Introduction

**This is a rock, paper, and scissor game project using python with Test Driven Development (TDD).**

**Objective of the program:**

* The main objective of the program is to create a game called rock paper scissors.
* A user should be able to make the move from one of the 3 options.
* The program can count the number of wins of the user as well as the number of wins by computer.
* The program can be terminated at any time.
* Users can play the game any number of times.
* The program decides the winner by the first player to 5 wins.
* Implement error handling to handle possible errors

**Requirement of the program:**

* User-friendly UI for the user to make a move
* Different variables for storing the user input as well as other variables required for processing.
* Random module for the move of computer which chooses one option of out three possible options.
* Control statements to determine the winner of a single game.
* Control statement to continue the game until the user terminates the game or a winner is decided.
* Error handling to handle unwanted inputs.
* Control statement to check if the user or computer won the game.
* Control statement to restart the game
* Unit Test of the function to compare the result for the win.

**Rock-Paper-Scissors:**

This is a simple game that can be played with two players. There are three possible options to choose from which a player can choose. Then the winner is decided according to the move that is chosen by the player. The three possible moves are paper, rock, and scissors.

**Conditions of the result of the game:**

* If player 1 chooses rock and player 2 chooses rock: - Game ends with a tie
* If player 1 chooses rock and player 2 chooses paper: - player 2 wins the game
* If player 1 chooses rock and player 2 chooses scissors: - player 1 wins the game
* If player 1 chooses paper and player 2 chooses paper: - Game ends with a tie
* If player 1 chooses paper and player 2 chooses scissors: - player 2 wins the game
* If player 1 chooses paper and player 2 chooses rock: - player 1 wins the game
* If player 1 chooses scissors and player 2 chooses scissors: - Game ends with a tie
* If player 1 chooses scissors and player 2 chooses paper: - player 1 wins the game
* If player 1 chooses scissors and player 2 chooses rock: - player 2 wins the game

**Modules used in the program:**

**random:** Random module is a built-in python module that is used to get randomness in the program. This has many methods which can be used according to the need of the programmer such as randint (), shuffle (), and choice (). In this program, only the choice () method is used. This method gets a random value from a list or any other sequence such as a tuple etc. This is used to select the move of the computer from the given list of possible moves.

**pytest:** This is the basic unit testing module of python. This has various methods to run the unit tests on any program. In this program, the unit test is used by creating a class by inheriting its TestCase class. Then, three functions are defined which test the function for all the possible inputs. Three methods are created that test input for each possible move by the user.

**pylint:** It is a static code analyzer that is widely used to test the python code without even running it. It checks the code according to the PEP8 standards which are official python standards that dictate python coding practices. This assigns a score to the python code after evaluation.

**flake8:** It is also similar to the pylint. It mainly checks for errors relating to spaces; while pylint checks for naming conventions.

**Why Python is used?**

Python is a high-level language that can be used for almost any purpose in programming. It is widely used for data science and in the field of machine learning. Apart from that, it is one the easiest language to write code and can have a rich library of built-in as well as community modules. These modules can be used to build the code easily. For this specific program, python is used for its easy syntax and easy availability of random modules which can be directly used without installing anything extra. Also, pylint and flake8 are used which are static code analyzers. These can be used to analyze the code for any possible bad coding practices and can help the programmer to avoid any runtime errors.

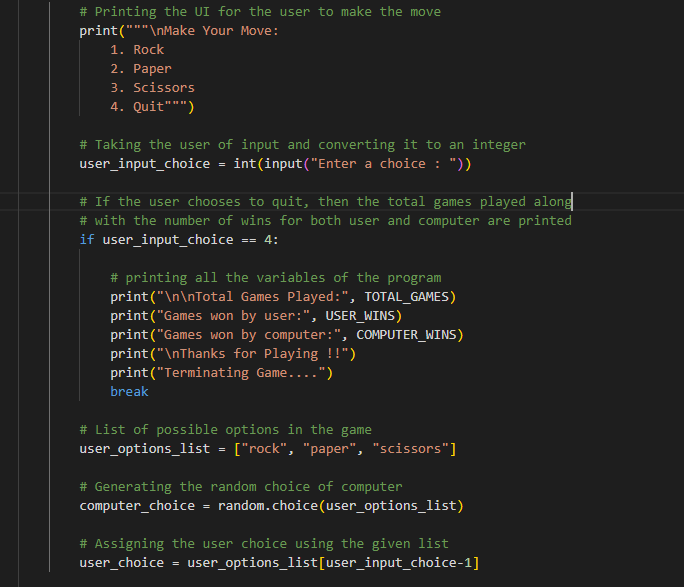
Process

**Implementation of each requirement**:

**1. User-friendly UI for the user to make a move**

In this program, I have chosen to implement a console-based UI. In this user have 4 options to choose from and these options are to be entered as an integer. Each option corresponds to a definite choice. Three of the options are for the three moves of the game and the fourth option is to exit or terminate the game. After that, input is used to accept the user input and convert the input to an integer. Then this integer is used to assign the user choice by using the values from the list of possible moves. It also prints the movement of the user along with the computer’s move.

**Coding implementation of this UI:**

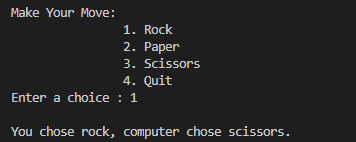


**Explanation:**

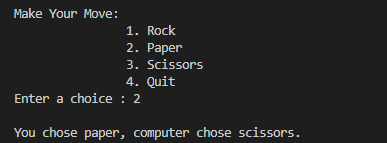
1. First, the possible inputs are printed to the console.
2. Then the user is prompted for the input. When a user enters the input, it is converted to an integer.
3. After that, if the user’s choice is 4, then the game is terminated and the required variables are printed.
4. Then, using the output user is assigned the move using the list of possible options.

**Screenshot of the working:**

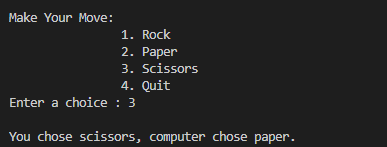
When the user enters 1:



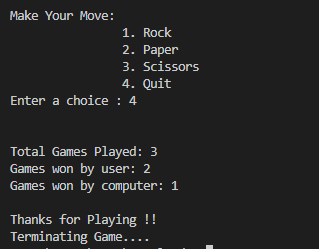
When the user enters 2:



When the user enters 3:



When the user enters 4:

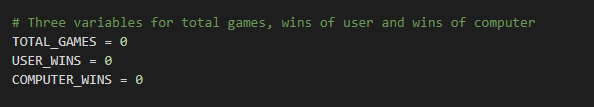


**2. Different variables for storing the user input as well as other variables required for processing.**

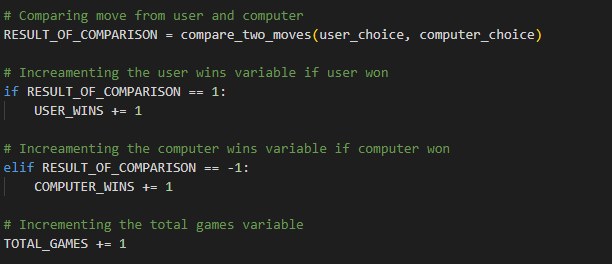
For this requirement mainly three variables are defined. These variables are used to store the total games played, games won by the user, and games won by the computer. These variables are from global scope so that they can be easily used and modified by the program.

**Implementation of the requirement:**

**Definition of the variables:**



**Manipulating variables in the code:**



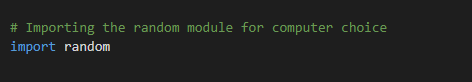
**Explanation of code:**

1. At first, all three variables are assigned to zero.
2. Then, the variable for total games is incremented with each iteration irrespective of the result of the game.
3. The other two variables are incremented according to the condition.
4. If the result of the method is 1 which means that the user has won the game; then the variable for user wins is incremented.
5. Otherwise, the variable for wins of the computer is incremented.
6. In this way, these variables are used to keep track of the important statistics of the game.

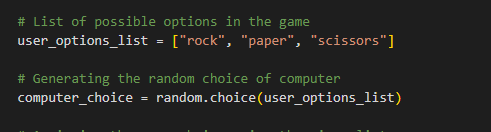
**3. Random module for the move of computer which chooses one option of out three possible options.**

The random module as already mentioned in the introduction is one of the built-in modules of python which is used in this code. This module is used to assign the computer’s move. In this program, the choice method of this module is used to choose a value from the list of possible options. This function selects one random value in each iteration. Then, this value is printed on the console along with the move of the user. Apart from that, this value is used to compare the user’s move to determine the winner of the game.

**Importing the module:**

****

**Using the function in the code:**

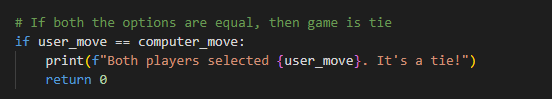
****

**4. Control statements to determine the winner of a single game.**

This is the major part of this game. This is enclosed in a method that receives two parameters. One parameter is the user’s move which is determined by the input entered by the user and the other is the computer’s move which is determined by the random module. After that, it returns one of three possible values. If both the moves are equal, then 0 is returned which means that game is a tie. However, it returns 1 if the user won the game; while -1 if the user loses the game. In this way, the decision of the particular is passed to the main driver code. Likewise, it also prints a message to tell the user who won the game so that game looks interesting.

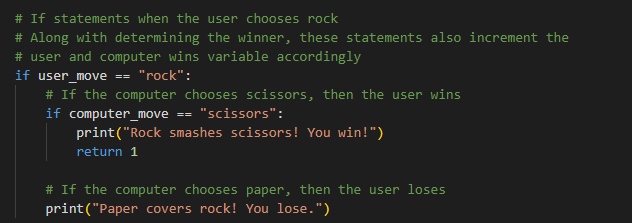
**Implementation of control block when both the moves are equal:**

When both the moves are equal, then first the message is printed which states that both players selected the same move and the game is a tie. After that, 0 is returned from the function.



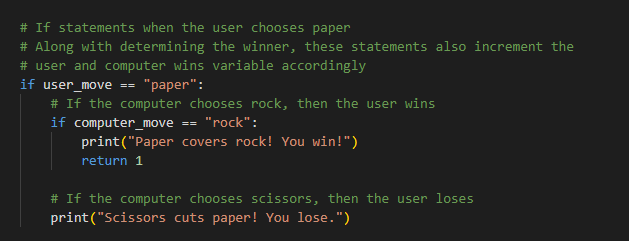
**Implementation of the control statement when the user’s choice is rock:**

For this, if the computer’s move is scissors, then the message is printed which states that the user won the game. Apart from that, 1 is returned and this also terminates this function. If the computer has selected the other move, then a message stating that the user lost the game is printed. In this case, -1 is not returned for every user choice, but a single return statement is used for the situations of user loss to make the code more efficient.



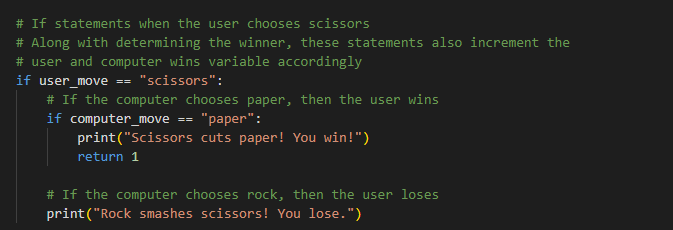
**Implementation of the control statement when the user’s choice is paper:**

This is also implemented in the same way as above. This follows the condition defined in the introduction of the game. The user wins the game if the rock is selected by the computer and the corresponding message is printed. However, the user loses the game if the computer has selected scissors. This also prints the message when the user loses, but does not returns -1 as it is shared by all the statements for the reasons stated above.

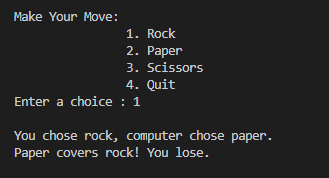


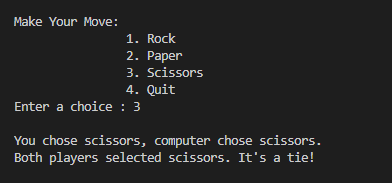
**Implementation of the control statement when the user’s choice is scissors:**

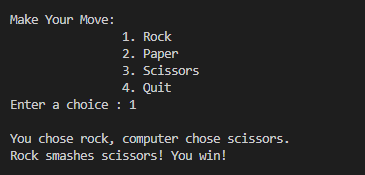
This is the implementation of the code when scissors are selected by the user. This also follows the same pattern as the above-mentioned statements. In this case, the user wins if the paper is chosen by the computer and the corresponding message is printed. It also only returns 1 for the win of the user but does not return anything for the user’s loss.

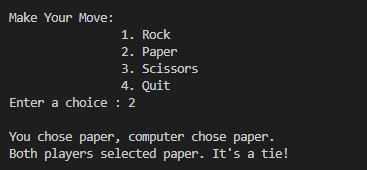


**Some sample executions of the method:**





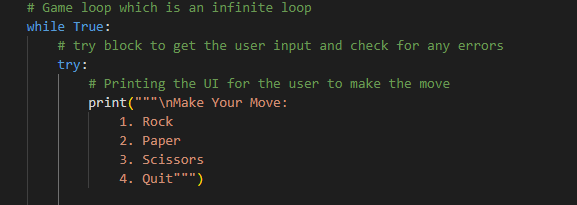




**5. Control statement to continue the game until the user terminates the game or a winner is decided.**

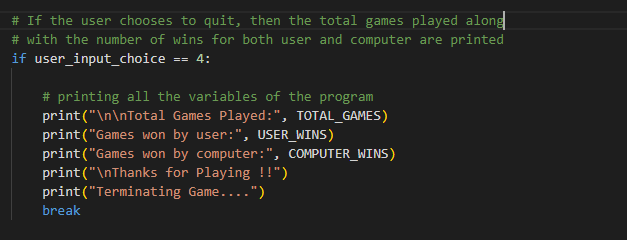
This is an infinite while loop which is a game loop. This loop first gives the user the UI and prompts the user for the user input. When the user enters the input is checked and processed accordingly. The processing of the input is already described in the first requirement. Then at the end of the iteration, the condition of winning the match is checked. This condition is if anyone reached 5 points.

**Implementation of the requirement:**

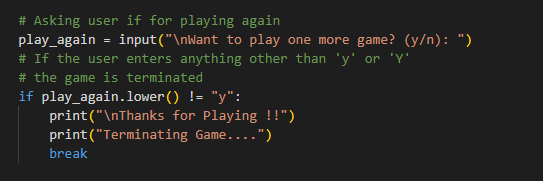
****

**Breaking the loop on both conditions of termination:**

1. The first condition of termination is when a user enters 4 as input which corresponds to the exit. This breaks the loop after printing the statistics about the game.

****

1. The second condition for termination is when the user chooses to not restart the game.

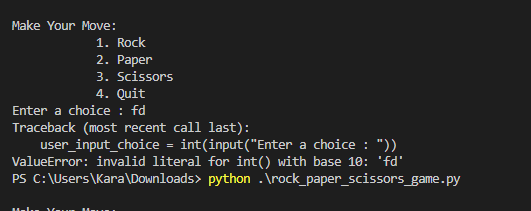
****

**6. Program must have error handling to handle unwanted inputs.**

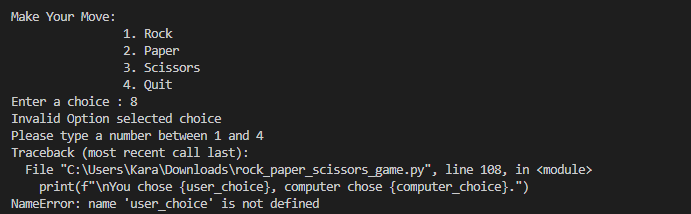
Error handling is required for the handling of any invalid inputs for the program. For example, suppose a user enters a string input while this program only expects an integer. This condition can cause the program to throw a runtime error. Another possible condition is when the user enters a number that is greater than 4.

Running the code without error handling:

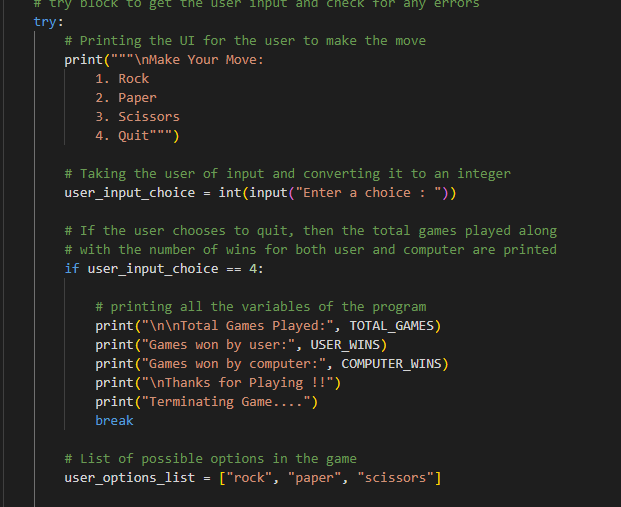
1. Error when the string is entered as an input



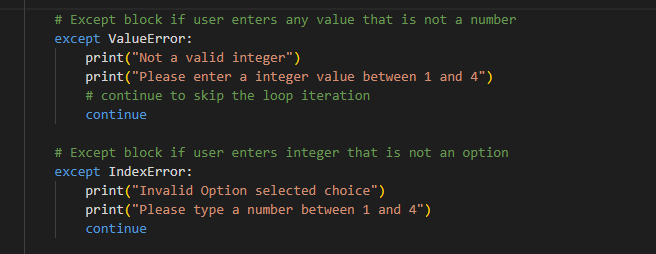
1. Error when a number greater than 4 is entered



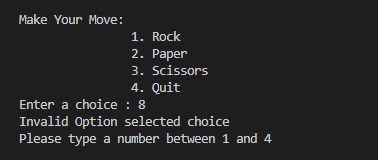
**Implementation of error handling**

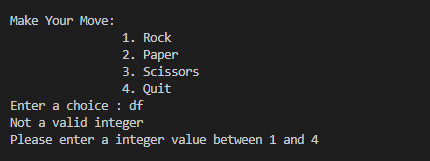


**Handling both errors:**



Running code after implementing error handling



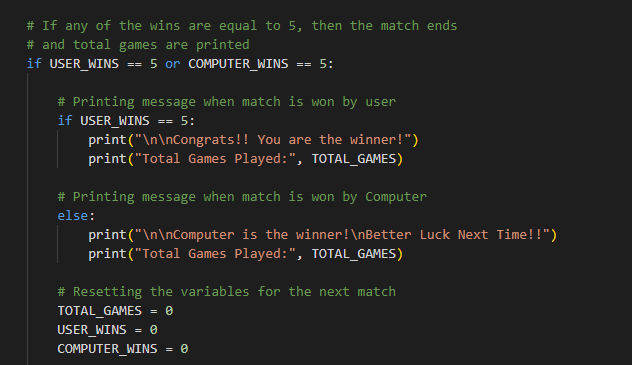


In this way, both errors and handled, and the user is given a useful message so that one can enter the required input.

**7. Control statement to check if the user or computer won the game.**

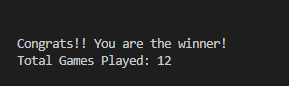
These are the control statements that determine who won the whole match. These statements check if the number of wins of any one player is equal to 5. If this condition matches then, a congratulatory message is printed. Apart from that, three statistics variables are reset to their initial value.

Implementation of these control statements:



Sample output:

Output when the user won the game:

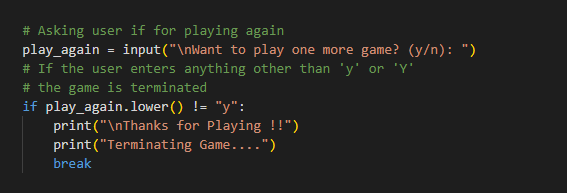


Similarly, the message when the computer won is also printed.

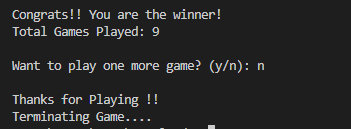
**8. Control statement to restart the game**

 This is the control statement to check and ask the user if wants to play the match again or not. This is part of the control statement block of the 7th requirement. This prompt the user to enter ‘y’ or ‘n’. If the user enters anything other than the letter Y, then the game is terminated. But if the user enters the letter Y then, the game restarts.

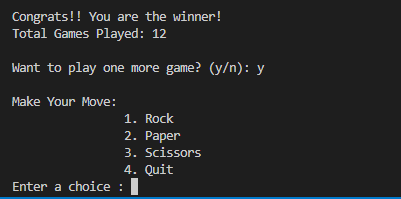
**Implementation of this requirement:**



Sample output when the user chose to quit the game:



Sample output when the user chose to restart the game:

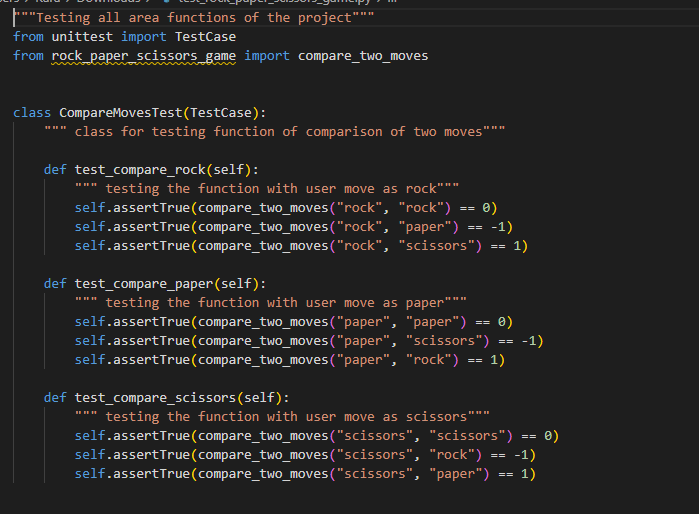


As can be seen from the output. When ‘y’ is entered game is started again.

**9. Unit Test of the function to compare the result for the win.**

This is a testing of the function which is used to compare the moves by both user and the computer. For this unit test module is used. This is implemented by creating a class that inherits the TestCase class of the unit test. Then it defines three functions, one for each user input.

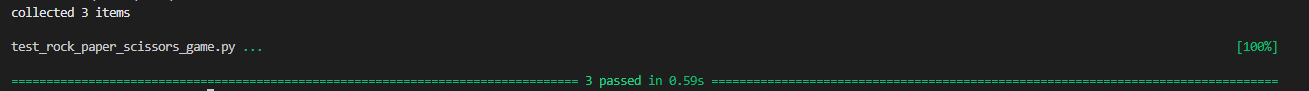
Implementation of unit Testing code:



**Explanation:**

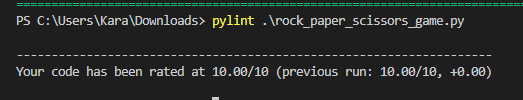
1. As it is clear from the above that three functions are defined.
2. The first function corresponds to the move of rock by the user. Then this function tests the output of this function by using all the possible inputs against their corresponding expected output.
3. In this way, all the required three inputs are tested.

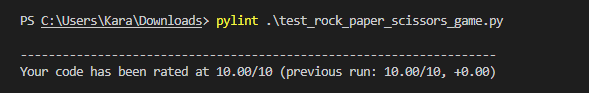
**Running the unit test using pytest:**



**Pylint static analysis:**

It is a static analysis tool that assigns the code a score that is given out of 10. The report for both the test code and implementation code of this analysis is given below:





As the score of both the codes is 100 %, these codes are up to the mark of the given code standards of python. It follows all the required coding standards and naming conventions are as required.

**Conclusion**

**Lessons learned:**

* Learned to use the TDD approach to test the code at regular intervals.
* Learned to use the random module and its various methods.
* Learned to use the infinite loop to process the data and create a console-based UI.
* This UI asks the user for input at the start of each loop iteration.
* Learned to test a function using pytest unit testing.
* Got to know about the static analyzers in python namely pylint and flake8.
* Learned the importance of static analysis and how it can help to improve the particular code and help to avoid common errors without running the code.
* Learned to use error handling on the code

**Possible improvements:**

* The UI can have made a graphical user interface that can be used by a user using the mouse cursor or keyboard input.
* The game can be played by two different users rather than playing against one player.
* The number of choices by the user can be increased to elevate the complexity of the game.

**How improvements can be done?**

**GUI implementation:**

The GUI implementation can be done using the various module available in python for GUI programming. One of the easiest and most used is the Tkinter module. This is a module that is very easy to implement and has a rich library of functions to implement the GUI interface.

**Playing a game with two different users:**

To make this possible user can be given turn one by one or the program can have two different interfaces each for both users.

**Adding more moves:**

This is one of the easiest improvements and it can make the game more fun. To make it possible new rules have to be made to compare with previously defined moves so that each input pair have a definite outcome.