In [1]:
```
import pandas as pd
import numpy as np
import math
import random
import matplotlib.pyplot as plt
print("Done Importing")
```

```
Done Importing
```

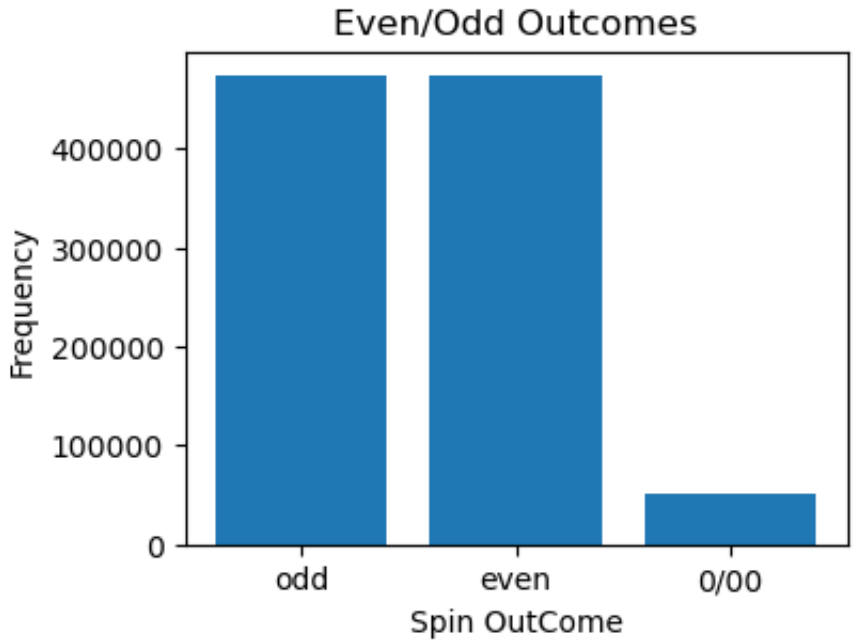
In [2]:
```
numOptions = [-1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
blacks = [2,4,6,8,10,11,13,15,17,19,20,22,24,29,29,31,33,35]
reds =   [1,3,5,7,9,12,14,16,18,21,23,25,27,28,30,32,34,36]
first18 = list(range(1,19))
second18 = list(range(19,37))

first12 = list(range(1,13))
second12 = list(range(13,25))
third12 = list(range(25,37))

col1 = [1,4,7,10,13,16,19,22,25,28,31,34]
col2 = [2,5,8,11,14,17,20,23,26,29,32,35]
col3 = [3,6,9,12,15,18,21,24,27,30,33,36]
rows = [[1,2,3],
        [4,5,6],
        [7,8,9],
        [10,11,12],
        [13,14,15],
        [16,17,18],
        [19,20,21],
        [22,23,24],
        [25,26,27],
        [28,29,30],
        [31,32,33],
        [34,35,36]]
```

In [5]:
```
# Lets simulate a multiple Rolls
sDict = {}
evenOddDict = {"odd":0,"even":0,"0/00":0}
redBlackDict = {"red":0,"black":0,"0/00":0}
less18more18 = {"1-18":0, "19-36":0,"0/00":0}
count12 = {"1-12":0, "13-24":0,"25-36":0,"0/00":0}
colsDict = {"col1":0,"col2":0,"col3":0, "0/00":0}

for i in range(1000000):
    random_element = random.choice(numOptions)

#Check for even or odd
    if random_element > 0:
        if random_element % 2 == 0:
            evenOddDict["even"] += 1
        else:
```

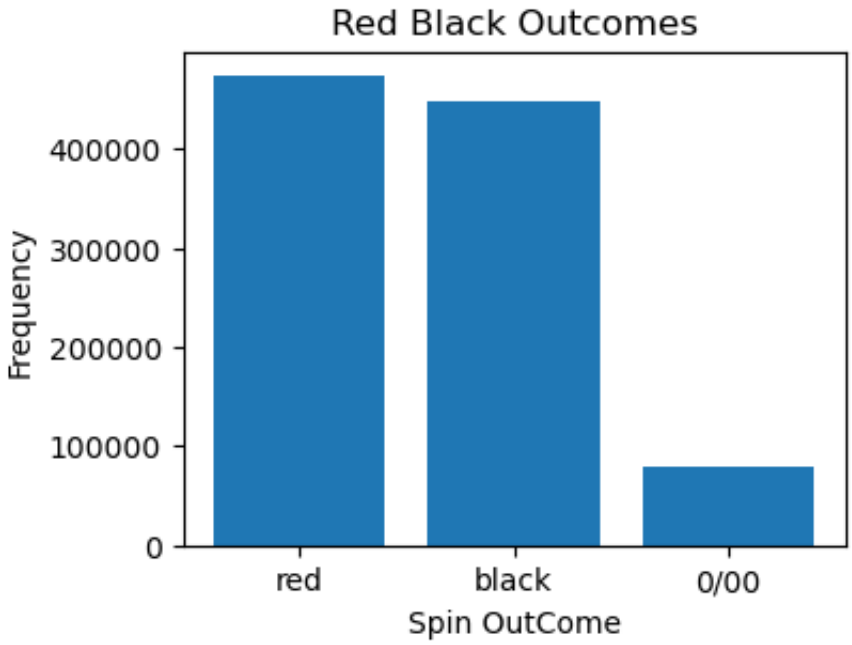
```
            evenOddDict["odd"] += 1
    else:
        evenOddDict["0/00"] += 1


    if random_element in blacks:
        redBlackDict["black"] += 1
    elif random_element in reds:
        redBlackDict["red"] += 1
    else:
        redBlackDict["0/00"] += 1

    if random_element in first18:
        less18more18["1-18"] +=1
    elif random_element in second18:
        less18more18["19-36"] +=1
    else:
        less18more18["0/00"] += 1


    if random_element in first12:
        count12["1-12"] +=1
    elif random_element in second12:
        count12["13-24"] +=1
    elif random_element in third12:
        count12["25-36"] +=1
    else:
        count12["0/00"] += 1


    if random_element in col1:
        colsDict["col1"] +=1
    elif random_element in col2:
        colsDict["col2"] +=1
    elif random_element in col3:
        colsDict["col3"] +=1
    else:
        colsDict["0/00"] += 1



    if random_element in sDict:
        sDict[random_element] = sDict[random_element] + 1
    else:
        sDict[random_element] = 1


print(evenOddDict)
plt.figure(figsize=(4, 3))
plt.bar(evenOddDict.keys(), evenOddDict.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('Even/Odd Outcomes')
```

```
# for i, value in enumerate(evenOddDict.values()):
#     plt.text(i, value + 0.5, str(value), ha='center', va='bottom')

plt.show()

print(redBlackDict)
plt.figure(figsize=(4, 3))
plt.bar(redBlackDict.keys(), redBlackDict.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('Red Black Outcomes')
plt.show()

print(less18more18)
plt.figure(figsize=(4, 3))
plt.bar(less18more18.keys(), less18more18.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('< 18 / > 18')
plt.show()

print(count12)
plt.figure(figsize=(4, 3))
plt.bar(count12.keys(), count12.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('12 Segments')
plt.show()
```

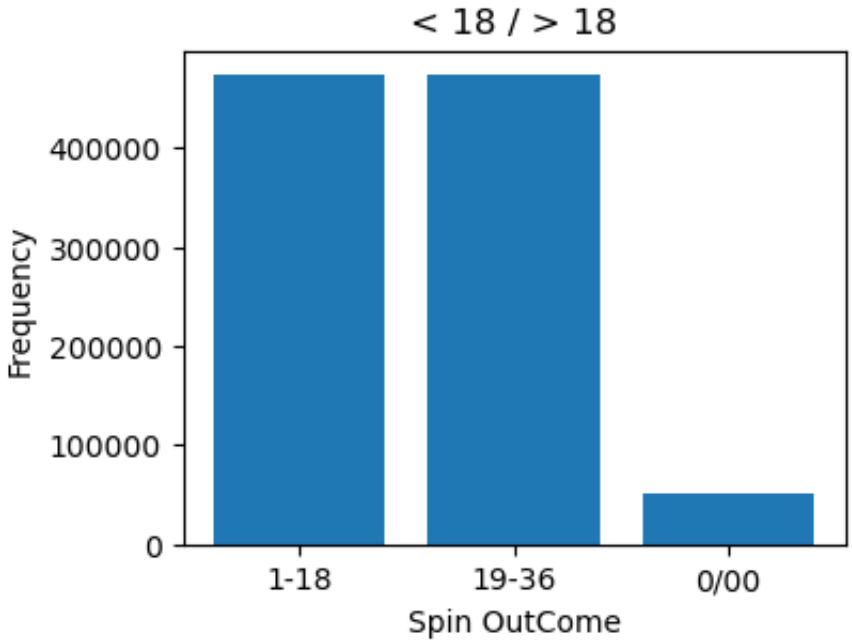
```
{'odd': 474432, 'even': 473409, '0/00': 52159}
```



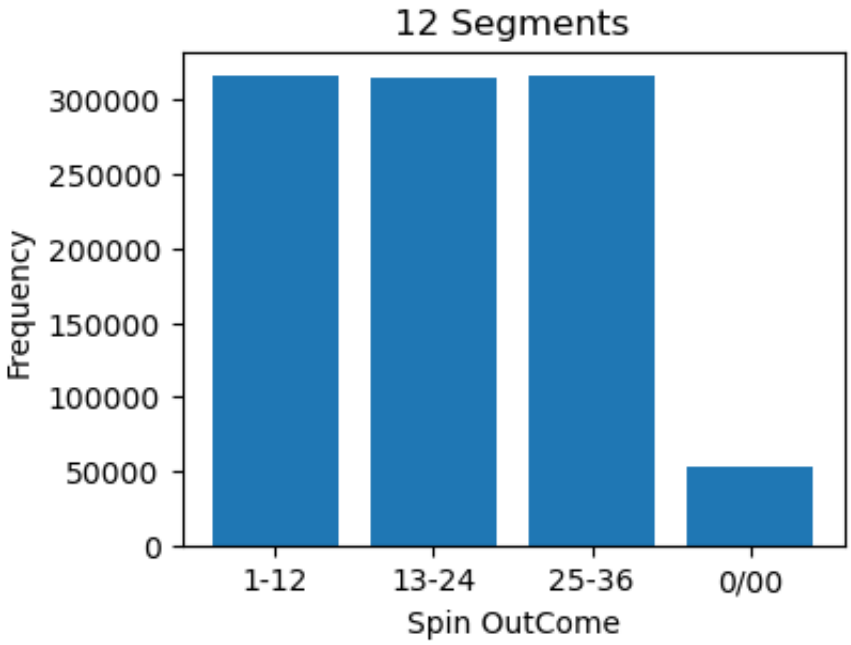

```
{'red': 474137, 'black': 447580, '0/00': 78283}
```




```
{'1-18': 473083, '19-36': 474758, '0/00': 52159}
```

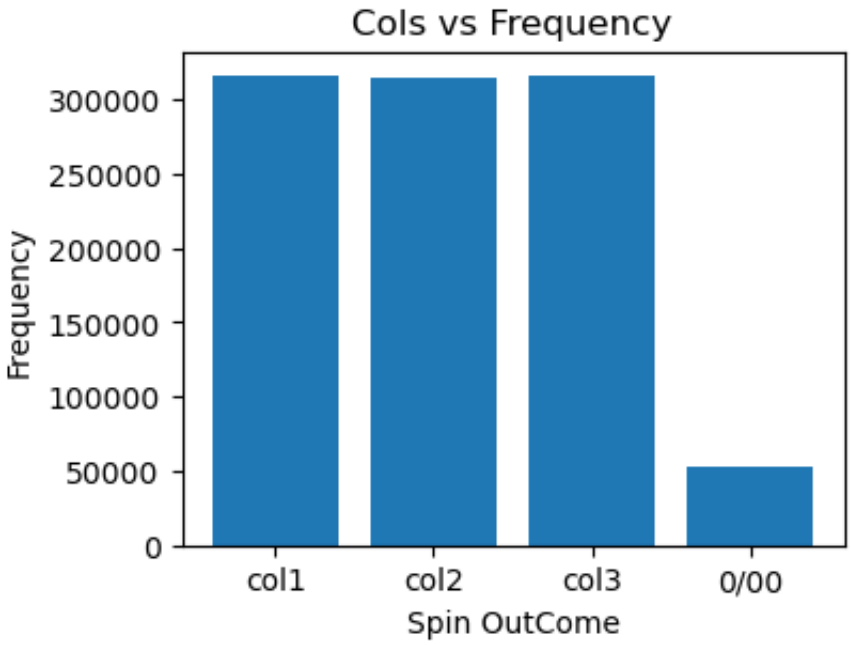



```
{'1-12': 315703, '13-24': 315365, '25-36': 316773, '0/00': 52159}
```

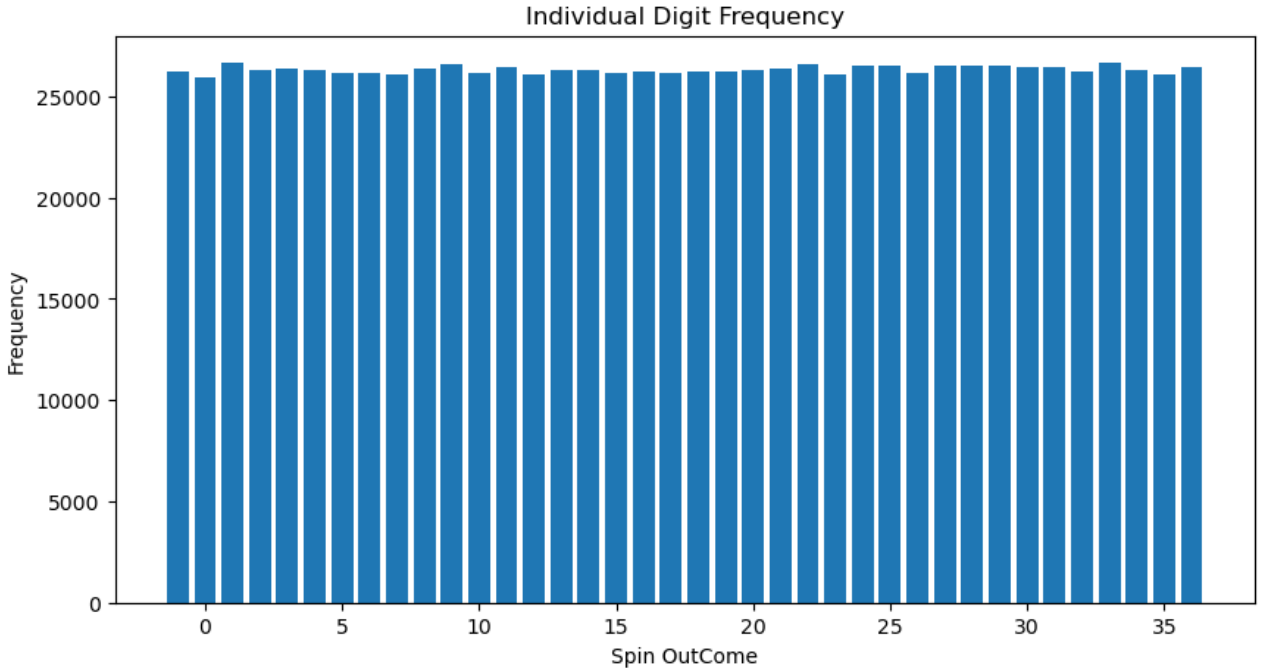


In [6]:
```
print(colsDict)
plt.figure(figsize=(4, 3))
plt.bar(colsDict.keys(), colsDict.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('Cols vs Frequency')
plt.show()

print(sDict)
plt.figure(figsize=(10, 5))
plt.bar(sDict.keys(), sDict.values())
plt.xlabel('Spin OutCome')
plt.ylabel('Frequency')
plt.title('Individual Digit Frequency')
plt.show()
```

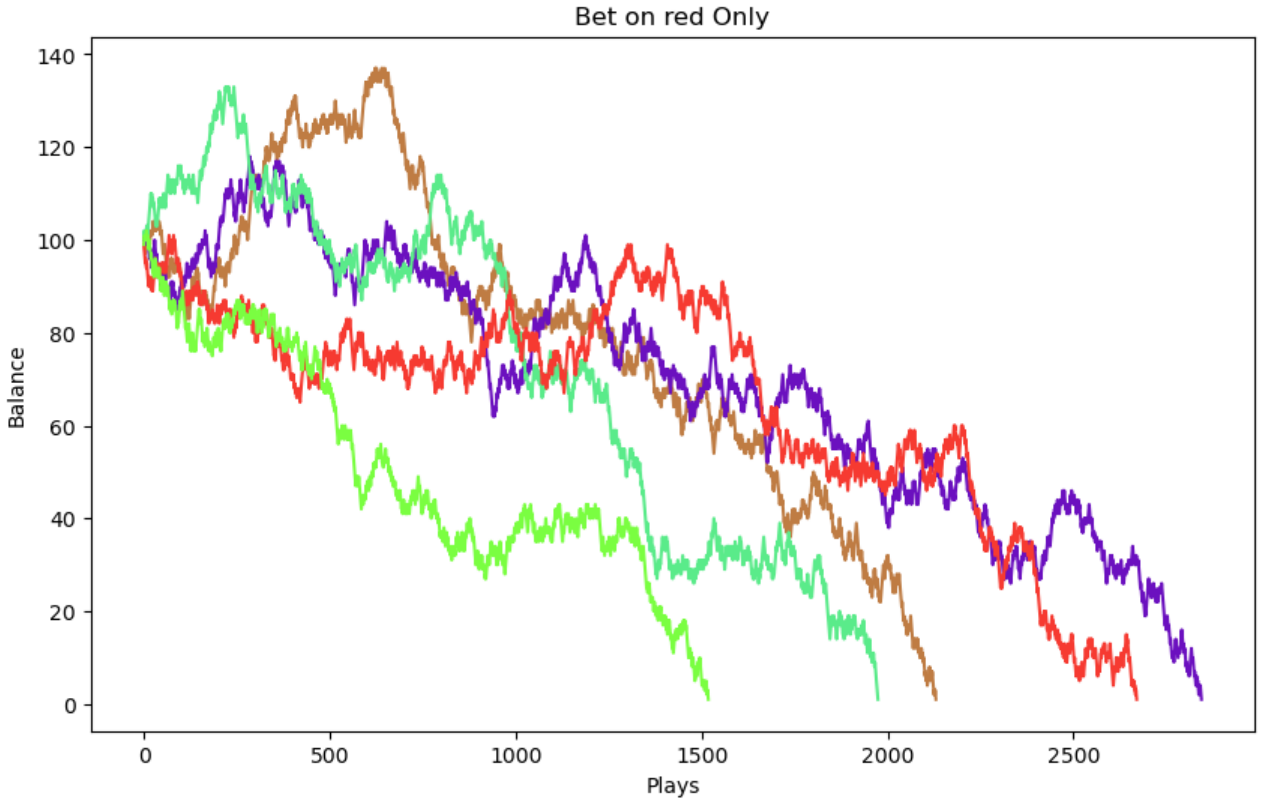
```
{'col1': 316220, 'col2': 315082, 'col3': 316539, '0/00': 52159}
```

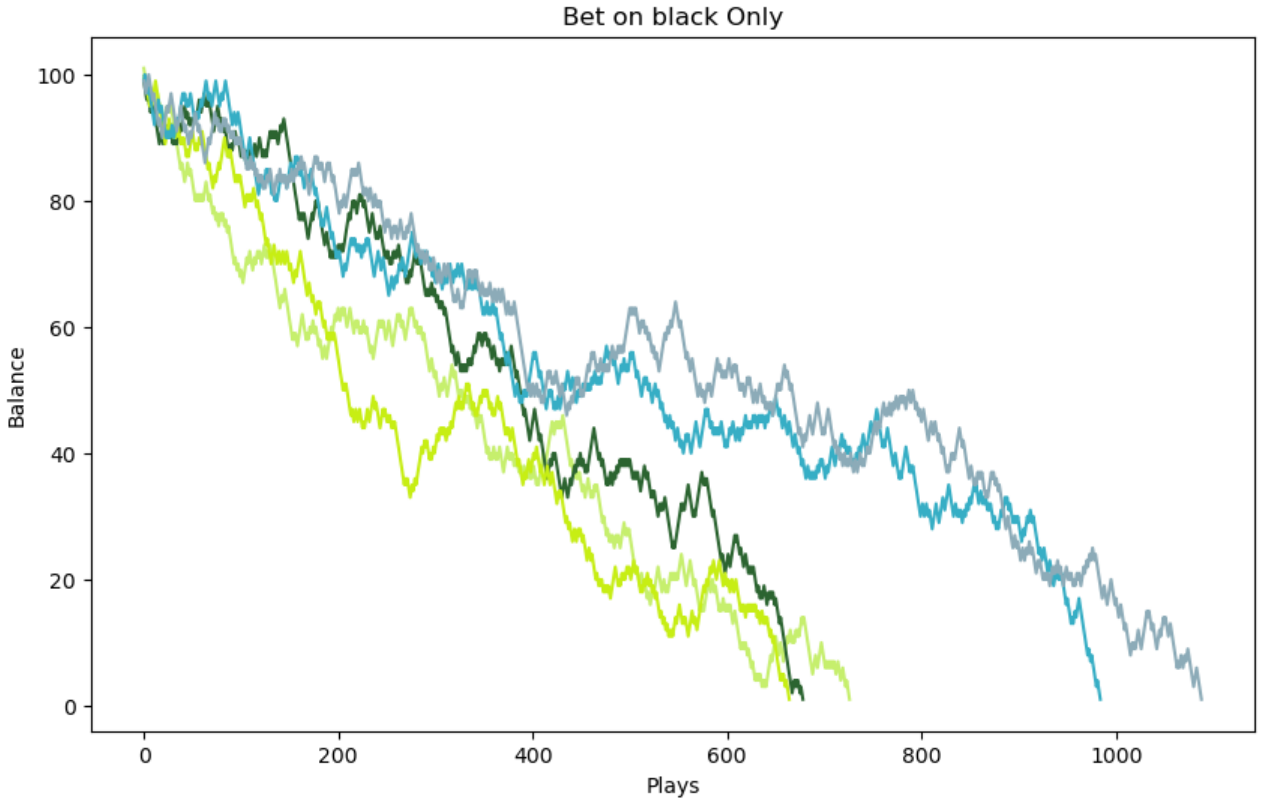


{12: 26066, 32: 26237, 9: 26584, 18: 26222, 27: 26549, 21: 26368, 15: 26175, -1: 26242, 2: 26271, 1: 26656, 5: 26167, 33: 26644, 30: 26450, 22: 26554, 23: 26091, 0: 25917, 19: 26205, 10: 26171, 11: 26468, 6: 26163, 3: 26405, 24: 26486, 36: 26427, 4: 26287, 25: 26516, 17: 26181, 35: 26110, 34: 26313, 28: 26501, 14: 26288, 13: 26314, 20: 26281, 31: 26406, 7: 26097, 16: 26200, 29: 26496, 8: 26368, 26: 26124}

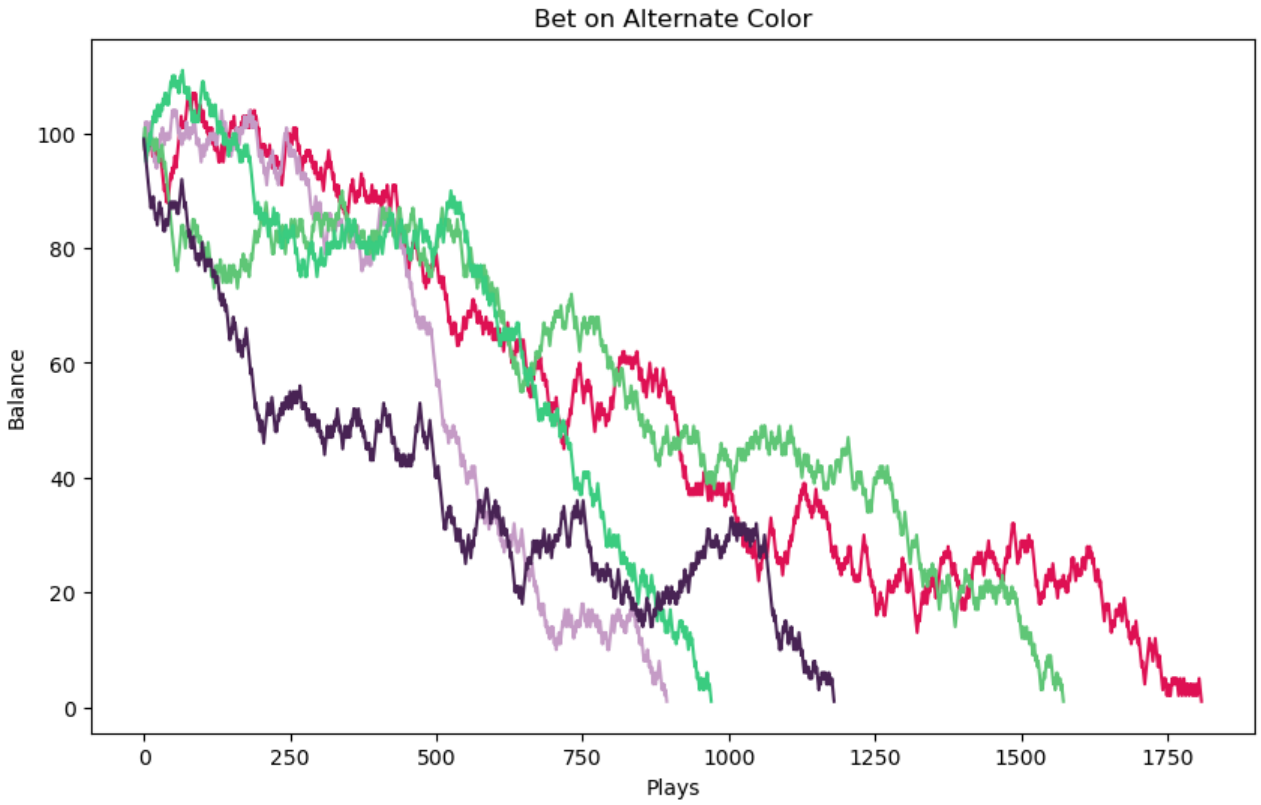


In [7]:
```
def random_color():
    return f'#{random.randint(0, 0xFFFFFF):06x}'

def simulateRedOrBlackOnly(color):
    options = []
    if color == 'red':
        options = reds
    else:
        options = blacks

    startingBalance = 100
    bet_amount = 1
    game_plays_amount = []
    game_play_count = 0
    while ((startingBalance > 1) & (game_play_count < 100000)):
        random_element = random.choice(numOptions)
        if random_element in options:
            startingBalance += bet_amount

        else:
            startingBalance -= bet_amount

        game_plays_amount.append(startingBalance)
        game_play_count += 1

    plt.plot(game_plays_amount, linestyle='-', color=random_color() , label=

    # Add labels and title
    plt.xlabel('Plays')
    plt.ylabel('Balance')
    plt.title(f'Bet on {color} Only')


plt.figure(figsize=(10, 6))
for i in range(5):
    simulateRedOrBlackOnly("red")

plt.figure(figsize=(10, 6))
for i in range(5):
    simulateRedOrBlackOnly("black")
```


Bet on red Only
Balance
Plays
0
20
40
60
80
100
120
140
0
500
1000
1500
2000
2500


Bet on black Only
Balance
Plays
0
20
40
60
80
100
0
200
400
600
800
1000

In [8]:
```
def simulateRedOrBlackAlternate():
    startingBalance = 100
    options = []
    bet_amount = 1
    game_plays_amount = []
    game_play_count = 0
    while ((startingBalance > 1) & (game_play_count < 100000)):
        random_element = random.choice(numOptions)
        if game_play_count % 2 == 0:
            options = reds
        else:
            options = blacks

        if random_element in options:
            startingBalance += bet_amount

        else:
            startingBalance -= bet_amount

        game_plays_amount.append(startingBalance)
        game_play_count += 1

    plt.plot(game_plays_amount, linestyle='-', color=random_color())

    # Add labels and title
    plt.xlabel('Plays')
    plt.ylabel('Balance')
    plt.title(f'Bet on Alternate Color')

plt.figure(figsize=(10, 6))
for i in range(5):
    simulateRedOrBlackAlternate()
```



In [135…
```
# Only bet on Even / Odd would yield similar outcomes as above
# Only bet on First 18 / last 18 would yield similar outcome as above
```

