

PROFESSIONAL TRAINING REPORT

At

**Sathyabama Institute of Science and
Technology**

(Deemed to be University)

**Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology Degree in**

Information Technology

By

**M.HARISH
(Reg.No.37120028)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY (DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

December 2020



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in



Department of Information Technology

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work **HARISH.M (Reg.No.37120028)** who underwent the professional training in "**GAME DEVELOPMENT USING PYTHON**" under our supervision from DEC 2020 TO JAN 2021

InternalGuide

MR.P.SARAVANAN

HeadoftheDepartment

DR.R.SUBHASHINI M.E.,Ph.D.,

submittedforVivavoceExaminationheldon_____

InternalExaminer

ExternalExaminer

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr.SUBHASHINI M.E., Ph.D.**, Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews. I would like to express my sincere and deep sense of gratitude to my Project Guide, **Mr. P.Saravanan M.E.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

Game development using Python is a basic project which uses different modules of Python to crack the logic behind different strategic games. The following project depicts a game center containing two different strategic games namely "Tic-Tac-Toe" and "Rock Paper Scissors". In this project it's about a single player strategy emphasizing logical thinking and planning. Tactical organization and execution are necessary and the decision-making skills and delivery of commands are left in the player's hand. Tic-Tac-Toe is a complete strategy game where the decision making is left in player's hand. Tic-tac-toe, noughts and crosses or Xs and Os is for two players, X and O, who take turns marking the spaces in a 3x3 grid who take turns marking the spaces in a 3x3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game while on the other hand Rock Paper Scissors, is a hand game usually played between two people, in which each player simultaneously chooses one the three traits. These traits are "rock", "paper", and "scissors". It has only two possible outcomes: a draw or a win for one player and a loss for the other. A player who decides to play rock will beat another player who has chosen scissors ("rock crushes scissors" or sometimes "blunts scissors"), but will lose to one who has played paper ("paper covers rock"); a play of paper will lose to a play of scissors. If both players choose the same shape, the game is tied and is usually immediately replayed to break the tie. Both the games a single player games with the other player being the computer.

CHAPTER 1

INTRODUCTION

1.1 OUTLINE OF THE PROJECT

The primary objective of this project is to develop a gaming centre using python which presents a menu displaying different games which can be played according to the user's choice. All the games displayed are single player games where the other player will be the computer itself. After the end of each game the player is asked if he/she wishes to play the game again. According to the desired output of the player i.e. "yes" or "no", either the game begins again from the start or else the player is taken back to the main menu happening according to the player's given input. The player can leave the anytime he wishes to by just pressing "exit" command and the game will terminate eventually. The use of various python modules makes the implementation of the project simpler compared to developing the project in any other programming language.

1.2 LITERATURE REVIEW

In this project, I am going to build Gaming Centre software which enables us to play different games with computer. With this program, user can play the game with bot when alone. Doing mathematics, and thinking about how you are doing it at the same time are not the easiest things to do. It is even more difficult if the player is not aware that he/she should be attempting both processes at the same time. They are likely to concentrate on the immediate task of "doing" the mathematics, rather than trying to access the deeper process. Yet it is this deeper process that is really at the heart of mathematics. In turn, accessing this deeper process requires in part some command of the appropriate rational/logical language so communication with yourself and others can proceed effectively and efficiently. This part discusses the possibilities of using player's explorations of the traditional strategy game "tic-tac-toe," and some extensions, to set up situations for player to discuss and examine this process.

1.3 DESIGNING THE SOFTWARE

Figure 1.0 shows a flowchart of the Tic-Tac-Toe program. The program starts by asking the player to choose their letter, X or O. Who takes the first turn is randomly chosen. Then the player and computer take turns making moves.

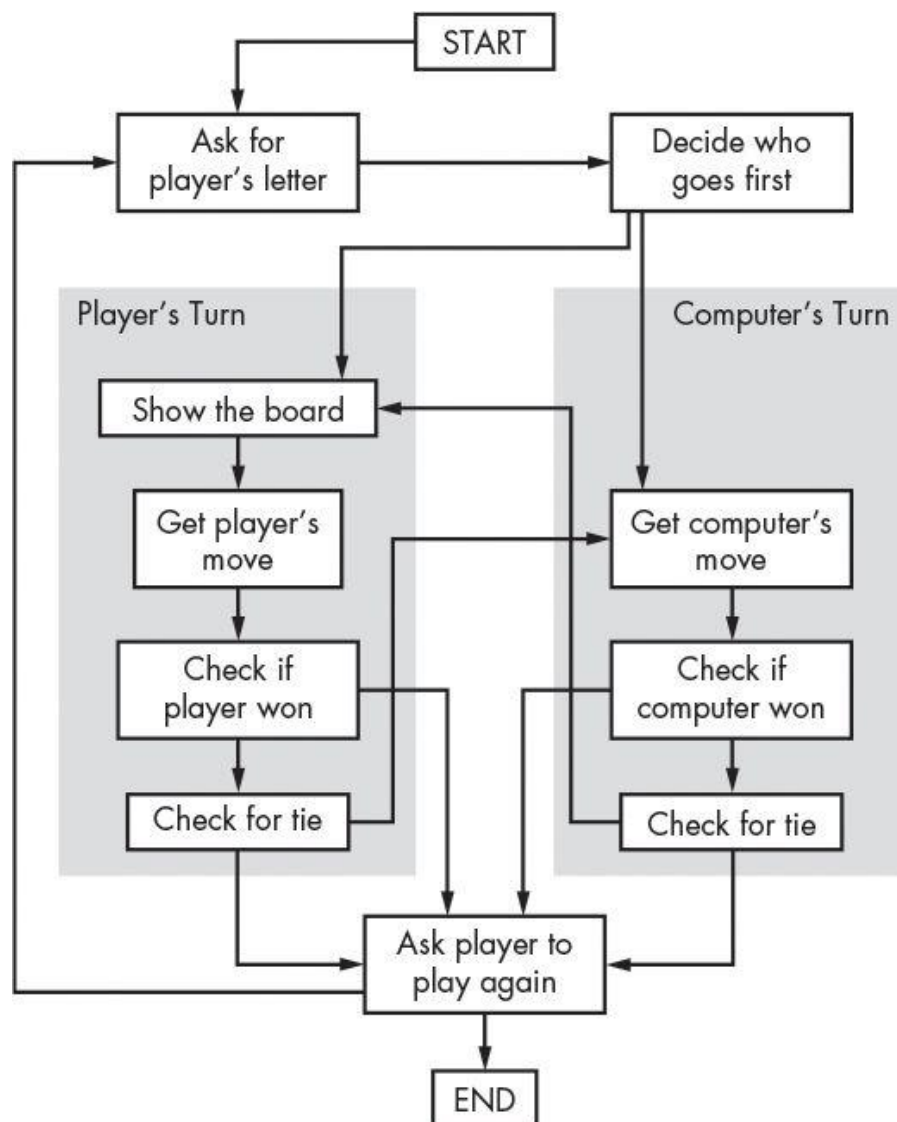


Fig 1.0 Flowchart of game's logic.

1.4 PROBLEM STATEMENT

Tic-tac-toe is a pencil-and-paper game for two players, X (ascii value 88) and O (ascii value 79), who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. Empty space is represented by (ascii value 95), and the X player goes first.

Here is an example figure 1.1 game won by the first player, X:



Fig 1.1 Trail of the game.

The function `nextMove` takes in a char player, and the 3x3 board as an array. Complete the function to print 2 space separated integers `r` and `c` which denote the row and column that will be marked in your next move. The top left position is denoted by (0,0).

How does it work? Your code is run alternately with the opponent bot for every move.

1.5 OBJECTIVE

The objective of Game Development using Python is as follows:

- To provide user with the trending skills
- The main aim of designing and developing
- Expand game play techniques as per cross-platform applications necessities.
- Make computer code that is supposed to create the game function smoothly.
- Plan and writing of computer code that controls and runs the graphics of a game on display.

CHAPTER 2

ALGORITHMS AND METHODS

2.1 GENERAL

Python is interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers are more compatible with python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program does not catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quietest way to debug a program is to add a few print statements to the source. The fast edit-test-debug cycle makes this simple approach very effective.

2.2 OVERVIEW

- **Python is Interpreted** – Python is processed at runtime by the interpreter.

You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

As shown in the above PYTHON plays a major role in all applications development and we are doing project based on basic syntax. As we can send anything according to our convenience.

2.3 MODULAR DESIGN

2.3.1 HARDWARE AND SOFTWARE REQUIREMENTS

- Programming language : Python 3.7.0 and above versions
- Hardware requirements : CPU
- Software requirements : Microsoft windows 10

2.3.2 DESIGN CRITERIA

- Simplicity : easily understood
- Efficiency : uses minimal resources
- Completeness : solves entire problem
- Not independent
- Simplicity by default

2.4 MODULES

2.4.1 ABOUT RANDOM MODULE

This module implements pseudo-random number generators for various distributions. For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement. On the real line, there are functions to compute uniform, normal (Gaussian), lognormal, negative exponential, gamma, and beta distributions. For generating distributions of angles, the von Mises distribution is available.

Almost all module functions depend on the basic function `random()`, which generates a random float uniformly in the semi-open range `[0.0, 1.0)`. Python uses the Mersenne Twister as the core generator. It produces 53-bit precision floats and has a period of $2^{19937}-1$. The underlying implementation in C is both fast and threadsafe. The Mersenne Twister is one of the most extensively tested random number generators in existence. However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes.

2.4.2 RANDINT IN PYTHON IMPLEMENTING RANDOM

`random.randint(a, b)` such that Return a random integer N such that $a \leq N \leq b$. Alias for `randrange(a, b+1)`. `randint()` is an inbuilt function of the random module in Python3. The random module gives access to various useful functions and one of them being able to generate random numbers, which is `randint()`. Parameters: (start, end) : Both of them must be integer type values.

2.5 SUMMARY

Python doesn't have pointers like other C-based languages, making it much more reliable. Along with that, errors never pass silently unless they're explicitly silenced. This allows you to see and read why the program crashed and where to correct your error, we can also use other programming languages like C, C++, Java but its effective to use and also have predefined functions to use in the Python language since the syntax for this programming language is simple to use and is more effective comparatively. So we have used Python for this project as python is fast enough for this project and allows one to produce maintainable features in record times.

CHAPTER 3

SYSTEM IMPLEMENTATION

3.1 GENERAL

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively. In-order to implement the Gaming Centre project firstly we need to install Python in our system. There are some steps to be followed to while installing Python in our system and the steps are as follows:

STEP 1:

Firstly, visit the link <https://www.python.org/downloads/windows/> and the window opened will show the Python Releases for Windows as well as Macs. There will be many versions of Python software and according to our requirement we need to click on that version of python software. And here in this project I have used the Python version of 3.7.3.

STEP 2:

After selecting our required version of python software we need to select the bit configuration of the system which will be displayed in the same window were the different versions are available. There will be different bit configurations for different

systems and in this project and according to my system bit configuration I have chosen 64 bit configuration.

STEP 3:

After clicking on the above two requirements the python software will start downloading and after downloading we need to follow the instructions that are shown by the software. After reading each and every instruction that are shown we need to click the Next button that is shown at the bottom of that window. In between the process we also need to set the destination directory that is where the python software needs to be stored or installed.

STEP 4:

After clicking the Next button finally the window with “Setup is successful” will be displayed and the installation of the Python project is successfully completed. This is not the final step of the Python installation and we still need to proceed further where the path should be set.

STEP 5:

After the completion of Python installation software we need to proceed to the next step which is to set the path. For setting the path we need to go the properties and in that Environment variables where we need to set the path. Setting the path is just copying the address of the Python where it is stored in the path location.

STEP 6:

The directory of the Python should be pasted in the path location and the changes we have made should be saved. This is the final step of the Python installation and after completion of these steps we can proceed to our project by using Python that we have just installed.

3.2 IMPLEMENTATION PHASE

First of all you need to print the menu depicting the available games that the player can play. Using “choice” as a switch statement one can take input from the user and run the game the user wishes to play. On the main menu screen the player

is provided with three options namely “Tic Tac Toe”, “Rock Paper Scissors” and an “Exit” button for terminating the program.*

3.2.1 Implementation of Rock Paper Scissors

First, we import randint from the random module. This is how our computer opponent will play. Then we create a list of play options:

```
t = ["Rock", "Paper", "Scissors"]
```

There are three possible plays you and the computer can make on each turn, “Rock”, “Paper” and “Scissors”. Next we setup our players, the computer and the user:

```
computer = t[randint(0,2)]
```

```
player = False
```

We assign a random play to the computer using our list, t, and the randint function. Why (0,2)? Remember that computers start counting at 0. So “Rock” is in the 0 position, “Paper” is in the 1, and so on. Unlike playing RPS with friends in meatspace, the computer has made its play and is waiting for you to take your turn. Also unlike playing RPS with friends in meat space, the computer isn’t go to cheat and change its play after you make yours. We set you, the player, to False. Why? I’m glad you asked. Let’s take a look at the body of our program the while loop:

Once the while loop starts, the computer will patiently wait for you to make a play. As soon as you take your turn, your status changes from False to True because any value assigned to the variable player makes player True. We use the input() function to pass the new value to the variable player. Your input will determine which statement is triggered below.

Using nested if/elif/else statements, we check every possible outcome of the game and return a message stating the winner, a tie, or an error.

We use else at the end to catch anything that isn’t “Rock”, “Paper” or “Scissors”. Finally we reset the player value to False to restart the while loop.

3.2.2 Implementation of Tic Tac Toe

REPRESENTING BOARD AS DATA

First, you must figure out how to represent the board as data in a variable. On paper, the Tic-Tac-Toe board is drawn as a pair of horizontal lines and a pair of vertical lines, with an X, O, or empty space in each of the nine spaces. In the program, the Tic-Tac-Toe board is represented as a list of strings like the ASCII art of Hangman. Each string represents one of the nine spaces on the board. The strings are either 'X' for the X player, 'O' for the O player, or a single space ' ' for a blank space.

Remember that we're laying out our board like a number pad on a keyboard. So if a list with 10 strings was stored in a variable named board, then board[7] would be the top-left space on the board, board[8] would be the top-middle space, board[9] would be the top-right space, and so on. The program ignores the string at index 0 in the list. The player will enter a number from 1 to 9 to tell the game which space they want to move on.

STRATEGIZING WITH THE GAME AI:

The AI needs to be able to look at the board and decide which types of spaces it will move on. To be clear, we will label three types of spaces on the Tic-Tac-Toe board: corners, sides, and the center. The chart in Figure 3.0 shows what each space is. The AI's strategy for playing Tic-Tac-Toe will follow a simple algorithm—a finite series of instructions to compute a result. A single program can make use of several different algorithms. An algorithm can be represented with a flowchart. The Tic-Tac-Toe AI's algorithm will compute the best move to make, as shown in Figure 3.1.

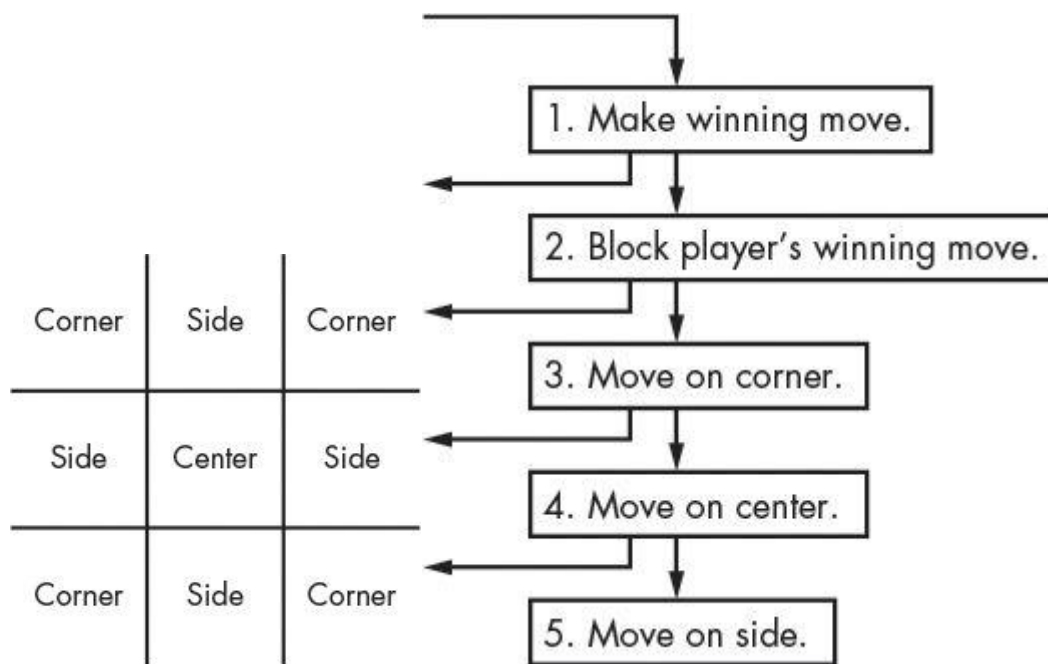


Fig 3.0Game's AI.**Fig 3.1Game's AI flowchart.**

PRINTING THE BOARD ON THE SCREEN:

The `drawBoard()` function prints the game board represented by the `board` parameter. Remember that the board is represented as a list of 10 strings, where the string at index 1 is the mark on space 1 on the Tic-Tac-Toe board, and so on. The string at index 0 is ignored. Many of the game's functions work by passing a list of 10 strings as the board. Be sure to get the spacing right in the strings; otherwise, the board will look funny when printed on the screen. Here are some example calls (with an argument for `board`) to `drawBoard()` and what the function would print. The program takes each string and places it on the board in number order according to the keyboard number pad from Figure 10-1, so the first three strings are the bottom row of the board, the next three strings are the middle, and the last three strings are the top.

LETTING THE PLAYER CHOOSE X OR O:

The `inputPlayerLetter()` function asks whether the player wants to be X or O. The while loop's condition contains parentheses, which means the expression inside the

parentheses is evaluated first. If letter has the value 'X' or 'O', then the loop's condition is False and lets the program execution continue past the while block. If the condition is true, the program will keep asking the player to choose a letter until the player enters an X or O. The string returned by the call to input() to uppercase letters with the upper() string method. A figure representing the alogorithm

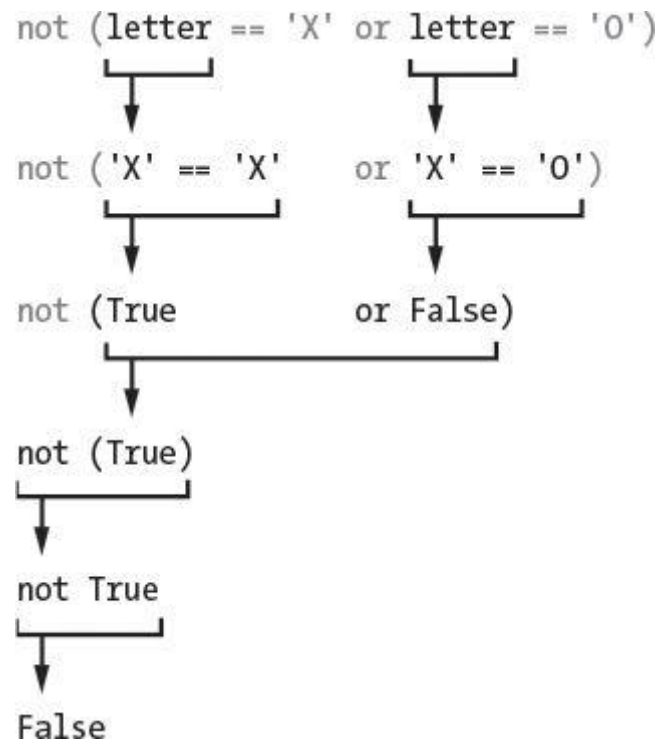


Fig 3.2 Selection of X and O.

DECIDING WHO GOES FIRST:

The whoGoesFirst() function does a virtual coin flip to determine whether the computer or the player goes first. The coin flip is done with a call to random.randint(0,). There is a 50 percent chance the function returns 0 and a 50 percent chance the function returns 1. If this function call returns a 0, the whoGoesFirst() function returns the string 'computer'. Otherwise, the function returns the string 'player'. The code that calls this function will use the return value to determine who will make the first move of the game.

CHECKING WHETHER THE PLAYER WON:

The `bo` and `le` names are shortcuts for the board and letter parameters. These shorter names mean you have less to type in this function. Remember, Python doesn't care what you name your variables. There are eight possible ways to win at Tic-Tac-Toe: you can have a line across the top, middle, or bottom rows; you can have a line down the left, middle, or right columns; or you can have a line across either of the two diagonals. Each line of the condition checks whether the three spaces for a given line are equal to the letter provided (combined with the `and` operator). You combine each line using the `or` operator to check for the eight different ways to win. This means only one of the eight ways must be true in order for us to say that the player who owns the letter in `le` is the winner.

DUPLICATING THE BOARD DATA:

The `getBoardCopy()` function allows you to easily make a copy of a given 10-string list that represents a Tic-Tac-Toe board in the game. When the AI algorithm is planning its moves, it will sometimes need to make modifications to a temporary copy of the board without changing the actual board. In those cases, we call this function to make a copy of the board's list. Right now, the list stored in `boardCopy` is just an empty list. The `for` loop will iterate over the board parameter, appending a copy of the string values in the actual board to the duplicate board. After the `getBoardCopy()` function builds up a copy of the actual board, it returns a reference to this new board in `boardCopy`, not to the original one in `board`.

CHECKING WHETHER A SPACE ON THE BOARD IS FREE:

Given a Tic-Tac-Toe board and a possible move, the simple `isSpaceFree()` function returns whether that move is available or not. Remember that free spaces in the board lists are marked as a single-space string. If the item at the space's index is not equal to ' ', then the space is taken.

LETTING THE PLAYER ENTER A MOVE:

The `getPlayerMove()` function asks the player to enter the number for the space they want to move on. The loop makes sure the execution doesn't continue until the

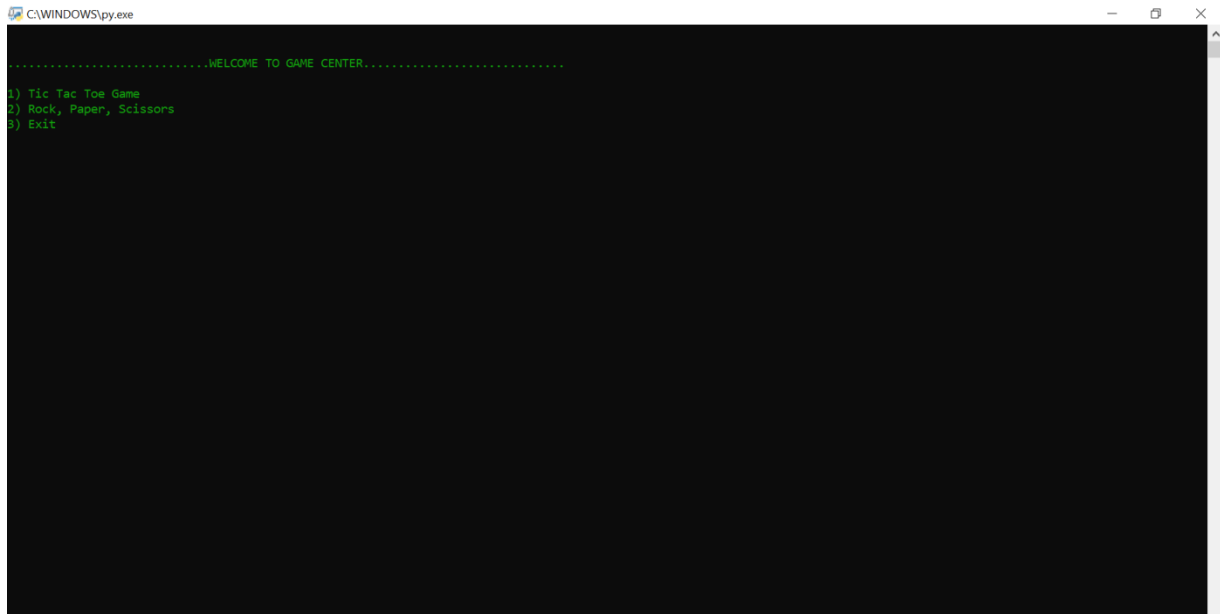
player has entered an integer between 1 and 9. It also checks that the space entered isn't already taken, given the Tic-Tac-Toe board passed to the function for the board parameter. The two lines of code inside the while loop simply ask the player to enter a number from 1 to 9. The expression on the left side checks whether the player's move is equal to '1', '2', '3', and so on up to '9' by creating a list with these strings (with the `split()` method) and checking whether move is in this list. In this expression, `'1 2 3 4 5 6 7 8 9'.split()` evaluates to `['1', '2', '3', '4', '5', '6', '7', '8', '9']`, but the former is easier to type. The expression on the right side checks whether the move the player entered is a free space on the board by calling `isSpaceFree()`. Remember that `isSpaceFree()` returns `True` if the move you pass is available on the board. Note that `isSpaceFree()` expects an integer for move, so the `int()` function returns an integer form of move. The not operators are added to both sides so that the condition is `True` when either of these requirements is unfulfilled. This causes the loop to ask the player again and again for a number until they enter a proper move. Finally, line 68 returns the integer form of whatever move the player entered. `input()` returns strings, so the `int()` function is called to return an integer form of the string.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 RESULT AND ANALYSIS

On executing the python code of GAMING CENTRE we get the output according to the corresponding code.



```
.....WELCOME TO GAME CENTER.....
1) Tic Tac Toe Game
2) Rock, Paper, Scissors
3) Exit
```

Fig 4.1 Output of Gaming centre program.

The above screenshot is the output of the code gaming centre, the functioning of the code is as follows, first we have compiled the main function of the program which on implementation display's the menu showed above displaying the menu. Then we use choice keyword to take input from the user and run the code of the game choice the user selected.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 SUMMARY

This game is very popular and is fairly simple by itself. It is a two player game. The object we are scripting is composed of 9 cubes with an X/O texture pre-set and a back pane prim as the root prim. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. The link order of the child prims tic -tac-toe is assumed to be in any sequential reading order (left to right, top to bottom or similar).The game can be generalized to an m, n, k -game in which two players alternate placing stones of their own colour or letter on an $m \times n$ board, with the goal of getting k of their own colour or letter in a row.Tic-tac-toe is the (3,3,3)-game. There is no such use of complex pseudo code and algorithm. It can also be generalized as n^d game. Tic-tac-toe is the game where n equals 3 and d equals 2. If played properly, the game will end in a draw making tic-tac-toe a futile game. Hence, tic-tac-toe is most often played by young children. Due to the simplicity of tic-tac-toe, it is often used as a pedagogical tool.

5.2 FUTURE WORKS

- Keyboard functions will be added.
- I want to design more complex boards for the game in future.
- In future in Android application development I would like to learn the OpenGL 2D graphics which can be implemented in the application which will improve the working and presentation of the application.
- Also, I would like to implement a 4x4 board game.
- Python programming language is best used for application development, web application or web development, game development, system administration, scientific computing and so I am looking forward to develop such games in future.

5.3 CONCLUSION

- In the end I would like to conclude that my aim to make this project was to research in the field of GAME DEVELOPMENT and also implementation of Artificial Intelligence by developing the logic for the game. Some scopes of improvements are also there in the project which will be rectified in the future advancements of the project.
- The design of project taught us about programming and also the documentation involved with creating this project. The limitations of this project were time constraints and limited testing time. Creating the whole project and documenting our design process went well.

REFERENCES

- [1] Python Basics, <https://docs.python.org/3/> , accessed on July 2019. Used to know the basics of python and installation on python software.
- [2] Basics of Tic Tac Toe, <http://inventwithpython.com/invent4thed/chapter10.html> , accessed on July 2019 which was used to develop the logic behind the game.
- [3] Basics of Rock Paper Scissors, <https://thehelloworldprogram.com/python/python-game-rock-paper-scissors/> , accessed on July 2019 which was used to develop the logic behind the game.
- [4] Game Development Using Python(Android+IOS), Udemy, <https://www.udemy.com/course/android-game-development-using-python-build-12-apps-games/learn> accessed on July 2019 which was used for better understanding of game development.

APPENDIX

A) SOURCE CODE

The following is the code of the project “Game Development Using Python”. The code contains all the packages that are required for the project and the functions that are required are also written in the code.

```
import random

import sys

from random import randint

def get_input():

    if sys.version_info >= (3, 0):

        return input()

    else:

        return raw_input()

def drawBoard(board):

    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])

    print('-----')

    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])

    print('-----')

    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])

def inputPlayerLetter():

    letter = "

    while not (letter == 'X' or letter == 'O'):

        print('Do you want to be X or O?')
```



```

letter = get_input().upper()

if letter == 'X':

    return ['X', 'O']

else:

    return ['O', 'X']

defwhoGoesFirst():

    ifrandom.randint(0, 1) == 0:

        return 'computer'

    else:

        return 'player'

defplayAgain():

    print('Do you want to play again? (yes or no)')

    returnget_input().lower().startswith('y')

defmakeMove(board, letter, move):

    ifisSpaceFree(board,move):

        board[move] = letter

    else:

        raise Exception("makeMove: the field is not empty!")

defisWinner(bo, le):

    return ((bo[7] == le and bo[8] == le and bo[9] == le) or

            (bo[4] == le and bo[5] == le and bo[6] == le) or

            (bo[1] == le and bo[2] == le and bo[3] == le) or

            (bo[7] == le and bo[4] == le and bo[1] == le) or

```

```

    (bo[8] == le and bo[5] == le and bo[2] == le) or

    (bo[9] == le and bo[6] == le and bo[3] == le) or

    (bo[7] == le and bo[5] == le and bo[3] == le) or

    (bo[9] == le and bo[5] == le and bo[1] == le))

defgetBoardCopy(board):

    dupeBoard = []

    for i in board:

        dupeBoard.append(i)

    returndupeBoard

defisSpaceFree(board, move):

    return board[move].isdigit()

defgetPlayerMove(board):

    move = ' '

    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board, int(move)):

        print('What is your next move? (1-9)')

        move = get_input()

    returnint(move)

defchooseRandomMoveFromList(board, movesList):

    possibleMoves = []

    for i in movesList:

        ifisSpaceFree(board, i):

            possibleMoves.append(i)

    iflen(possibleMoves) > 0:

```

```

return random.choice(possibleMoves)

else:

return None

def getComputerMove(board, computerLetter):

if computerLetter == 'X':

playerLetter = 'O'

else:

playerLetter = 'X'

for i in range(1, 10):

copy = getBoardCopy(board)

if isSpaceFree(copy, i):

makeMove(copy, computerLetter, i)

if isWinner(copy, computerLetter):

return i

for i in range(1, 10):

copy = getBoardCopy(board)

if isSpaceFree(copy, i):

makeMove(copy, playerLetter, i)

if isWinner(copy, playerLetter):

return i

move = chooseRandomMoveFromList(board, [1, 3, 7, 9])

if move != None:

return move

```

```

if isSpaceFree(board, 5):

    return 5

    return chooseRandomMoveFromList(board, [2, 4, 6, 8])

def isBoardFull(board):

    for i in range(1, 10):

        if isSpaceFree(board, i):

            return False

    return True

def main():

    d=0

    while(d!=1):

        print("")

        print("")

        print(".....WELCOME TO GAME CENTER.....")

        print("")

        print('1) Tic Tac Toe Game')

        print("2) Rock, Paper, Scissors")

        print("3) Exit")

        print("")

        choice= input()

        if choice=='2':

            t=['Rock','Paper','Scissors']

```

```
comp=t[randint(0,2)]

player=False

x=0

print('you dont want to continue press "n"')

while player==False and x!=1:

player=input('Rock,Paper,Scissors?')

if player==comp:

print("Tie")

elif player=="Rock":

if comp=="paper":

print("you lose!")

else:

print("You WIN")

elif player=="Paper":

if comp=="Scissors":

print("you lose!")

else:

print("You WIN")

elif player=="Scissors":

if comp=="Rock":

print("you lose!")

else:

print("You WIN")
```

```
elif player=='n':  
    x=1  
  
else:  
  
print("Not VALID")  
  
player=False  
  
comp=t[randint(0,2)]  
  
elif choice=='1':  
  
print('Welcome to Tic Tac Toe!')  
  
random.seed()  
  
while True:  
  
theBoard = [' '] * 10  
  
for i in range(9,0,-1):  
  
theBoard[i] = str(i)  
  
playerLetter, computerLetter = inputPlayerLetter()  
  
turn = whoGoesFirst()  
  
print('The ' + turn + ' will go first.')  
  
gamelsPlaying = True  
  
while gamelsPlaying:  
  
if turn == 'player':  
  
drawBoard(theBoard)  
  
move = getPlayerMove(theBoard)  
  
makeMove(theBoard, playerLetter, move)  
  
if isWinner(theBoard, playerLetter):
```

```
drawBoard(theBoard)

print('Hooray! You have won the game!')

gameIsPlaying = False

else:

    if isBoardFull(theBoard):

        drawBoard(theBoard)

        print('The game is a tie!')

        break

    else:

        turn = 'computer'

    else:

        move = getComputerMove(theBoard, computerLetter)

        makeMove(theBoard, computerLetter, move)

        if isWinner(theBoard, computerLetter):

            drawBoard(theBoard)

            print('The computer has beaten you! You lose.')

            gameIsPlaying = False

        else:

            if isBoardFull(theBoard):

                drawBoard(theBoard)

                print('The game is a tie!')

                break

            else:
```

```
turn = 'player'
```

```
if not playAgain():
```

```
    break
```

```
elif choice=='3':
```

```
    d=1
```

```
else:
```

```
    print("NOT A VALID INPUT")
```

```
if __name__ == "__main__":
```

```
    main()
```


B)SCREENSHOTS:

A screenshot of a Python IDE window titled '*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is written in Python and uses syntax highlighting. It defines several functions: 'get__input' which checks the Python version to use either 'input' or 'raw_input'; 'drawBoard' which prints a 3x3 board state using a list 'board'; 'inputPlayerLetter' which prompts the user to choose 'X' or 'O' and returns the chosen letters; 'whoGoesFirst' which uses 'random.randint' to decide if the computer or player goes first; and 'playAgain' which asks the user if they want to play again. The status bar at the bottom right shows 'Ln: 34 Col: 51'.

```
import random
import sys
from random import randint
def get__input () :
    if sys.version__info >= ( 3, 0) :
        return input ()
    else:
        return raw__input ()

def drawBoard ( board) :
    print ( ' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print ( '-----')
    print ( ' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print ( '-----')
    print ( ' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])


def inputPlayerLetter () :
    letter = "
    while not ( letter == 'X' or letter == 'O') :
        print ( 'Do you want to be X or O?')
        letter = get__input () .upper ()
    if letter == 'X':
        return ['X', 'O']
    else:
        return ['O', 'X']

def whoGoesFirst () :
    if random.randint ( 0, 1) == 0:
        return 'computer'
    else:
        return 'player'

def playAgain () :
    print ( 'Do you want to play again? ( yes or no ) ' ) |
    return get__input () .lower () .startswith ( 'y')
```

Ln: 34 Col: 51

Fig B.1 Screenshot of Actual Code



The screenshot shows a Python IDE window titled "*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code is as follows:

```
def makeMove ( board, letter, move) :  
    if isSpaceFree ( board,move) :  
        board[move] = letter  
    else:  
        raise Exception ( "makeMove: the field is not empty!")  
  
def isWinner ( bo, le) :  
    return ( ( bo[7] == le and bo[8] == le and bo[9] == le) or  
            ( bo[4] == le and bo[5] == le and bo[6] == le) or  
            ( bo[1] == le and bo[2] == le and bo[3] == le) or  
            ( bo[7] == le and bo[4] == le and bo[1] == le) or  
            ( bo[8] == le and bo[5] == le and bo[2] == le) or  
            ( bo[9] == le and bo[6] == le and bo[3] == le) or  
            ( bo[7] == le and bo[5] == le and bo[3] == le) or  
            ( bo[9] == le and bo[5] == le and bo[1] == le) )  
  
def getBoardCopy ( board) :  
    dupeBoard = []  
    for i in board:  
        dupeBoard.append ( i)  
    return dupeBoard  
  
def isSpaceFree ( board, move) :  
    return board[move].isdigit ()  
  
def getPlayerMove ( board) :  
    move = ''  
    while move not in '1 2 3 4 5 6 7 8 9'.split () or not isSpaceFree ( board, int ( move ) ):  
        print ( 'What is your next move? ( 1-9) ' )  
        move = get__input ()  
    return int ( move)  
  
def chooseRandomMoveFromList ( board, movesList) :  
    possibleMoves = []
```

The status bar at the bottom right indicates "Ln: 69 Col: 47".

Fig B.2 Screenshot of Actual Code



```
*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...
File Edit Format Run Options Window Help
possibleMoves = []
for i in movesList:
    if isSpaceFree ( board, i) :
        possibleMoves.append ( i)
if len ( possibleMoves) > 0:
    return random.choice ( possibleMoves)
else:
    return None

def getComputerMove ( board, computerLetter) :
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'
    for i in range ( 1, 10) :
        copy = getBoardCopy ( board)
        if isSpaceFree ( copy, i) :
            makeMove ( copy, computerLetter, i)
            if isWinner ( copy, computerLetter) :
                return i
    for i in range ( 1, 10) :
        copy = getBoardCopy ( board)
        if isSpaceFree ( copy, i) :
            makeMove ( copy, playerLetter, i)
            if isWinner ( copy, playerLetter) :
                return i
    move = chooseRandomMoveFromList ( board, [1, 3, 7, 9])
    if move != None:
        return move
    if isSpaceFree ( board, 5) :
        return 5
    return chooseRandomMoveFromList ( board, [2, 4, 6, 8])

def isBoardFull ( board) :
    for i in range ( 1, 10) :
        if isSpaceFree ( board, i) :
```

Ln: 104 Col: 26

Fig B.3 Screenshot of Actual Code



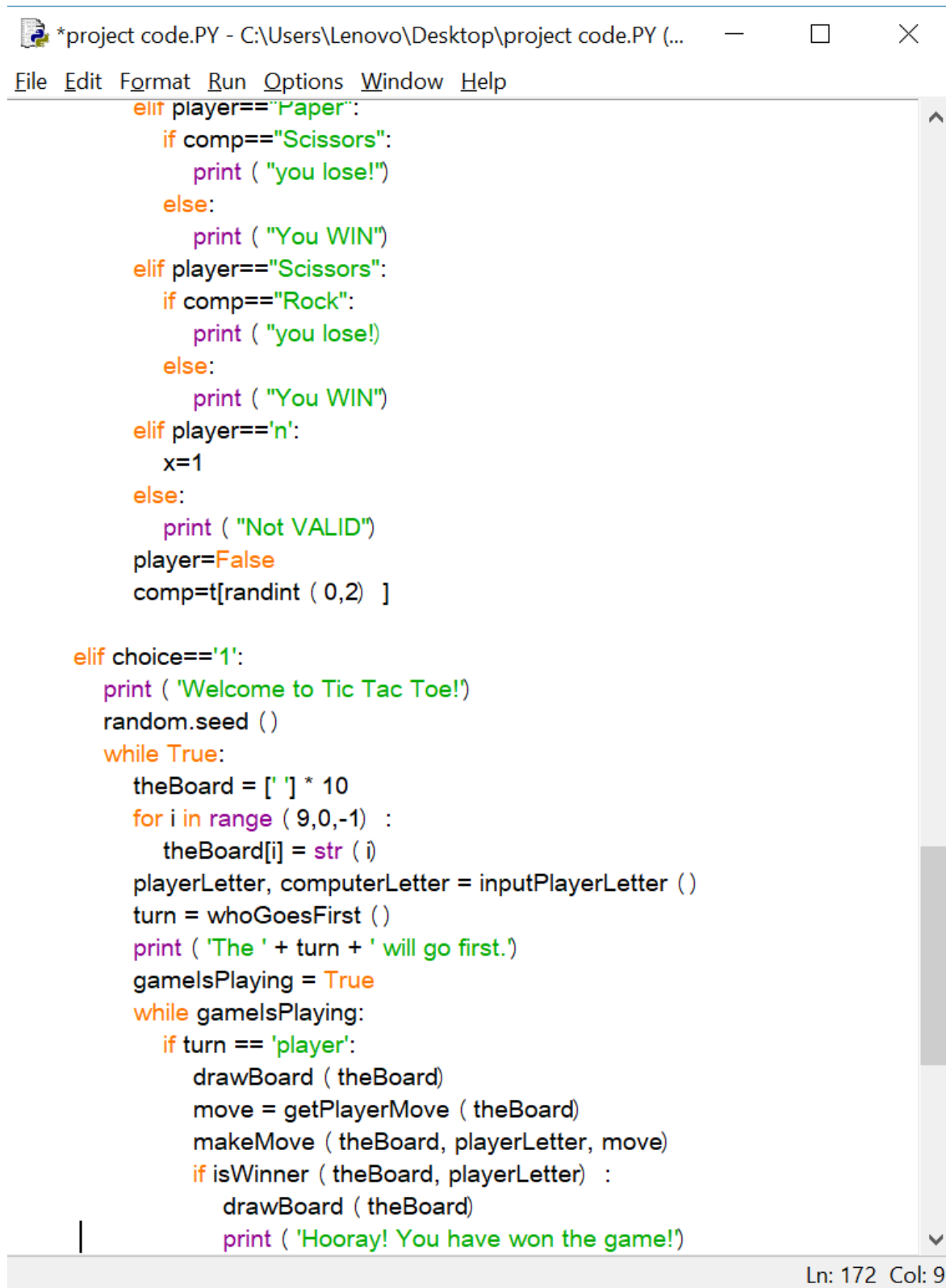
```
*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...)
```

```
File Edit Format Run Options Window Help
```

```
def isBoardFull ( board) :  
    for i in range ( 1, 10) :  
        if isSpaceFree ( board, i) :  
            return False  
    return True  
  
def main () :  
    d=0  
    while ( d!=1) :  
        print ( "  
        print ( "  
        print ( ".....WELCOME TO GAME CENTER.....  
        print ( "  
        print ( '1)  Tic Tac Toe Game'  
        print ( "2)  Rock, Paper, Scissors")  
        print ( "3)  Exit")  
        print ( "  
        choice= input ( )  
  
        if choice=='2':  
            t=['Rock','Paper','Scissors']  
            comp=t[randint ( 0,2) ]  
            player=False  
            x=0  
            print ( 'you dont want to continue press "n")  
            while player==False and x!=1:  
                player=input ( 'Rock,Paper,Scissors?')  
                if player==comp:  
                    print ( "Tie")  
                elif player=="Rock":  
                    if comp=="paper":  
                        print ( "you lose!")  
                    else:  
                        print ( "You WIN")  
                elif player=="Paper":  
                    if comp=="Scissors":
```

Ln: 138 Col: 9

Fig B.4 Screenshot of Actual Code



```
*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...
File Edit Format Run Options Window Help
elif player=="Paper":
    if comp=="Scissors":
        print ( "you lose!")
    else:
        print ( "You WIN")
elif player=="Scissors":
    if comp=="Rock":
        print ( "you lose!")
    else:
        print ( "You WIN")
elif player=='n':
    x=1
else:
    print ( "Not VALID")
player=False
comp=t[randint ( 0,2) ]

elif choice=='1':
    print ( 'Welcome to Tic Tac Toe!')
    random.seed ( )
    while True:
        theBoard = [ ' ' ] * 10
        for i in range ( 9,0,-1) :
            theBoard[i] = str ( i)
        playerLetter, computerLetter = inputPlayerLetter ( )
        turn = whoGoesFirst ( )
        print ( 'The ' + turn + ' will go first.')
        gamelsPlaying = True
        while gamelsPlaying:
            if turn == 'player':
                drawBoard ( theBoard)
                move = getPlayerMove ( theBoard)
                makeMove ( theBoard, playerLetter, move)
                if isWinner ( theBoard, playerLetter) :
                    drawBoard ( theBoard)
                    print ( 'Hooray! You have won the game!')
```

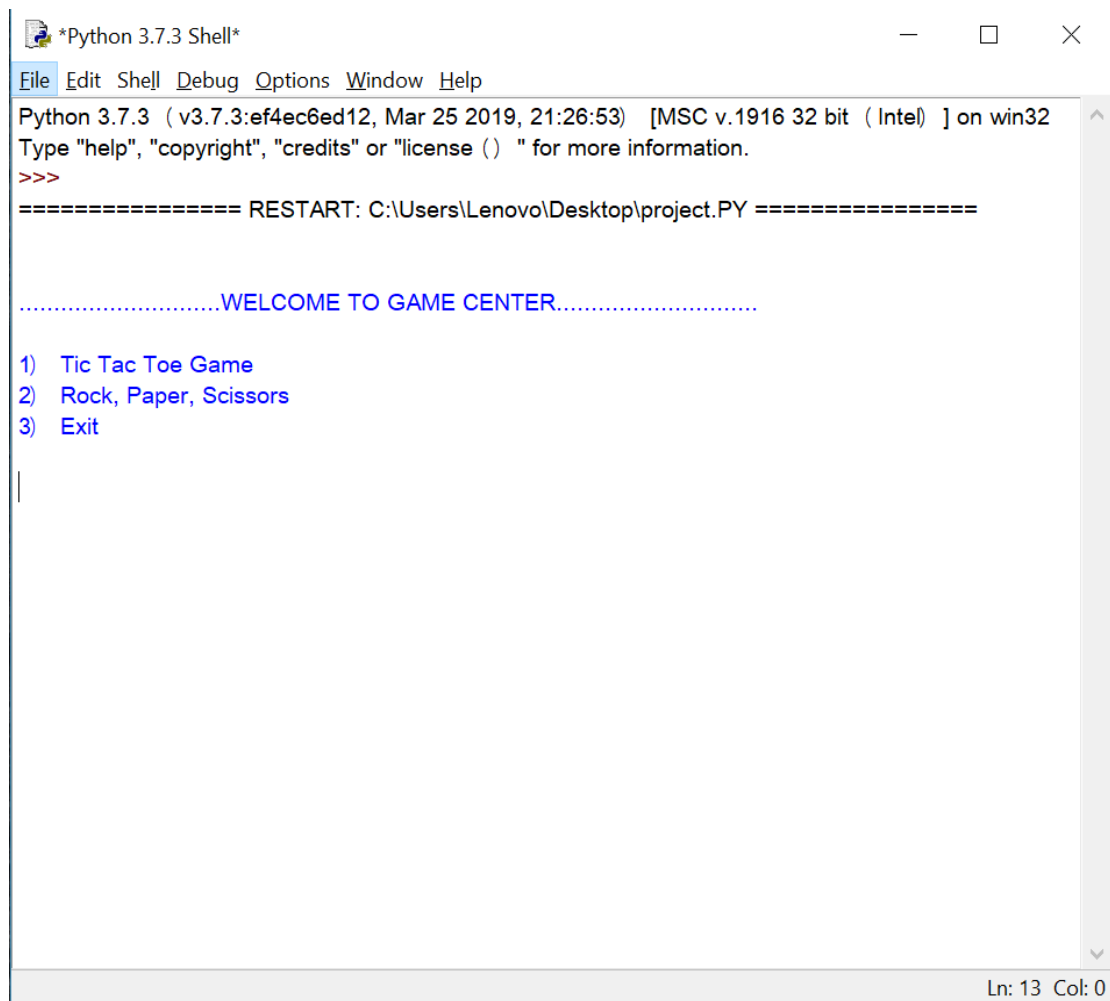
Ln: 172 Col: 9

Fig B.5 Screenshot of Actual Code



```
*project code.PY - C:\Users\Lenovo\Desktop\project code.PY (...
File Edit Format Run Options Window Help
drawBoard ( theBoard)
move = getPlayerMove ( theBoard)
makeMove ( theBoard, playerLetter, move)
if isWinner ( theBoard, playerLetter) :
    drawBoard ( theBoard)
    print ( 'Hooray! You have won the game!')
    gamelsPlaying = False
else:
    if isBoardFull ( theBoard) :
        drawBoard ( theBoard)
        print ( 'The game is a tie!')
        break
    else:
        turn = 'computer'
else:
    move = getComputerMove ( theBoard, computerLetter)
    makeMove ( theBoard, computerLetter, move)
    if isWinner ( theBoard, computerLetter) :
        drawBoard ( theBoard)
        print ( 'The computer has beaten you! You lose.')
        gamelsPlaying = False
    else:
        if isBoardFull ( theBoard) :
            drawBoard ( theBoard)
            print ( 'The game is a tie!')
            break
        else:
            turn = 'player'
    if not playAgain ( ) :
        break
elif choice=='3':
    d=1
else:
    print ( "NOT A VALID INPUT")
if __name__ == "__main__" :
    main ()
Ln: 172 Col: 9
```

Fig B.6 Screenshot of Actual Code



The image shows a screenshot of a Python 3.7.3 Shell window. The title bar reads '*Python 3.7.3 Shell*'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following content:

```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Lenovo\Desktop\project.PY =====

.....WELCOME TO GAME CENTER.....

1) Tic Tac Toe Game
2) Rock, Paper, Scissors
3) Exit
|
```

The status bar at the bottom right indicates 'Ln: 13 Col: 0'.

Fig B.7 USER INPUT SCREEN