Task 1 - Improving the Sample Message Input

The function GetTypeOfGame() is responsible for getting this message from the user.

Rather than giving an error message the program should provide suitable feedback to the user and allow them to try again:

1. Improve this function so that all valid input (y,Y,Yes,n,N,No etc.) is accepted.
   Answer:

```
def GetTypeOfGame():
  valid = False
  while not valid:
    TypeOfGame = input("Do you want to play the sample game
(enter Y for Yes)? ")
    TypeOfGame = TypeOfGame.lower()[0]
    if TypeOfGame == "y" or TypeOfGame == "n":
      valid = True
    else:
      print("Please enter Y or N")
  return TypeOfGame
```

1. Ensure that this function is being used in the main program to get value for `SampleGame`.
   Answer: `SampleGame = GetTypeOfGame()`

Task 2 - Move Validation

1. Explain what happens when a piece is moved off the right-hand edge of the board.
   -The program crashes because the index is out of range
2. Explain what happens when a piece is move off the left-hand edge of the board.
   -The program outputs
   `That is not a legal move - please try again`
   `Enter coordinates of square containing piece to move (file first):`
3. Explain what happens when a piece is moved off the top edge of the board.
   -The piece goes off the board
4. Explain what happens when a piece is moved off the bottom edge of the board.
   -The program crashes because the index is out of range
5. Explain any differences you encountered whilst attempting the above questions (1-4).
   (1,4)When you attempt to move the piece off the right hand and the bottom edge

the program crashes.
(2)If you attempt to move the piece off the left-hand edge of the board the program will output a message asking for a valid input.
(3)If you attempt to move a piece of the top of the board then it will work.

6. Identify the function responsible for validating whether a move is acceptable or not. `def CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn)`
7. Improve the function identified in question 6 so that an appropriate message is displayed if the move is not valid. The user should then be prompted to reenter the co-ordinates of the move.

```
def CheckMoveIsLegal(Board, StartRank, StartFile, FinishRank,
FinishFile, WhoseTurn):
  MoveIsLegal = True
  if (FinishFile == StartFile) and (FinishRank == StartRank):
    MoveIsLegal = False
  elif 0 > FinishFile > 9 or 0 > FinishRank > 9: #This is what I
added
    MoveIsLegal = False
  else:
    PieceType = Board[StartRank][StartFile][1]
    PieceColour = Board[StartRank][StartFile][0]
    if WhoseTurn == "W":
      if PieceColour != "W":
        MoveIsLegal = False
      if Board[FinishRank][FinishFile][0] == "W":
        MoveIsLegal = False
    else:
      if PieceColour != "B":
        MoveIsLegal = False
      if Board[FinishRank][FinishFile][0] == "B":
        MoveIsLegal = False
    if MoveIsLegal == True:
      if PieceType == "R":
        MoveIsLegal = CheckRedumMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile, PieceColour)
      elif PieceType == "S":
        MoveIsLegal = CheckSarrumMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile)

      elif PieceType == "M":
        MoveIsLegal = CheckMarzazPaniMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile)
      elif PieceType == "G":
```

```
        MoveIsLegal = CheckGisgigirMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile)
      elif PieceType == "N":
        MoveIsLegal = CheckNabuMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile)
      elif PieceType == "E":
        MoveIsLegal = CheckEtluMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile)
  return MoveIsLegal
```

Task 3 - Rank and File Validation

1. Identify the function responsible for getting the move from the user.
   `def GetMove(StartSquare, FinishSquare):`
2. Improve this function so that the start and end positions are validated separately. This means that an appropriate error message should be displayed as soon as invalid data has been entered:

```
def GetMove(StartSquare, FinishSquare):
  valid = False
  while not valid:
    StartSquare = int(input("Enter coordinates of square
containing piece to move (file first): "))
    if StartSquare >10 and StartSquare <89:
      valid = True
    else:
      print("Please provide both FILE and RANK for this move")
  valid = False
  while not valid:
    FinishSquare = int(input("Enter coordinates of square to move
piece to (file first): "))
    if FinishSquare >10 and FinishSquare <89:
      valid = True
    else:
      print("Please provide both FILE and RANK for this move")
  return StartSquare, FinishSquare
```

Task 4 - Move Confirmation

1. Create a new function called ConfirmMove() that will take StartSquare and FinishSquare as parameters and return whether the move was confirmed or not.

```
def ConfirmMove(StartSquare, FinishSquare):
  StartSquare = str(StartSquare)
  FinishSquare = str(FinishSquare)
  print("Move from Rank {0}, File {1} to Rank {2}, File
{3}?".format(StartSquare[1],StartSquare[0],FinishSquare[1],Finish
Square[0]))
  question = input("Confirm move?(Yes/No): ")
  question = question.lower()[0]
  if question == "y":
    confirm = True
  elif question == "n":
    confirm = False
  else:
    print("Please enter a valid answer!")
  return confirm
```

1. Amend the main program to confirm the move before it is made.

```
if __name__ == "__main__":
  Board = CreateBoard() #0th index not used
  StartSquare = 0
  FinishSquare = 0
  PlayAgain = "Y"
  while PlayAgain == "Y":
    WhoseTurn = "W"
    GameOver = False
    SampleGame = GetTypeOfGame()
    if ord(SampleGame) >= 97 and ord(SampleGame) <= 122:
      SampleGame = chr(ord(SampleGame) - 32)
    InitialiseBoard(Board, SampleGame)
    while not(GameOver):
      DisplayBoard(Board)
      DisplayWhoseTurnItIs(WhoseTurn)
      MoveIsLegal = False
      while not(MoveIsLegal):
        StartSquare, FinishSquare = GetMove(StartSquare,
FinishSquare)
        StartRank = StartSquare % 10
        StartFile = StartSquare // 10
        FinishRank = FinishSquare % 10
```

```
        FinishFile = FinishSquare // 10
        MoveIsLegal = CheckMoveIsLegal(Board, StartRank,
StartFile, FinishRank, FinishFile, WhoseTurn)
        if not(MoveIsLegal):
          print("That is not a legal move - please try again")
        if not(MoveIsLegal):
          confirm = True
        else:
          confirm = ConfirmMove(StartSquare, FinishSquare)
#amending my function
        if not confirm:
          MoveIsLegal = False #restarts the while loop
      GameOver = CheckIfGameWillBeWon(Board, FinishRank,
FinishFile)
      MakeMove(Board, StartRank, StartFile, FinishRank,
FinishFile, WhoseTurn)
      if GameOver:
        DisplayWinner(WhoseTurn)
      if WhoseTurn == "W":
        WhoseTurn = "B"
      else:
        WhoseTurn = "W"
    PlayAgain = input("Do you want to play again (enter Y for
Yes)? ")
    if ord(PlayAgain) >= 97 and ord(PlayAgain) <= 122:
      PlayAgain = chr(ord(PlayAgain) - 32)
```

Task 5 - Game Piece Removal Confirmation

1. Describe what is returned by the game when a position on the board containing a piece is selected e.g. Board[4][3] (if there where a piece in that position).
   The colour of the piece and the type of the piece is being returned.
2. Create a new function called GetPieceName() that takes the value identified in question 1 and returns the full name of both the colour and type of piece.

```
def GetPieceName(StartRank, StartFile, Board):
  #Gets the colour of the piece in the board
  piece_colour = Board[StartRank][StartFile][0]
  if piece_colour == "W":
    piece_colour = "White"
  elif piece_colour == "B":
```

```
    piece_colour = "Black"
  #Gets the name of the piece in the board
  piece_name = Board[StartRank][StartFile][1]
  if piece_name == "R":
    piece_name = "Redum"
  elif piece_name == "S":
    piece_name = "Sarrum"
  elif piece_name == "M":
    piece_name = "MarzazPani"
  elif piece_name == "G":
    piece_name = "Gisgigir"
  elif piece_name == "N":
    piece_name = "Nabu"
  elif piece_name == "E":
    piece_name = "Etlu"
  return piece_colour,piece_name
```

1. Identify the function responsible for moving the pieces on the board.
   `def MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):`
2. Improve the function identified in question 3 to make use of the new function GetPieceName() to present the user with a message similar to the one in the screenshot above.

```
def MakeMove(Board, StartRank, StartFile, FinishRank,
FinishFile, WhoseTurn):
  if WhoseTurn == "W" and FinishRank == 1 and Board[StartRank]
[StartFile][1] == "R":
    Board[FinishRank][FinishFile] = "WM"
    Board[StartRank][StartFile] = "  "
    #Just needs a print statement :D
    print("White Redum promoted to Marzaz Pani")
  elif WhoseTurn == "B" and FinishRank == 8 and Board[StartRank]
[StartFile][1] == "R":
    Board[FinishRank][FinishFile] = "BM"
    Board[StartRank][StartFile] = "  "
    print("Black Redum promoted to Marzaz Pani")
  else:
    piece_colour, piece_name = GetPieceName(FinishRank,
FinishFile, Board)
    piece_colour2, piece_name2 = GetPieceName(StartRank,
StartFile, Board)
    if piece_colour == "White" or piece_colour == "Black":
      print("{0} {1} has been taken by {2}
```

```
{3}".format(piece_colour, piece_name, piece_colour2,
piece_name2))
    Board[FinishRank][FinishFile] = Board[StartRank][StartFile]
    Board[StartRank][StartFile] = "  "
```

Task 6 - Redum Promotion Confirmation

1. Identify the function where the Redum is promoted to the marzaz Pani.
   `def MakeMove(Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn):`
2. Improve the function identified in question one so that an appropriate message is displayed when the Redum piece is promoted.

```
def MakeMove(Board, StartRank, StartFile, FinishRank,
FinishFile, WhoseTurn):
  if WhoseTurn == "W" and FinishRank == 1 and Board[StartRank]
[StartFile][1] == "R":
    Board[FinishRank][FinishFile] = "WM"
    Board[StartRank][StartFile] = "  "
    #Just needs a print statement :D
    #Thank you for having mercy after that last task QQ
    print("White Redum promoted to Marzaz Pani")
  elif WhoseTurn == "B" and FinishRank == 8 and Board[StartRank]
[StartFile][1] == "R":
    Board[FinishRank][FinishFile] = "BM"
    Board[StartRank][StartFile] = "  "
    print("Black Redum promoted to Marzaz Pani")
  else:
    piece_colour, piece_name = GetPieceName(FinishRank,
FinishFile, Board)
    print("{0} takes {1}".format(piece_colour, piece_name))
    Board[FinishRank][FinishFile] = Board[StartRank][StartFile]
    Board[StartRank][StartFile] = "  "
```

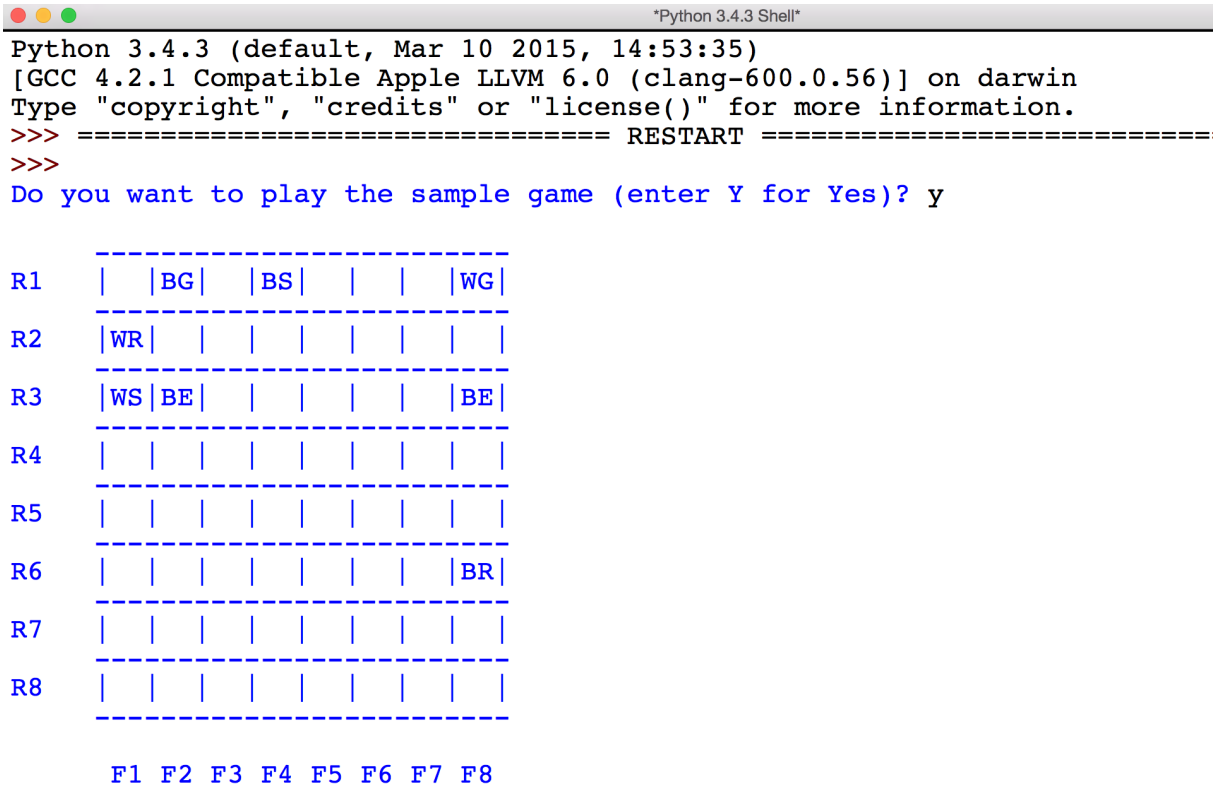
Task 7 - Board Layout

If I get picked I hope its for this!

1. Identify the function where the board is generated to be displayed.

`def DisplayBoard(Board):`

2. Amend this function so that the board closely resembles the one in the screenshot below.




```
#Trial and error
def DisplayBoard(Board):
  print()
  for RankNo in range(1, BOARDDIMENSION + 1):
    print("    ------------------------")
    #changed 'print(RankNo), end="   ")
    print("R{0}".format(RankNo), end="   ")
    for FileNo in range(1, BOARDDIMENSION + 1):
      print("|" + Board[RankNo][FileNo], end="")
    print("|")
  print("    ------------------------")
  print()
  print("      F1 F2 F3 F4 F5 F6 F7 F8")
  print()
  print()
```

Task 8 - Variable roles

1. Describe each variable role in your own words.
2. Give an example of variable from the program code for each variable role (if possible).

Role	Description	Example
Fixed Value	A Variable that's value does not change.	`Board[1][2] = "BG"`
Stepper	A stepper goes through a succession of values in a systematic way.	Iteration
Most recent holder	This variable is used to hold the latest value encountered or obtained.	`question = input("Confirm move? (Yes/No): ")`
Most wanted holder	This variable holds the most appropriate data gained so far	
Gather	This gathers the effect of individual values	
Transformation	This variable gets new values from a fixed calculation	
Follower	This variable gains new data from the value of other data	
Temporary	This variable only holds value for a short amount of time	

Task 9 - Functions and parameters

1. Describe the difference between passing by value and passing by reference in your own words.
   When a parameter is passed by reference the variable uses the same memory as the parameter. This means that any changes made to the variable that has been passes will make changes to the variable.
   When a parameter is being passed be value it is making a copy of the value. This means that any changes to the parameter will not change the original value of the variable.

2. For each function in the program identify the mechanism using to pass each parameter. Note: this task will take a while but it will improve your understanding of the program and by useful for the exam.

Data Type	Passing Mechanism
Integer	by value
Float	by value
String	by value
Boolean	by value
List	by reference
Record	by reference

Function	Parameter(s)	Passing Method
CreateBoard	NA	NA
DisplayWhoseTurnItIs	WhoseTurn	Value
GetPieceName	Rank,File,Board	Value x2,Reference
GetTypeOfGame	NA	NA
DisplayWinner	WhoseTurn	Value
CheckIfGameWillBeWon	Board, FinishRank, FinishFile	Reference,Value x2
DisplayBoard	Board	Reference
CheckRedumMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile, ColourOfPiece	Reference,Value x5
CheckSarrumMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
CheckGisgigirMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4

Function	Parameter(s)	Passing Method
CheckNabuMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
CheckMarzazPaniMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
CheckEtluMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
CheckMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn	Reference, Value x5
InitialiseBoard	Board, SampleGame	Reference, Value
GetMove	StartSquare, FinishSquare	Value x2
ConfirmMove	StartSquare, FinishSquare	Value x2
MakeMove	Board, StartRank, StartFile, FinishRank, FinishFile, WhoseTurn	Reference, Value x5