

TESTING AND DEBUGGING PYTHON

A PROJECT REPORT

Submitted by

SHOBIKA R
(Reg. No: 19S034)

of

5 Year Integrated M.Sc., (Data Science)

in

**DEPARTMENT OF APPLIED MATHEMATICS
AND COMPUTATIONAL SCIENCE**



THIAGARAJAR COLLEGE OF ENGINEERING

(A Govt. Aided Autonomous Institution

affiliated to Anna University)

MADURAI – 625015

June 2020

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI



DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report "TESTING AND DEBUGGING PYTHON" is the bonafide work of SHOBICA R (19S034) Second Semester student of 5 Year Integrated MSc (Data Science) Degree Programme, who carried out the project under my supervision from 17.06.2020 to 30.06.2020 during the academic year 2019-2020.

The project report was submitted to the department on 30/06/2020 for evaluation/assessment.

Dr. S. Parthasarathy

Professor & Head

**Department of Applied Mathematics
and Computational Science**

Prof. B. Ramprakash

Project Guide

**Assistant Professor in Data Science
Department of Applied Mathematics
and Computational Science**

COURSE CERTIFICATE



TABLE OF CONTENTS

LIST OF TABLES

Table index	Name of the table	Page no
5.1	Significance of testing and debugging	xvii

LIST OF FIGURES

Figure index	Title of the figure	Page no
4.1	Code stub	xii
4.2	Unittest framework	xii
4.3	Function code	xiii
4.4	Debugging	xiii
4.5	Adding Tests	xiv
4.6	Sample Code	xiv
4.7	Result	xv

LIST OF ABBREVIATIONS

S.No	Abbreviation /Acronym	Description
1.	TDD	Test Driven Development

	Page No
1. Introduction	vi
2. Objective of the Project	vii
3. Description of the Project	ix
4. Implementation	xii
5. Significance of the project	xvi
6. Conclusion	xviii
7. Appendices	-
8. Project Worksheet / Diary	xix

1. Introduction:

Software bugs cost the economy billions of dollars per year as well as costing lives. The ability to test the code is probably one of the most important skills to know as a developer. Knowing how to track down and fix bugs is also critical. Testing can mean adding print statements to output data at certain points in a program to visually check it. Testing can also involve developing separate programs to test the functionality of the code itself. Debugging is the process of identifying and removing error from computer hardware or software.

The Title of the project is TESTING AND DEBUGGING PYTHON. It revolves around how to test and debug a simple code using Testing Frameworks and Python Debugger. The Aim of the project is to bring about a better level of understanding about Python error correction and to pursue all its objectives.

Python is an interpreted high-level open source programming language that is used by programmers in Web programming , Data science , Artificial intelligence , Machine learning and many other projects and applications. Python allows the programmer to focus on solving problems rather than syntax. Its relative size and simplified syntax gives it an edge over other languages like Java and C++, yet the abundance of libraries gives it the power to accomplish solutions.

Python has successfully emerged as general non-compiler program. Python is a challenging and powerful yet versatile programming platform that inspires to solve complex problem creatively. Successfully accomplishing two courses with Python under the guidance of The University Of Michigan in coursera paved the way for choosing this project.

2. Objective of the project :

This project covers three main objective :

- ✓ Using Testing Framework
- ✓ Testing a Code
- ✓ Using a Debugger To Debug a Code

It allows to read , graph and analyze data which is an important part of Data Science.

a) Using Testing Framework:

Python offers several modules for Testing Frameworks one of its best being Unit Testing. Each framework is categorized based on their test simplicity and their special features. The first objective of this project is to learn how to use a testing framework and use it in the test case.

There are a lot of testing frameworks like:

- unittest
- pytest
- nose2
- doctest
- robot
- testify

b) Testing a code:

One of the essential act to become a successful tester and developer is to know how to test the code with all the possible outcomes.

Automated testing is a well-known context in the world of testing. It is where the test plans are being executed using script instead of a human. Python comes with the tools and libraries that support automated testing for

the system. Python Test cases are comparatively easy to write. With the increased use of Python, Python-based test automation frameworks are also becoming popular.

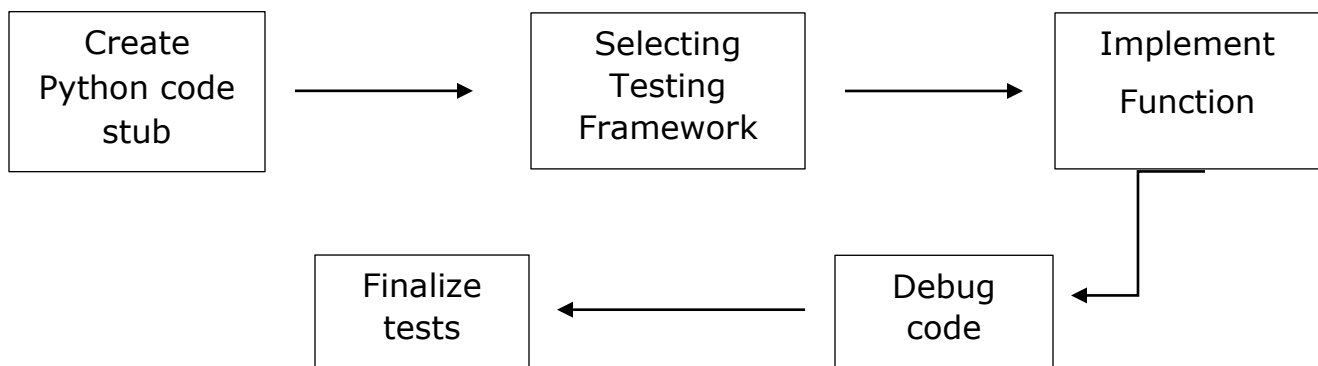
This project serves the objective of how to test a code with all tests passed.

c) Use a debugger to debug the code:

The code doesn't yield any result unless its errors are corrected and rerun. A Python debugger serves this purpose. One of the objectives of this project is to learn how to use a debugger and debug the code. Python debugger is the most efficient debugger which aims at producing the result in a short time.

Problem statement:

This project aims to test and debug errors which arises during reversal of digits using Bubble sort in Python. Overcome this problem through tasks that includes



Quiz:

To test the ability to understand and to make sure that all the topics are covered, the project includes a pack of 10 questions based on testing and debugging in Python.

3. Description of the project:

Rhyme:

The online platform used is called Rhyme. On Rhyme, projects are done in a hands-on manner in the browser. An instant access to pre-configured cloud desktops that have all the software and data needed. So, just focus on the learning. For this project, this means instant access to a cloud desktop with Python, Jupyter, and TensorFlow pre-installed. The following actions are carried out in Rhyme.

Visual Studio Code:

The Visual Studio Code Ide is used to write the code. The main Visual Studio icons used are

- Explorer
- Run -> Run without Debugging
- Code editor window
- Debugging controls
- Status bar (Run tests)

Testing Framework:

Unittest in Python is done to identify bugs early in the development stage of the application when bugs are less recurrent and less expensive to fix. So enable the unittest test framework in the project. A test is designed with test input and expected results based on function specs. The given function is likely to fail the test. Nevertheless import unittest from the Python library and required file and run a few test cases.

Unit testing with Python focuses mostly on testing a particular component without accessing any dependent objects. This is often difficult to do and a bit unrealistic as code is inherently dependent on other piece of

code. Python developers can use techniques such as stubs and mocks to separate code into “units” and run unit level testing on the individual pieces.

Technique used:

TDD - Unit testing should be run along with the Python development; it should execute unit tests as soon as developing each new feature. To test the application robustly development teams follow so-called test-driven development (TDD) approach: for each new functionality that the application must have, first design Python unit tests and only then carry on writing the code that will implement this feature.

Framework	: Unittest
Library	: Part of Python standard library
Category	: Unit Testing
Category Special feature	: Fast text collection and flexible text execution

Python unit testing framework supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The unittest module provides classes that make it easy to support these qualities for a set of tests.

Unittest is the very first Python-based automated unit test framework that was designed to work with the Python standard library. It Supports the reuse of test suits and test organization. It was inspired by JUnit and supports test automation including test collections, test independence, setup code for tests, etc.... It is also being called as PyUnit.

Standard Workflow Of Unittest:

- a) Import unittest module from Python library in the code.
- b) Define a class.
- c) Create function bubblesort.
- d) Place unittest.main() which is the main method to run tests.

After running, the project executes with result one out of one test case has failed.

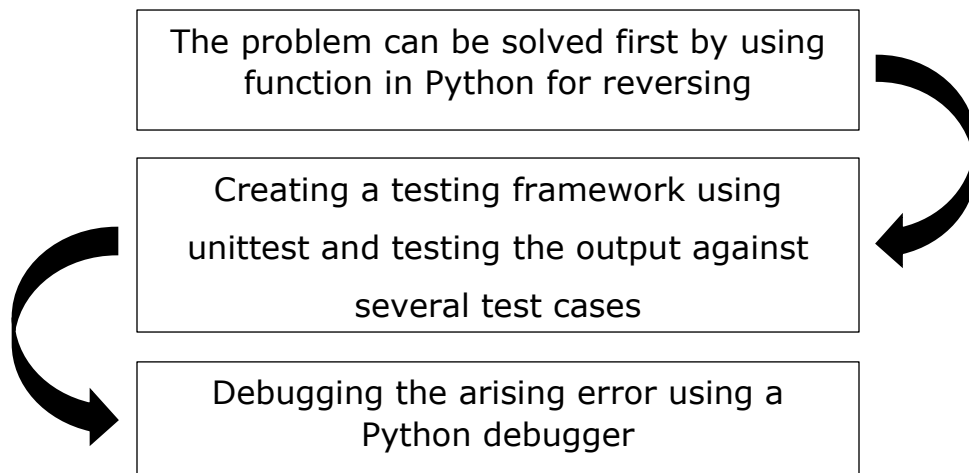
Python debugger:

Python debugger supports setting conditional breakpoints, stepping through the source code one line at a time, stack inspection etc... In this project :

Set breakpoints – Used to pause program to analyze the variables and error. Use the code debug controls to step through code and bring a zero on error using:

1. Continue.
2. Step over.
3. Step into.
4. Step out.

Solution:



4. Implementation:

Execution of project : Completion of 5 tasks

Create Python code stub : A function stub is a place holder for a function. Create a piece of stub which does nothing but defines the bubble sort function with a return statement. It is a simple test to observe using print.

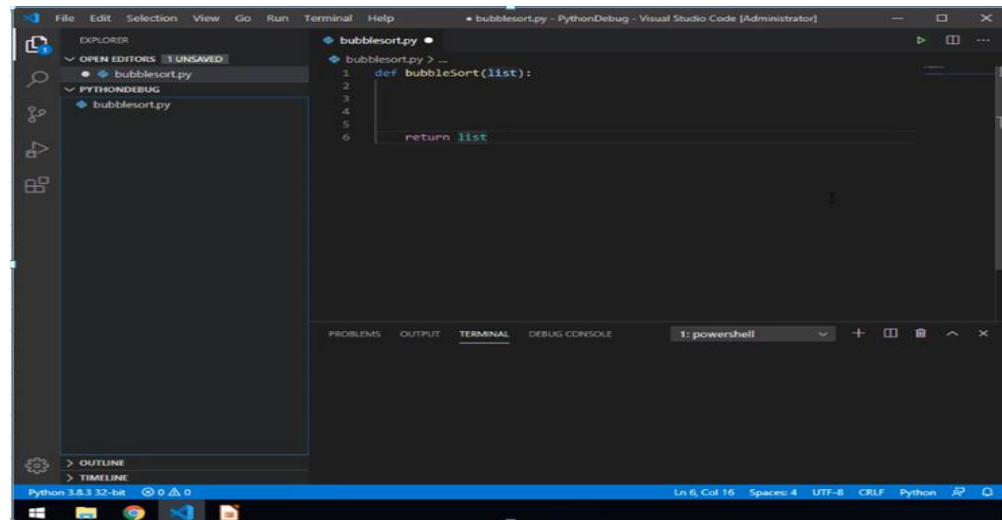


Fig 4.1 : Code stub

Selecting testing framework : Unittest framework is imported along with the file bubblesort and the function and tests to be performed is written.

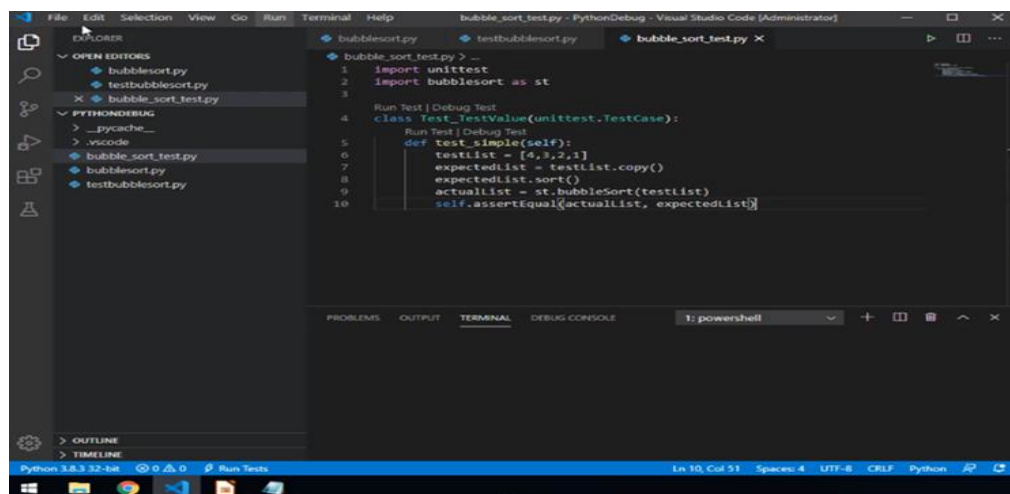
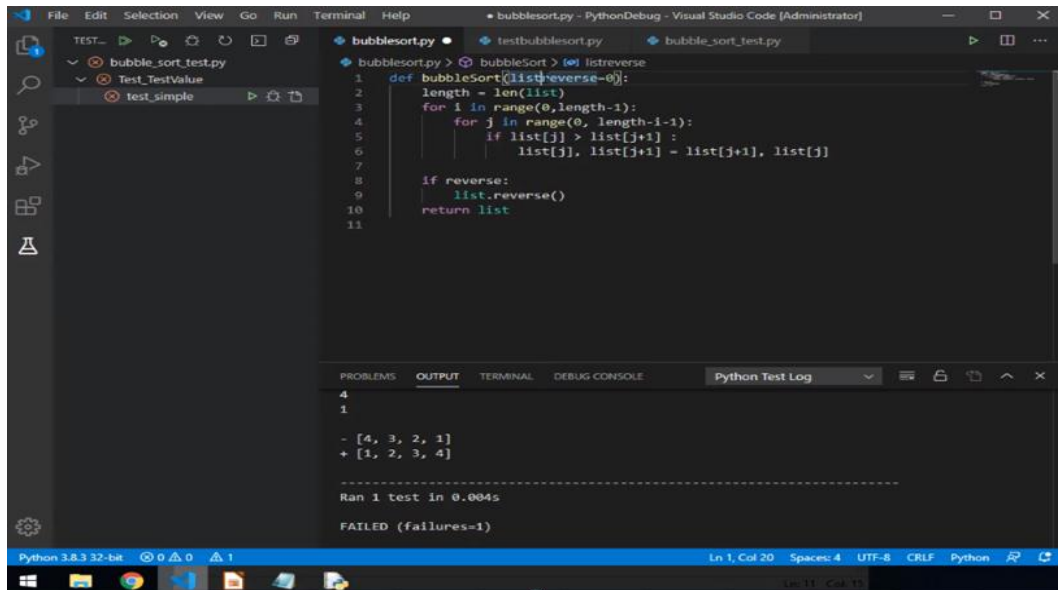


Fig 4.2 : Unittest framework

Implement Function : The code for reversing the digits in Python using bubblesort function is implemented in this block. Only implementation is carried out, the test being in a failed case.

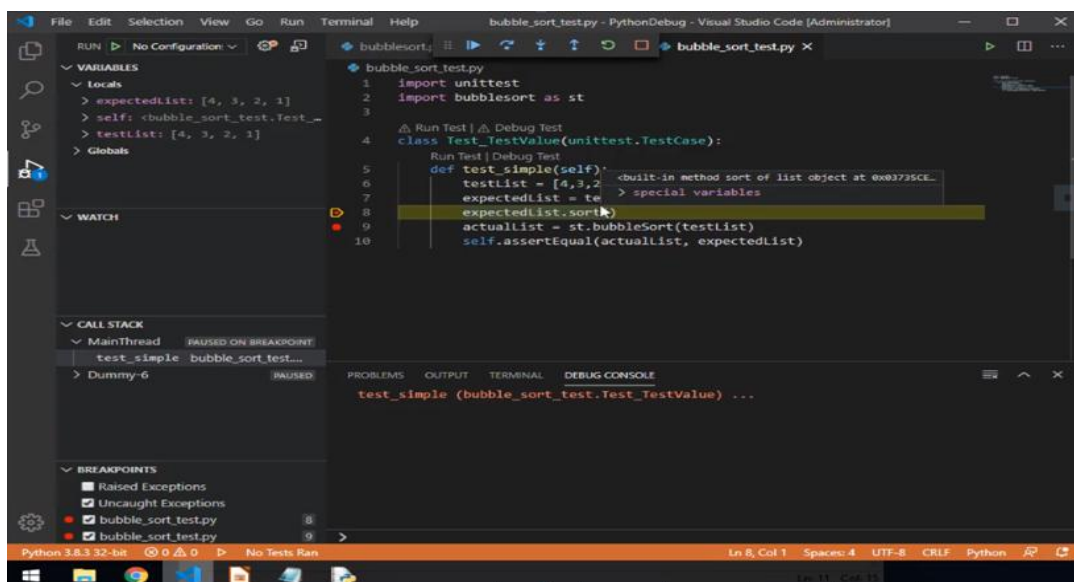


```
def bubbleSort(list, reverse=False):
    length = len(list)
    for i in range(0, length-1):
        for j in range(0, length-i-1):
            if list[j] > list[j+1]:
                list[j], list[j+1] = list[j+1], list[j]
    if reverse:
        list.reverse()
    return list
```

```
4
1
- [4, 3, 2, 1]
+ [1, 2, 3, 4]
-----
Ran 1 test in 0.004s
FAILED (failures=1)
```

Fig 4.3 : Function code

Debug code : The code is debugged using Python debugger by adding some breakpoints and correcting the error.



```
1 import unittest
2 import bubblesort as st
3
4 class Test_TestValue(unittest.TestCase):
5     def test_simple(self):
6         testlist = [4,3,2,1]
7         expectedlist = testlist
8         expectedlist.sort()
9         actuallist = st.bubbleSort(testlist)
10        self.assertEqual(actuallist, expectedlist)
```

```
test_simple (bubble_sort_test.Test_TestValue) ...
```

Fig 4.4 : Debugging

Finalize tests : Repair the tests and add more tests like corner-case tests, empty array, presorted array and also some random tests. The program is implemented after tests are run and corrections are made using Python.

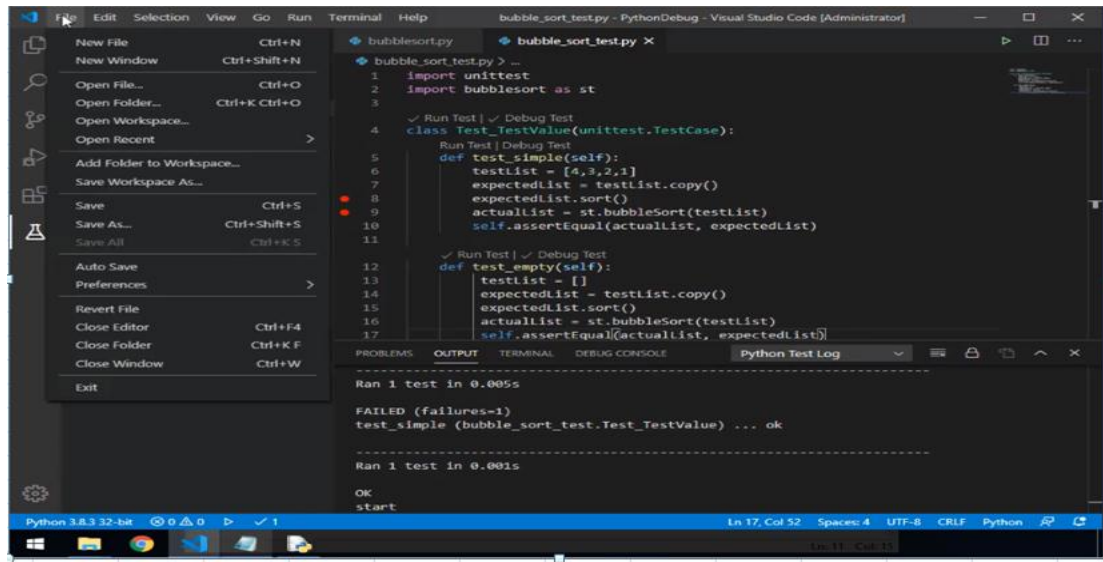


Fig 4.5 : Adding Tests

Implementing : Creating sample source code

Creating a simple calculator : Writing a stub of code to define simple functions in a normal calculator like add, subtract, multiply, divide, power, remainder, etc.

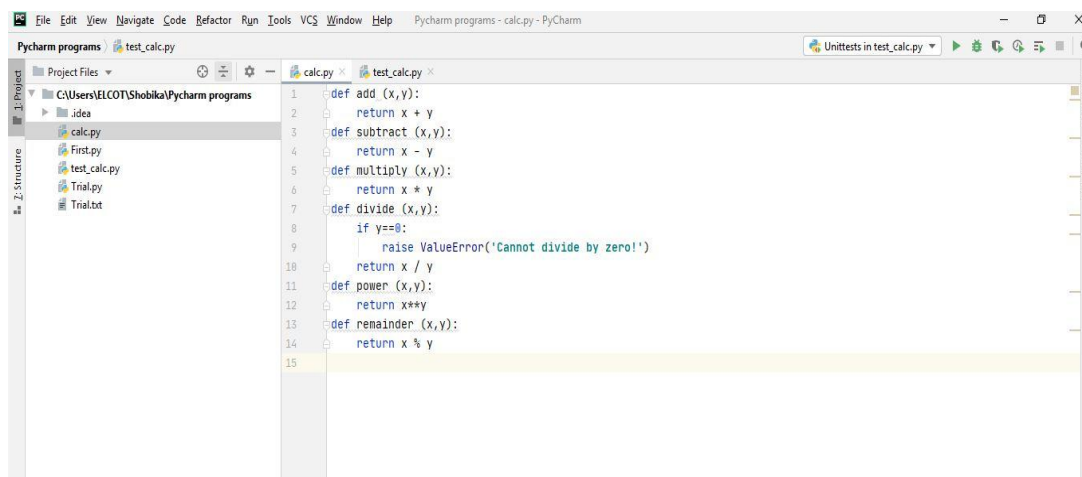


Fig 4.6 : Sample Code

Platform : The sample project is created using Pycharm IDE from JetBrains. It provides easy and friendly platform for both testing and debugging using Python.

Creating a unittest extension : Work out all the sample test cases and desired result using assertEquals() and check the output. If the code is correct all the test cases gets passed.

Failure cases : Failure occurs due to error in code or changes between original value and expected value in a test case.

Debugging : Debug the code using breakpoints in the error like step over and step through.

Solution : After debugging the code is implemented and output obtained with all test cases passed.

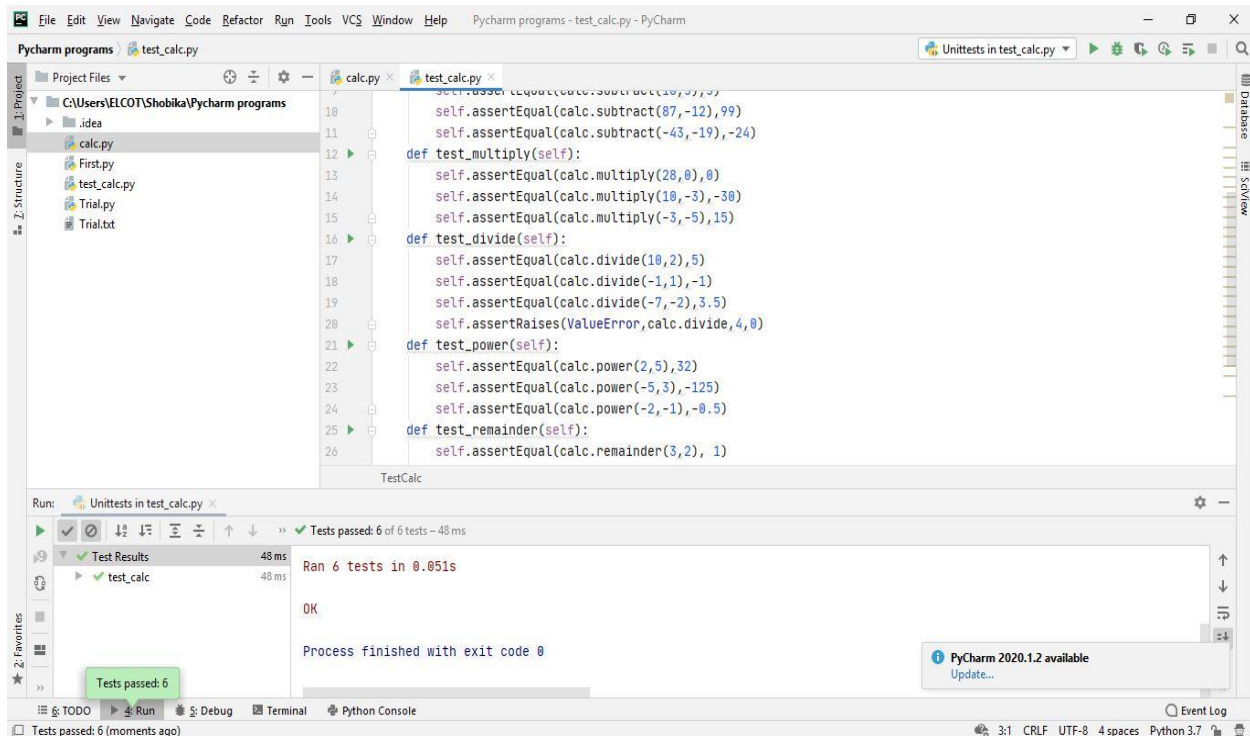


Fig 4.7 : Result

5. Significance of the project:

Any given code is unable to run unless it gets tested and debugged. The project serves this very purpose of enlightening programmers with how to test and get the program debugged.

Benefits:

Detecting problems early - Tests disclose problems early into the development.

Mitigating change - Allows the developer to refactor the source code during the testing stage and later on, while still making sure the module works as expected.

Simplifying integration - By testing the separate components of an application first and then testing them altogether, integration testing becomes much easier.

Debugging - It reports an error condition immediately. This allows earlier detection of an error and makes the process of software development stress-free and unproblematic. It also provides maximum useful information of data structures and allows easy interpretation.

Unique features:

TDD may seem like developing Python applications upside-down, but it has some advantages. For example, it ensures that it won't overlook unit testing for some feature and forces the radical simplification of the given code. Further, developing test-first will help to focus first on the tasks a certain function, test suite, or test case should achieve, and only afterwards to deal with how to implement that function, test suite, or test case. It gives fast feedback and creates a detailed specification.

A stub is used to fill in some dependency that is required for a unit to run correctly. So, in order to test an application component, which may be dependent on other components not yet developed, will need to write stubs, which basically are fake implementations of various components interfaces that give correct responses in cases needed to test other units.

Challenges:

1) Speed Limitations - Testing code is executed line by line. But since Python is interpreted, it often results in slow execution.

2) Run-time Errors - The Python language is dynamically typed so it has many design restrictions that cannot be debugged that are reported by some Python developers.

3) Underdeveloped Database Access Layers - As compared to the popular technologies like JDBC and ODBC, the Python's database access layer is found to be bit underdeveloped and primitive.

There is no need of design knowledge in the testing process.	Debugging can't be done without proper design knowledge.
Testing can be done by insider as well as outsider.	Debugging is done only by insider. Outsider can't do debugging.
Testing can be manual or automated.	Debugging is always manual. Debugging can't be automated.
It is based on different testing levels i.e. unit testing, integration testing, system testing etc.	Debugging is based on different types of bugs.
Testing is a stage of software development life cycle (SDLC).	Debugging is not an aspect of software development life cycle, it occurs as a consequence of testing.
Testing is composed of validation and verification of software.	While debugging process seeks to match symptom with cause, by that it leads to the error correction.
Testing is initiated after the code is written.	Debugging commences with the execution of a test case.

Table 5.10 : Significance of testing and debugging.

6. Conclusion:

This project gives a more clear understanding of the errors occurring during compilation of the program, reversing the digits in a list using bubblesort and the methods to solve the errors. As the objective states, Python testing framework was implemented using unittest to test all the possible results and interpreting the err using breakpoints. Python is a complete solution for obtaining result in no time. This documentation shows how to write a code, use testing frameworks in it, implement the function, detecting the errors and debugging it to get the desired output in a compile time of 0.006 secs and testing time of 0.004 secs.

Furthermore, accurate testing and debugging can be performed in future by various Python testing frameworks like Nose2, PytTest and Robot on several programs and production can be differentiated. Also the code skeleton can be increased and enhanced output can be obtained by testing and debugging.

Reference Links and URLs:

Course: <https://www.coursera.org/projects/testing-and-debugging-python>

Python testing frameworks: <https://www.softwaretestinghelp.com/python-testing-frameworks/>

Debugging: <https://code.visualstudio.com/docs/python/debugging>

Reference books:

Python Unit Test Automation : Practical Techniques for Python developers and testers by Pajankar, Ashwin – Chapter 3 : Unittest.

Download Links:

Visual Studio Code : <https://code.visualstudio.com/download>

8. Project Worksheet/Diary:

WEEK 1	Date	Topics learned / Activity carried out / Task completed / Online / E-resources accessed
	17-06-2020	Course enrollment and getting started in Testing and debugging python
	18-06-2020	A study about Python testing frameworks
	19-06-2020	How to use a Python debugger
	20-06-2020	Using rhyme to listen and work out in coursera
	21-06-2020	Python debugging and testing / 5 Tasks completed
	22-06-2020	Graded quiz attended and certificate received

WEEK 2	Date	Topics learned / Activity carried out / Task completed / Online / E-resources accessed
	23-06-2020	Creating own sample code using PyCharm
	24-06-2020	Testing the code using unittest framework
	25-06-2020	Using Python debugger for fixing and correcting the code
	26-06-2020	Obtaining output and recording the results
	27-06-2020	Completion of Project Diary and Consultation and approval by the faculty guide
	28-06-2020	Submission of Project Report and Handling in of Video Presentation

**Signature of the student
(with date)**

**Signature of the faculty guide
(with date)**