In [ ]:

```
import random

def main():
    stoneList = []
    pile = random.randint(2, 5)
    stone = random.randint(1, 9)

    players = 1

    Nimboard(stoneList, pile, stone, players) # Set initial board
    play_again(stoneList, pile, stone, players)

#declare board
def Nimboard(stoneList, pile, stone, players):

    # Get initial board
    print("Let's look at the board now.")
    print("-" * 25)
    for i in range(0, pile):
        stone = random.randint(1, 8)
        print('Pile {}: {}'.format(i + 1, stone))
        stoneList.append(stone)
    print("-" * 25)

    nim_sum(stoneList, pile)

#set the validation
def valid_move(stoneList, pile, players):

    # Begin loop that tests for valid input - if valid, break loop - if not, keep ask
ing
    while True:
        stones = input('Player{}, how many stones to remove? '.format(players))
        piles = input('Pick a pile to remove from: ')

        # correct condition
        if (stones and piles) and (stones.isdigit()) and (piles.isdigit()):
            if (int(stones) > 0) and (int(piles) <= len(stoneList)) and (int(piles) >
0):
                if (int(stones) <= stoneList[int(piles) - 1]) and (int(stones) <= (su
m(stoneList)-1)):
                    if (int(stones) != 0) and (int(piles) != 0):
                        break

        # if its not met the condition
        print("illegal move. Try again, {}.".format(players))

    # Update state
    stoneList[int(piles) - 1] -= int(stones)

    # Keep playing game
    continue_game(stoneList, pile, players)

def continue_game(stoneList, pile, players):
    print("Let's look at the board now.")
    print("-" * 25)
    for i in range(0, pile):
        print("Pile {}: {}".format(i + 1, stoneList[i]))
    print("-" * 25)

    # In the case when game is over, do not display computer hint for empty board
    if stoneList != [0] * len(stoneList):
```

```
        nim_sum(stoneList, pile)

def play_again(stoneList, pile, stone,players):

    # Begin loop to initiate player switching
    while True:
        print(stoneList)
        print(sum(stoneList))
        valid_move(stoneList, pile, players)

        # To determine winner, check if rockList contains all 0's on that player's tu
rn
        if sum(stoneList) == 1:
            print("Player{} is the winner of this round!".format(players))
            print ("Game is over!")
            user = input("Do you want to play again? Enter y for yes, anything for n
o: ")

            if user.lower() == 'y':
                # reset all conditions, start the game again
                stoneList = []
                pile = random.randint(2, 5)
                player = 1
                Nimboard(stoneList, pile, stone, players)
                valid_move(stoneList, pile, players)

            else:
                break

        # switch players 2->1, 1->2
        if players == 1:
            players = 2

        else:
            players = 1

def nim_sum(stoneList, pile):
    nim = 0

    # Calculate nim sum for all elements in the rockList
    for i in stoneList:
        nim = nim ^ i

    print("Hint: nim sum is {}.".format(nim))

    # Determine how many rocks to remove from which pile
    stones_to_remove = max(stoneList) - nim
    stones_to_remove = abs(stones_to_remove)

    # Logic for certain configurations on determining how many stones to remove from
 which pile
    # "rockList.index(max(rockList))+ 1 )" determines the index in rockList at which
 the biggest
    # pile of stones exists.
    if (nim > 0) and (len(stoneList) > 2) and (nim != max(stoneList)) and (nim !=1):
        print("Pick {} stones from pile {}".format(stones_to_remove, stoneList.index(
max(stoneList))+ 1 ))

    if (nim > 0) and (len(stoneList) > 2) and (nim == max(stoneList)) and (nim !=1):
        print("Pick {} stones from pile {}.".format(nim, stoneList.index(max(stoneLis
t))+ 1 ))

    if nim > 0 and len(stoneList) <= 2 and (stones_to_remove != 0):
```

```
        print("Pick {} stones from pile {}".format(stones_to_remove, stoneList.index(
max(stoneList))+ 1 ))

    if nim > 0 and len(stoneList) <= 2 and (stones_to_remove == 0):
        print("Pick {} stones from pile {}".format(nim, stoneList.index(max(stoneList
))+ 1 ))

    elif (nim == 1) and (len(stoneList) <= 2):
        print("Pick {} stones from pile {}".format(nim, stoneList.index(max(stoneList
))+ 1 ))

    if (nim == 1) and (nim == max(stoneList)) and (nim != 0) and (len(stoneList) > 2
):
        print("Pick {} stones from pile {}".format(nim, stoneList.index(max(stoneList
))+ 1))

    if nim == 0:
        print("Pick all stones from pile {}.".format(stoneList.index(max(stoneList))+
1 ))
main()
```

```
Let's look at the board now.
-------------------------
Pile 1: 3
Pile 2: 1
-------------------------
Hint: nim sum is 2.
Pick 1 stones from pile 1
[3, 1]
4
Player1, how many stones to remove? 1
Pick a pile to remove from: 1
Let's look at the board now.
-------------------------
Pile 1: 2
Pile 2: 1
-------------------------
Hint: nim sum is 3.
Pick 1 stones from pile 1
[2, 1]
3
Player2, how many stones to remove? 1
Pick a pile to remove from: 1
Let's look at the board now.
-------------------------
Pile 1: 1
Pile 2: 1
-------------------------
Hint: nim sum is 0.
Pick all stones from pile 1.
[1, 1]
2
Player1, how many stones to remove? 1
Pick a pile to remove from: 1
Let's look at the board now.
-------------------------
Pile 1: 0
Pile 2: 1
-------------------------
Hint: nim sum is 1.
Pick 1 stones from pile 2
Player1 is the winner of this round!
Game is over!
```

In [ ]: