Task Sheet 2 - Game Improvements

Introduction

This series of tasks focuses on improving the user interface and functionality of the Capture the Sarrum game.

Task 11 - Main Menu

When the user first loads the program they are presented with the following option:

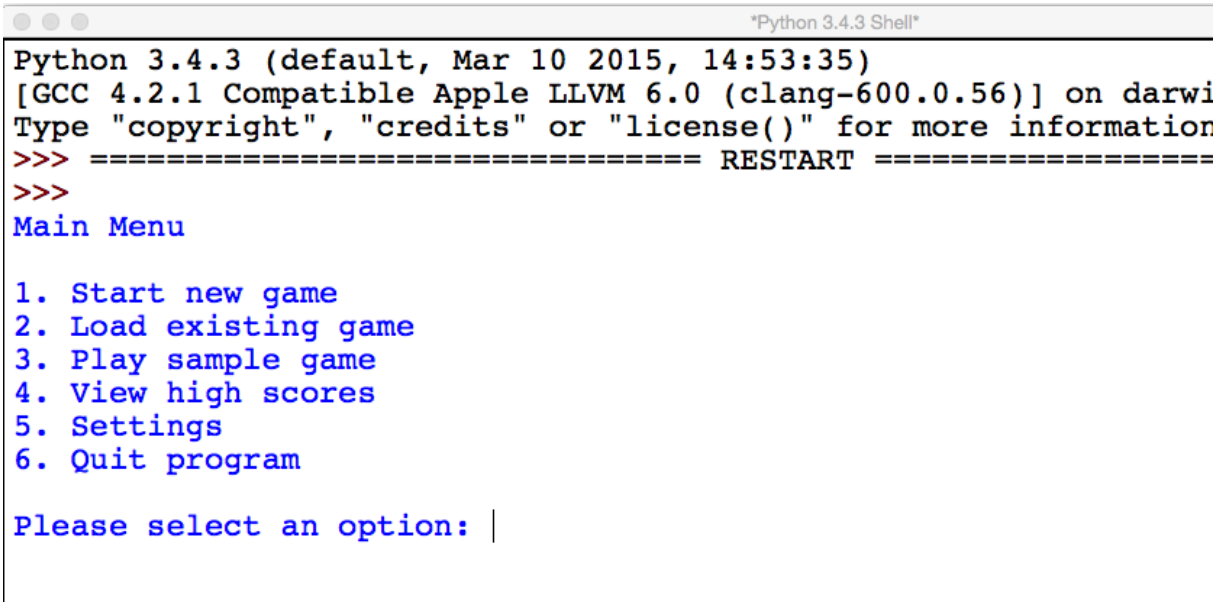
This is fine at the moment as the game has limited functionality. If however, the program is to be improved so that it has additional options available it would be better if a **menu** is presented instead:

To do this the following additional functions are required:

- `display_menu()`
- `get_menu_selection()`
- `make_selection()`
- `play_game()`

Attempt the **exercises** below.

Task 11 - Main Menu



1. Attempt to explain the purpose of each of the new functions listed below and indicate any parameters or return values that each of them will require.

Functions	Purpose	Parameters	Return Values
`display_menu()`	Displays a menu for the User to see	None	None
`get_menu_selection()`	Gets information for the menu	None	selection
`make_selection()`	Processes the options from the menu	selection	None
`play_game()`	To Play the sample game	None	None

2. Write the function `display_menu()` and add it to the existing program.

```
def display_menu(): #Task 11
  print("Main Menu")
  print("1. Start new game")
  print("2. Load exsiting game")
  print("3. Play sample game")
  print("4. View high scores")
  print("5. Settings")
  print("6. Quit Program")
```

1. Write the function `get_menu_selection()` including any validation that is

necessary and add it to the existing program.

```
def get_menu_selection(): #Task 11
  valid = False
  while not valid:
    selection = int(input("Please select an option"))
    if selection >0 and selection <=6:
      valid = True
    else:
      print("Please enter a valid number")
      valid = False
  return selection
```

1. Write the function `make_selection()` and add it to the existing program.

```
def make_selection(selection): #Task 11
  if selection == 1: #Start new game
    pass
  elif selection == 2: #Load exsiting game
    pass
  elif selection == 3: #Play sample game
    sample_game = play_game(Board)
  elif selection == 4: #View high scores
    pass
  elif selection == 5: #Settings
    pass
  elif selection == 6: #Quit Program
    pass
```

1. Write the function `play_game()`, refactoring the existing program to make use of it.

```
def play_game(Board):
  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")
        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 12 - In-game Menu

At the moment once a game has been started there is no way to interrupt it aside from quitting the program and starting it again. Clearly this is unsatisfactory. There are many instances where being able to interrupt the game would be useful, for example:

- Saving the game
- Quiting the game

An option to recognise a **rogue value** could be added so that an in-game menu could be displayed:

Attempt the **exercises** below.

---

1. **Identify** the functions that will require modification to make it possible for an in-game menu to be presented.
   `def GetMove()`
2. **Write** the functions necessary to **display** the menu and get an **option** from the user and then add them to the existing program.

```
def display_in_game_options():
  print("Options")
  print("1. Save Game")
  print("2. Quit to Menu")
  print("3. Return to Game")

def get_option(StartSquare, FinishSquare):
  print()
  option = int(input("Please select an option: "))
  if option == 1: #Save Game
    pass
  elif option == 2: #Quit to Menu
    display_menu()
    selection = get_menu_selection()
    choice = make_selection(selection)
  elif option == 3: #Return to Game
    StartSquare, FinishSquare = GetMove(StartSquare, FinishSquare)
```

1. **Modify** the program so that the it can make use of the new functions you have written to present the in-game menu.

```
def GetMove(StartSquare, FinishSquare):
  valid = False
  while not valid:
    StartSquare = int(input("Enter coordinates of square
containing piece to move (file first) or type '-1' for menu: "))
    if StartSquare >10 and StartSquare <89:
      valid = True
    elif StartSquare = -1: #Task 12
      valid = True
    else:
      print("Please provide both FILE and RANK for this move")
  if StartSquare = -1: #I did this so that it would not ask for
the Finish Square
    FinishSquare = " "
  else:
    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

def play_game(Board):
  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)
        if StartSquare == -1: #Task 12
          display_in_game_options()
          options = get_option(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")
        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 13 - Surrendering

One option that is common in games like Chess is the ability to surrender or forfeit a game. This may be necessary as the player could be in position where there is no way to win but it may require several more moves to actually get to that game state or simply that the

player has to go and has no more time to play!

In the above screenshot the option to surrender has been added to the **in-game** menu and the message displayed to show the winner is appropriate to the situation.

Attempt the **exercises** below.

---

1. **Identify** the functions that will require modification to make it possible to surrender during the game. **Explain** why each function will require modification.

Functions	Role
`def display_in_game_options`	Displays the surrender option
`def get_option`	Surrenders the game if the option is chosen

2. **Modify** the functions identified above so that surrendering is possible.

```
def display_in_game_options():
  print()
  print("Options")
  print("1. Save Game")
  print("2. Quit to Menu")
  print("3. Return to Game")
  print("4. Surrender") #Task 13

def get_option(StartSquare, FinishSquare,WhoseTurn):
  print()
  option = int(input("Please select an option: "))
  if option == 1: #Save Game
    pass
  elif option == 2: #Quit to Menu
    display_menu()
    selection = get_menu_selection()
    choice = make_selection(selection)
  elif option == 3: #Return to Game
    StartSquare, FinishSquare = GetMove(StartSquare, FinishSquare)
  elif option == 4: #Task 13
    if WhoseTurn == "W":
      print("White surrendered. Black wins!")
```

```
    else:
      print("Black surrendered. White wins!")
```

Task 14 - Refactoring

The function `InitialiseBoard()` contains the following code:

```
def InitialiseBoard(Board, SampleGame):
  if SampleGame == "Y":
    for RankNo in range(1, BOARDDIMENSION + 1):
      for FileNo in range(1, BOARDDIMENSION + 1):
        Board[RankNo][FileNo] = "  "
    Board[1][2] = "BG"
    Board[1][4] = "BS"
    Board[1][8] = "WG"
    Board[2][1] = "WR"
    Board[3][1] = "WS"
    Board[3][2] = "BE"
    Board[3][8] = "BE"
    Board[6][8] = "BR"
  else:
    for RankNo in range(1, BOARDDIMENSION + 1):
      for FileNo in range(1, BOARDDIMENSION + 1):
        if RankNo == 2:
          Board[RankNo][FileNo] = "BR"
        elif RankNo == 7:
          Board[RankNo][FileNo] = "WR"
        elif RankNo == 1 or RankNo == 8:
          if RankNo == 1:
            Board[RankNo][FileNo] = "B"
          if RankNo == 8:
            Board[RankNo][FileNo] = "W"
          if FileNo == 1 or FileNo == 8:
            Board[RankNo][FileNo] = Board[RankNo][FileNo] + "G"
          elif FileNo == 2 or FileNo == 7:
            Board[RankNo][FileNo] = Board[RankNo][FileNo] + "E"
          elif FileNo == 3 or FileNo == 6:
            Board[RankNo][FileNo] = Board[RankNo][FileNo] + "N"
          elif FileNo == 4:
```

```
            Board[RankNo][FileNo] = Board[RankNo][FileNo] + "M"
          elif FileNo == 5:
            Board[RankNo][FileNo] = Board[RankNo][FileNo] + "S"
        else:
          Board[RankNo][FileNo] = "  "
```

It could be **refactored** so that there are three functions instead:

- `InitialiseBoard()`
- `InitialiseNewBoard()`
- `InitialiseSampleBoard()`

Attempt the **exercises** below.

---

1. **Explain** what is meant by the term refactoring and why it is sometimes useful to refactor sections of code.
   Refactoring is when the design of existing software code is changed to make it clearing however refactoring doesn't change the action of the code.
2. **Refactor** the `InitialiseBoard()` function so that there are the three functions identified in the list above.

```
def InitialiseBoard(Board, SampleGame):
  if SampleGame == "Y":
    game_board = InitialiseSampleBoard(Board, SampleGame)
  else:
    game_board = InitialiseNewBoard(Board)

def InitialiseSampleBoard(Board, SampleGame):
  for RankNo in range(1, BOARDDIMENSION + 1):
    for FileNo in range(1, BOARDDIMENSION + 1):
      Board[RankNo][FileNo] = "  "
  Board[1][2] = "BG"
  Board[1][4] = "BS"
  Board[1][8] = "WG"
  Board[2][1] = "WR"
  Board[3][1] = "WS"
  Board[3][2] = "BE"
  Board[3][8] = "BE"
  Board[6][8] = "BR"

def InitialiseNewBoard(Board):
  for RankNo in range(1, BOARDDIMENSION + 1):
```

```
        for FileNo in range(1, BOARDDIMENSION + 1):
          if RankNo == 2:
            Board[RankNo][FileNo] = "BR"
          elif RankNo == 7:
            Board[RankNo][FileNo] = "WR"
          elif RankNo == 1 or RankNo == 8:
            if RankNo == 1:
              Board[RankNo][FileNo] = "B"
            if RankNo == 8:
              Board[RankNo][FileNo] = "W"
            if FileNo == 1 or FileNo == 8:
              Board[RankNo][FileNo] = Board[RankNo][FileNo] + "G"
            elif FileNo == 2 or FileNo == 7:
              Board[RankNo][FileNo] = Board[RankNo][FileNo] + "E"
            elif FileNo == 3 or FileNo == 6:
              Board[RankNo][FileNo] = Board[RankNo][FileNo] + "N"
            elif FileNo == 4:
              Board[RankNo][FileNo] = Board[RankNo][FileNo] + "M"
            elif FileNo == 5:
              Board[RankNo][FileNo] = Board[RankNo][FileNo] + "S"
          else:
            Board[RankNo][FileNo] = "  "
```

Task 15 - Variable Roles

Section B of the COMP1 exam focuses on your understanding of the program source code. Often the questions will focus on the **role of variables** in the program. There are several different roles that a variable can have: they are described on **page 66** of the AS textbook.

Answer the following questions.

---

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

Role	Description	Example

Role	Description	Example
Fixed Value	A Variable that’s value does not change.	Board Dimension
Stepper	A stepper goes through a succession of values in a systematic way.	count
Most recent holder	This variable is used to hold the latest value encountered or obtained.	StartSquare
Most wanted holder	This variable holds the most appropriate data gained so far	PieceType
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 16 - Improve Marzaz Pani

Currently the **Marzaz Pani** piece can move in the following ways:

It can move **one space** horizontally or vertically in any direction. An improvement would be to make it so that it could also move **diagonally**:

Attempt the **exercise** below.

---

1. Make any necessary **modifications**so that the Marzaz Pani piece can move one space diagonally in any direction, in addition to its current movement range.

Task 17 - Improve Nabu

Currently the **Nabu** piece can move in the following ways:

It can move **one space** diagonally in any direction. An improvement would be to make it so that it could move **any number of spaces** diagonally (so that it was like a Bishop in Chess):

Attempt the **exercise** below.

---

1. Make any necessary **modifications** so that the Nabu piece can move any number of spaces diagonally in any direction.

Task 18 - Improve Etlu

Currently the **Etlu** piece can move in the following ways:

It can move **exactly two spaces** in a horizontal or vertical direction and **jump** over pieces that are in its way. An improvement would be to make it so that its movement was like that of a **Knight** in Chess:

Attempt the **exercise** below.

---

1. Make any necessary **modifications** so that the Nabu piece can move like a Knight in Chess.

Task 19 - Improve Redum

The **Redum** piece in Capture the Sarrum is similar to the **Pawn** in Chess. Once difference is that a pawn can move **upto** two spaces on its first move. Currently the **Redum** can only move one:

Each **Redum** piece should have the option of moving either one or two spaces on its opening move **only**:

Attempt the **exercises** below.

---

1. Make any necessary **modifications** so that the Redum piece can move up to two spaces on its opening move.

Task 20 - Functions and Parameters

When **binding** arguments to parameters they are passed into the function either *by value* or *by reference*. In some programming languages you can specify which method to use but in Python this is done automatically for you. Some values are passed by value and others by reference - it depends on the value's **data type**.

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

Data Type	Passing Mechanism
Record	by reference

The AS textbook has a good section on passing by value and passing by reference on **pages 63 to 65**.

Answer the following questions.

---

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.

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

Function	Parameter(s)	Passing Method
CheckSarrumMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
CheckGisgigirMoveIsLegal	Board, StartRank, StartFile, FinishRank, FinishFile	Reference,Value x4
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
display_menu	NA	NA
get_menu_selection	NA	NA
make_selection	selection	Value
InitialiseSampleBoard	Board, SampleGame	Reference, Value
InitialiseNewBoard	Board	Reference