Duel of sorcerers

Intro

You are witnessing an epic battle between two powerful sorcerers: Gandalf and Saruman. Each sorcerer has 10 spells of variable power in their mind and they are going to throw them one after the other. The winner of the duel will be the one who wins more of those clashes between spells. Spells are represented as a list of 10 integers whose value equals the power of the spell.

```
gandalf = [10, 11, 13, 30, 22, 11, 10, 33, 22, 22]
saruman = [23, 66, 12, 43, 12, 10, 44, 23, 12, 17]
```

For example:

1. The first clash is won by Saruman: 10 against 23, wins 23
2. The second clash wins Saruman: 11 against 66, wins 66
3. etc.

You will create two variables, one for each sorcerer, where the sum of clashes won will be stored. Depending on which variable is greater at the end of the duel, you will show one of the following three results on the screen:

- Gandalf wins
- Saruman wins
- Tie

LINKS:

- zip: https://www.programiz.com/python-programming/methods/built-in/zip (https://www.programiz.com/python-programming/methods/built-in/zip)

Solution

In [1]:

```
# Assign spell power lists to variables
gandalf = [10, 11, 13, 30, 22, 11, 10, 33, 22, 22]
saruman = [23, 66, 12, 43, 12, 10, 44, 23, 12, 17]
```

In [2]:

```
# Assign 0 to each variable that stores the victories
```

In [16]:

```
# Execution of spell clashes

def play_game(player_1,player_2):
    victories_player1 = 0
    victories_player2 = 0
    ties = 0
    winner = ""
    battles = list(zip(player_1, player_2))
    for battle in battles:
        if battle[0] > battle[1]:
            victories_player1 += 1
        if battle[0] == battle[1]:
            ties += 1
        if battle[0] < battle[1]:
            victories_player2 += 1
    if victories_player1 > victories_player2:
        winner = "winner is player_1"
    elif victories_player1 == victories_player2:
        winner = "game is a tie"
    else:
        winner = "winner is player2"
    return """
    player1 wins {victories_player1}
    player2 wins {victories_player2}
    {ties} ties
    The {winner}
    """.format(player_1 = player_1, victories_player1 = victories_player1, pla
yer_2 = player_2, victories_player2 = victories_player2, ties = ties, winner =
winner)

print(play_game(gandalf, saruman))
```

```
    player1 wins 6
    player2 wins 4
    0 ties
    The winner is player_1
```

Expected output:

```
6
4
```

In [17]:

```
# We check who has won, do not forget the possibility of a draw.
# Print the result based on the winner.
```

Expected output:

```
Gandalf wins
```

Goals

1. Treatment of lists
2. Use of **for loop**
3. Use of conditional **if-elif-else**
4. Use of the functions **range(), len()**
5. print()
6. zip()

Bonus

1. Spells now have a name and there is a dictionary that relates that name to a power.
2. A sorcerer wins if he succeeds in winning 3 spell clashes in a row.
3. Average of each of the spell lists.
4. Standard deviation of each of the spell lists.

```
POWER = {
    'Fireball': 50,
    'Lightning bolt': 40,
    'Magic arrow': 10,
    'Black Tentacles': 25,
    'Contagion': 45
}

gandalf = ['Fireball', 'Lightning bolt', 'Lightning bolt', 'Magic arrow
', 'Fireball',
           'Magic arrow', 'Lightning bolt', 'Fireball', 'Fireball', 'Fi
reball']
saruman = ['Contagion', 'Contagion', 'Black Tentacles', 'Fireball', 'Bl
ack Tentacles',
           'Lightning bolt', 'Magic arrow', 'Contagion', 'Magic arrow',
'Magic arrow']
```

Good luck!

In [44]:

```
# 1. Spells now have a name and there is a dictionary that relates that name t
o a power.
# variables

power = {
    'Fireball': 50,
    'Lightning bolt': 40,
    'Magic arrow': 10,
    'Black Tentacles': 25,
    'Contagion': 45
}

gandalf = ['Fireball', 'Lightning bolt', 'Lightning bolt', 'Magic arrow', 'Fir
eball',
           'Magic arrow', 'Lightning bolt', 'Fireball', 'Magic arrow', 'Fireba
ll']
saruman = ['Contagion', 'Contagion', 'Black Tentacles', 'Fireball', 'Black Ten
tacles',
           'Lightning bolt', 'Magic arrow', 'Contagion', 'Magic arrow', 'Magic
arrow']
```

In [47]:

```
# Assign spell power lists to variables
gandalf_powers = []
for x in gandalf:
    gandalf_powers.append(power[x])
saruman_powers = []
for x in saruman:
    saruman_powers.append(power[x])

print(gandalf_powers,saruman_powers)
```

```
[50, 40, 40, 10, 50, 10, 40, 50, 10, 50] [45, 45, 25, 50, 25, 40,
10, 45, 10, 10]
```

Expected output:

```
([50, 40, 40, 10, 50, 10, 40, 50, 10, 50],
 [45, 45, 25, 50, 25, 40, 10, 45, 10, 10])
```

In [50]:

```
# 2. A sorcerer wins if he succeeds in winning 3 spell clashes in a row.



# Execution of spell clashes

# check clashes

# check for 3 wins in a row

def play_game(player_1,player_2):
    player1_battles_in_a_row = []
    player2_battles_in_a_row = []
    battles = zip(player_1, player_2)
    for battle in battles:
        if battle[0] > battle[1]:
            player1_battles_in_a_row.append(1)
            player2_battles_in_a_row.append(0)
        if battle[0] == battle[1]:
            player1_battles_in_a_row.append(0)
            player2_battles_in_a_row.append(0)
        if battle[0] < battle[1]:
            player1_battles_in_a_row.append(0)
            player2_battles_in_a_row.append(1)
    winner = ""
    while winner == "":
        for x in range(len(player1_battles_in_a_row)):
            if (sum(player1_battles_in_a_row[:x]) == 3) and (sum(player1_battl
es_in_a_row[:x]) > sum(player2_battles_in_a_row[:x])):
                winner = "player1"
            if (sum(player2_battles_in_a_row[:x]) == 3) and (sum(player2_battl
es_in_a_row[:x]) > sum(player1_battles_in_a_row[:x])):
                winner = "player2"
    return "winner is {winner}".format(winner = winner)


play_game(gandalf_powers,saruman_powers)


# check the winner

# count sum of clashes
```

Out[50]:

```
'winner is player1'
```

Expected output:

```
   Gandalf wins
```

In [52]:

```
# 3. Average of each of the spell lists.

gandalf_mean = sum(gandalf_powers) / len(gandalf_powers)

print(gandalf_mean)
```

```
35.0
```

Expected output:

```
35.0
30.5
```

In [9]:

```
# 4. Standard deviation of each of the spell lists.
```

Expected output:

```
16.881943016134134
15.56438241627338
```