GCSE Computer Science

EDULiTO

# A Complete Guide to Python Programming Part 4

9. Creating a Simple Board Game
10. Creating a Simple Database

## Photocopiable Resources

# Terms and Conditions of Use

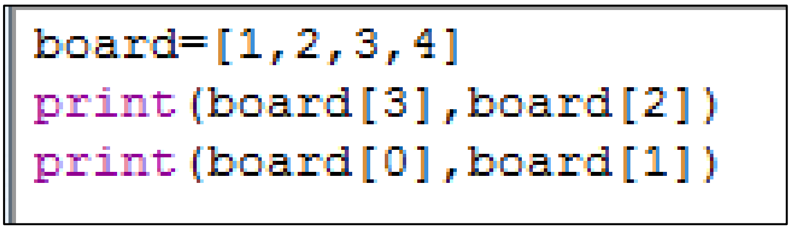
# 9 - Creating a simple board game using Python

## 9.1 Creating a game board

A simple way to create a game board is using a list with every number from the game board. If you want a board like the one below you will need 4 numbers.

| | |
|---|---|
| 4 | 3 |
| 1 | 2 |

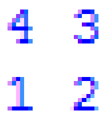
Copy this code into Python:



```
board=[1,2,3,4]
print(board[3],board[2])
print(board[0],board[1])
```

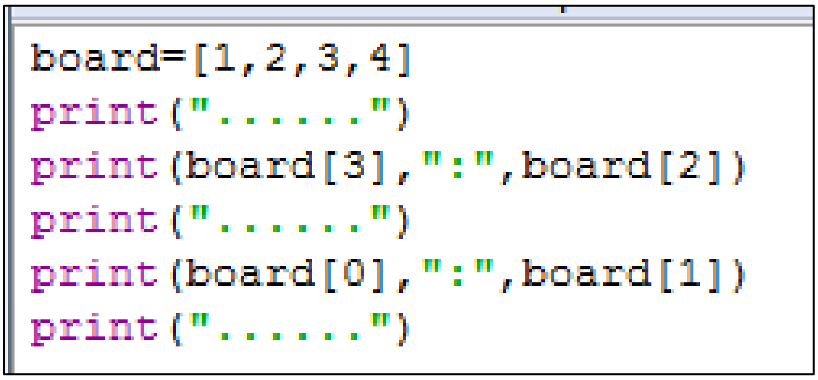
Remember the positions in a list start from 0, and so number 1 in the list is found in the position 0 and number two in the position 1.

The output from this program is:



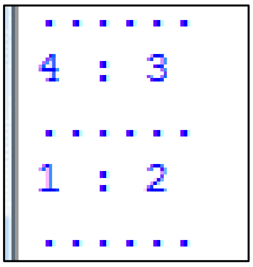
```
4 3
1 2
```

You can use print statements to make it more like a game board.
Add this to your Python code.



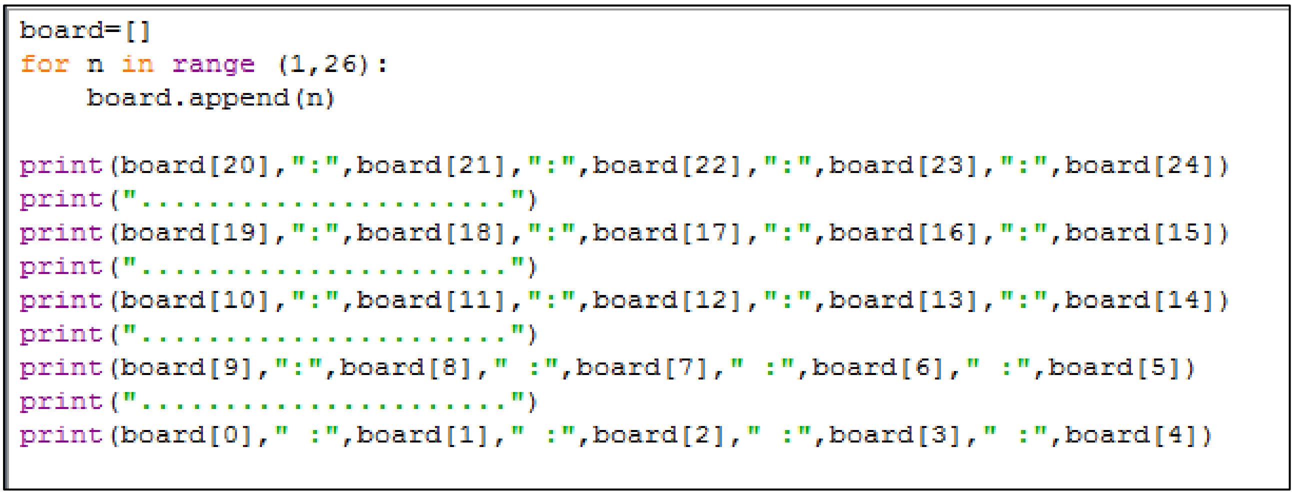
```
board=[1,2,3,4]
print("......")
print(board[3],":",board[2])
print("......")
print(board[0],":",board[1])
print("......")
```

The output from this program is:



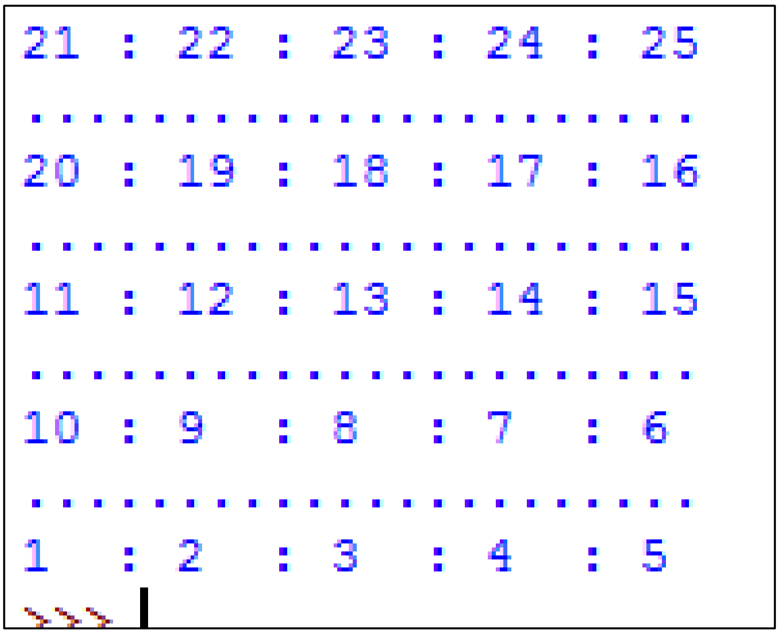
```
......
4 : 3
......
1 : 2
......
```

If you want to create a bigger board then it is easier if you use a FOR loop to add the numbers to the list.
Now create this game board:



```
board=[]
for n in range (1,26):
    board.append(n)

print(board[20],":",board[21],":",board[22],":",board[23],":",board[24])
print("......................")
print(board[19],":",board[18],":",board[17],":",board[16],":",board[15])
print("......................")
print(board[10],":",board[11],":",board[12],":",board[13],":",board[14])
print("......................")
print(board[9],":",board[8]," :",board[7]," :",board[6]," :",board[5])
print("......................")
print(board[0]," :",board[1]," :",board[2]," :",board[3]," :",board[4])
```

The output from this program is:



```
21 : 22 : 23 : 24 : 25
......................
20 : 19 : 18 : 17 : 16
......................
11 : 12 : 13 : 14 : 15
......................
10 : 9  : 8  : 7  : 6
......................
1  : 2  : 3  : 4  : 5
>>> 
```

## Activity 1

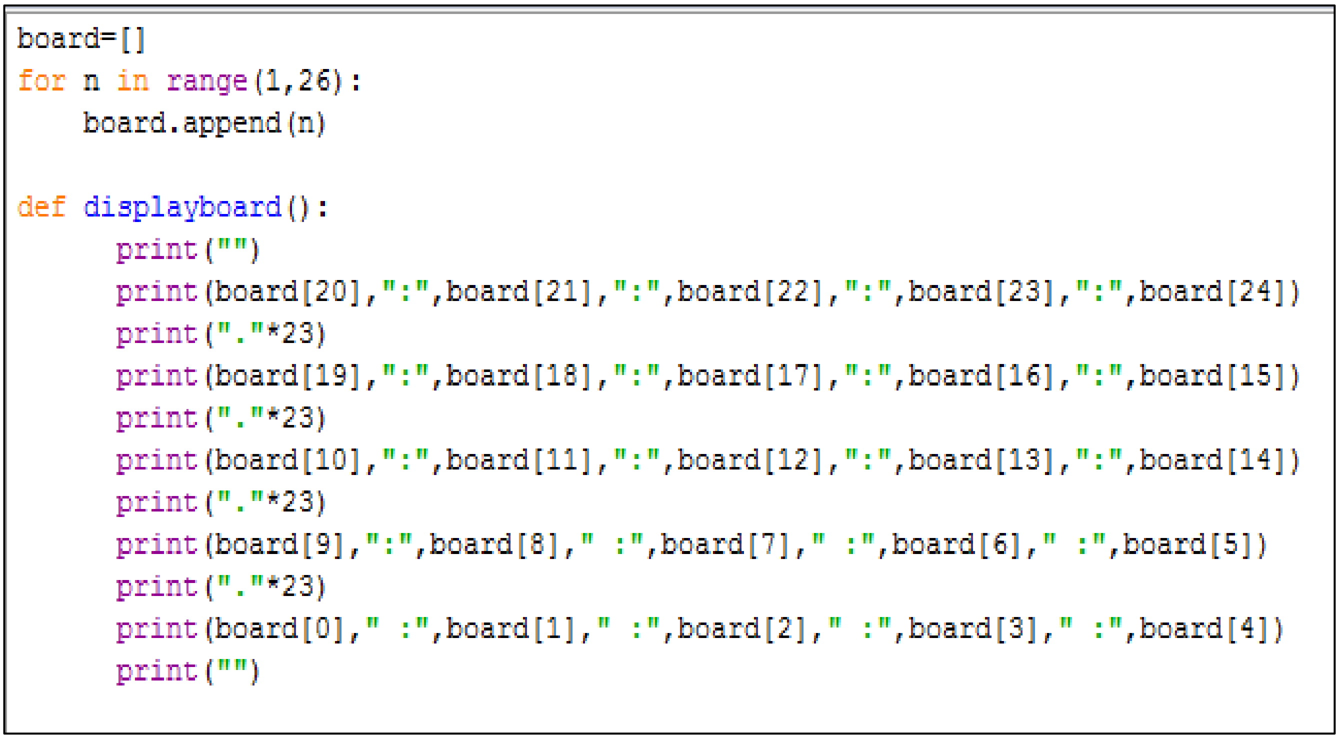
Clark is designing a game that allows players to move around a 7x7 grid, each position in the board has a number to represent the space, as shown in Fig 1.
Create this game board.

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| 43 | 44 | 45 | 46 | 47 | 48 | 49 |
| 42 | 41 | 40 | 39 | 38 | 37 | 36 |
| 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| 28 | 27 | 26 | 25 | 24 | 23 | 22 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Fig 1**

### 9.2 Creating a game board function

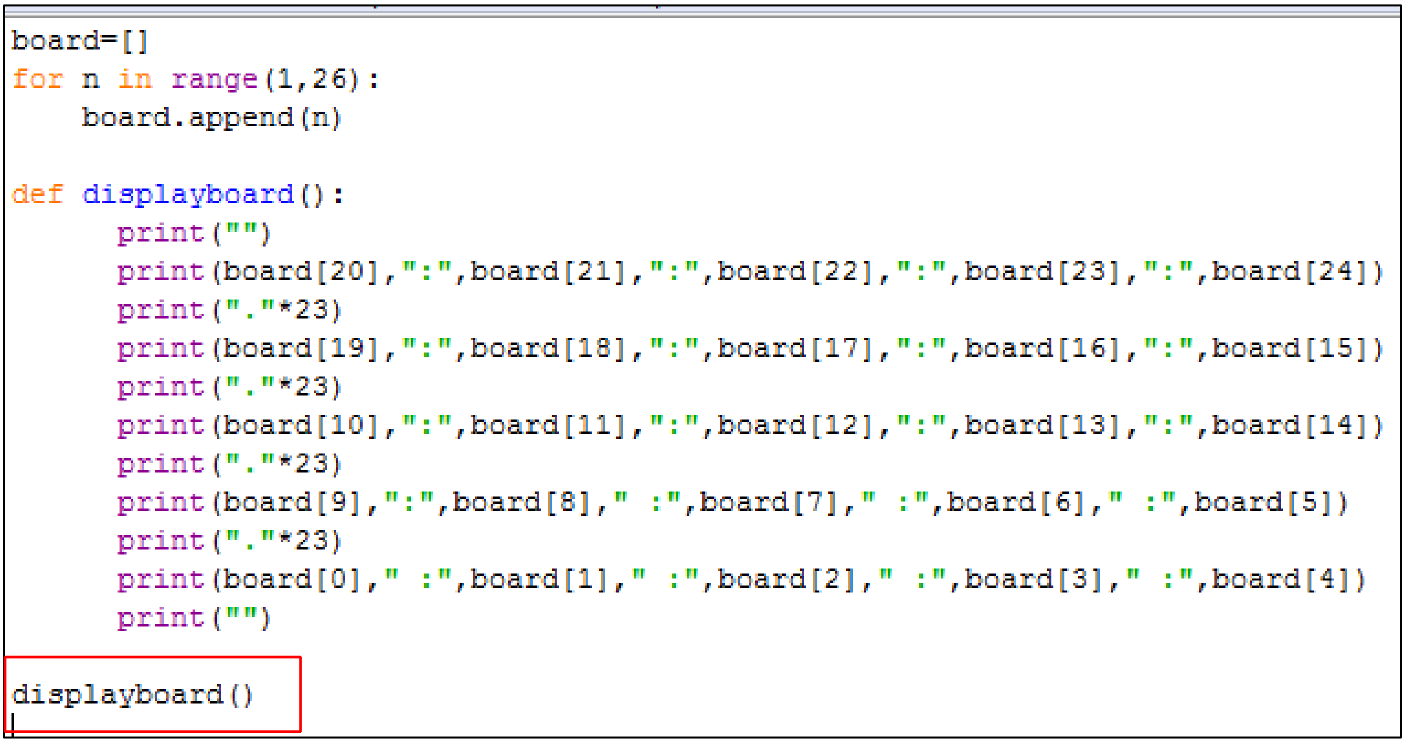
When writing code for a whole game it’s a good idea to decompose the game into sub-tasks. One of these sub-tasks could be making the game board into a function that can be repeatedly used.
Edit the Python code you have already used to turn the board into a function:



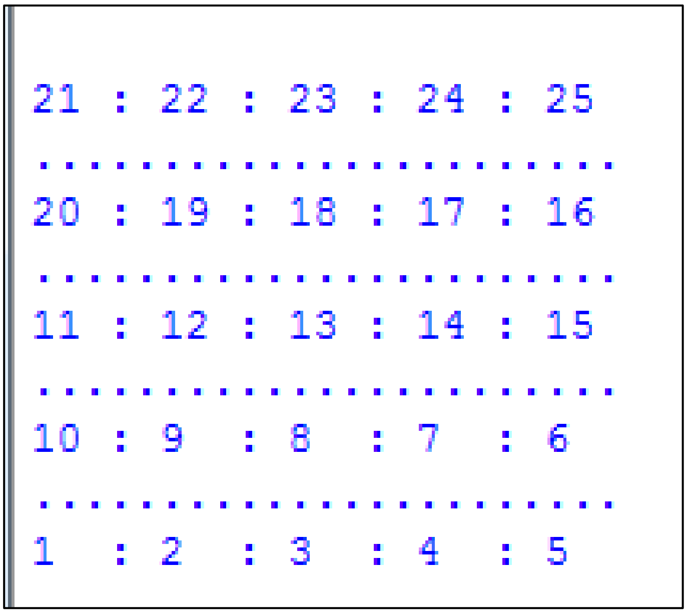
```
board=[]
for n in range(1,26):
    board.append(n)

def displayboard():
      print("")
      print(board[20],":",board[21],":",board[22],":",board[23],":",board[24])
      print("."*23)
      print(board[19],":",board[18],":",board[17],":",board[16],":",board[15])
      print("."*23)
      print(board[10],":",board[11],":",board[12],":",board[13],":",board[14])
      print("."*23)
      print(board[9],":",board[8]," :",board[7]," :",board[6]," :",board[5])
      print("."*23)
      print(board[0]," :",board[1]," :",board[2]," :",board[3]," :",board[4])
      print("")
```

When you run the code nothing happens, because for a function to work you need to call it.

Add an extra line of code to call the function (see below).



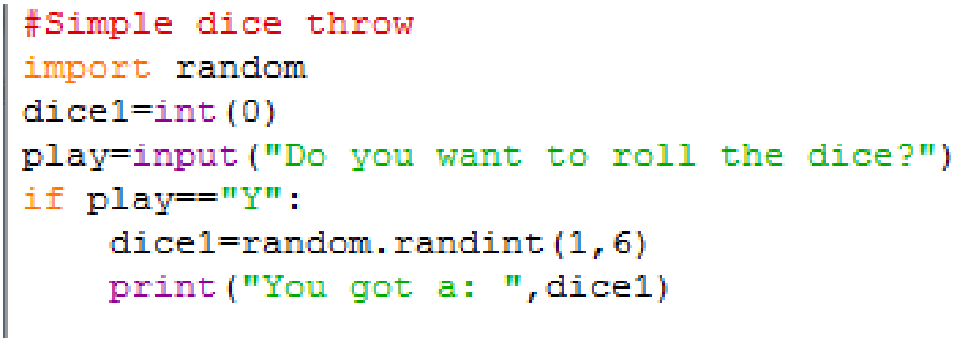
```
board=[]
for n in range(1,26):
    board.append(n)

def displayboard():
      print("")
      print(board[20],":",board[21],":",board[22],":",board[23],":",board[24])
      print("."*23)
      print(board[19],":",board[18],":",board[17],":",board[16],":",board[15])
      print("."*23)
      print(board[10],":",board[11],":",board[12],":",board[13],":",board[14])
      print("."*23)
      print(board[9],":",board[8]," :",board[7]," :",board[6]," :",board[5])
      print("."*23)
      print(board[0]," :",board[1]," :",board[2]," :",board[3]," :",board[4])
      print("")

displayboard()
```



```

21 : 22 : 23 : 24 : 25
.......................
20 : 19 : 18 : 17 : 16
.......................
11 : 12 : 13 : 14 : 15
.......................
10 : 9  : 8  : 7  : 6
.......................
1  : 2  : 3  : 4  : 5
```

## 9.3 How to roll a dice

To roll a dice you must import the random module.
Copy this code into Python:



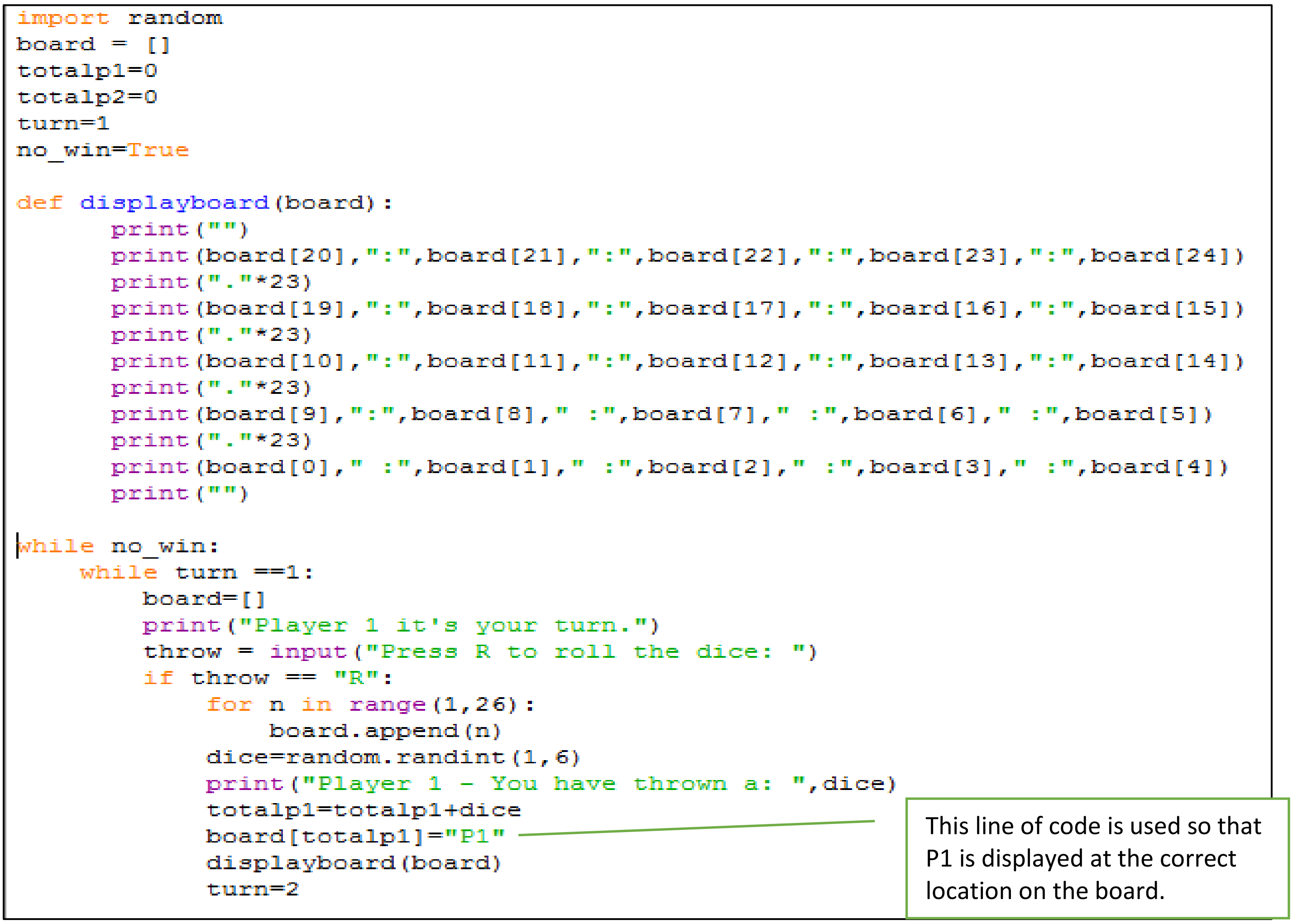
```
#Simple dice throw
import random
dice1=int(0)
play=input("Do you want to roll the dice?")
if play=="Y":
    dice1=random.randint(1,6)
    print("You got a: ",dice1)
```

### Activity 2

Create a program that enables a player to throw **two** dice. The program then tells the player their score from each dice. The program then tells the player the total score (the score when the dice throws are added together).

## 9.4 Creating a two-player game

If you create a two-player game then you need each player to take it in turn to throw the dice. The code below is for the first player. Copy this code and then create the code for the second player.



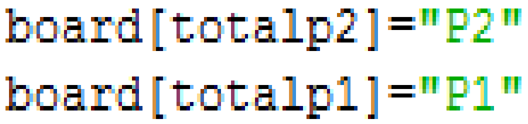
```
import random
board = []
totalp1=0
totalp2=0
turn=1
no_win=True

def displayboard(board):
      print("")
      print(board[20],":",board[21],":",board[22],":",board[23],":",board[24])
      print("."*23)
      print(board[19],":",board[18],":",board[17],":",board[16],":",board[15])
      print("."*23)
      print(board[10],":",board[11],":",board[12],":",board[13],":",board[14])
      print("."*23)
      print(board[9],":",board[8]," :",board[7]," :",board[6]," :",board[5])
      print("."*23)
      print(board[0]," :",board[1]," :",board[2]," :",board[3]," :",board[4])
      print("")

while no_win:
    while turn ==1:
        board=[]
        print("Player 1 it's your turn.")
        throw = input("Press R to roll the dice: ")
        if throw == "R":
            for n in range(1,26):
                board.append(n)
            dice=random.randint(1,6)
            print("Player 1 - You have thrown a: ",dice)
            totalp1=totalp1+dice
            board[totalp1]="P1"
            displayboard(board)
            turn=2
```

This line of code is used so that P1 is displayed at the correct location on the board.

When you create this two-player game there are some limitations.

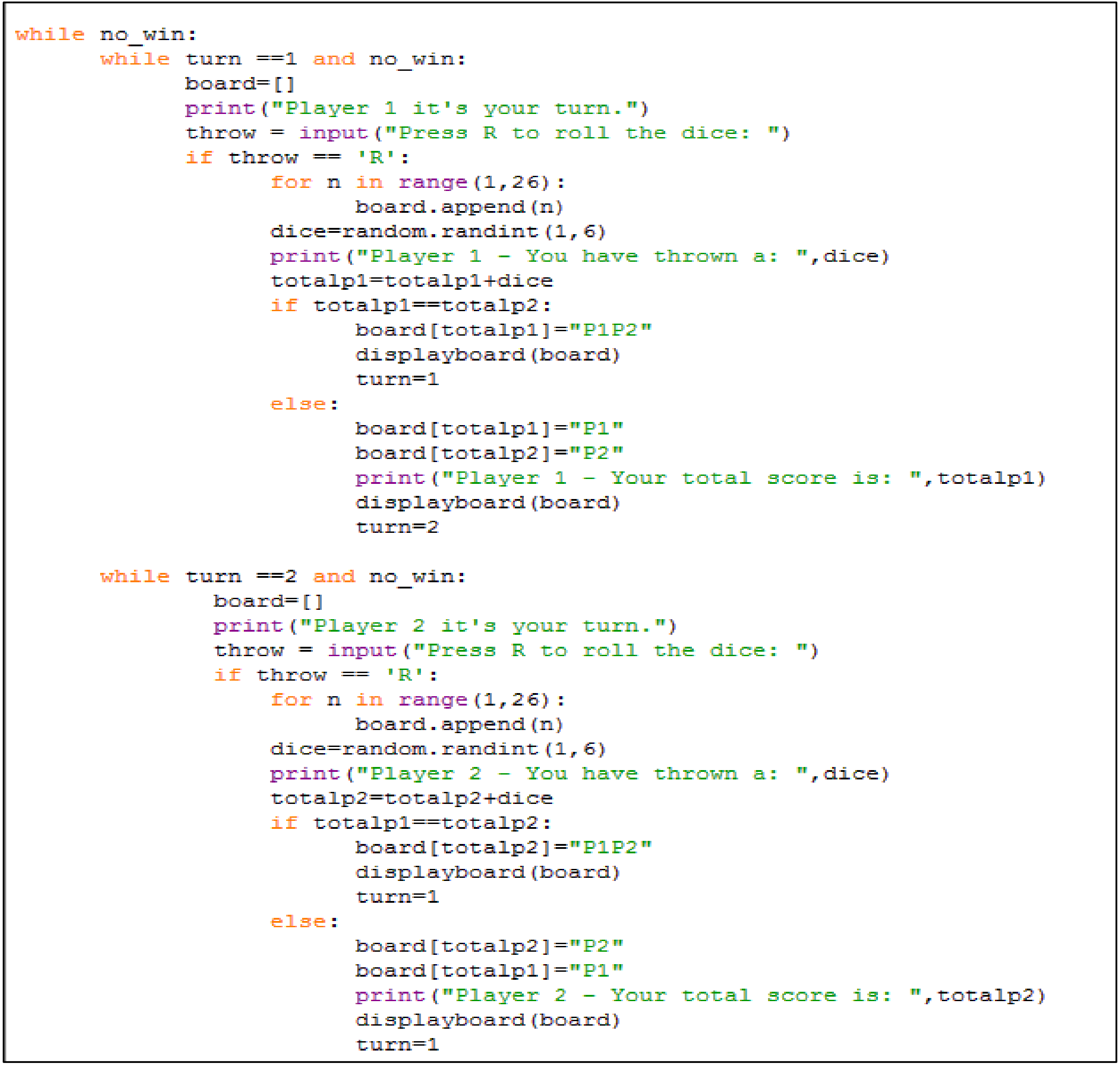
- The board only displays the player (e.g. P1) on the game board when it is the players turn.
- The game does not take into account what would happen if two players landed on the same spot on the board.

To overcome these problems the code will need to be adjusted. To overcome the first limitation, you will need to add a line of code to the program for each player.



```
board[totalp2]="P2"
board[totalp1]="P1"
```

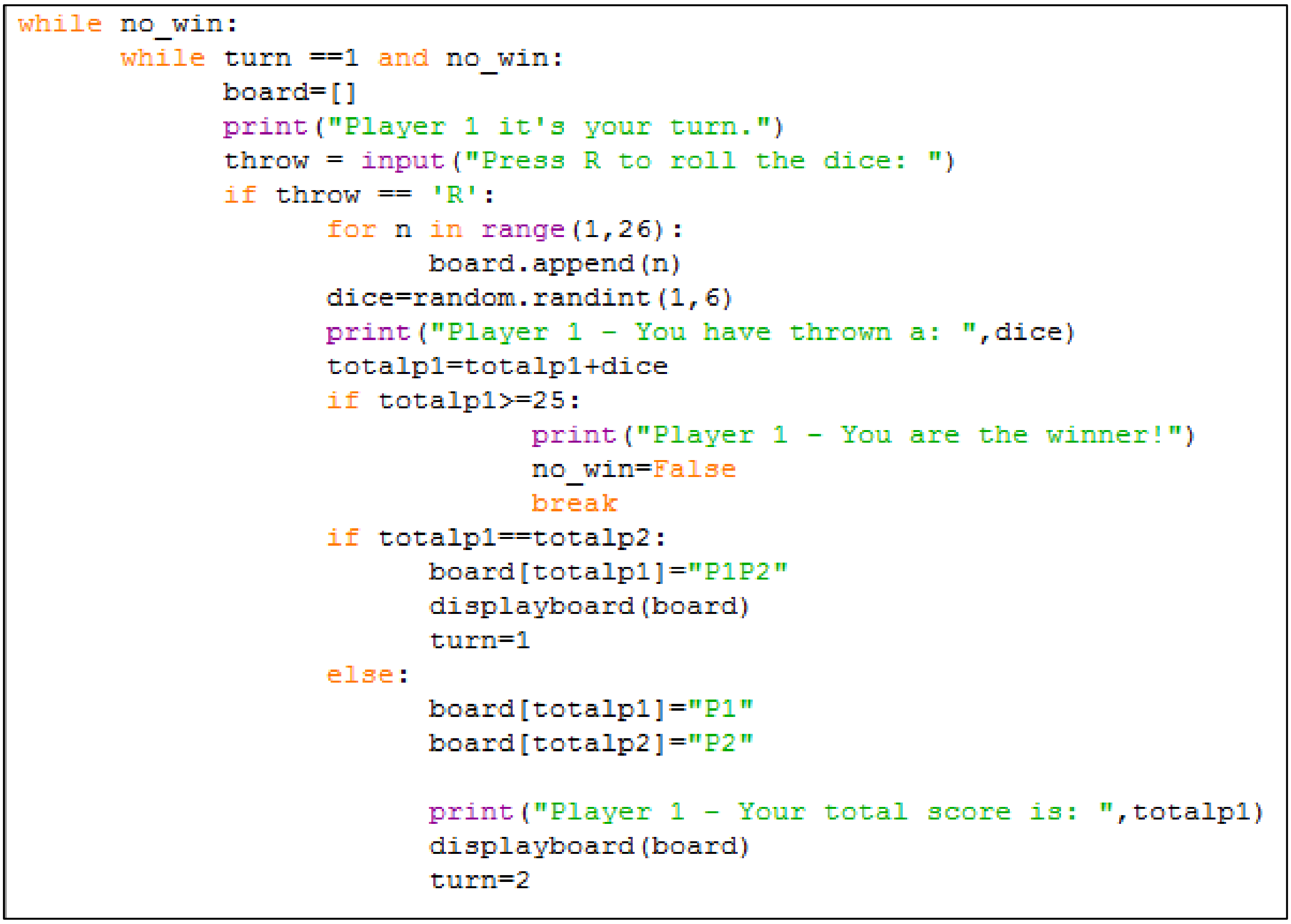
This means that both players will be displayed on the board at the same time.
The second limitation is a little more complicated to resolve. The total score for each player would have to be identical and then you would need to display P1P2 in the same square.
Adapt you code for player 1 and 2 so that it is as shown below.



```
while no_win:
      while turn ==1 and no_win:
            board=[]
            print("Player 1 it's your turn.")
            throw = input("Press R to roll the dice: ")
            if throw == 'R':
                  for n in range(1,26):
                        board.append(n)
                  dice=random.randint(1,6)
                  print("Player 1 - You have thrown a: ",dice)
                  totalp1=totalp1+dice
                  if totalp1==totalp2:
                        board[totalp1]="P1P2"
                        displayboard(board)
                        turn=1
                  else:
                        board[totalp1]="P1"
                        board[totalp2]="P2"
                        print("Player 1 - Your total score is: ",totalp1)
                        displayboard(board)
                        turn=2

      while turn ==2 and no_win:
              board=[]
              print("Player 2 it's your turn.")
              throw = input("Press R to roll the dice: ")
              if throw == 'R':
                  for n in range(1,26):
                        board.append(n)
                  dice=random.randint(1,6)
                  print("Player 2 - You have thrown a: ",dice)
                  totalp2=totalp2+dice
                  if totalp1==totalp2:
                        board[totalp2]="P1P2"
                        displayboard(board)
                        turn=1
                  else:
                        board[totalp2]="P2"
                        board[totalp1]="P1"
                        print("Player 2 - Your total score is: ",totalp2)
                        displayboard(board)
                        turn=1
```

## 9.5 Winning the game

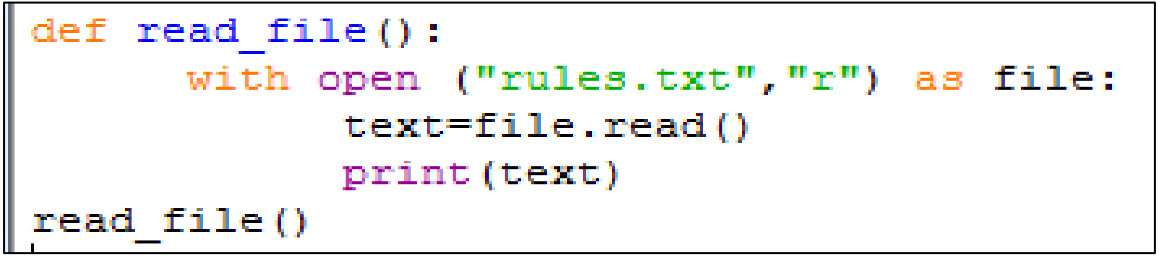
The first player who gets to the final number on the board wins the game. As the game goes on you are keeping a record of the total score for each player. If there are 25 numbers on the grid then it is the first player to reach or exceed 25 on the board.
Adapt your code to show the winner of the game. Here is the code for player 1.



```
while no_win:
      while turn ==1 and no_win:
            board=[]
            print("Player 1 it's your turn.")
            throw = input("Press R to roll the dice: ")
            if throw == 'R':
                  for n in range(1,26):
                        board.append(n)
                  dice=random.randint(1,6)
                  print("Player 1 - You have thrown a: ",dice)
                  totalp1=totalp1+dice
                  if totalp1>=25:
                              print("Player 1 - You are the winner!")
                              no_win=False
                              break
                  if totalp1==totalp2:
                        board[totalp1]="P1P2"
                        displayboard(board)
                        turn=1
                  else:
                        board[totalp1]="P1"
                        board[totalp2]="P2"

                        print("Player 1 - Your total score is: ",totalp1)
                        displayboard(board)
                        turn=2
```

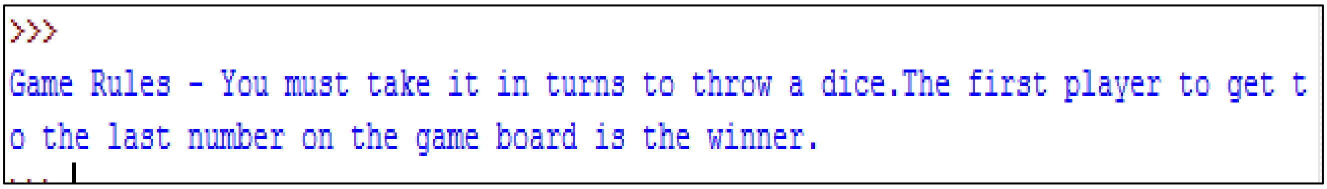
## 9.6 Importing a message that is stored externally.

Create a notepad file called **rules** and enter the rules of the game in this file (This notepad file must be in the same folder as your Python program).
Add this code to your game and then run the program:



```
def read_file():
      with open ("rules.txt","r") as file:
            text=file.read()
            print(text)
read_file()
```

The output from this part of the program is:

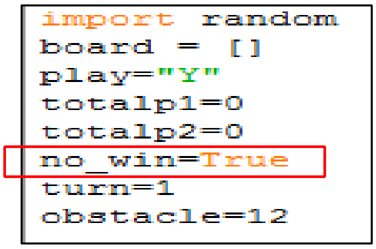


```
>>>
Game Rules - You must take it in turns to throw a dice.The first player to get t
o the last number on the game board is the winner.
```

## 9.7 Adding obstacles to the game

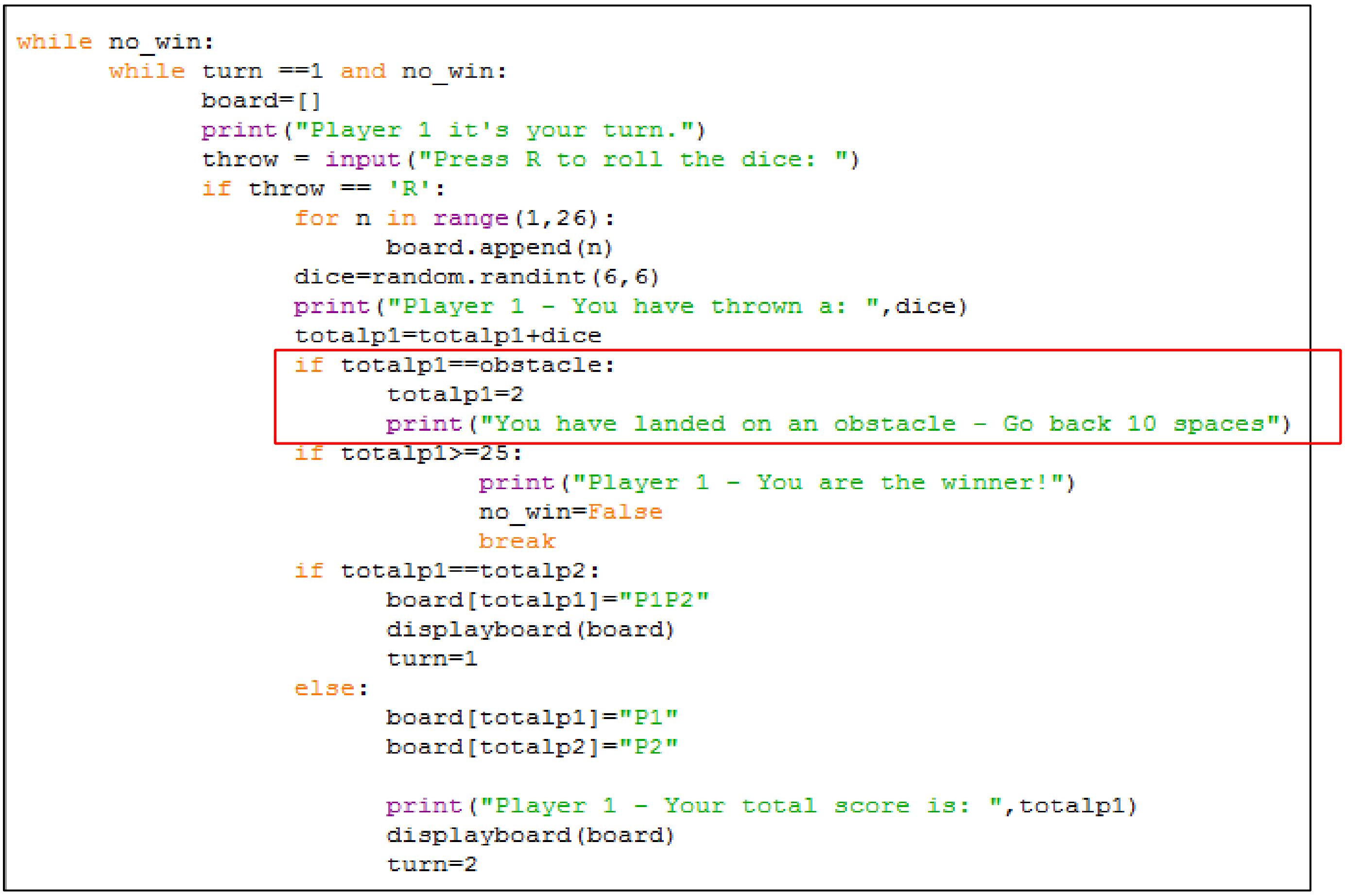
Games like this e.g. Snakes and Ladders, often have obstacles. If you land on an obstacle you move backwards a certain number of spaces.
To do this you will need to create a variable that stores information about where the obstacle is on the board.

Start by adding the variable and assigning it a value.



```
import random
board = []
play="Y"
totalp1=0
totalp2=0
no_win=True
turn=1
obstacle=12
```

Now edit the code for player 1 (you will need to repeat this for player 2)



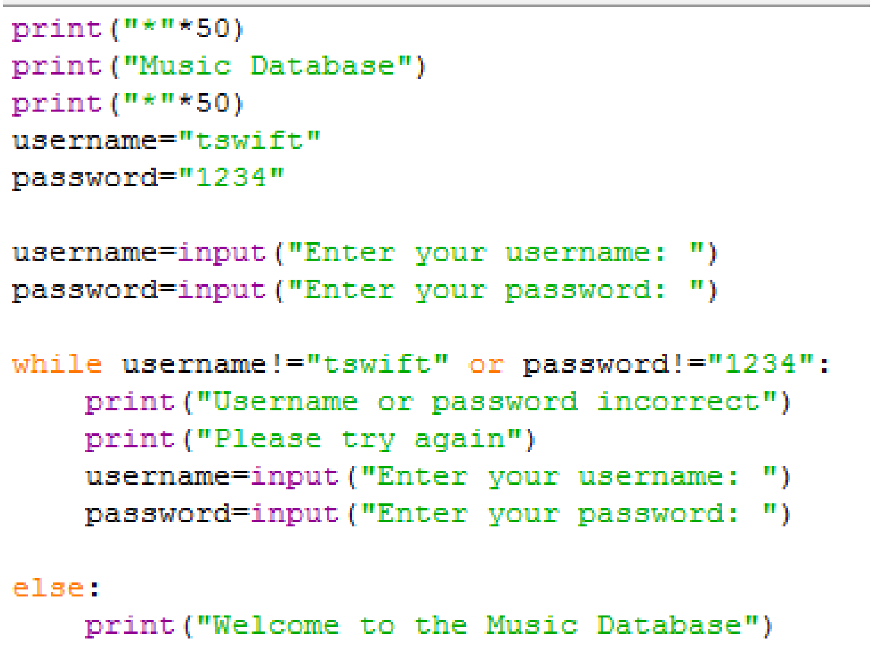
```
while no_win:
    while turn ==1 and no_win:
        board=[]
        print("Player 1 it's your turn.")
        throw = input("Press R to roll the dice: ")
        if throw == 'R':
            for n in range(1,26):
                board.append(n)
            dice=random.randint(6,6)
            print("Player 1 - You have thrown a: ",dice)
            totalp1=totalp1+dice
            if totalp1==obstacle:
                totalp1=2
                print("You have landed on an obstacle - Go back 10 spaces")
            if totalp1>=25:
                    print("Player 1 - You are the winner!")
                    no_win=False
                    break
            if totalp1==totalp2:
                board[totalp1]="P1P2"
                displayboard(board)
                turn=1
            else:
                board[totalp1]="P1"
                board[totalp2]="P2"

                print("Player 1 - Your total score is: ",totalp1)
                displayboard(board)
                turn=2
```

# 10. Creating a Simple Database using Python

## 10.1. Database Login

Databases often store personal and sensitive information and so a secure login is essential to authenticate that the user has permission to access the database. The database used in this example is a database of Music Artists. The person who wishes to use the database has the username: **tswift** and the password: **1234** (This password is not recommended but can be used whilst testing the database).

Copy this code into Python:



```
print("*"*50)
print("Music Database")
print("*"*50)
username="tswift"
password="1234"

username=input("Enter your username: ")
password=input("Enter your password: ")

while username!="tswift" or password!="1234":
    print("Username or password incorrect")
    print("Please try again")
    username=input("Enter your username: ")
    password=input("Enter your password: ")

else:
    print("Welcome to the Music Database")
```

**Activity 1**
Complete the table below:

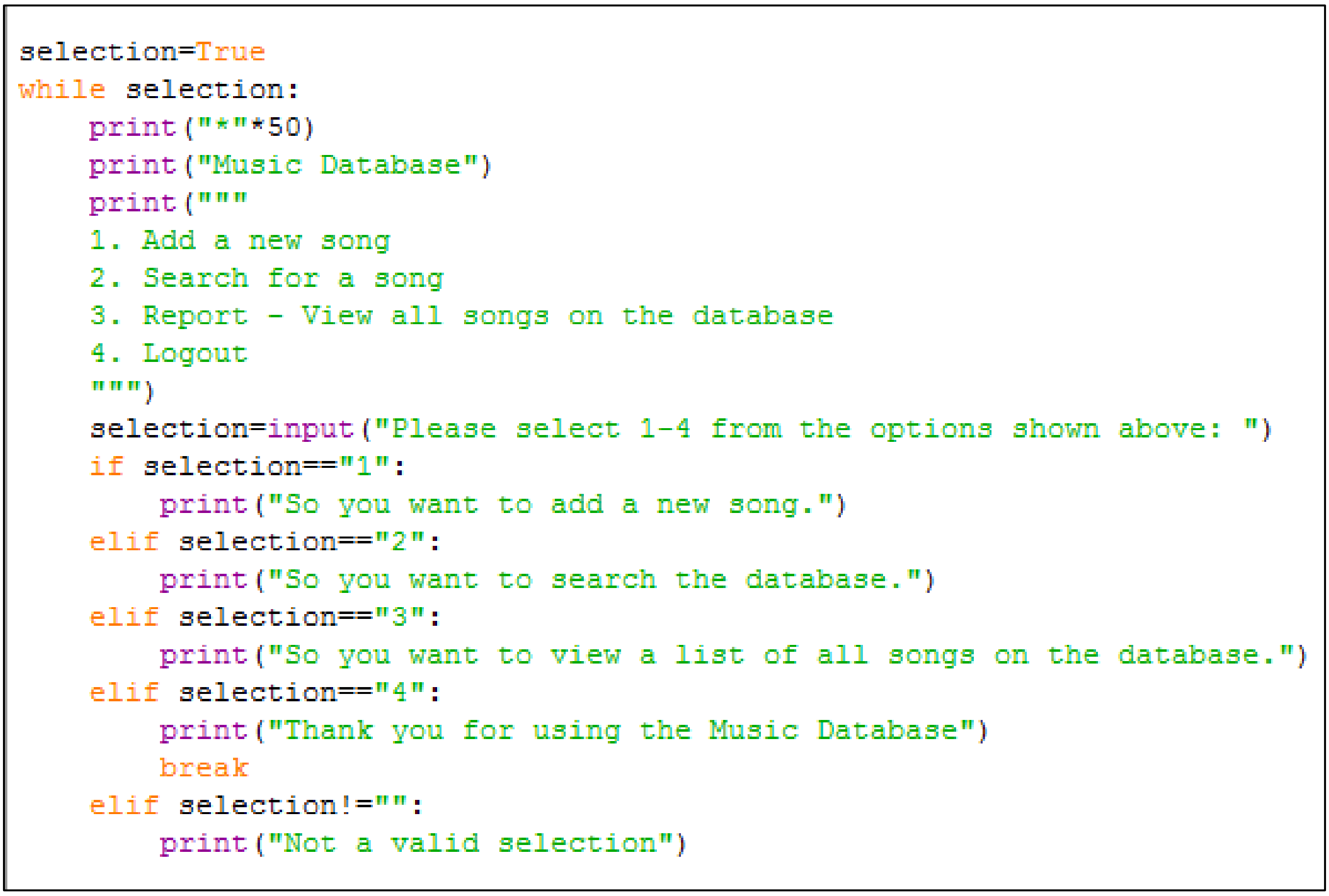
| Test | Expected Result | Actual Result |
|---|---|---|
| A correct username but an incorrect password | | |
| An incorrect username but a correct password | | |
| A correct password and username | | |
| An incorrect password and an incorrect username | | |

Why is it important to carry out these tests?

## 10.2 Database Menu

When you create a database, you need to have a menu system to be able to select different aspects of the database. We need a menu that lets you add a new song to the database and search for a particular song.

Add this code to the Python program you have started:



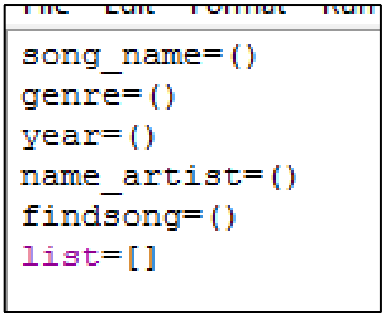
```
selection=True
while selection:
    print("*"*50)
    print("Music Database")
    print("""
    1. Add a new song
    2. Search for a song
    3. Report - View all songs on the database
    4. Logout
    """)
    selection=input("Please select 1-4 from the options shown above: ")
    if selection=="1":
        print("So you want to add a new song.")
    elif selection=="2":
        print("So you want to search the database.")
    elif selection=="3":
        print("So you want to view a list of all songs on the database.")
    elif selection=="4":
        print("Thank you for using the Music Database")
        break
    elif selection!="":
        print("Not a valid selection")
```

### Activity 2

Add another suitable choice to the menu and run the program to test whether this choice can be selected.

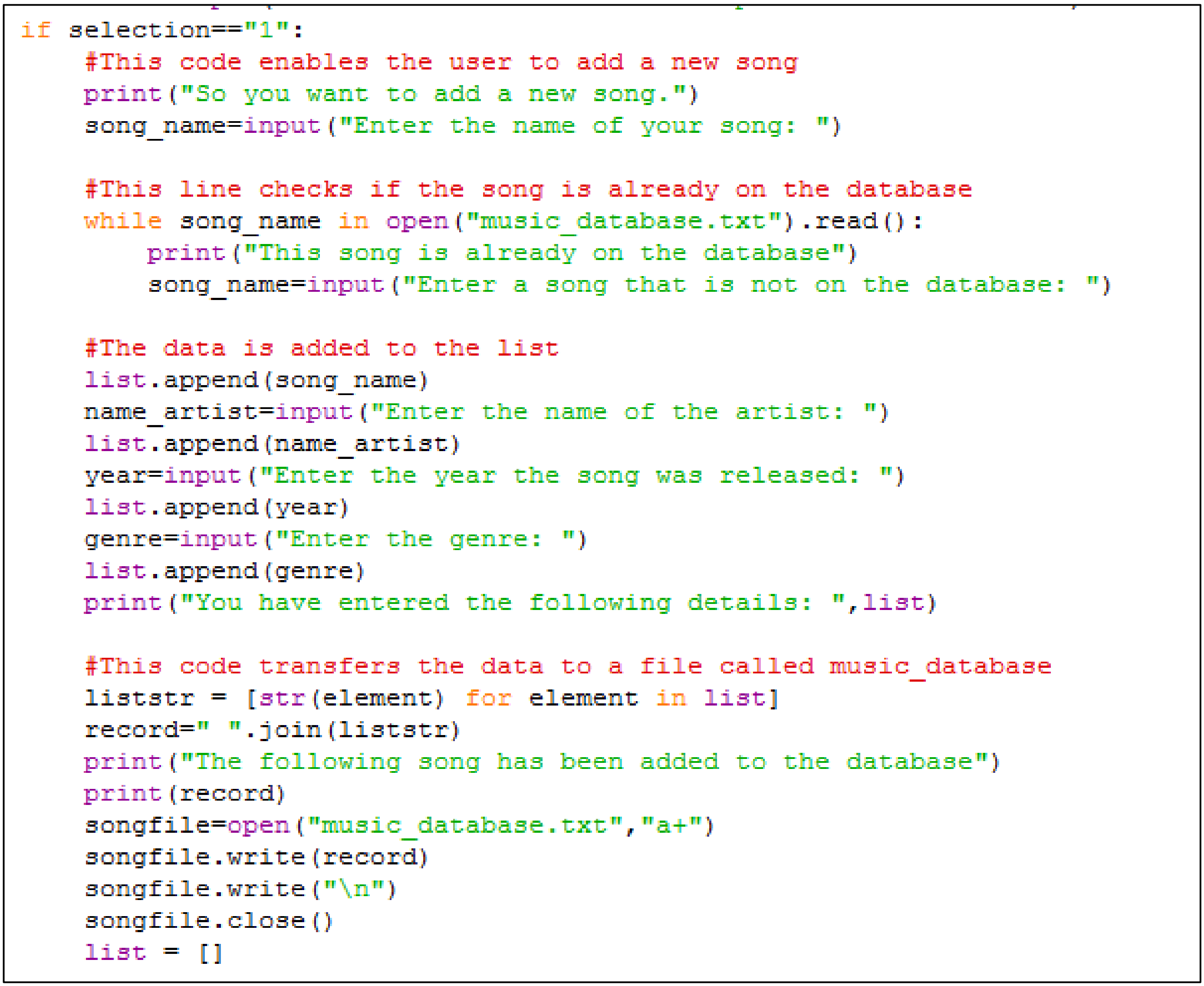
## 10.3 Adding a Record to the Database

To add a record to the database you must check whether the song is already on the database and then get the user to input the information about a song. Once you have gathered the information you can then transfer it to the external data file. Create a Notepad file called **music database** and save it in the save folder as your Python file.
Add this code to the top of the Python program you have started (you need to declare these variables and the list):



```
song_name=()
genre=()
year=()
name_artist=()
findsong=()
list=[]
```

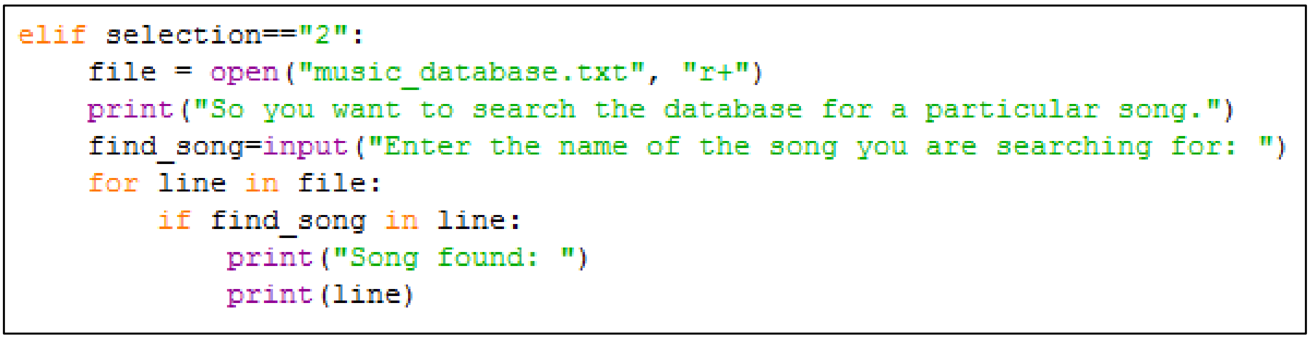
Add this code to the Python program for menu choice 1:



```
if selection=="1":
    #This code enables the user to add a new song
    print("So you want to add a new song.")
    song_name=input("Enter the name of your song: ")

    #This line checks if the song is already on the database
    while song_name in open("music_database.txt").read():
        print("This song is already on the database")
        song_name=input("Enter a song that is not on the database: ")

    #The data is added to the list
    list.append(song_name)
    name_artist=input("Enter the name of the artist: ")
    list.append(name_artist)
    year=input("Enter the year the song was released: ")
    list.append(year)
    genre=input("Enter the genre: ")
    list.append(genre)
    print("You have entered the following details: ",list)

    #This code transfers the data to a file called music_database
    liststr = [str(element) for element in list]
    record=" ".join(liststr)
    print("The following song has been added to the database")
    print(record)
    songfile=open("music_database.txt","a+")
    songfile.write(record)
    songfile.write("\n")
    songfile.close()
    list = []
```

## 10.4 Searching the Database for a Particular Record

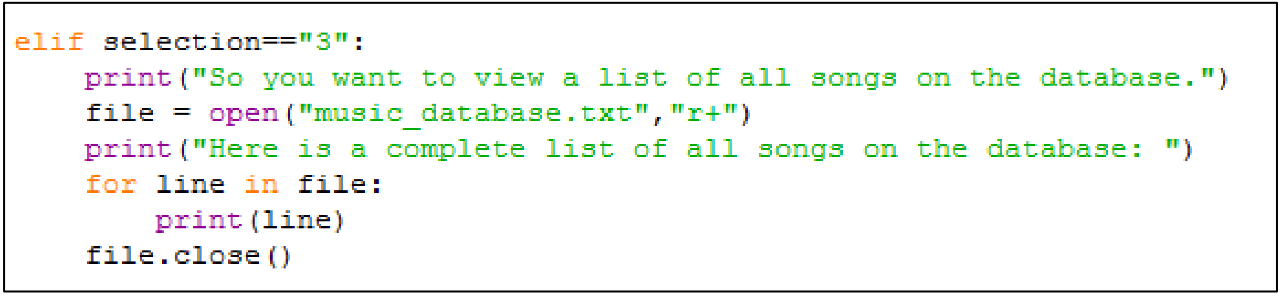
A database is only useful if you can search for the information you are looking for.
Add this code to the Python program for menu choice 2:



```
elif selection=="2":
    file = open("music_database.txt", "r+")
    print("So you want to search the database for a particular song.")
    find_song=input("Enter the name of the song you are searching for: ")
    for line in file:
        if find_song in line:
            print("Song found: ")
            print(line)
```

## 10.5 Display the Database

It is sometimes useful to display the records in a database.
Add this code to the Python program for menu choice 3:



```
elif selection=="3":
    print("So you want to view a list of all songs on the database.")
    file = open("music_database.txt","r+")
    print("Here is a complete list of all songs on the database: ")
    for line in file:
        print(line)
    file.close()
```

### Activity 3

Add a new menu item that lets the user choose a music genre and then all of the records of that genre are displayed.
For example, the user chooses the genre “pop” and then all of the “pop” songs on the database are displayed.