# PYTHON FUNCTIONS PROJECT

**TEAM MATES: Sandra Hidalgo, Iker Fariñas, Unai Zelaia-Zugadi and Julen Herrero**

In this project we have created a python program with several games. We have used git as our version control system with several contributors.

We have used python as our programming language using pycharm, a development environment specially for python, which also includes git integration. We have used the tools that pycharm gave us like the python debugger and python interpreter, as well as the git tool to commit, push and merge our project. Our project is hosted on a github git repository that we created and branched out.

For the python project itself, we have created modules with functions that can be called from a main file to be executed. The functions are a key part of this project, since all the code is contained within them. We have used different python features to make this project, such as lists, tuples and time and date.

## BRANCHES AND TASKS

Each project member has developed a String game using Python's list and time methods. We have created 5 different branches on GitHub, one is the main branch and the other 4 are the branches of each member. This is the distribution and the task that were assigned to the members.

### Main Branch

1. Unai's Branch => Connect 4 game
2. Iker F's Branch => Guess the word game
3. Sandra's Branch => Quiz game
4. Julen H's Branch => Hangman game

## GAMES

### Connect 4

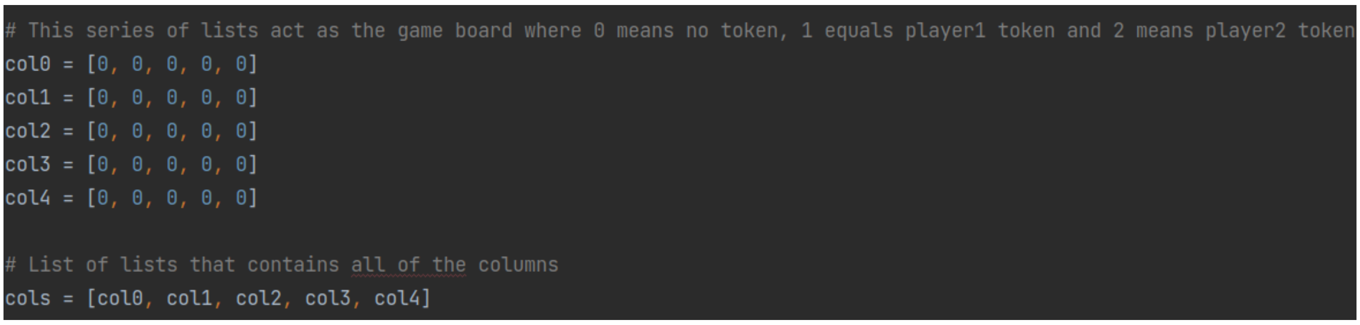
This game is based on the famous connect 4 game. The program prints a board formed by 5 rows and 5 columns. Is designed to play 1v1 so they ask the players the name and the board is printed.

Each round the program asks the players the column that wants to put the token. For each round that is played the program checks if there are 4 tokens straight on three positions; vertically, horizontally and diagonally. If the program detects this pattern it will show the winner.

### functions of connect4

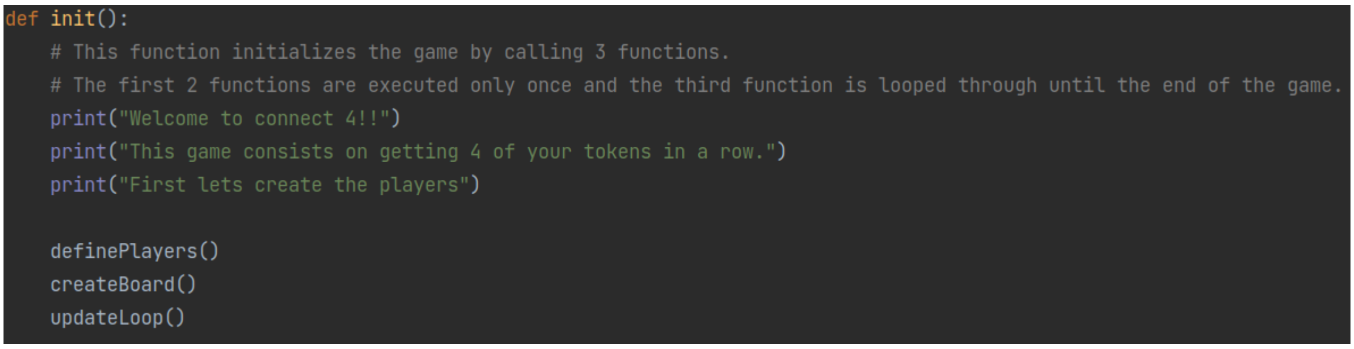
This section will contain screenshots of all of the functions in the connect4 game. The functions are explained in the comments contained within the code.

First a screenshot of the **first variables declaration**:



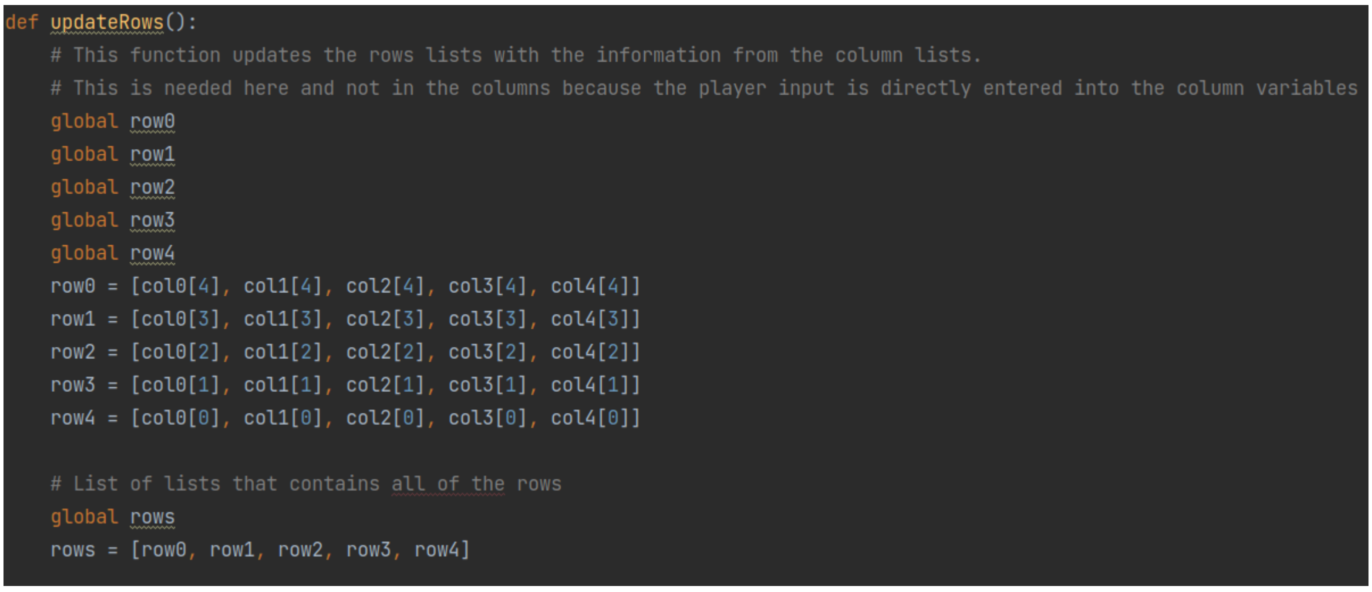
```python
# This series of lists act as the game board where 0 means no token, 1 equals player1 token and 2 means player2 token
col0 = [0, 0, 0, 0, 0]
col1 = [0, 0, 0, 0, 0]
col2 = [0, 0, 0, 0, 0]
col3 = [0, 0, 0, 0, 0]
col4 = [0, 0, 0, 0, 0]

# List of lists that contains all of the columns
cols = [col0, col1, col2, col3, col4]
```

**init()** function:



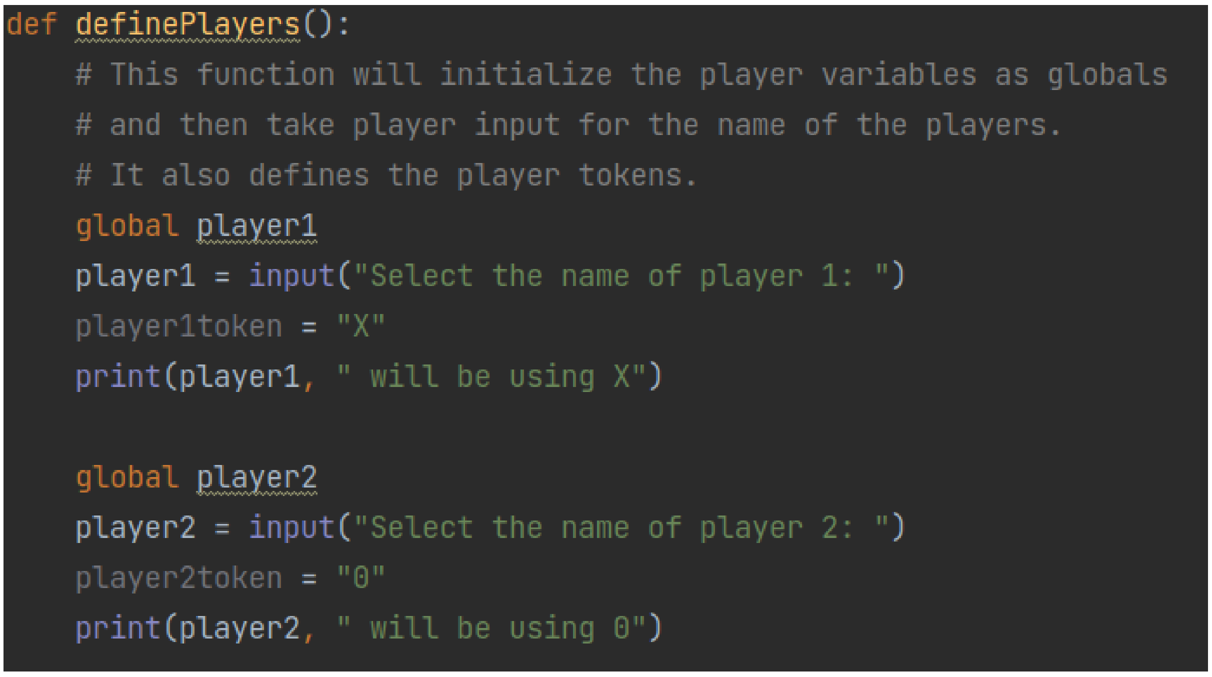
```python
def init():
    # This function initializes the game by calling 3 functions.
    # The first 2 functions are executed only once and the third function is looped through until the end of the game.
    print("Welcome to connect 4!!")
    print("This game consists on getting 4 of your tokens in a row.")
    print("First lets create the players")

    definePlayers()
    createBoard()
    updateLoop()
```

**updateRow()** function:



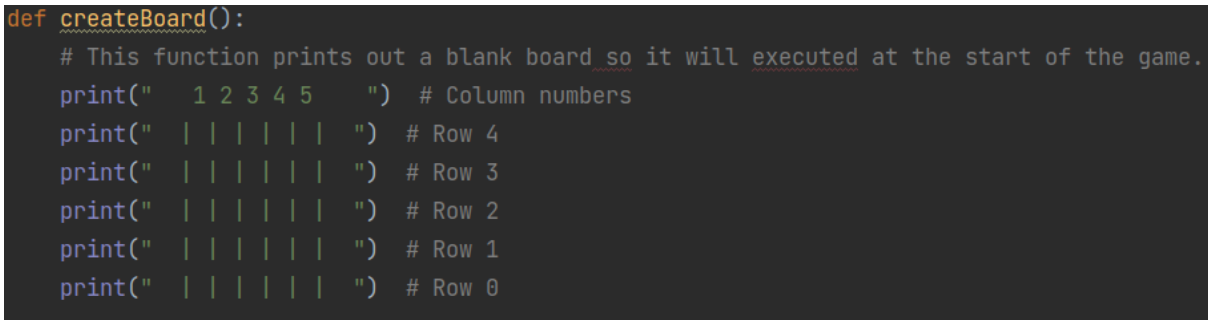
```python
def updateRows():
    # This function updates the rows lists with the information from the column lists.
    # This is needed here and not in the columns because the player input is directly entered into the column variables
    global row0
    global row1
    global row2
    global row3
    global row4
    row0 = [col0[4], col1[4], col2[4], col3[4], col4[4]]
    row1 = [col0[3], col1[3], col2[3], col3[3], col4[3]]
    row2 = [col0[2], col1[2], col2[2], col3[2], col4[2]]
    row3 = [col0[1], col1[1], col2[1], col3[1], col4[1]]
    row4 = [col0[0], col1[0], col2[0], col3[0], col4[0]]

    # List of lists that contains all of the rows
    global rows
    rows = [row0, row1, row2, row3, row4]
```

**definePlayers()** function:

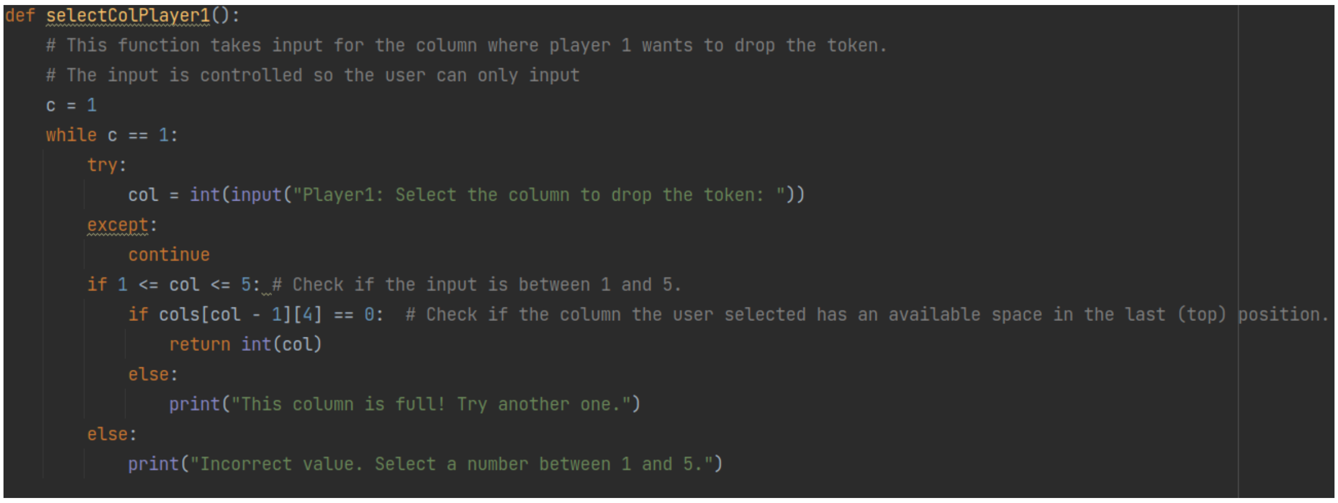


```python
def definePlayers():
    # This function will initialize the player variables as globals
    # and then take player input for the name of the players.
    # It also defines the player tokens.
    global player1
    player1 = input("Select the name of player 1: ")
    player1token = "X"
    print(player1, " will be using X")

    global player2
    player2 = input("Select the name of player 2: ")
    player2token = "0"
    print(player2, " will be using 0")
```

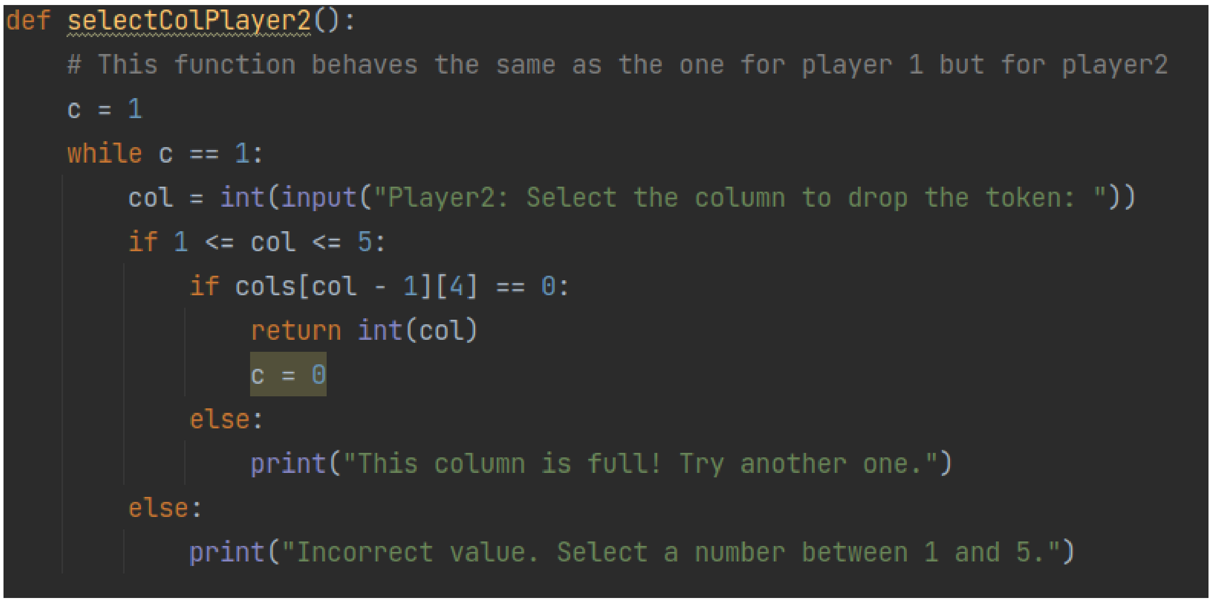
**createBoard()** function:


```
def createBoard():
    # This function prints out a blank board so it will executed at the start of the game.
    print("   1 2 3 4 5    ")  # Column numbers
    print("  | | | | | |  ")  # Row 4
    print("  | | | | | |  ")  # Row 3
    print("  | | | | | |  ")  # Row 2
    print("  | | | | | |  ")  # Row 1
    print("  | | | | | |  ")  # Row 0
```

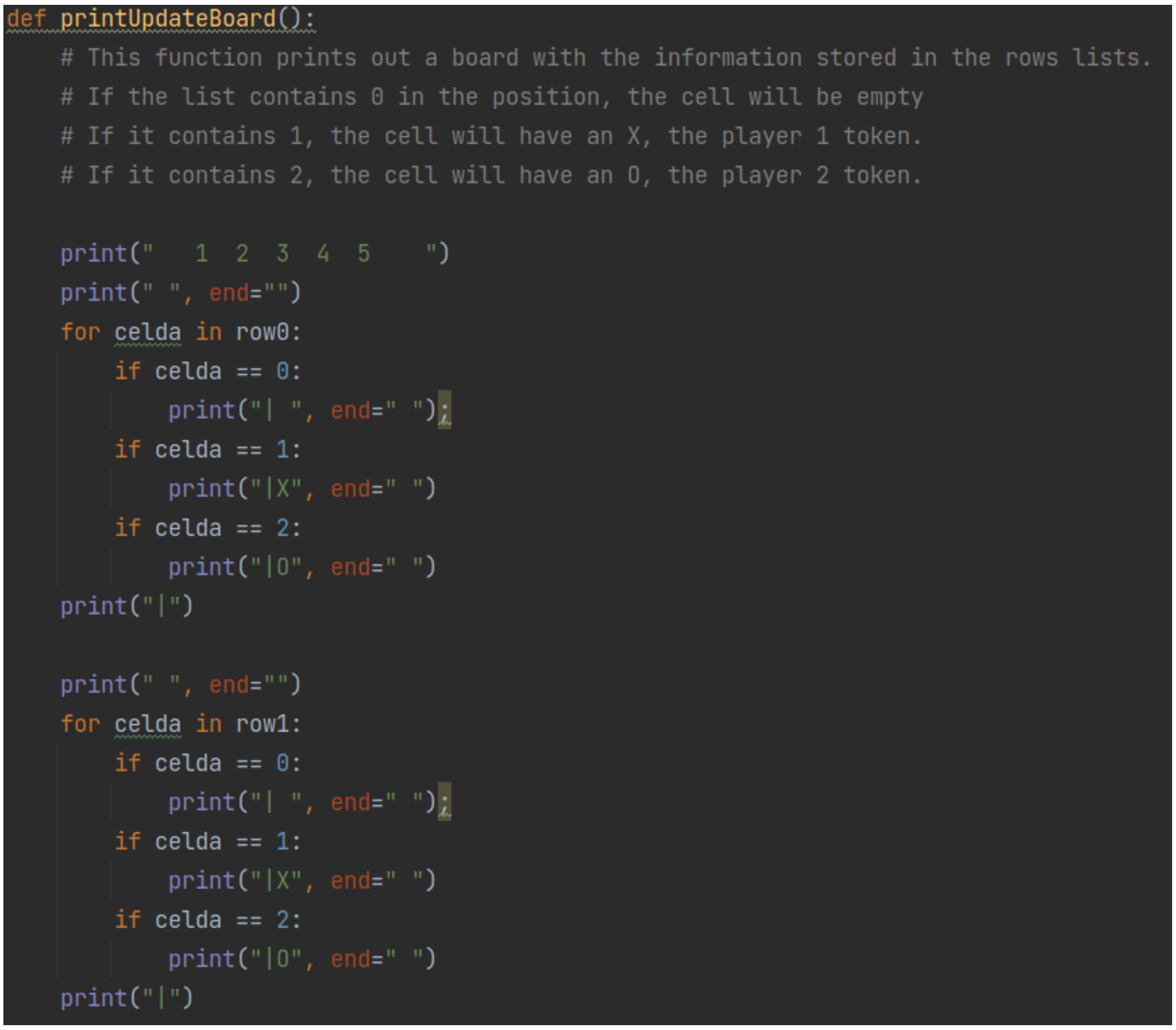
**selectColPlayer1()** function:


```
def selectColPlayer1():
    # This function takes input for the column where player 1 wants to drop the token.
    # The input is controlled so the user can only input
    c = 1
    while c == 1:
        try:
            col = int(input("Player1: Select the column to drop the token: "))
        except:
            continue
        if 1 <= col <= 5: # Check if the input is between 1 and 5.
            if cols[col - 1][4] == 0:  # Check if the column the user selected has an available space in the last (top) position.
                return int(col)
            else:
                print("This column is full! Try another one.")
        else:
            print("Incorrect value. Select a number between 1 and 5.")
```

**selectColPlayer2()** function:


```
def selectColPlayer2():
    # This function behaves the same as the one for player 1 but for player2
    c = 1
    while c == 1:
        col = int(input("Player2: Select the column to drop the token: "))
        if 1 <= col <= 5:
            if cols[col - 1][4] == 0:
                return int(col)
                c = 0
            else:
                print("This column is full! Try another one.")
        else:
            print("Incorrect value. Select a number between 1 and 5.")
```

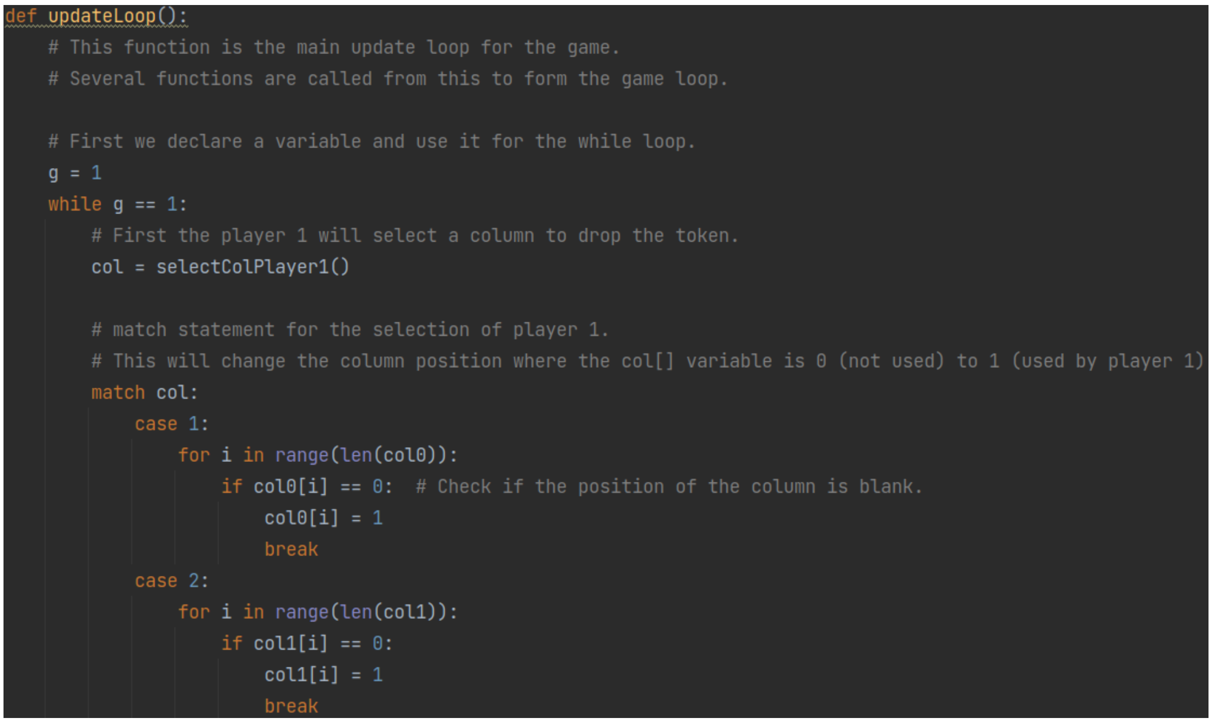
**printUpdateBoard()** function:



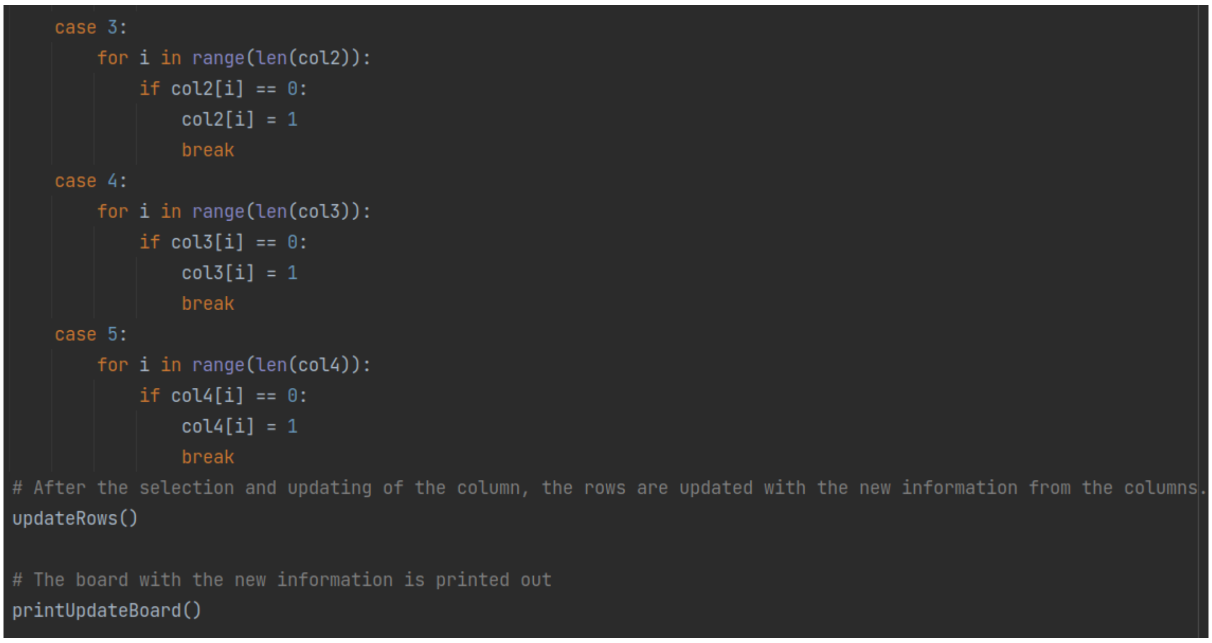
```python
def printUpdateBoard():
    # This function prints out a board with the information stored in the rows lists.
    # If the list contains 0 in the position, the cell will be empty
    # If it contains 1, the cell will have an X, the player 1 token.
    # If it contains 2, the cell will have an O, the player 2 token.

    print("   1  2  3  4  5    ")
    print(" ", end="")
    for celda in row0:
        if celda == 0:
            print("| ", end=" ");
        if celda == 1:
            print("|X", end=" ")
        if celda == 2:
            print("|O", end=" ")
    print("|")

    print(" ", end="")
    for celda in row1:
        if celda == 0:
            print("| ", end=" ");
        if celda == 1:
            print("|X", end=" ")
        if celda == 2:
            print("|O", end=" ")
    print("|")
```

This function continues to print out each of the rows in the same manner.

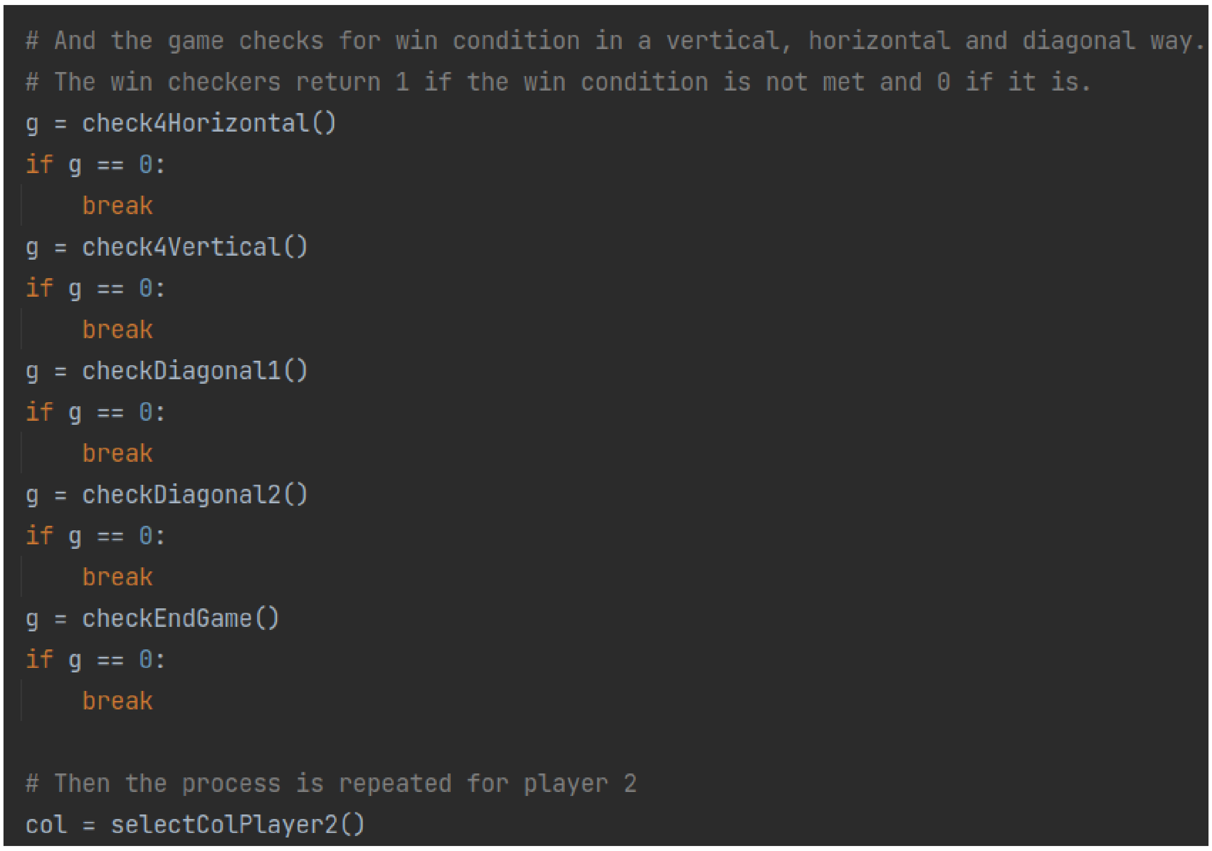
**updateLoop()** function:



```python
def updateLoop():
    # This function is the main update loop for the game.
    # Several functions are called from this to form the game loop.

    # First we declare a variable and use it for the while loop.
    g = 1
    while g == 1:
        # First the player 1 will select a column to drop the token.
        col = selectColPlayer1()

        # match statement for the selection of player 1.
        # This will change the column position where the col[] variable is 0 (not used) to 1 (used by player 1)
        match col:
            case 1:
                for i in range(len(col0)):
                    if col0[i] == 0:  # Check if the position of the column is blank.
                        col0[i] = 1
                        break
            case 2:
                for i in range(len(col1)):
                    if col1[i] == 0:
                        col1[i] = 1
                        break
```



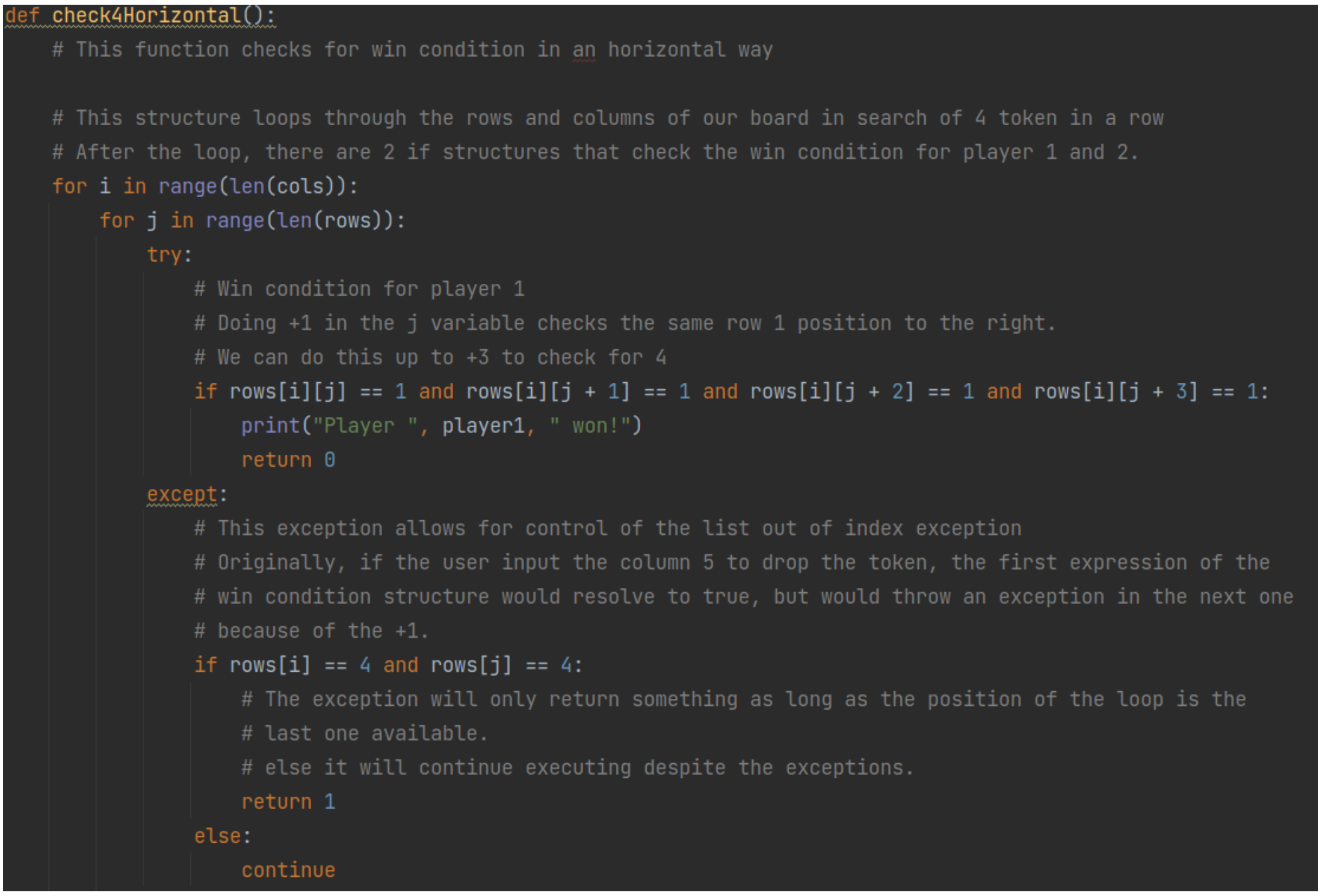
```python
    case 3:
        for i in range(len(col2)):
            if col2[i] == 0:
                col2[i] = 1
                break
    case 4:
        for i in range(len(col3)):
            if col3[i] == 0:
                col3[i] = 1
                break
    case 5:
        for i in range(len(col4)):
            if col4[i] == 0:
                col4[i] = 1
                break
# After the selection and updating of the column, the rows are updated with the new information from the columns.
updateRows()

# The board with the new information is printed out
printUpdateBoard()
```



```python
# And the game checks for win condition in a vertical, horizontal and diagonal way.
# The win checkers return 1 if the win condition is not met and 0 if it is.
g = check4Horizontal()
if g == 0:
    break
g = check4Vertical()
if g == 0:
    break
g = checkDiagonal1()
if g == 0:
    break
g = checkDiagonal2()
if g == 0:
    break
g = checkEndGame()
if g == 0:
    break

# Then the process is repeated for player 2
col = selectColPlayer2()
```

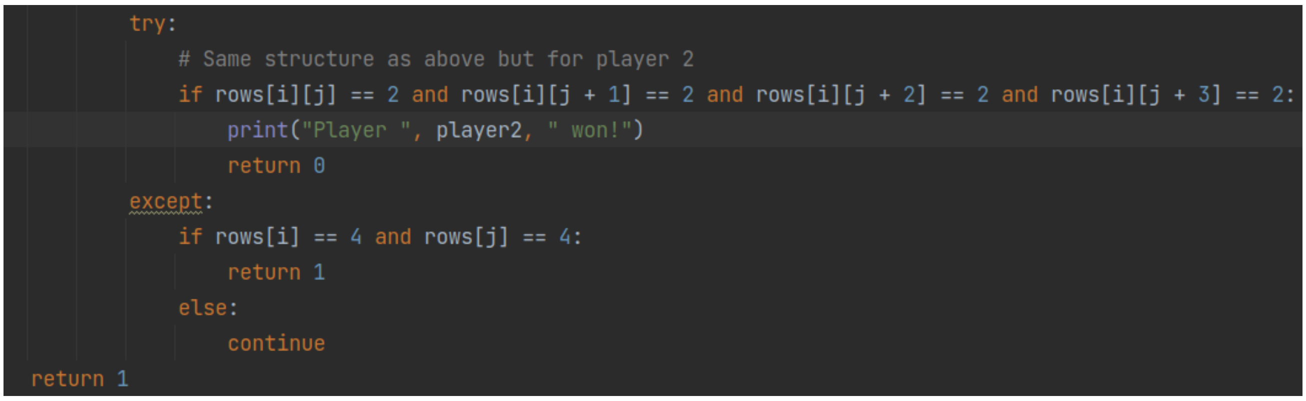
The same code is then executed but this time for the player 2.

**check4Horizontal()** function:



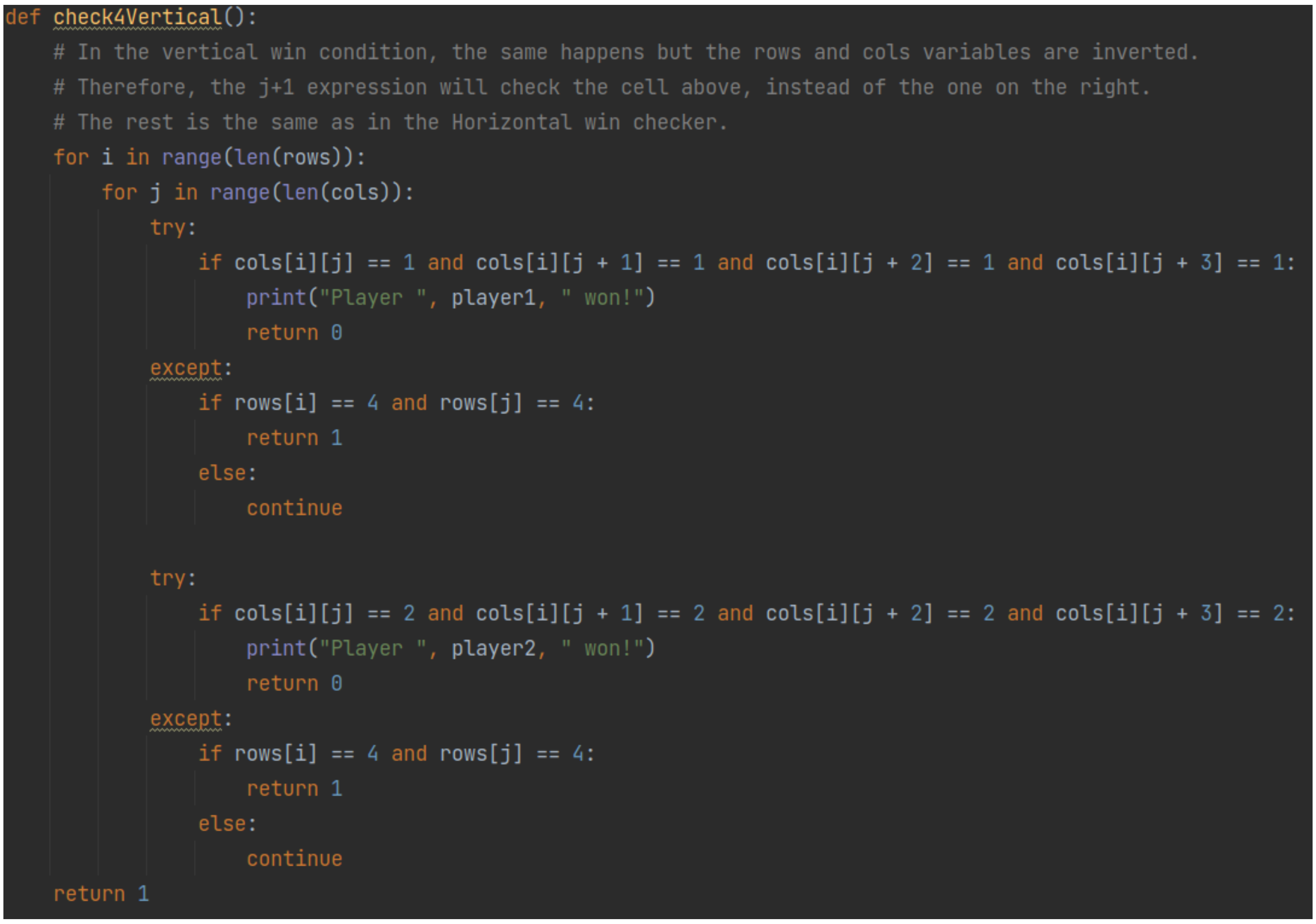
```python
def check4Horizontal():
    # This function checks for win condition in an horizontal way

    # This structure loops through the rows and columns of our board in search of 4 token in a row
    # After the loop, there are 2 if structures that check the win condition for player 1 and 2.
    for i in range(len(cols)):
        for j in range(len(rows)):
            try:
                # Win condition for player 1
                # Doing +1 in the j variable checks the same row 1 position to the right.
                # We can do this up to +3 to check for 4
                if rows[i][j] == 1 and rows[i][j + 1] == 1 and rows[i][j + 2] == 1 and rows[i][j + 3] == 1:
                    print("Player ", player1, " won!")
                    return 0
            except:
                # This exception allows for control of the list out of index exception
                # Originally, if the user input the column 5 to drop the token, the first expression of the
                # win condition structure would resolve to true, but would throw an exception in the next one
                # because of the +1.
                if rows[i] == 4 and rows[j] == 4:
                    # The exception will only return something as long as the position of the loop is the
                    # last one available.
                    # else it will continue executing despite the exceptions.
                    return 1
                else:
                    continue
```

Then the same code is executed for the player 2:



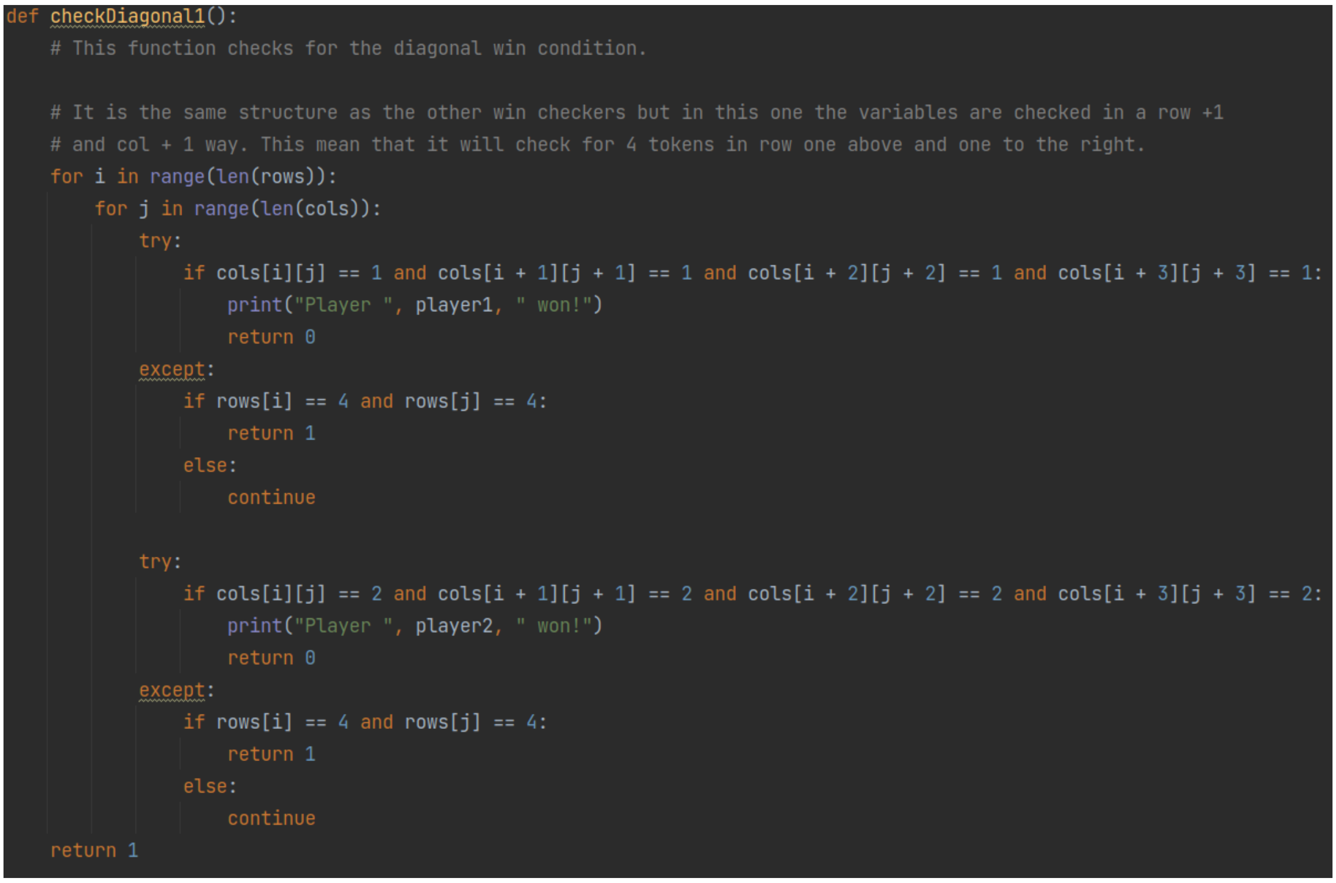
```
            try:
                # Same structure as above but for player 2
                if rows[i][j] == 2 and rows[i][j + 1] == 2 and rows[i][j + 2] == 2 and rows[i][j + 3] == 2:
                    print("Player ", player2, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue
    return 1
```

**check4Vertical()** function:



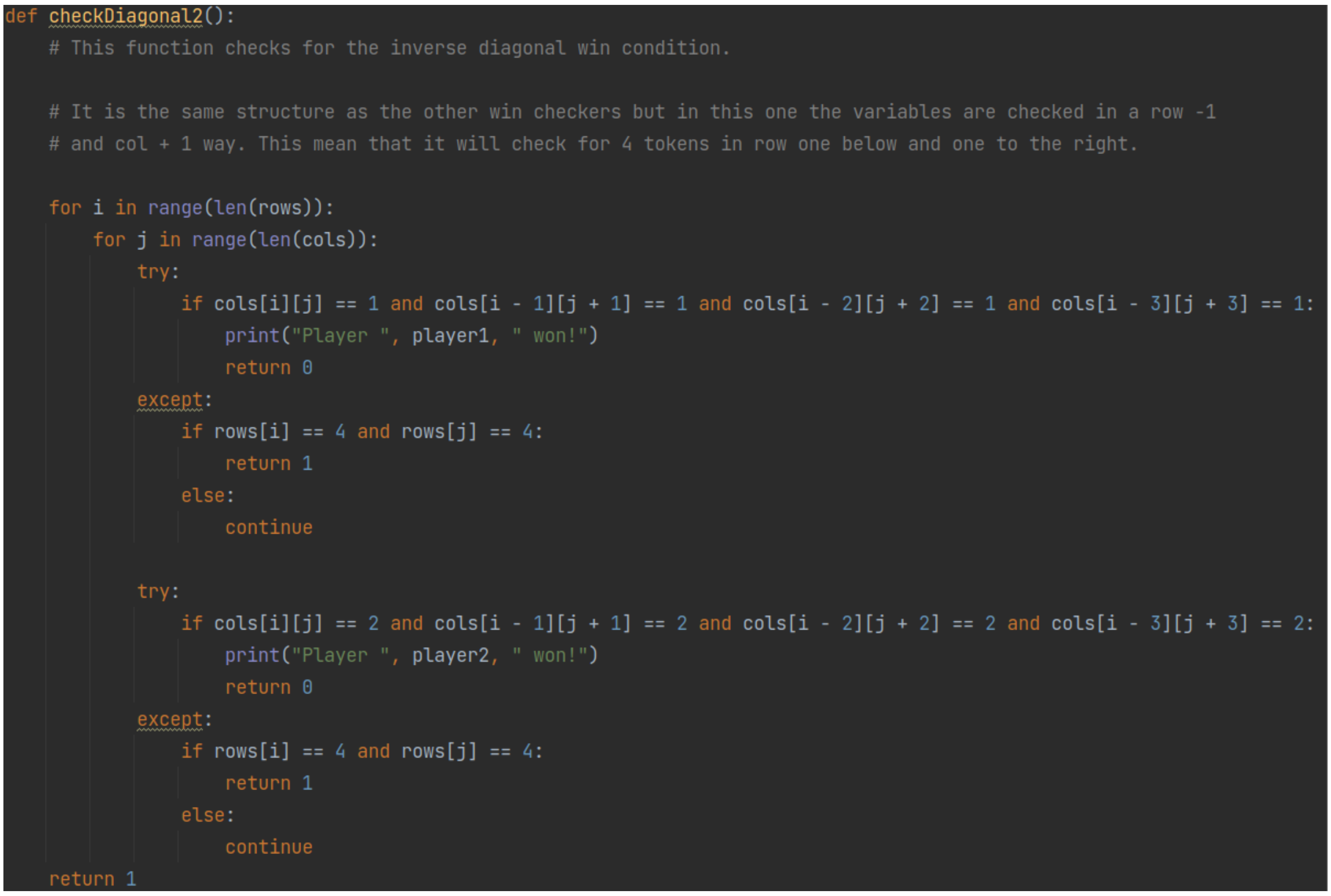
```
def check4Vertical():
    # In the vertical win condition, the same happens but the rows and cols variables are inverted.
    # Therefore, the j+1 expression will check the cell above, instead of the one on the right.
    # The rest is the same as in the Horizontal win checker.
    for i in range(len(rows)):
        for j in range(len(cols)):
            try:
                if cols[i][j] == 1 and cols[i][j + 1] == 1 and cols[i][j + 2] == 1 and cols[i][j + 3] == 1:
                    print("Player ", player1, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue

            try:
                if cols[i][j] == 2 and cols[i][j + 1] == 2 and cols[i][j + 2] == 2 and cols[i][j + 3] == 2:
                    print("Player ", player2, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue
    return 1
```

**check4Diagonal1()** function:



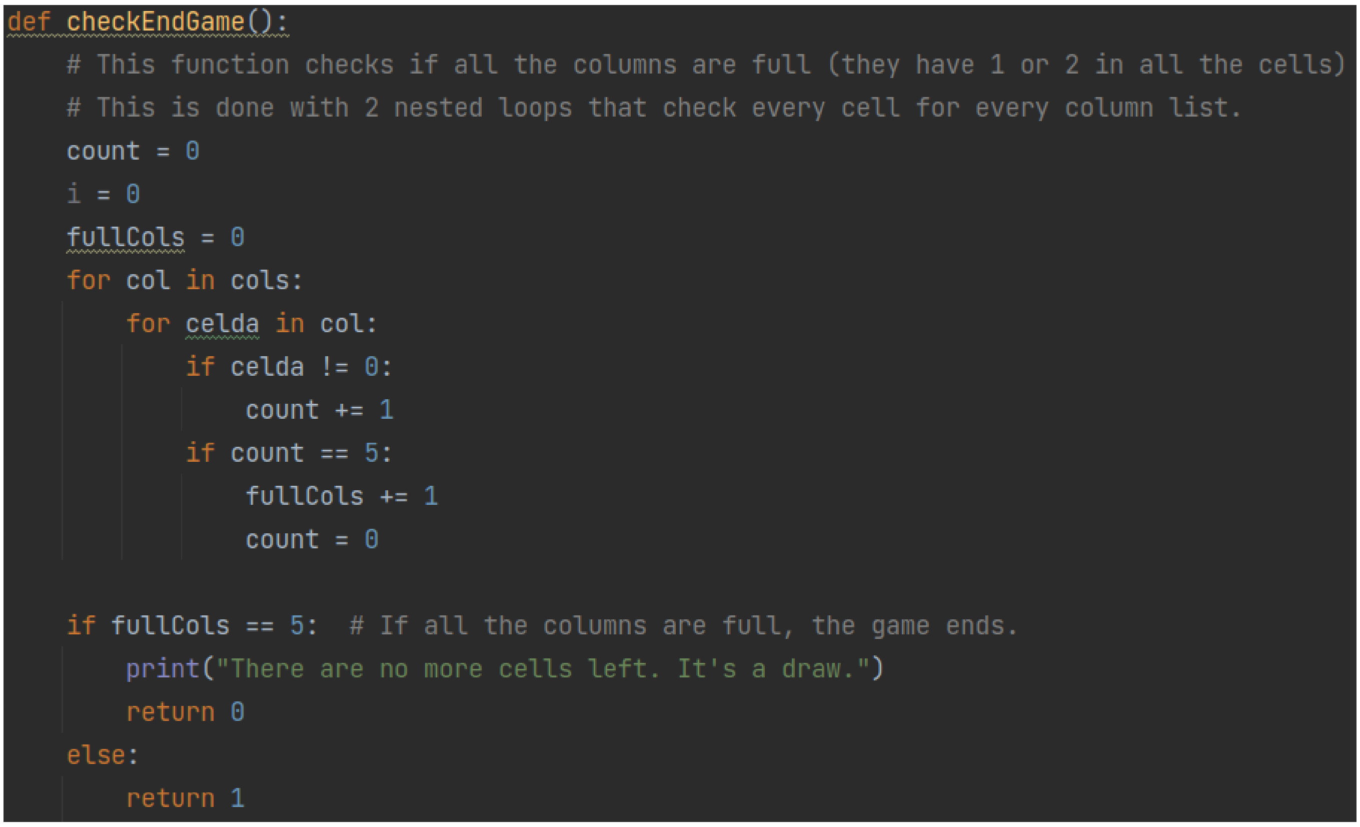
```python
def checkDiagonal1():
    # This function checks for the diagonal win condition.

    # It is the same structure as the other win checkers but in this one the variables are checked in a row +1
    # and col + 1 way. This mean that it will check for 4 tokens in row one above and one to the right.
    for i in range(len(rows)):
        for j in range(len(cols)):
            try:
                if cols[i][j] == 1 and cols[i + 1][j + 1] == 1 and cols[i + 2][j + 2] == 1 and cols[i + 3][j + 3] == 1:
                    print("Player ", player1, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue

            try:
                if cols[i][j] == 2 and cols[i + 1][j + 1] == 2 and cols[i + 2][j + 2] == 2 and cols[i + 3][j + 3] == 2:
                    print("Player ", player2, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue
    return 1
```

**check4Diagonal2()** function:



```python
def checkDiagonal2():
    # This function checks for the inverse diagonal win condition.

    # It is the same structure as the other win checkers but in this one the variables are checked in a row -1
    # and col + 1 way. This mean that it will check for 4 tokens in row one below and one to the right.

    for i in range(len(rows)):
        for j in range(len(cols)):
            try:
                if cols[i][j] == 1 and cols[i - 1][j + 1] == 1 and cols[i - 2][j + 2] == 1 and cols[i - 3][j + 3] == 1:
                    print("Player ", player1, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue

            try:
                if cols[i][j] == 2 and cols[i - 1][j + 1] == 2 and cols[i - 2][j + 2] == 2 and cols[i - 3][j + 3] == 2:
                    print("Player ", player2, " won!")
                    return 0
            except:
                if rows[i] == 4 and rows[j] == 4:
                    return 1
                else:
                    continue
    return 1
```

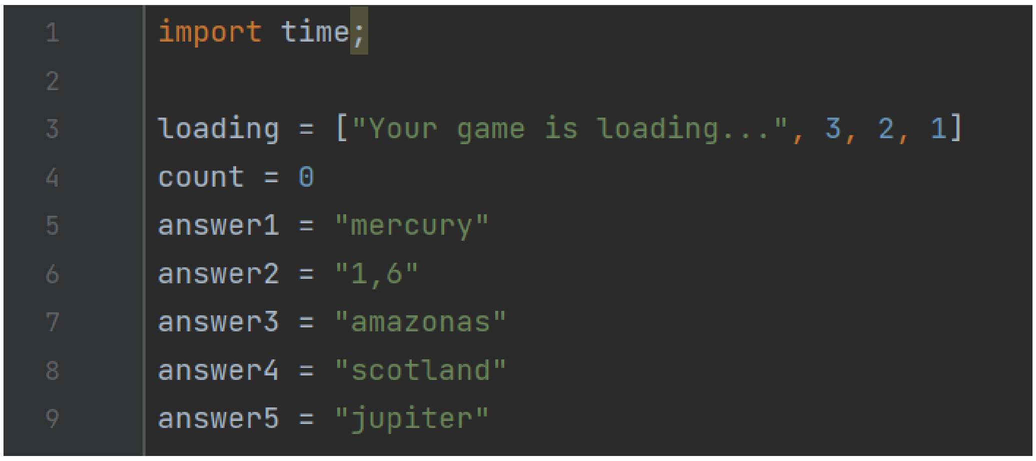
**checkEndGame()** function:



```python
def checkEndGame():
    # This function checks if all the columns are full (they have 1 or 2 in all the cells)
    # This is done with 2 nested loops that check every cell for every column list.
    count = 0
    i = 0
    fullCols = 0
    for col in cols:
        for celda in col:
            if celda != 0:
                count += 1
            if count == 5:
                fullCols += 1
                count = 0

    if fullCols == 5:  # If all the columns are full, the game ends.
        print("There are no more cells left. It's a draw.")
        return 0
    else:
        return 1
```

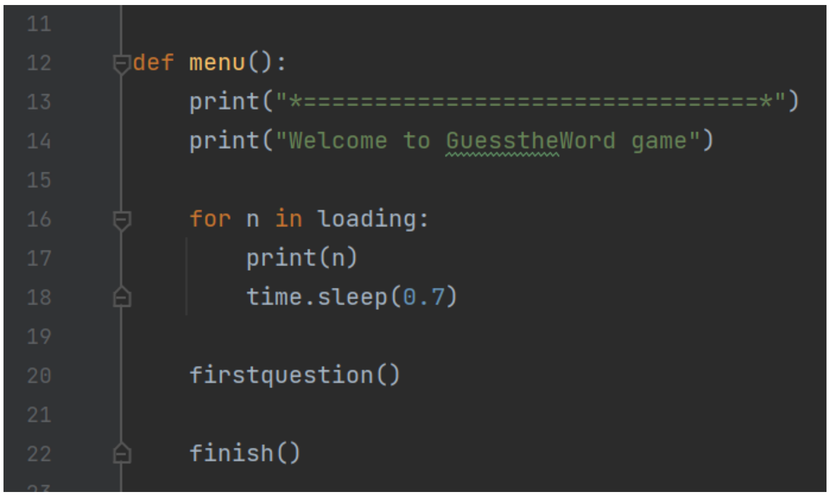
**Guess the word**

The main purpose of this game is to guess the word. Some questions are going to be asked and the user has to guess them. If the word entered is not the correct answer a hint is going to be given. Each question has two hints and when the answer entered is the correct one is going to pass to the next one.



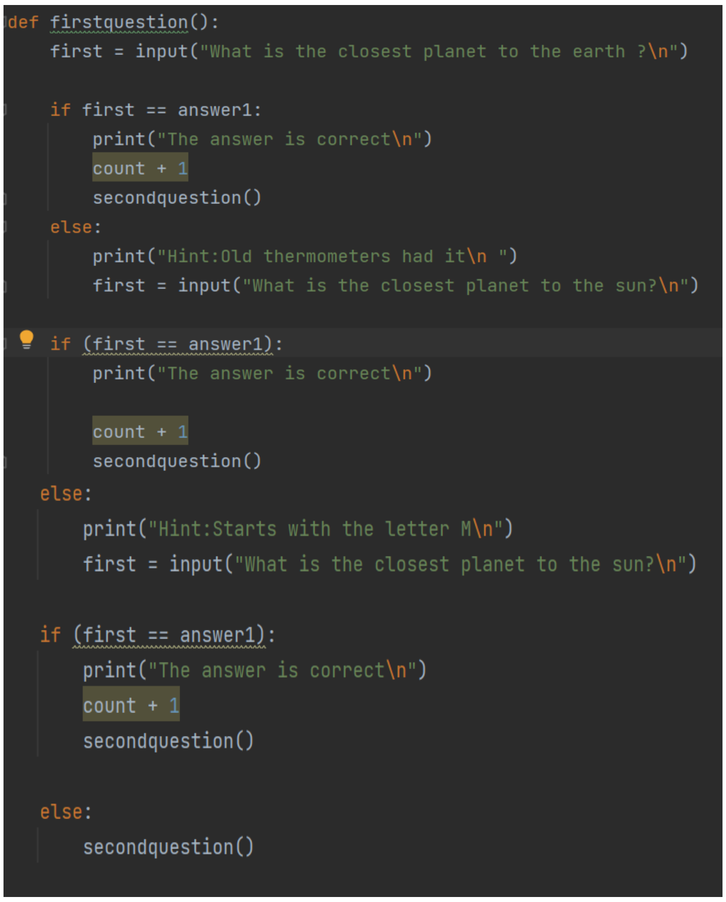
```python
import time;

loading = ["Your game is loading...", 3, 2, 1]
count = 0
answer1 = "mercury"
answer2 = "1,6"
answer3 = "amazonas"
answer4 = "scotland"
answer5 = "jupiter"
```

First we have all the strings declared and imported time.These are going to be used later on on each question.



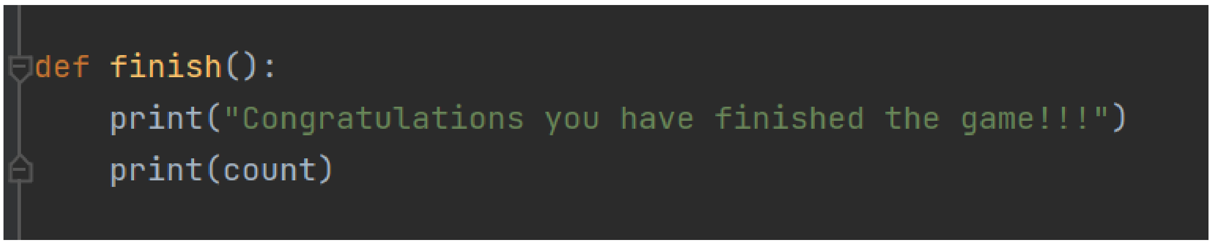
```python

def menu():
    print("*================================*")
    print("Welcome to GuesstheWord game")

    for n in loading:
        print(n)
        time.sleep(0.7)

    firstquestion()

    finish()
```

Then we have the main menu.This one contains greetings for the user and it is going to do a countdown to load the game.
After that the first question is going to be executed.



```python
def firstquestion():
    first = input("What is the closest planet to the earth ?\n")

    if first == answer1:
        print("The answer is correct\n")
        count + 1
        secondquestion()
    else:
        print("Hint:Old thermometers had it\n ")
        first = input("What is the closest planet to the sun?\n")

    if (first == answer1):
        print("The answer is correct\n")

        count + 1
        secondquestion()
    else:
        print("Hint:Starts with the letter M\n")
        first = input("What is the closest planet to the sun?\n")

    if (first == answer1):
        print("The answer is correct\n")
        count + 1
        secondquestion()

    else:
        secondquestion()
```

Now we enter the first question.All the questions have the same structure. If the answer is correct it is going to do count + 1 and get to the next question. If it is not correct it is going to pass to the hint and like that until it passes the second hint. If the user does not guess the word it will pass to the next question.



```python
def finish():
    print("Congratulations you have finished the game!!!")
    print(count)
```

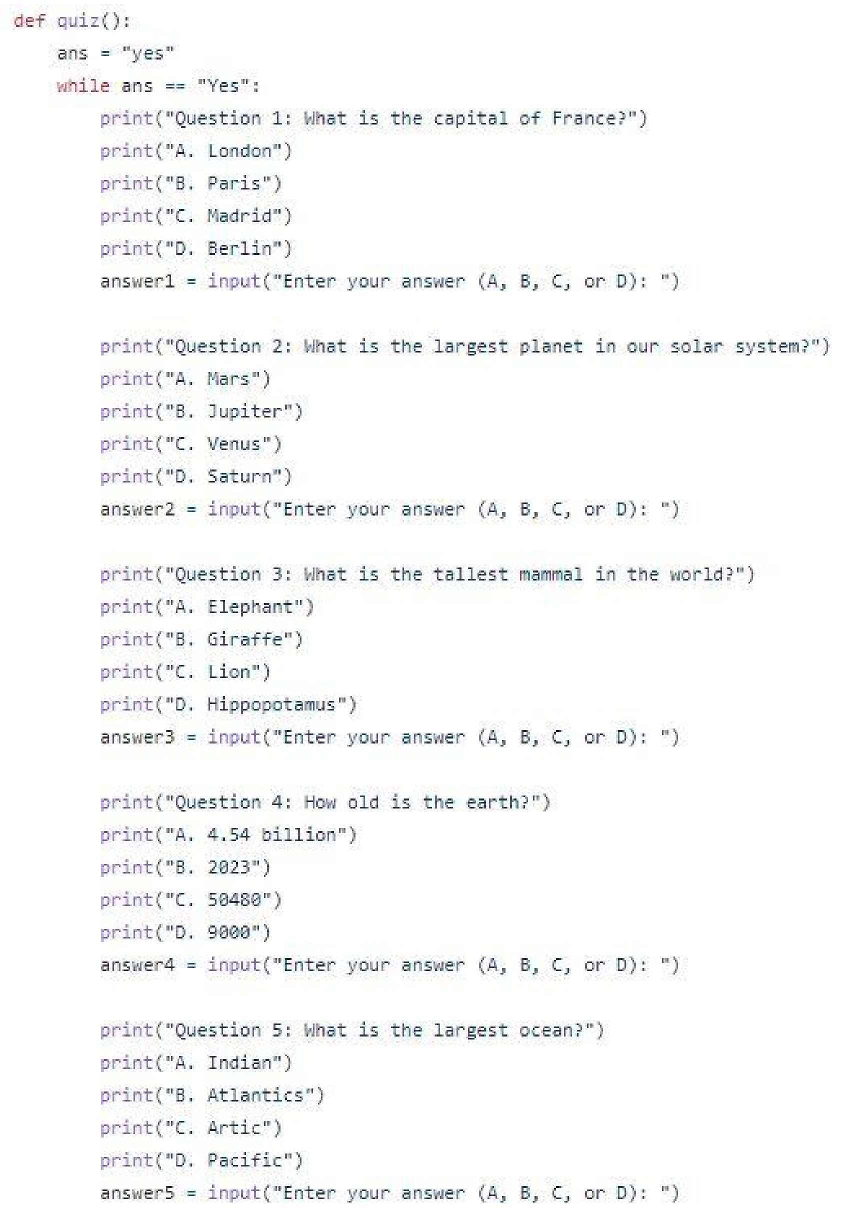
In the last question instead of count + 1 it is going to be the finish function.This one is going to congratulate the user and give him the points that he has scored.

## Quiz

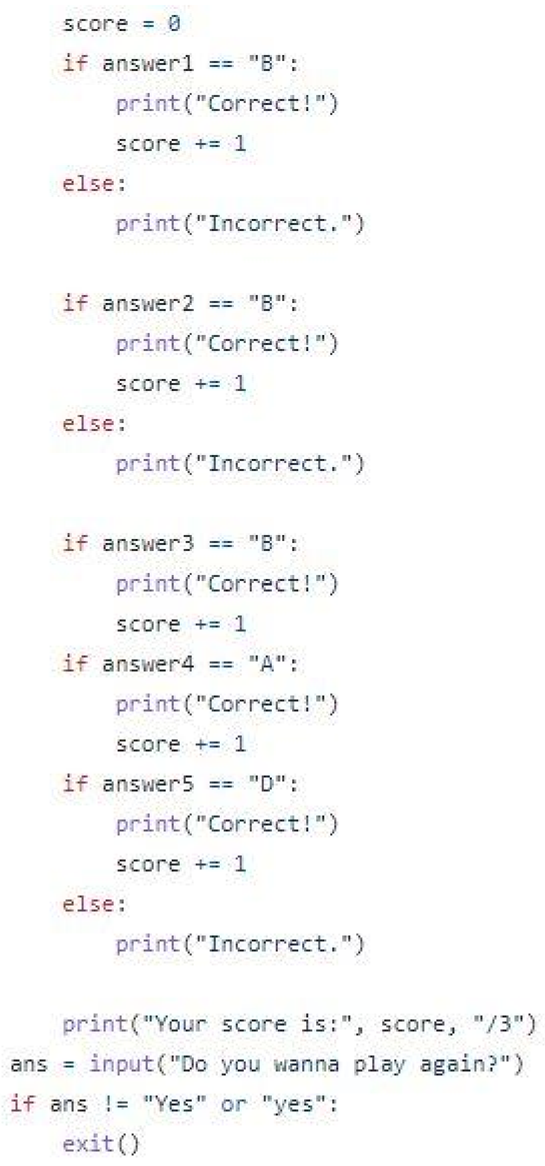
This game is a quiz with questions. The program shows some questions and it also shows different possible answers. The user has to try to guess which is the correct answer for those questions.

For each question the program shows four different possible answers, when the user starts entering the answer the program will start to verify if the answer is the correct one. If the answer is the correct one it will start counting to show at the end of the game the final score. Finally the programs will ask the user if he wants to play again.

## Functions of Quiz



```python
def quiz():
    ans = "yes"
    while ans == "Yes":
        print("Question 1: What is the capital of France?")
        print("A. London")
        print("B. Paris")
        print("C. Madrid")
        print("D. Berlin")
        answer1 = input("Enter your answer (A, B, C, or D): ")

        print("Question 2: What is the largest planet in our solar system?")
        print("A. Mars")
        print("B. Jupiter")
        print("C. Venus")
        print("D. Saturn")
        answer2 = input("Enter your answer (A, B, C, or D): ")

        print("Question 3: What is the tallest mammal in the world?")
        print("A. Elephant")
        print("B. Giraffe")
        print("C. Lion")
        print("D. Hippopotamus")
        answer3 = input("Enter your answer (A, B, C, or D): ")

        print("Question 4: How old is the earth?")
        print("A. 4.54 billion")
        print("B. 2023")
        print("C. 50480")
        print("D. 9000")
        answer4 = input("Enter your answer (A, B, C, or D): ")

        print("Question 5: What is the largest ocean?")
        print("A. Indian")
        print("B. Atlantics")
        print("C. Artic")
        print("D. Pacific")
        answer5 = input("Enter your answer (A, B, C, or D): ")
```



```
    score = 0
    if answer1 == "B":
        print("Correct!")
        score += 1
    else:
        print("Incorrect.")

    if answer2 == "B":
        print("Correct!")
        score += 1
    else:
        print("Incorrect.")

    if answer3 == "B":
        print("Correct!")
        score += 1
    if answer4 == "A":
        print("Correct!")
        score += 1
    if answer5 == "D":
        print("Correct!")
        score += 1
    else:
        print("Incorrect.")

    print("Your score is:", score, "/3")
ans = input("Do you wanna play again?")
if ans != "Yes" or "yes":
    exit()
```

**Hangman**

This game is based on the well known Hangman game. The program is designed to run it on the terminal. The game is based on a list of animals. When the program starts to run it randomly takes a word from the list and it shows horizontally, but it shows only a “-” for each character.

The player has to start entering characters and the program will check every time the user enters a character if this one is on the word that the program has taken randomly from the animal list.

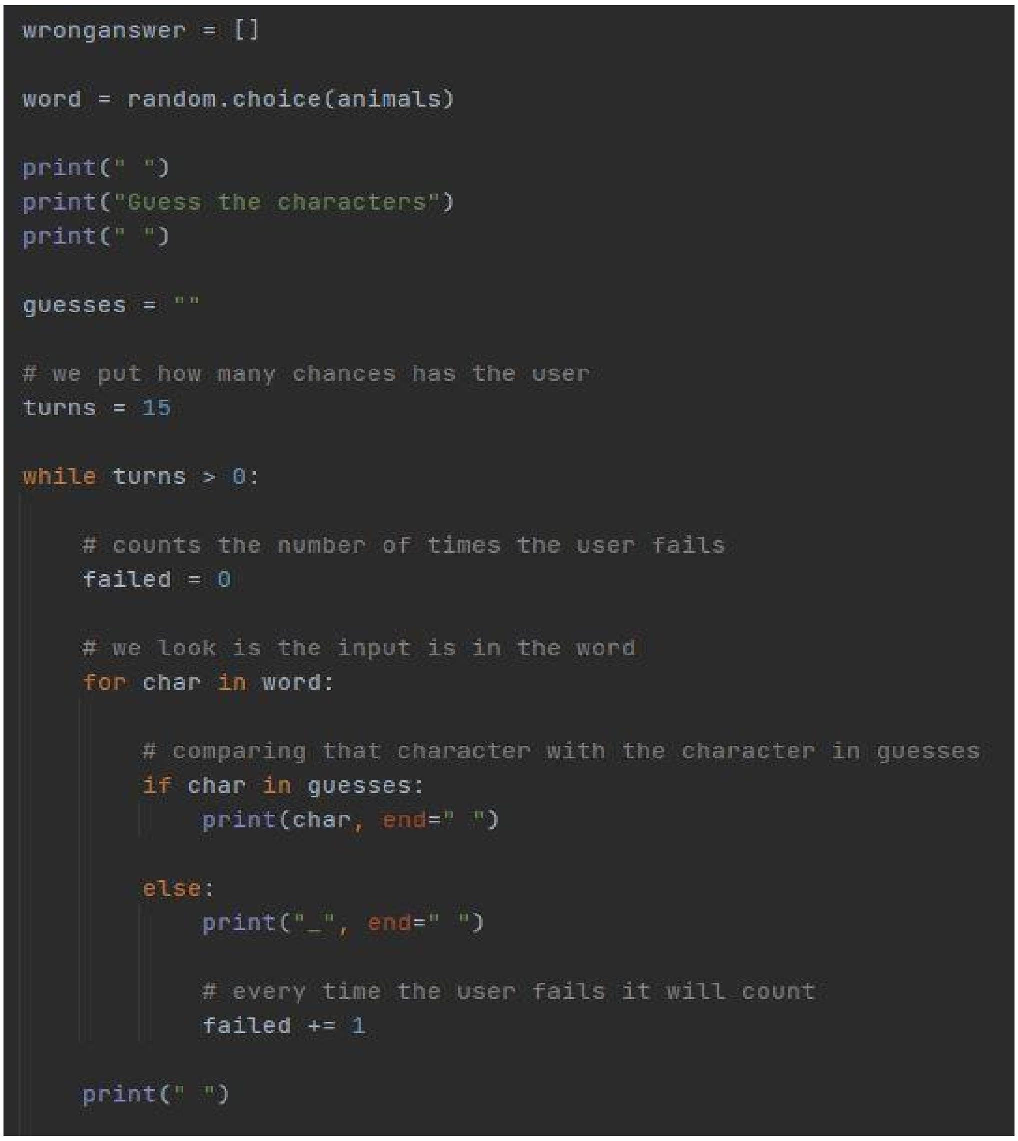
If the character is on the word it will print that character on the specific position that it has on the word, else the character will be added on a list of wrong guesses and the list will be shown at the end of each round to give the player information about the guesses that has entered it wrong. The game gives the user 15 chances and each time it fails a character it rest a chance.

**Functions of Hangman**

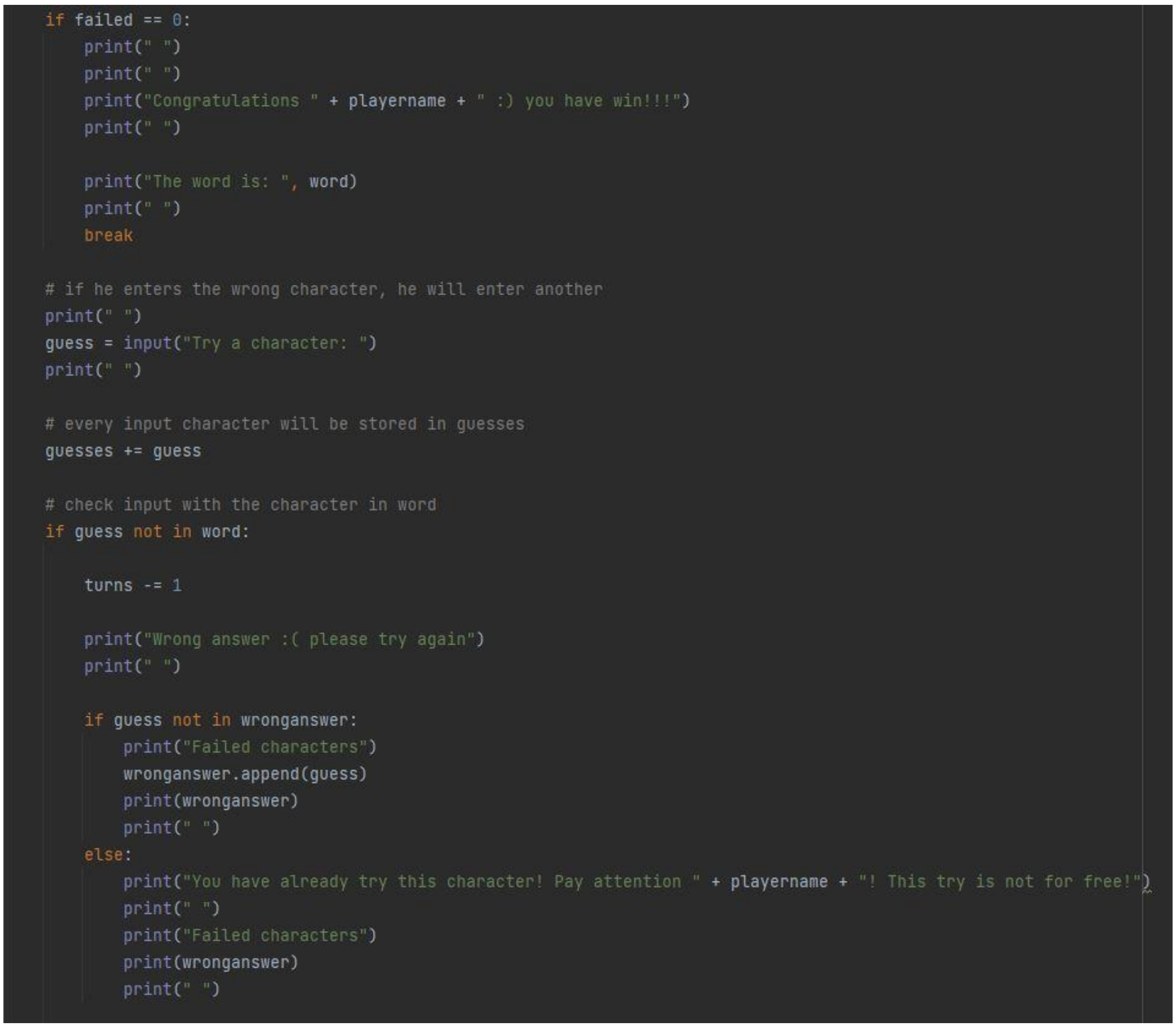
**name() => Asks the player his name**

```
def name():

   playername = input("What's your name? ")
   hangmangame(playername)
```

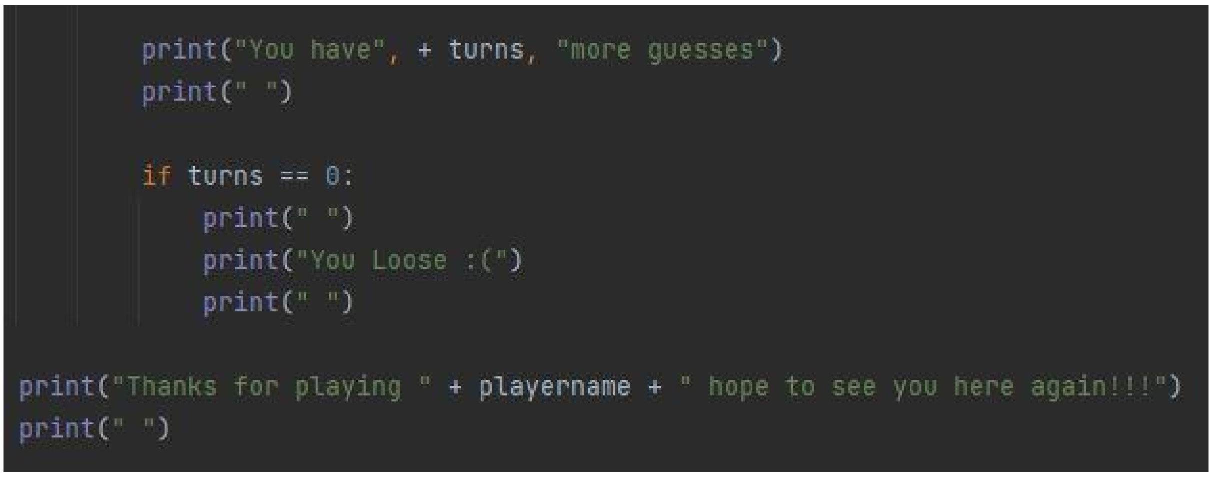
**hangmangame() => Runs the entire game**



```
wronganswer = []

word = random.choice(animals)

print(" ")
print("Guess the characters")
print(" ")

guesses = ""

# we put how many chances has the user
turns = 15

while turns > 0:

    # counts the number of times the user fails
    failed = 0

    # we look is the input is in the word
    for char in word:

        # comparing that character with the character in guesses
        if char in guesses:
            print(char, end=" ")

        else:
            print("_", end=" ")

            # every time the user fails it will count
            failed += 1

    print(" ")
```



```python
    if failed == 0:
        print(" ")
        print(" ")
        print("Congratulations " + playername + " :) you have win!!!")
        print(" ")

        print("The word is: ", word)
        print(" ")
        break

    # if he enters the wrong character, he will enter another
    print(" ")
    guess = input("Try a character: ")
    print(" ")

    # every input character will be stored in guesses
    guesses += guess

    # check input with the character in word
    if guess not in word:

        turns -= 1

        print("Wrong answer :( please try again")
        print(" ")

        if guess not in wronganswer:
            print("Failed characters")
            wronganswer.append(guess)
            print(wronganswer)
            print(" ")
        else:
            print("You have already try this character! Pay attention " + playername + "! This try is not for free!")
            print(" ")
            print("Failed characters")
            print(wronganswer)
            print(" ")
```



```python
        print("You have", + turns, "more guesses")
        print(" ")

        if turns == 0:
            print(" ")
            print("You Loose :(")
            print(" ")

print("Thanks for playing " + playername + " hope to see you here again!!!")
print(" ")
```