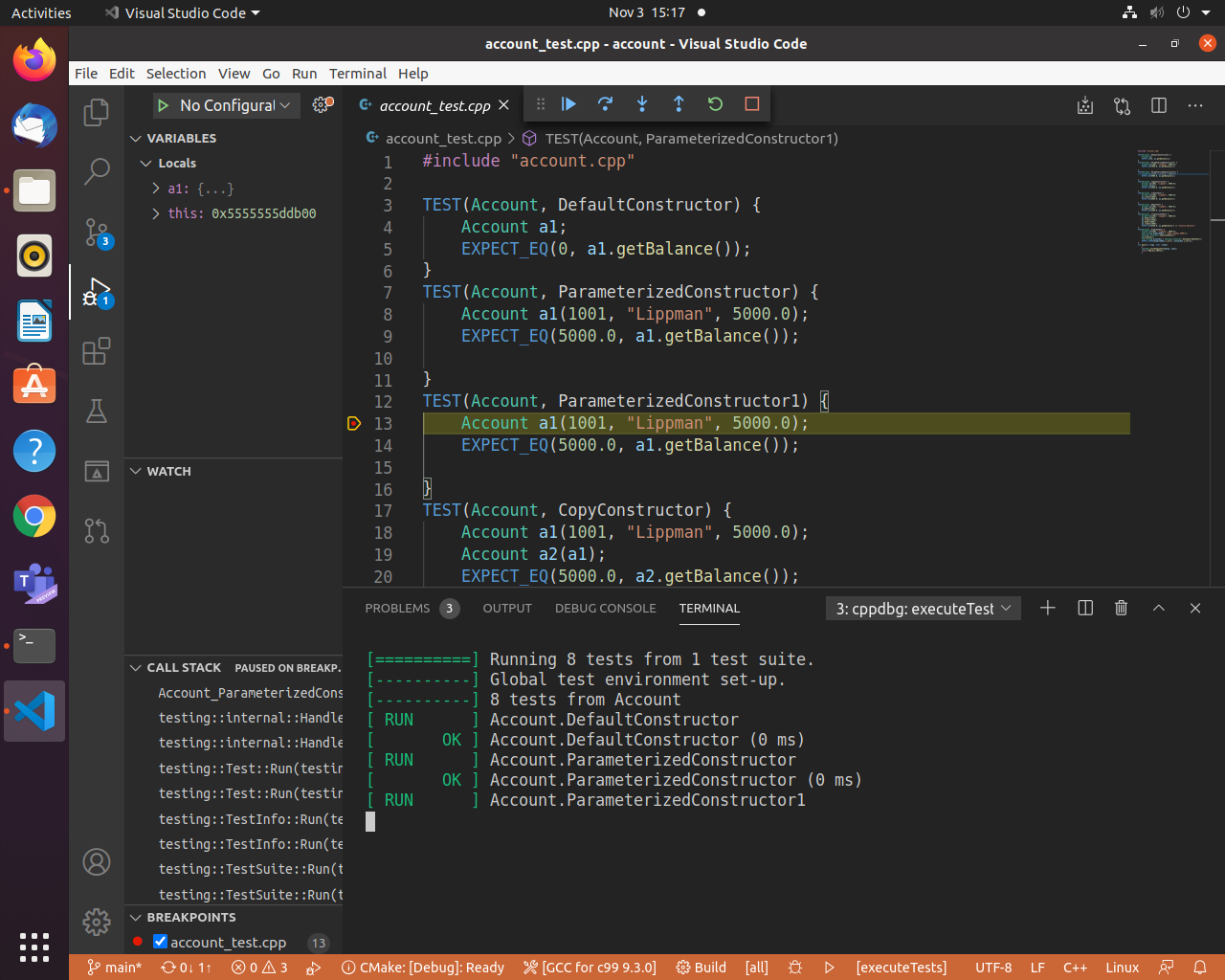


Fig 9 . Debugging

**VS Code has built-in debugging support,can debug any language.For debugging**

 any languages and runtimes (including PHP, Ruby, C#, Python, C++ and many

      others), look for Debuggers extensions in VS Code marketplace or select Install

      Additional Debuggers in the top-level Run menu.

**Steps to be followed:**

1. Debugger extensions

      VS Code has built-in debugging support,can debug any language.For debugging

      any languages and runtimes (including PHP, Ruby, C#, Python, C++ and many

      others), look for Debuggers extensions in VS Code marketplace or select Install

      Additional Debuggers in the top-level Run menu.

1. Start Debugging

      ->To bring up the Run view, select the Run icon in the Activity Bar on the side of VS

          Code or use the keyboard shortcut(ctrl+shift+D).

      -> The Run view displays all information related to running and debugging and has

           a top bar with debugging commands and configuration settings.

1. Launch Configuration

    ->To run or debug a code in VS Code, press F5.

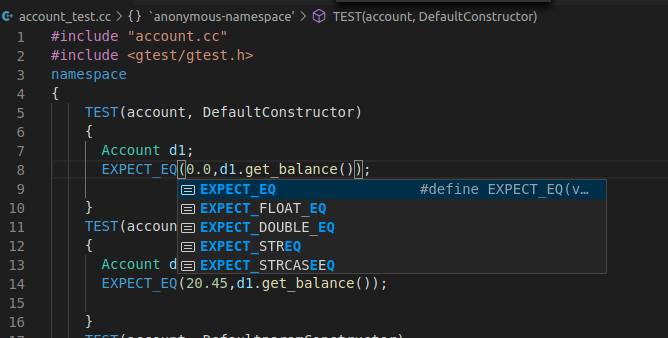
    ->VS Code automatically detect the debug environment, but if it fails, we have to

       choose it manually.

    ->As soon as a debugging session starts, the DEBUG CONSOLE panel is displayed

       and shows debugging output, and the Status Bar changes color.

### **2. AUTO COMPLETE:**

Auto Complete works based on installing the required extensions.

### **3. SYMBOL VIEWER:**

**Go to Symbol**

You can navigate **symbols** inside a file with Ctrl+Shift+O. By typing : the **symbols** will be grouped by category. Press Up **or** Down and navigate to the place you want.

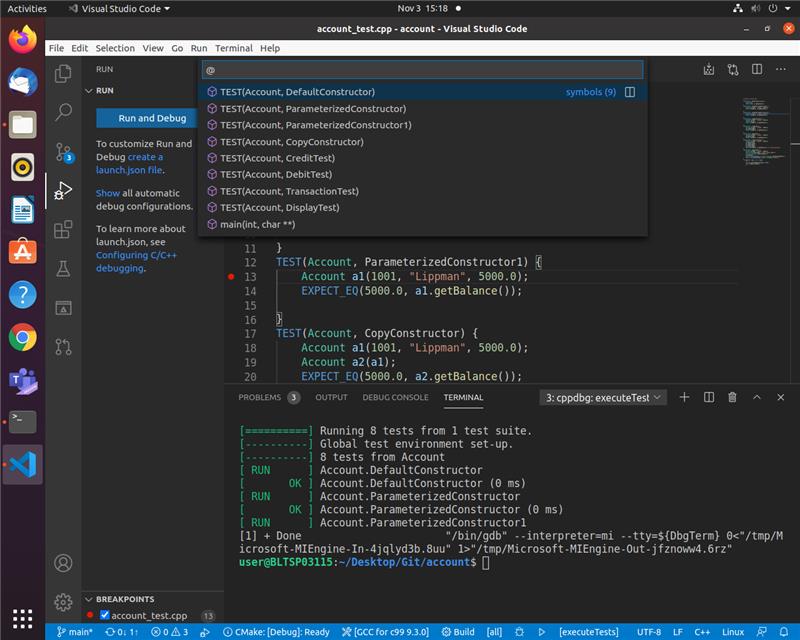


Fig 10 : Symbol Viewer

### **4. BUILD**

It builds the entire code and checks for any errors.

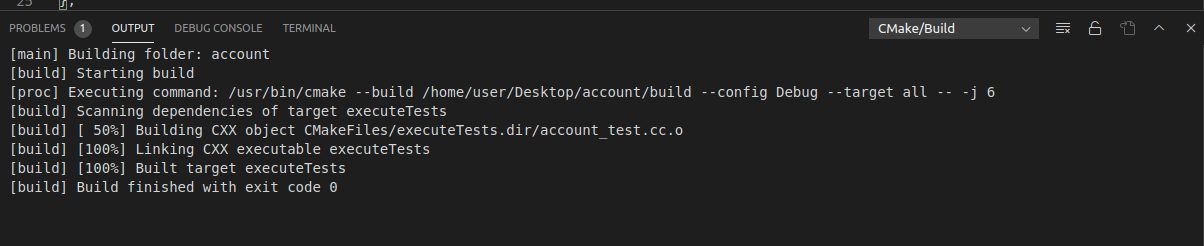
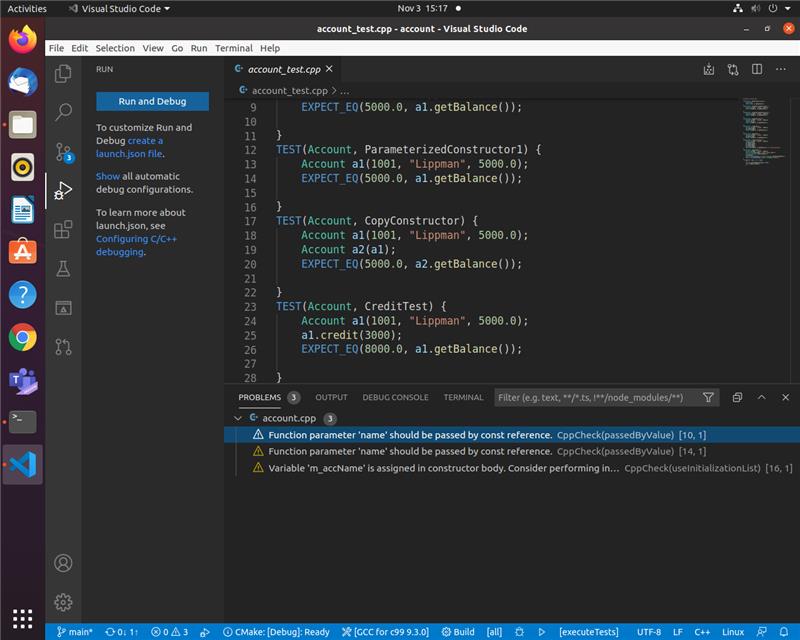


Fig 11: Build

### **5. CPP CHECK**

It is used for static code analysis, unused variables, code quality, coding rules, etc.



**Fig 12: Cpp Check**

### **6. VALGRIND**

Valgrind is a programming tool for memory debugging, memory leak detection, and profiling.The package used for memory debugging is “vs code valgrind”.

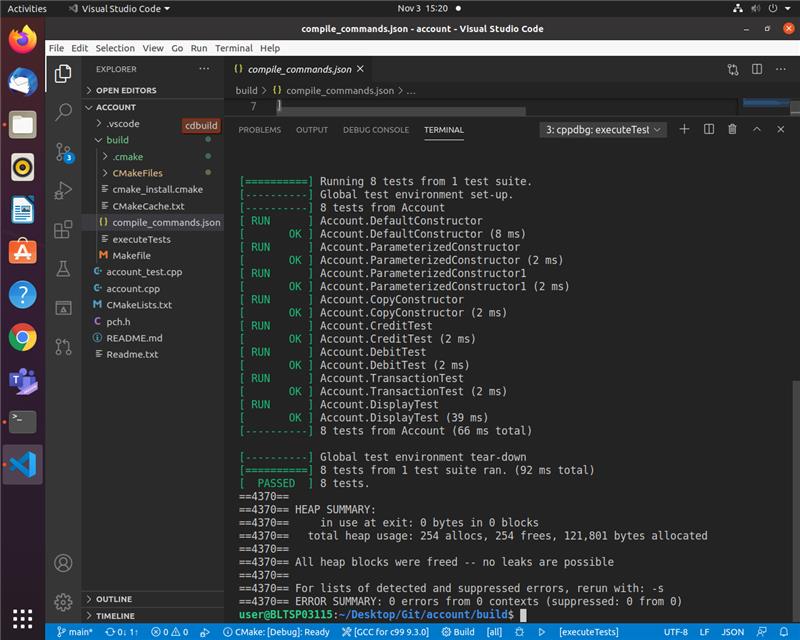


Fig 12: Cpp Check

## **REFERENCES:-**

<https://docs.microsoft.com/en-us/visualstudio/debugger/what-is-debugging?view=vs-2019>

<https://marketplace.visualstudio.com/items?itemName=Atishay-Jain.All-Autocomplete>

<https://docs.microsoft.com/en-us/visualstudio/debugger/how-to-use-the-modules-window?view=vs-2019>

<https://marketplace.visualstudio.com/items?itemName=Alexium.Cppcheckadd-in>

## **PYTHON PROJECT**

### **1.PYLINT**

Linting highlights syntactical and stylistic problems in your Python source code, which oftentimes helps you identify and correct subtle programming errors or unconventional coding practices that can lead to errors.

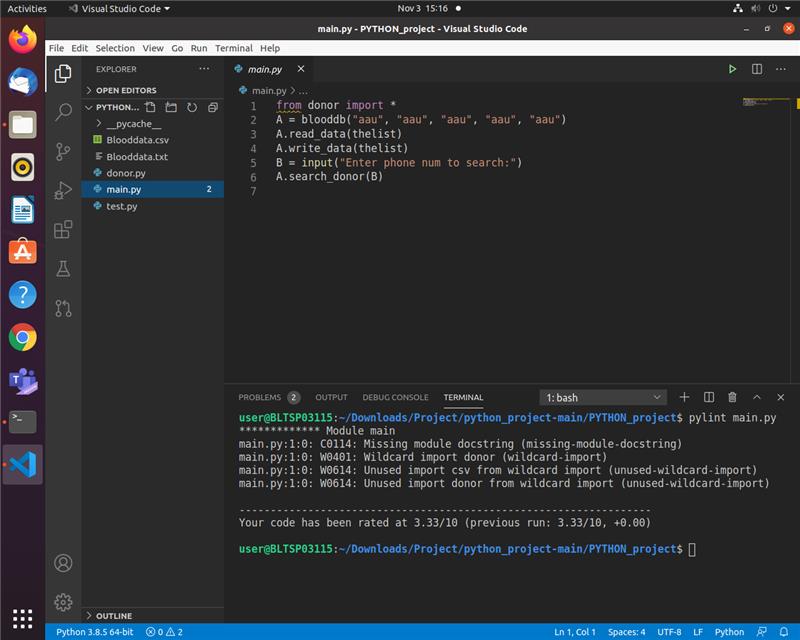


Fig 13: Pylint

### **2. UNIT TESTING IN PYTHON**

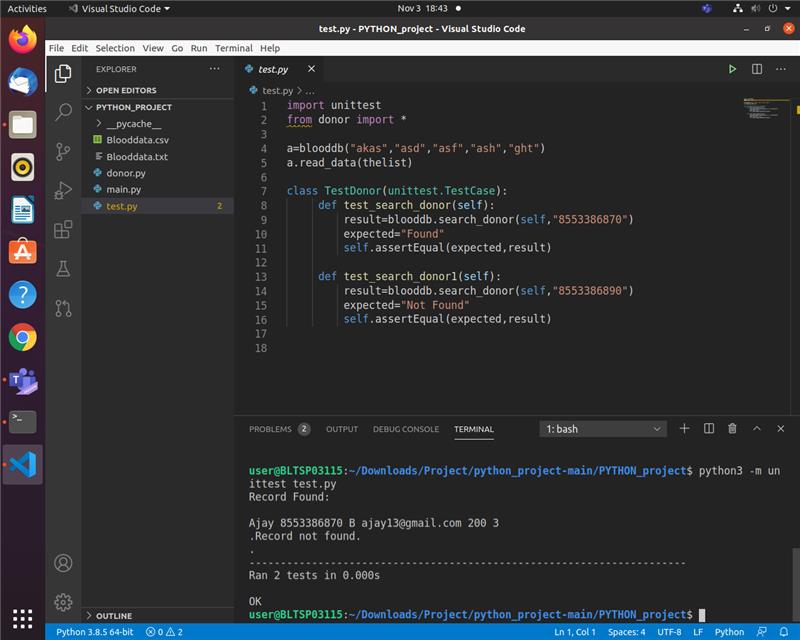


Fig 14: Unit testing in Python

## **REFERENCES:-**

<https://code.visualstudio.com/docs/python/linting>