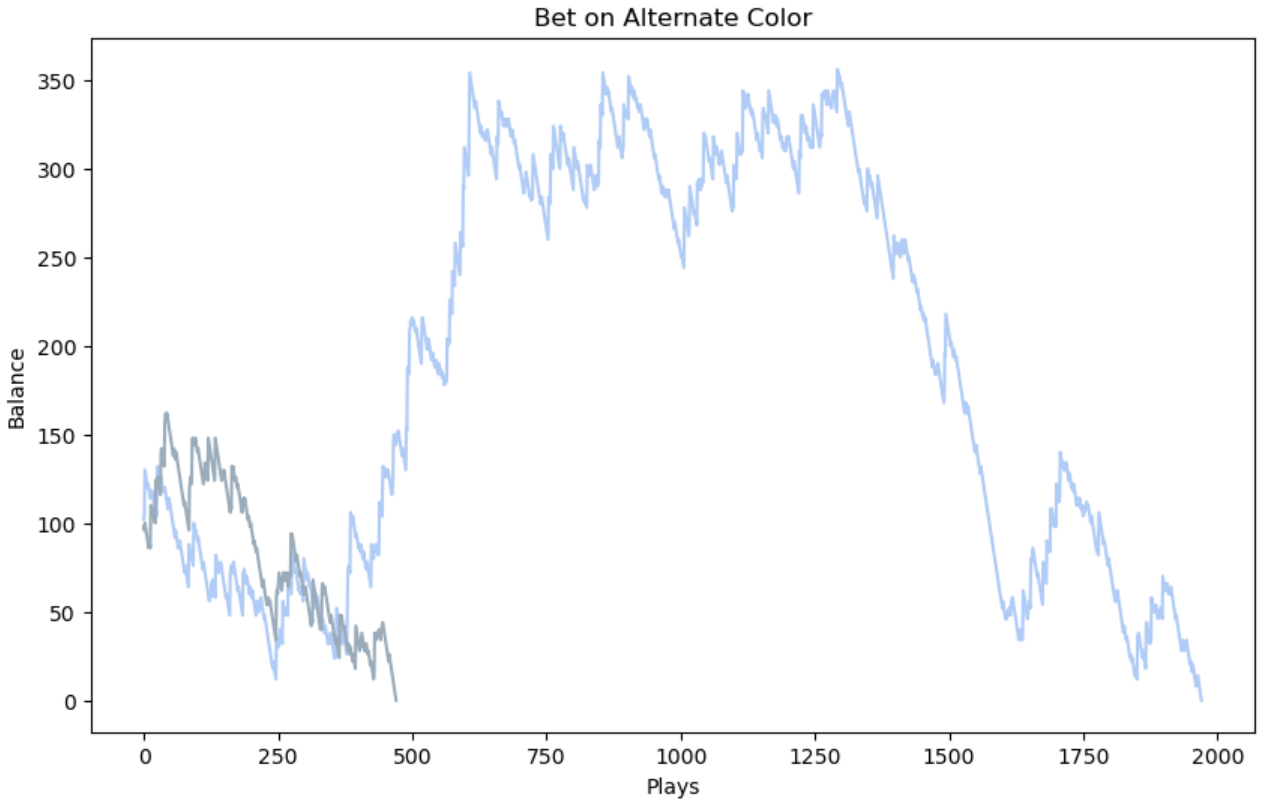
In [ ]:
```
```

In [19]:
```
# Simulate a game where the player puts bets on random places as seen someti
gamePlayOptions = ["single_num",
                   "odd","even",
                   "red","black",
                   "row1","row2","row3","row4","row5","row6","row7","row8",
                   "col1","col2","col3",
                   "first12","second12","third12"]
def simulateRandomPlays(numberOfBetsPerPlay):

    startingBalance = 100
    bet_amount = 1
    balance_history = []
    game_play_count = 0

    while ((startingBalance > 1) & (game_play_count < 200000)):
        random_element = random.choice(numOptions)
        game_plays = []
        for i in range(numberOfBetsPerPlay):
            game_play = random.choice(gamePlayOptions)
            game_plays.append(game_play)
```

```
        for gPlay in game_plays:
            if game_play == "single_num":
                bet_num = random.choice(numOptions)
                if bet_num == random_element:
                    startingBalance += (bet_amount * 35)
                else:
                    startingBalance -= bet_amount
            elif game_play == "even":
                if random_element % 2 == 0:
                    startingBalance += bet_amount
                else:
                    startingBalance -= bet_amount
            elif game_play == "odd":
                if random_element % 2 == 1:
                    startingBalance += bet_amount
                else:
                    startingBalance -= bet_amount
            elif game_play == "red":
                if random_element in reds:
                    startingBalance += bet_amount
                else:
                    startingBalance -= bet_amount
            elif game_play == "black":
                if random_element in blacks:
                    startingBalance += bet_amount
                else:
                    startingBalance -= bet_amount
            elif game_play == "row1":
                if random_element in rows[0]:
                    startingBalance += (bet_amount * 11)
                else:
                    startingBalance -= bet_amount
            elif game_play == "row2":
                if random_element in rows[1]:
                    startingBalance += (bet_amount * 11)
                else:
                    startingBalance -= bet_amount
            elif game_play == "row3":
                if random_element in rows[2]:
                    startingBalance += (bet_amount * 11)
                else:
                    startingBalance -= bet_amount
            elif game_play == "row4":
                if random_element in rows[3]:
                    startingBalance += (bet_amount * 11)
                else:
                    startingBalance -= bet_amount
            elif game_play == "row5":
                if random_element in rows[4]:
                    startingBalance += (bet_amount * 11)
                else:
                    startingBalance -= bet_amount
            elif game_play == "row6":
```

```
            if random_element in rows[5]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row7":
            if random_element in rows[6]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row8":
            if random_element in rows[7]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row9":
            if random_element in rows[8]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row10":
            if random_element in rows[9]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row11":
            if random_element in rows[10]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "row12":
            if random_element in rows[11]:
                startingBalance += (bet_amount * 11)
            else:
                startingBalance -= bet_amount
        elif game_play == "col1":
            if random_element in col1:
                startingBalance += (bet_amount * 2)
            else:
                startingBalance -= bet_amount
        elif game_play == "col2":
            if random_element in col2:
                startingBalance += (bet_amount * 2)
            else:
                startingBalance -= bet_amount
        elif game_play == "col3":
            if random_element in col3:
                startingBalance += (bet_amount * 2)
            else:
                startingBalance -= bet_amount

        elif game_play == "first12":
            if random_element in first12:
                startingBalance += (bet_amount * 2)
```

```
                else:
                    startingBalance -= bet_amount
            elif game_play == "second12":
                if random_element in second12:
                    startingBalance += (bet_amount * 2)
                else:
                    startingBalance -= bet_amount
            elif game_play == "third12":
                if random_element in third12:
                    startingBalance += (bet_amount * 2)
                else:
                    startingBalance -= bet_amount

        balance_history.append(startingBalance)
        game_play_count += 1

    plt.plot(balance_history, linestyle='-', color=random_color())
    plt.xlabel('Plays')
    plt.ylabel('Balance')
    plt.title(f'Bet on Alternate Color')



plt.figure(figsize=(10, 6))
for i in range(2):
    simulateRandomPlays(2)
```



In [125…

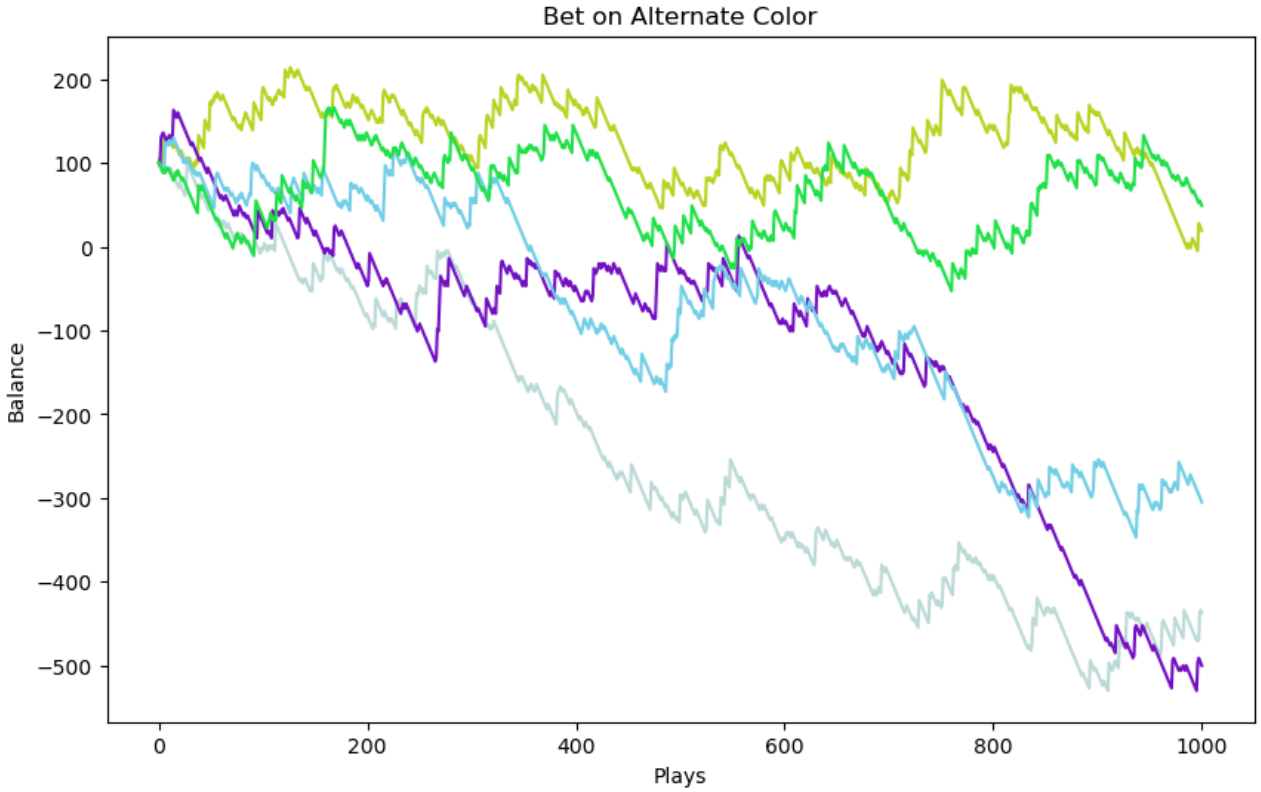
```
gamePlayOptionsNoSingleDigits = [
                    "odd","even",
                    "red","black",
                    "row1","row2","row3","row4","row5","row6","row7","row8",
                    "col1","col2","col3",
                    "first12","second12","third12"]

def simulateHouseVsMultiPlayer(numOfPlayers,numOfBetsPerPlayer, numOfGames):
    houseBalance = 0
    players = []
    bet_amount = 1
    for i in range(numOfPlayers):
        playerDict = {"playerName":f"Player-{i}",
                      "playerBalance":100,
                       "playerBalanceHistory":[100]
                      }
        players.append(playerDict)

    for game in range(numOfGames):
        random_element = random.choice(numOptions)

        for player in players:
            game_plays = []
            for i in range(numOfBetsPerPlayer):
                game_play = random.choice(gamePlayOptionsNoSingleDigits)
                game_plays.append(game_play)


            for gPlay in game_plays:
                if game_play == "single_num":
                    bet_num = random.choice(numOptions)
                    if bet_num == random_element:
                        player["playerBalance"] += (bet_amount * 35)
                    else:
                        player["playerBalance"] -= bet_amount

                elif game_play == "even":
                    if random_element % 2 == 0:
                        player["playerBalance"] += bet_amount
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "odd":
                    if random_element % 2 == 1:
                        player["playerBalance"] += bet_amount
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "red":
                    if random_element in reds:
                        player["playerBalance"] += bet_amount
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "black":
                    if random_element in blacks:
```

```
                player["playerBalance"] += bet_amount
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row1":
            if random_element in rows[0]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row2":
            if random_element in rows[1]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row3":
            if random_element in rows[2]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row4":
            if random_element in rows[3]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row5":
            if random_element in rows[4]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row6":
            if random_element in rows[5]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row7":
            if random_element in rows[6]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row8":
            if random_element in rows[7]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row9":
            if random_element in rows[8]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
        elif game_play == "row10":
            if random_element in rows[9]:
                player["playerBalance"] += (bet_amount * 11)
            else:
                player["playerBalance"] -= bet_amount
```

```
                elif game_play == "row11":
                    if random_element in rows[10]:
                        player["playerBalance"] += (bet_amount * 11)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "row12":
                    if random_element in rows[11]:
                        player["playerBalance"] += (bet_amount * 11)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "col1":
                    if random_element in col1:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "col2":
                    if random_element in col2:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "col3":
                    if random_element in col3:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount

                elif game_play == "first12":
                    if random_element in first12:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "second12":
                    if random_element in second12:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount
                elif game_play == "third12":
                    if random_element in third12:
                        player["playerBalance"] += (bet_amount * 2)
                    else:
                        player["playerBalance"] -= bet_amount

            player["playerBalanceHistory"].append(player["playerBalance"])

#     print(players)
    for player in players:
        print(player["playerName"],player["playerBalance"])
        plt.plot(player["playerBalanceHistory"], linestyle='-', color=random
        plt.xlabel('Plays')
        plt.ylabel('Balance')
        plt.title(f'Bet on Alternate Color')
```

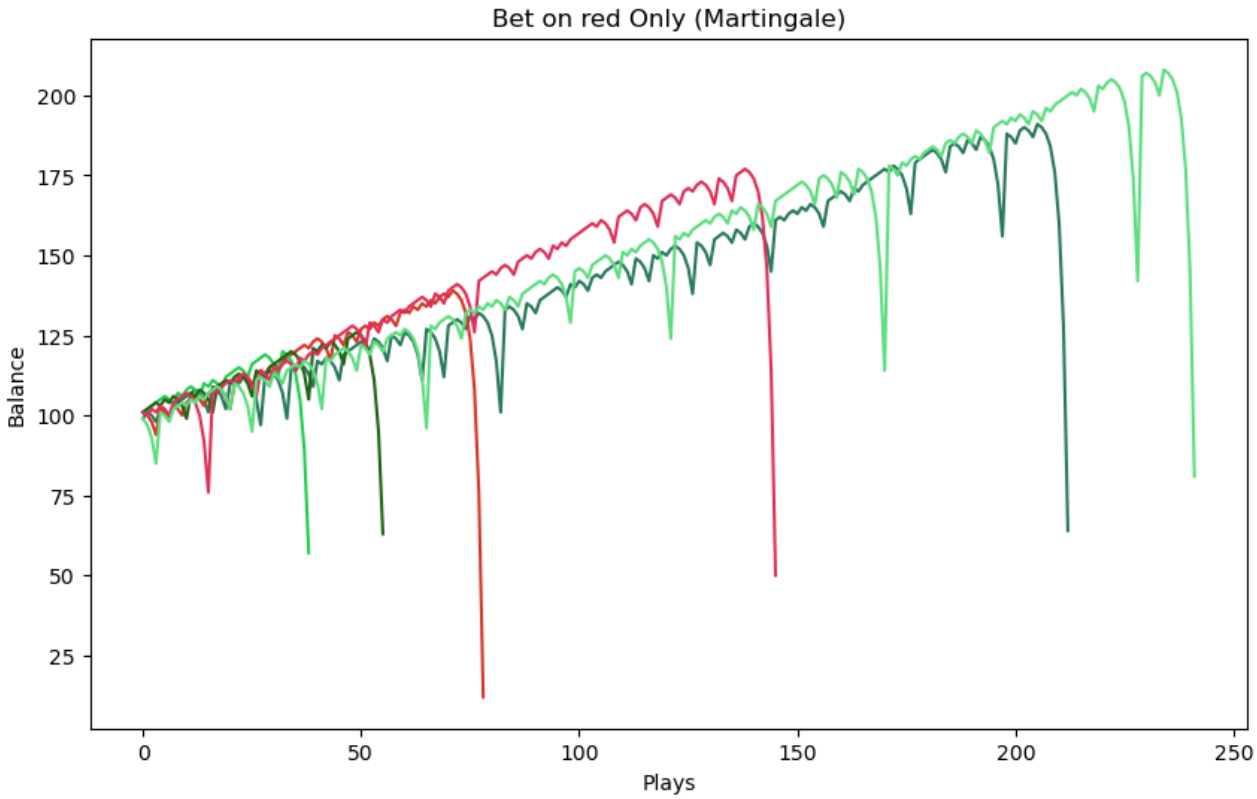
```
#def simulateHouseVsMultiPlayer(numOfPlayers,numOfBetsPerPlayer, numOfGames)
plt.figure(figsize=(10, 6))
simulateHouseVsMultiPlayer(5,3,1000)
```

```
Player-0 19
Player-1 -437
Player-2 -500
Player-3 -305
Player-4 49
```



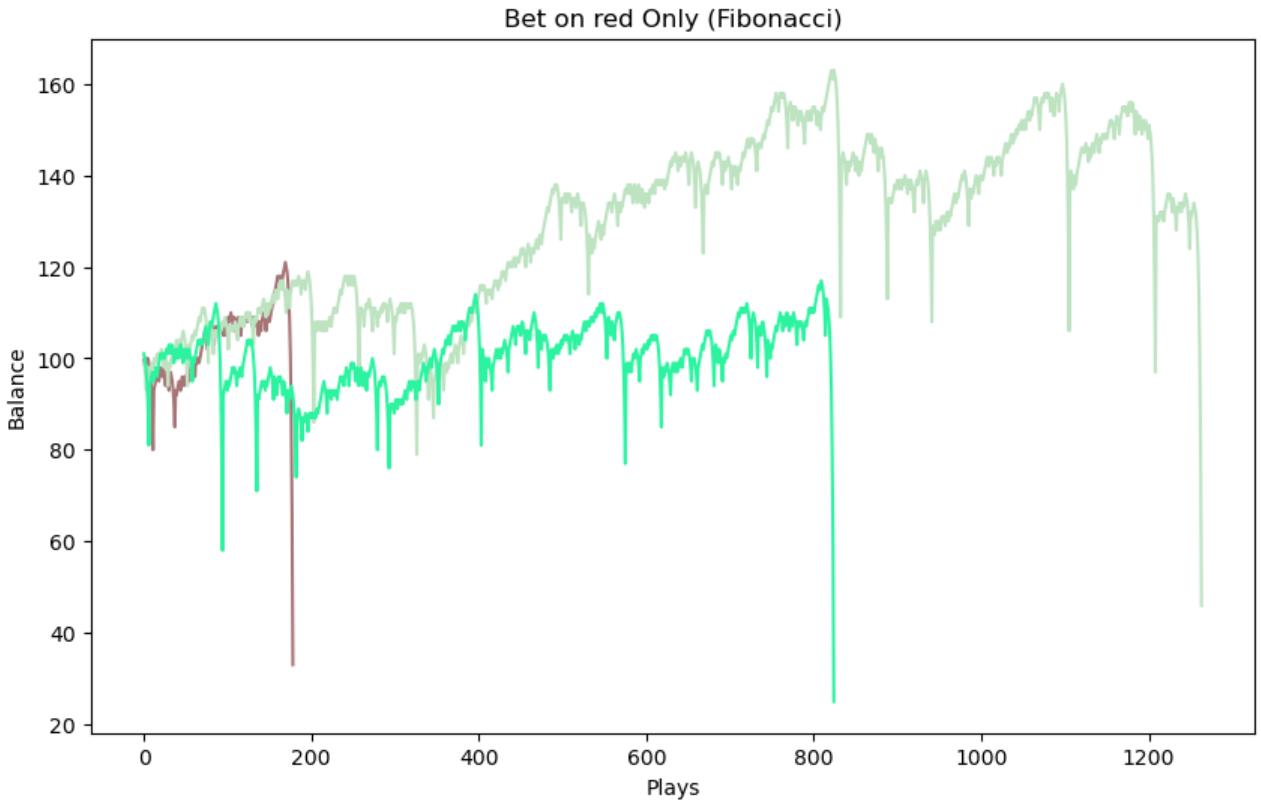
In [124…

```
def simulateRedOrBlackOnlyDoubleBetsIfLost(color):
    options = []
    if color == 'red':
        options = reds
    else:
        options = blacks

    startingBalance = 100
    bet_amount = 1
    balance_history = []
    game_play_count = 0
    while ((startingBalance > bet_amount) & (game_play_count < 100000)):
        random_element = random.choice(numOptions)
#         print(f"bet_amount = {bet_amount} startingBalance = {startingBalan
        if random_element in options:
            startingBalance += bet_amount
            bet_amount = 1

        else:
            startingBalance -= bet_amount
            bet_amount = bet_amount * 2

        balance_history.append(startingBalance)
        game_play_count += 1

    plt.plot(balance_history, linestyle='-', color=random_color() , label='D

    # Add labels and title
    plt.xlabel('Plays')
    plt.ylabel('Balance')
    plt.title(f'Bet on {color} Only (Martingale)')


plt.figure(figsize=(10, 6))
for i in range(6):
    simulateRedOrBlackOnlyDoubleBetsIfLost("red")
```


Bet on red Only (Martingale)
Balance
Plays
200
175
150
125
100
75
50
25
0
50
100
150
200
250

In [122…

```
def simulateBetsInFibonacciSeries(color):
    options = []
    if color == 'red':
        options = reds
    else:
        options = blacks

    startingBalance = 100
    old_bet = 1
    new_bet = 1
    balance_history = []
    game_play_count = 0

    while ((startingBalance > new_bet) & (game_play_count < 100000)):
        random_element = random.choice(numOptions)
        if random_element in options:
            startingBalance += old_bet
            old_bet = 1
            new_bet = 1
        else:
            startingBalance -= old_bet
            temp = new_bet
            new_bet = temp + old_bet
            old_bet = temp


        balance_history.append(startingBalance)
        game_play_count += 1

    plt.plot(balance_history, linestyle='-', color=random_color() , label='D

    # Add labels and title
    plt.xlabel('Plays')
    plt.ylabel('Balance')
    plt.title(f'Bet on {color} Only (Fibonacci)')

plt.figure(figsize=(10, 6))
for i in range(3):
    simulateBetsInFibonacciSeries("red")
```


Bet on red Only (Fibonacci)
Balance
Plays
160
140
120
100
80
60
40
20
0
200
400
600
800
1000
1200

In [ ]: