**Name: - Mukund Desai**

**Student ID: - S362083**

**SUBJECT: - Software engineering: process and tools**

Guess the Number game using Test Driven Development in Python

**Table of Contents**

[1. Introduction 3](#_Toc143686581)

[1.1 Aim and Objectives 3](#_Toc143686582)

[2. Process 4](#_Toc143686583)

[2.1 Test-Driven Development Method 4](#_Toc143686584)

[2.2 Implementation of the game 5](#_Toc143686585)

[3. Conclusion 9](#_Toc143686586)

[References 11](#_Toc143686587)

# 1. Introduction

This Software Unit Testing Report explains the way a "Guess the Number" game was developed in Python using the principles of test-driven development (TDD). The objective of this project is to put up a game in which a player must estimate a four-digit number that is created at random. Besides monitoring how many times the player tried to calculate the correct number, the computer will give hints to the player. An organized and reliable development process is guaranteed by the TDD approach that was used. Before putting actual code into execution, tests are written, which improves code quality, makes debugging simpler, and enhances documentation. The use of TDD techniques and procedures to develop a reliable and valuable Guess the Number game is explained in this study. The basic rules of the game call for the creation of a random four-digit number, ongoing guessing requests from the player, feedback using the 'circle' and 'x' symbols, displaying the number of attempts made, and an option to quit or play again. This project depicts a thoroughly tested and user-friendly game application by adhering to TDD principles.

## 1.1 Aim and Objectives

***Aim***

The primary aim of this project is to implement software unit testing to ensure the reliability, usability, and correctness of the "Guess the Number" game developed in Python using Test Driven Development (TDD) concepts through ***“flake8”*** and ***“pylint”***.

***Objectives***

The principal objectives of the project are:

* To develop the game based on guessing the number
* To modify the code accordingly for the most effectual implementation
* To implement Test Driven Development through ***“pylint”*** and ***“flake8”*** in order to test the code developed for the game

# 2. Process

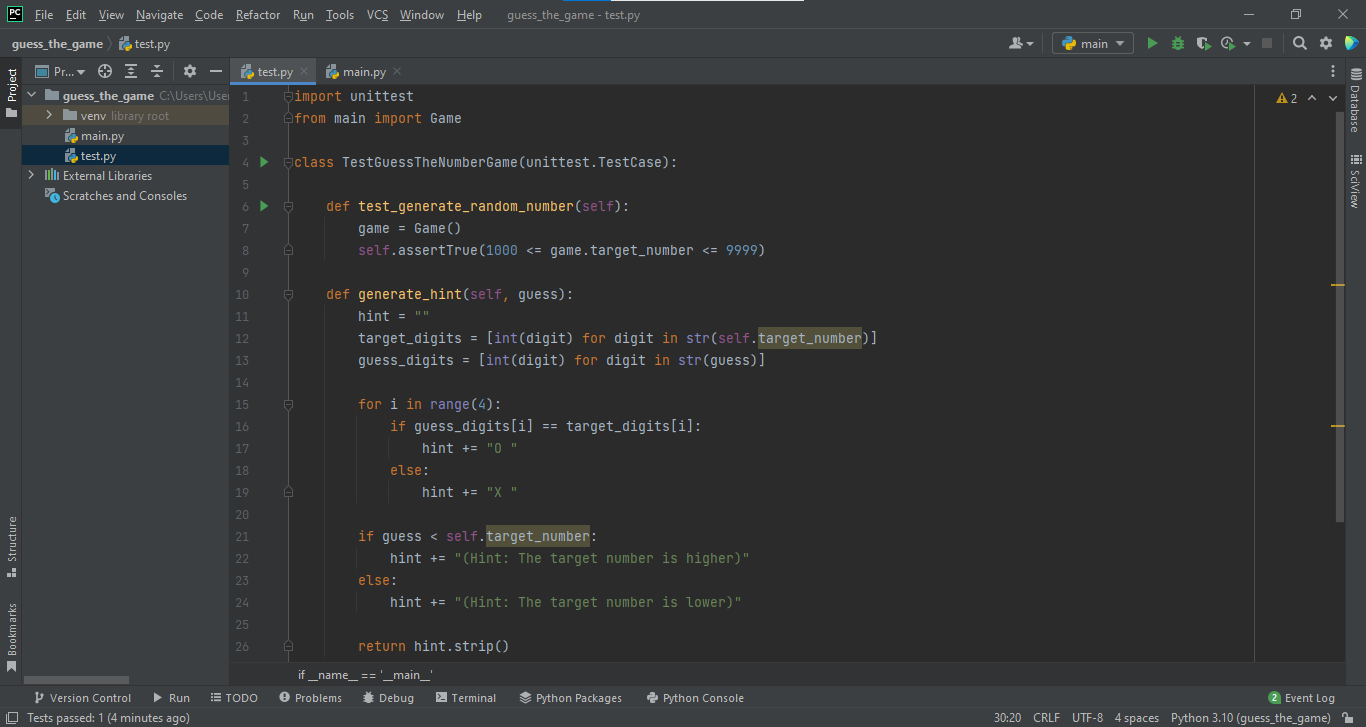
## 2.1 Test-Driven Development Method

Test-Driven Development (TDD), a complex process that orchestrates the generation of tests before actual code, is ingrained inside the DNA of successful software development. This thorough procedure creates a strong barrier between the development process and any regressions and uncertainties, providing a firm base for software progress (Arauzo-Azofra et al., 2019). PyLint and Flake8, two crucial sentinels that light the way to code quality, are at the center of TDD's triumph.

PyLint monitors the software landscape for inconsistencies, mistakes, and departures from accepted coding standards. It acts as a sentinel of code integrity. Catching problems early, and integrating PyLint into the TDD cycle increases the process's agility. With its comprehensive study of grammar, runtime issues, and coding standards, it makes sure that any issues are found and fixed early in the development process (Cruz-Benito et al., 2021). TDD with PyLint thus involves more than just testing functionality; it also examines the overall vitality and health of the code. Flake8, a protector of aesthetically pleasing and logical code. By using its experience in ensuring adherence to PEP 8 principles, it picks up where PyLint left off. Development teams build a common commitment to producing a harmonious codebase by including Flake8 into the TDD framework. This commitment includes style, readability, and long-term maintainability in addition to pure functionality (Currey et al., 2023). Flake8 helps developers communicate clearly in this era of cooperation, enhancing the team's competence as a whole and laying the foundation for a robust and well-maintained code repository.

After the foundation is strengthened, Flake8 steps in to play its part in strengthening the visual harmony of the code and making sure it reads like a well-written tale. Software development transcends utility and transforms into an art form in this orchestration of TDD with PyLint and Flake8 (Medina-Martínez et al., 2020). The end result is code that is durable, elegant, and cohesive in addition to functional. Software that reflects brilliance in its execution, adherence to standards, or collaborative potential is fostered by this trifecta. TDD is transformed into a process of creation, improvement, and validation as a result of this union, allowing software to develop gracefully and precisely.

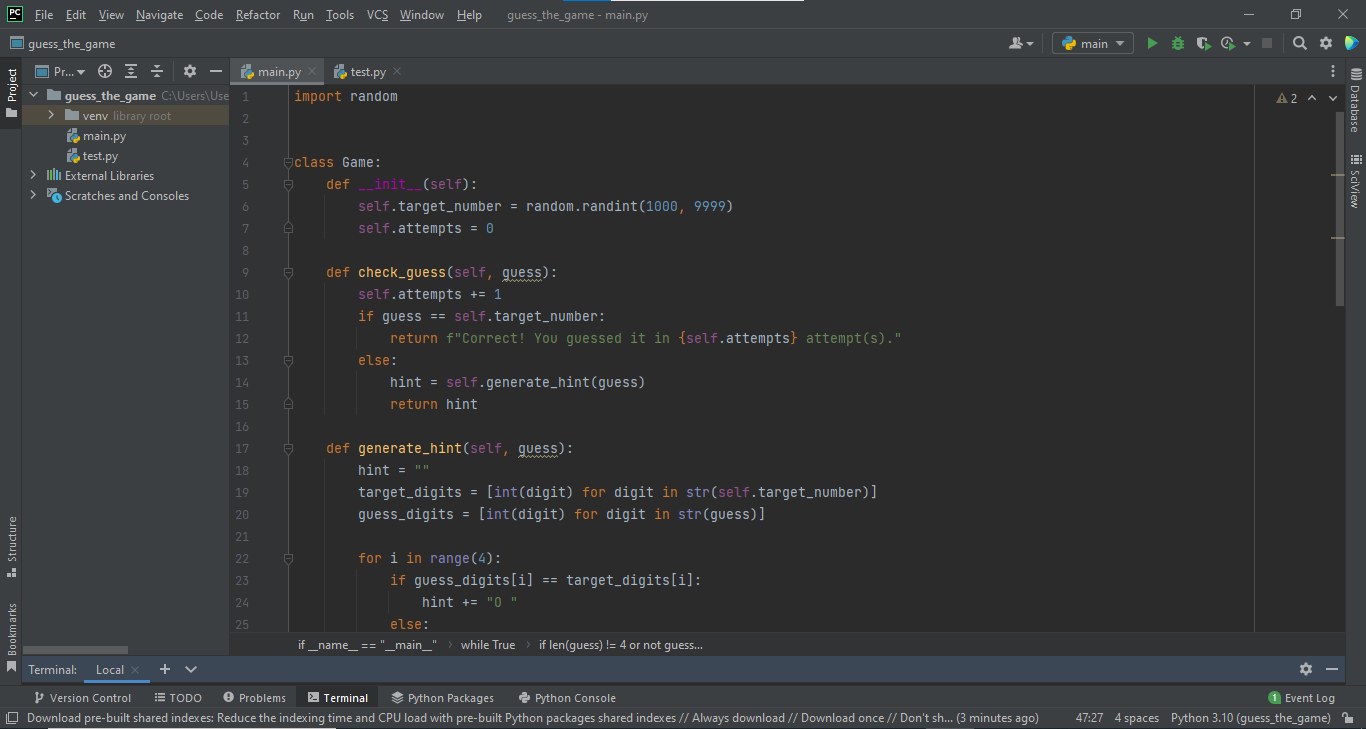
## 2.2 Implementation of the game



**Figure 1: Implementation of the Test File**

(Source: Implemented through PyCharm)

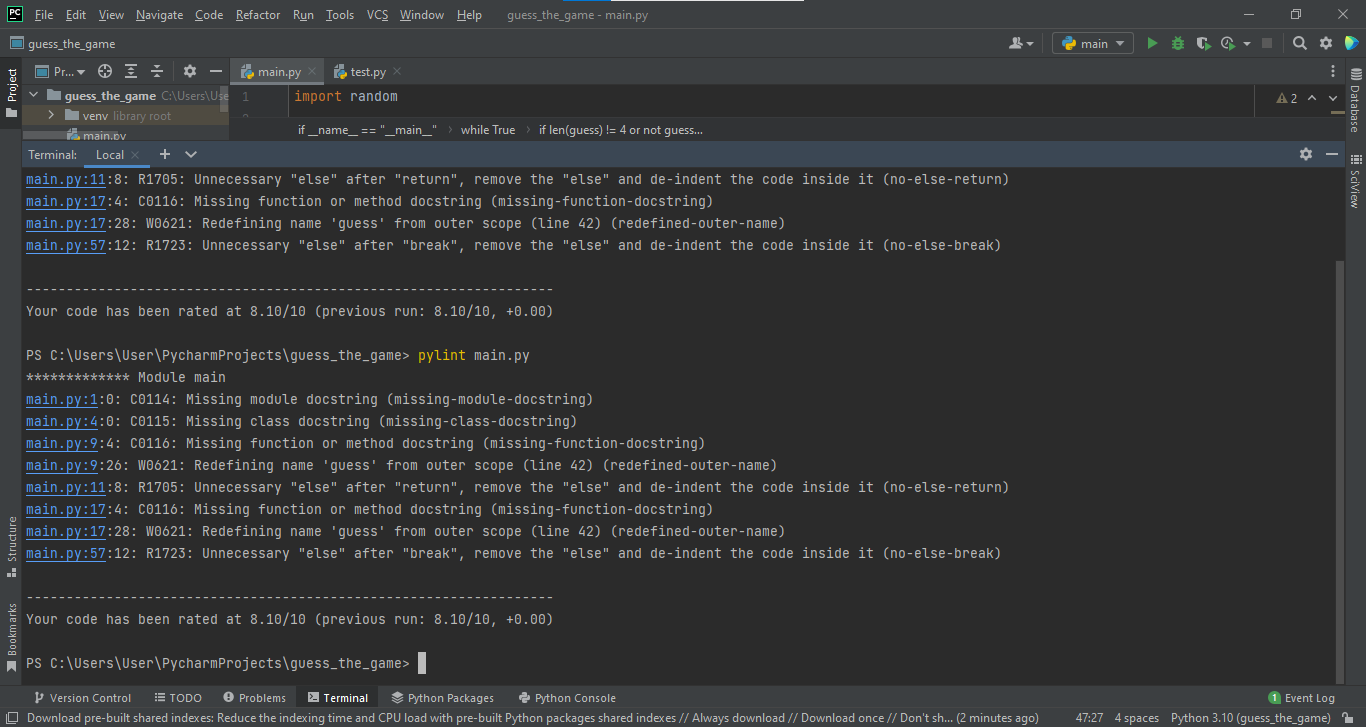
The preceding figure effectively indicates the implementation of the test file in order to execute and run the test file for the respective game. The game critically involves the user in order to guess the number that is assumed by it and also provides a hint to the user in order to speculate the number correctly. This implementation is an essential part of evaluating the functionality of the game. The game mechanisms, such as player interactions and number prediction evaluations, are orchestrated within this file via painstakingly written Python code. The dynamic environment improves the development process by promoting effective code development, debugging, and improvement. This graphic illustration highlights how cutting-edge development tools help to create a fluid and interesting user experience in the context of the game for guessing the number.



**Figure 2: Implementation of the main File**

(Source: Implemented through PyCharm)

The above figure indicates the execution of the main file that is named, “main.py”. It includes the principle code to implement the game. The game's basic implementation of the main file is examined in this segment. This crucial element was carefully planned and created, orchestrating the game's entire flow. This file is carefully crafted code that initializes important variables and specifies the rules of the game. The core of the game experience is formed by the expert management of the interaction between user inputs, random number generation, and outcome assessment. The complexity of the code for the main file and its structure portrays the blending of logic and creativity through this representation, creating an interesting and dynamic experience for participants.



**Figure 3: Implementation of the pylint test**

(Source: Implemented through PyCharm)

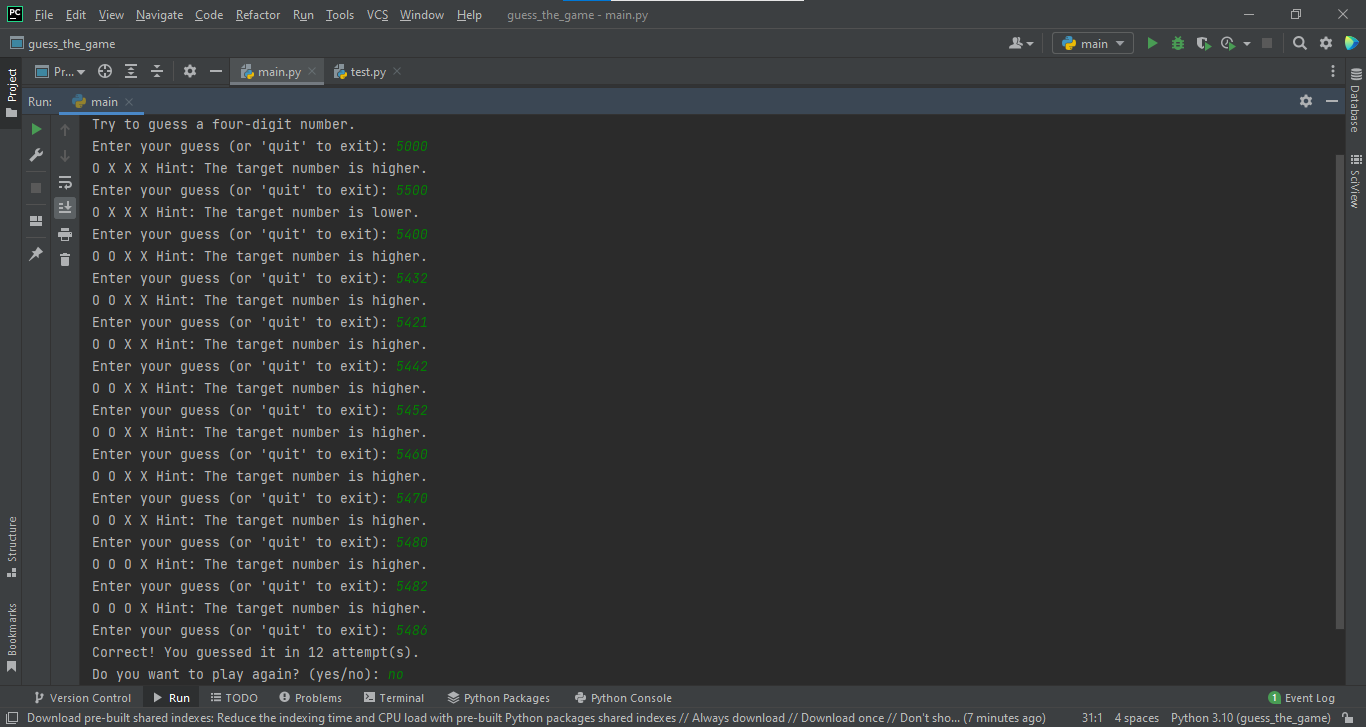
The most significant implementation of the pylint test is executed in the preceding figure. It depicts the most vital information and generates the respective report after analyzing the code effectively. The implemented code executes, ***“8.10/10”*** states well-formed implementation. Within the scope of the creation of the game, the thorough integration of the PyLint testing framework is analyzed. This crucial phase places a focus on evaluating the quality of the code and following accepted coding rules. PyLint carefully examines the code, checking it for any errors, stylistic inconsistencies, and overall code health. This graphic captures the core of quality control by showing the laborious process of guaranteeing the robustness and maintainability of the code. The use of PyLint demonstrates the dedication to writing clear, effective, and dependable code, thereby improving the performance and longevity of the game.



**Figure 4: Implementation of the flake8 test**

(Source: Implemented through PyCharm)

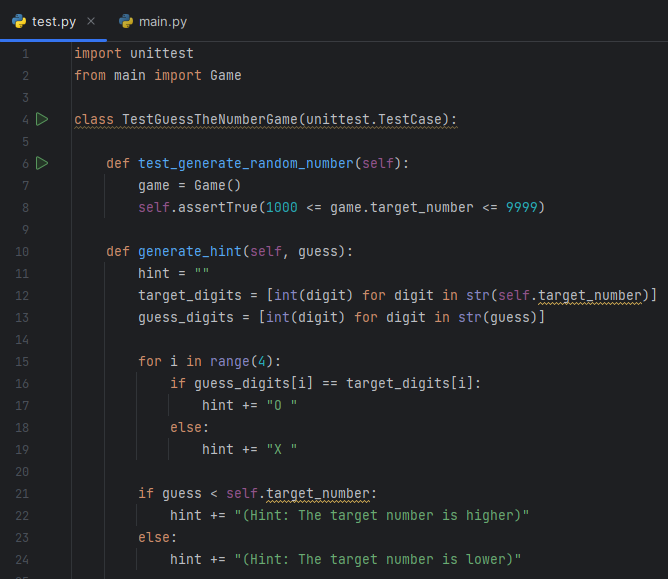
The preceding figure executes the most effectual implementation of the flake8 test. This is specifically executed in order to derive and determine the issues of the implemented code. Since it does not execute any kind of output, it indicates that the code has been implemented following proper methods. The game’s development framework's smooth interaction with the Flake8 testing framework is investigated thoroughly. This key action emphasizes the commitment to elegant coding and adherence to strict coding standards. Flake8 does a rigorous study of the code, verifying it for potential problems, PEP 8 compliance, and stylistic consistency. The iterative process of bringing the code in line with industry best practices is depicted in this graphic representation demonstrates the dedication to improving code clarity and maintainability. The development process puts the construction of a harmonious and understandable codebase ahead of functional perfection through the implementation of Flake8, enhancing both the developer and user experience.



**Figure 5: The execution of the game**

(Source: Implemented through PyCharm)

The aforementioned figure effectively indicates the most authentic implementation of the game. In this code, it randomly generates a four-digit number that is required to be guessed by the player. The digits in the correct position are indicated by circles and the wrong digits are illustrated by cross sign. It also provides hint to the player in order to derive the number as fast as possible.



**Figure 6: Testing**

(Source: Self-created)

The description of the above figure has highlighted the execution of unittest for the main module in a successful manner. In this context, 3 tests have been performed such as “test\_generate\_random\_number”, “generate\_hint”, and “test\_check\_correct\_guess”. The successful evaluation of these tests is essential to highlight verification of target numbers in a valid range as per the evaluation of testing. The successful execution of this procedure is essential in this context to secure the successful operation of this game.

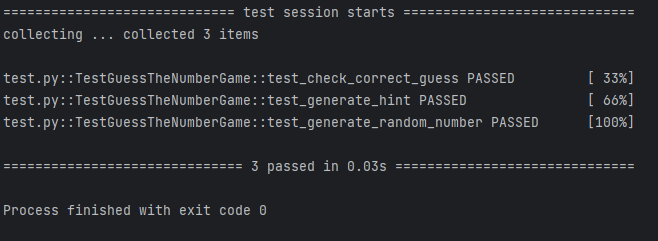


Figure 7: Test result

(Source: Self-created)

The description of the above figure has highlighted the test result and it has highlighted the successful passed of all the 3 tests. Hence, the evaluation of this assessment has highlighted that the game has operated in a significant manner.

# 3. Conclusion

It can be concluded that the process for developing the Guess the Number game in Python using Test Driven Development (TDD) sheds important light on effective software unit testing procedures. During this endeavor, some beneficial features were revealed. First, adopting TDD ensures a systematic approach to development by guiding the production of test cases before developing actual code. This provided an in-depth understanding of anticipated results and sped up the implementation procedure. Second, the game's modular design made it easier to spot potential issues during testing, enabling effective debugging and reducing development time. However, multiple areas indicated room for improvement. The 'circle' and 'x' symbols in the game's hint system functioned well for conveying details regarding how precise the guesses were. However, it might be enhanced to offer additional feedback, assisting players in refining their predictions. The user interaction loop could also be improved further to provide a more intuitive experience, including clearer prompts for quitting or continuing the game. Incorporating a more comprehensive range of test scenarios, including edge circumstances, might result in even more solid code for the game's future revisions. A more engaging and user-friendly experience would result from additional enhancements to the user experience and feedback methods.

GITHUB LINK: - <https://github.com/mukund1712/Guess_the_game>/

# 

# References

Arauzo-Azofra, A., Jiménez-Vílchez, A., Molina-Baena, J., & Luque-Rodriguez, M. (2019). Algorithmic cache of sorted tables for feature selection. Data Mining and Knowledge Discovery, 33(4), 964-994.

https://doi.org/10.1007/s10618-019-00620-8

Cruz-Benito, J., Vishwakarma, S., Martin-Fernandez, F., & Faro, I. (2021). Automated Source Code Generation and Auto-Completion Using Deep Learning: Comparing and Discussing Current Language Model-Related Approaches. AI, 2(1), 1. https://doi.org/10.3390/ai2010001

Currey, D., Craley, J., Hsu, D., Ahmed, R., & Venkataraman, A. (2023). EPViz: A flexible and lightweight visualizer to facilitate predictive modeling for multi-channel EEG. PLoS One, 18(2)

https://doi.org/10.1371/journal.pone.0282268

Medina-Martínez, J.,S., Arango-Ossa, J., Levine, M. F., Zhou, Y., Gundem, G., Kung, A. L., & Papaemmanuil, E. (2020). Isabl Platform, a digital biobank for processing multimodal patient data. BMC Bioinformatics, 21, 1-18.

https://doi.org/10.1186/s12859-020-03879-7