

FROG LEAP GAME

The objective of the puzzle is to switch the positions of all the colored frogs, so that all three green frogs are on the right and all three brown frogs are on the left. When all the frogs have reached the other side of the pond, you win!

Prepared by



Rajesh

<https://rajesh9943.github.io>



rajeshalagupandian@gmail.com



Frog Leap Game Documentation

Problem statement

Create famous 'Frog leap' puzzle game. Try completing the game before starting to get an idea about its working. [Enjoy Playing](#) 🎮 the Frog Leap game!

Rules

1. The left set of frogs can only move right, the right set of frogs can only move left.
2. Frogs can move forward one space, or move two spaces by jumping over another frog from opposite side.
3. The puzzle is solved when the two sets of frogs have switched positions.

Introduction

Welcome to the documentation for the Frog Leap puzzle game! This game is a simulation of the classic puzzle in which you must move frogs between two sets of positions until they switch places. The left set of frogs can only move right, and the right set can only move left. Frogs can move one space forward or leap over another frog to move two spaces.

Game Display

The game is displayed with green frogs ('🐸') on the left side, brown frogs ('🐸') on the right side, and empty spaces represented by leaves ('🍁'). The initial display is as follows:

Steps to solve the problem:

Step1:-

- Display green and brown frogs on the left and right sides initially.

Initial Display :-

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]  
['G', 'G', 'G', '-', 'B', 'B', 'B']
```

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]  
['🐸', '🐸', '🐸', '🍁', '🐸', '🐸', '🐸']
```

Here 'G' represents Green frogs on the left side and 'B' represents brown frogs on the right side. The '-' defines the position of empty leaf. (You can change display according to your imagination or convinience)

Step2:-

Accept positions of the frog that you want to move.

Example: If we enter position 2 then the game will look like this:-

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]  
['G', 'G', '-', 'G', 'B', 'B', 'B']
```

Step3:-

Define Invalid moves and add conditional 'if' statements accordingly

Rules

1. Entered position should be between 0 to 6. Or a character 'q' to quit the game.
2. Entered position cannot be the position of empty leaf.
3. If the selected frog position cannot perform the constraints given in rule 2 then the move is invalid.

Step4:-

Make the appropriate move by changing the game display.

Step 1

First create a list `positions` which contains the characters 'G','B' and '-' in the same sequence as given in the initial display state.

```
Positions = list(['G', 'G', 'G', '-', 'B', 'B', 'B'])
```

Now print this string `[0 , 1 , 2 , 3 , 4 , 5 , 6]` and after that print the list `positions`

```
print("##### Always enter an integer less than 7 #####")
```

```
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
```

```
print(Positions)
```

```
##### Always enter an integer less than 7 #####
```

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
```

```
['G', 'G', 'G', '-', 'B', 'B', 'B']
```

Take position as input

```
pos = input("Press q to quit else \nEnter position of piece:")
```

Now the taken input is in string format. So first check if the input is 'q' character. If input is 'q' then the person is quitting the game so print 'You Lose'.

```
if pos == 'q':  
    print("YOU LOSE!!")
```

Next if input character is not 'q' then it has to be some integer. so convert input to integer format.

```
pos = int(pos)
```

Step 2

Now we have to check validity of the selected positions or move.
If the entered number isn't between 0 and 6, then print 'Invalid move'.

```
if pos<0 or 6<pos:  
    print("Invalid move")
```

A frog should be present on the selected position to make a move. If leaf is selected then it doesn't make sense. Therefore, if entered position is same as the position of empty leaf then the move is invalid and print **Invalid Move**

```
if Positions[pos] == '-':  
  
    print("Invalid move")
```

Check if the selected frog is 'G':

(Inside if when it's 'G'. As 'G' is selected frog can move to right only.)

! condition 1

If ****selected_position + 1**** is less than or equal to 6 and ****curent_position + 1**** contains '-'

then it's a valid move and store that position in `pos2`.

! condition2

Else if `**selected_position + 2**` is less than or equal to 6 and if `**current_position + 2**` contains '-' and if `**selected_position + 1**` contains 'B' then it's a valid move and store that position in `pos2`.

! condition3:

Else remaining all are invalid, so print `Invalid Move`

```
if Positions[pos] == 'G':
    if (pos + 1) <= 6 and Positions[pos + 1] == '-':
        pass

    elif (pos + 2) <= 6 and Positions[pos + 2] == '-' and Positions[pos + 1] == 'B':
        pass
    else:
        print("Invalid move")
```

Check if the selected frog is 'B':

Check if the selected frog is 'B':

(Inside if when it's 'B'. As 'B' is selected frog can move to left only.)

! condition1:

If `**selected_position - 1**` is more than or equal to 0 and `**current_position - 1**` contains '-' then it's a valid move and store that position in `pos2`.

! condition2:

Else if `**selected_position - 2**` is more than or equal to 0 and if `**current_position - 2**` contains '-' and if `**selected_position - 1**` contains 'G' then it's a valid move and store that position in `pos2`.

! condition3:

Else remaining all are invalid,, so print 'Invalid Move'.

```
if Positions[pos] == 'B':
    if (pos - 1) >= 0 and Positions[pos - 1] == '-':
        pass
    elif (pos - 2) >= 0 and Positions[pos - 2] == '-' and Positions[pos - 1] == 'G':
        pass
    else:
        print("Invalid move")
```

Step 3

Now we have to make appropriate move.

Check if the selected frog is 'G':

(Inside if when it's 'G')

condition1:

If selected position +1 is empty leaf select it as positions2 by assigning it to some variable.

condition2:

If selected position +2 is empty leaf select it as positions2 by assigning it to some variable.

```
pos2 = 0
if Positions[pos] == 'G':
    if Positions[pos + 1] == '-':
        pos2 = (pos+1)
    elif Positions[pos + 2] == '-':
        pos2 = (pos+2)
```

Check if the selected frog is 'B': (Inside if when it's 'B') condition1: If selected position -1 is empty leaf select it as positions2 by assigning it to some variable.

condition2:

If selected position -2 is empty leaf select it as positions2 by assigning it to some variable.

(You can assign these values while writing invalid moves but the code will look complicated so we are assigning position in different 'if' statement.)

```
if Positions[pos] == 'B':  
    if Positions[pos - 1] == '-':  
        pos2 = (pos-1)  
    elif Positions[pos - 2] == '-':  
        pos2 = (pos-2)
```

Swap the element at selected positions and calculated position2 in the list. So basically we are moving the frog to next valid position by swapping elements of array.

```
Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]
```

Now print the display of the game again to see the change.

If we enter position 2 then the output will look like this:-

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
```

```
['G', 'G', '-', 'G', 'B', 'B', 'B']
```

```
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")  
print(Positions)
```

```
[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
```

```
['G', 'G', 'G', 'B', '-', 'B', 'B']
```

Check for winning condition by comparing the elements of list. If player has won the game print 'You Win'

```
if Positions == ['B', 'B', 'B', '-', 'G', 'G', 'G']:  
    print("Congratulations.! YOU WIN 🎉")
```

Now the game should keep running until the player quits, so place all conditional statements inside an infinite loop.

1. We have to 'break' the loop if the player presses 'q' and quits.
2. If the move made by player is 'Invalid Move' then we have to 'continue' without executing remaining part of the selected iteration.
3. If player wins the game we have to break the loop.

Infinite loop:

(inside loop)

- 1.taking inputs
- 2.Chek all conditions invalid and quite
- 3.Make the appropriate move by calculating position2.
- 4.Display game
- 4.Check winning condition

```
Positions = list(['G', 'G', 'G', '-', 'B', 'B', 'B'])
print("##### Always enter an integer less than 7 #####")
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

while True:
    pos = input("Press q to quit else \nEnter position of piece:")
    if pos == 'q':
        print("YOU LOSE!!")
        break
    pos = int(pos)

    if pos < 0 or 6 < pos:
        print("Invalid move")
        continue
    if Positions[pos] == '-':
        print("Invalid move")
        continue

    if Positions[pos] == 'G':
        if (pos + 1) <= 6 and Positions[pos + 1] == '-':
            pass
        elif (pos + 2) <= 6 and Positions[pos + 2] == '-' and
Positions[pos + 1] == 'B':
            pass
        else:
```



```

        print("Invalid move")
        break
    if Positions[pos] == 'B':
        if (pos - 1) >= 0 and Positions[pos - 1] == '-':
            pass
        elif (pos - 2) >= 0 and Positions[pos - 2] == '-' and
Positions[pos - 1] == 'G':
            pass
        else:
            print("Invalid move")

    pos2 = 0
    if Positions[pos] == 'G':
        if Positions[pos + 1] == '-':
            pos2 = (pos+1)
        elif Positions[pos + 2] == '-':
            pos2 = (pos+2)
    if Positions[pos] == 'B':
        if Positions[pos - 1] == '-':
            pos2 = (pos-1)
        elif Positions[pos - 2] == '-':
            pos2 = (pos-2)
    Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

    print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
    print(Positions)

    if Positions == ['B','B','B','-','G','G','G']:
        print("Congratulations.! YOU WIN 🎉")
        break

```

Always enter an integer less than 7

[0 , 1 , 2 , 3 , 4 , 5 , 6]

['G','G','G','-','B','B','B']

Press q to quit else

Enter position of piece:

To avoid printing the output too many times, clear the screen after every iteration.

1. Copy the following at the top of your code: `from IPython.display import clear_output`

2. We have to clear screen before displaying new state in the game. Add following function right before displaying output: `clear_output()`

Algorithm with `clear_output()`

Infinite loop:

(inside loop)

1. taking inputs

2. Check all conditions invalid and quite

3. Make the appropriate move by calculating position2.

`clear_output()`

4. Display game

4. Check winning condition

```
from IPython.display import clear_output
Positions = list(['G', 'G', 'G', '-', 'B', 'B', 'B'])
print("##### Always enter an integer less than 7 #####")
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

while True:
    pos = input("Press q to quit else \nEnter position of piece:")
    if pos == 'q':
        print("YOU LOSE!!")
        break
    pos = int(pos)

    if pos < 0 or 6 < pos:
        print("Invalid move")
        continue
    if Positions[pos] == '-':
        print("Invalid move")
        continue

    if Positions[pos] == 'G':
        if (pos + 1) <= 6 and Positions[pos + 1] == '-':
            pass
```

```

        elif (pos + 2) <= 6 and Positions[pos + 2] == '-' and
Positions[pos + 1] == 'B':
            pass
        else:
            print("Invalid move")
            break
    if Positions[pos] == 'B':
        if (pos - 1) >= 0 and Positions[pos - 1] == '-':
            pass
        elif (pos - 2) >= 0 and Positions[pos - 2] == '-' and
Positions[pos - 1] == 'G':
            pass
        else:
            print("Invalid move")

pos2 = 0
if Positions[pos] == 'G':
    if Positions[pos + 1] == '-':
        pos2 = (pos+1)
    elif Positions[pos + 2] == '-':
        pos2 = (pos+2)
if Positions[pos] == 'B':
    if Positions[pos - 1] == '-':
        pos2 = (pos-1)
    elif Positions[pos - 2] == '-':
        pos2 = (pos-2)
Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

clear_output()
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

if Positions == ['B','B','B','-','G','G','G']:
    print("Congratulations.! YOU WIN 🎉")
    break

```

Always enter an integer less than 7

```

[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
['G','G','G','-','B','B','B']

```

The game is now functionally complete.

Keeping everything inside the main loop is bad practice. Sometimes, it also introduces redundancy in the code. So let's create functions. Create 2 functions `is_invalid()` and `make_a_move()` pass selected position to both functions. Define these functions right after variable declaration

`is_invalid()`

- 1.Pass the selected position to this function.
- 2.Copy all conditions for validating a move in this function.
- 3.Remove all print statements from the conditions in the function.
- 4.In place of all pass statements we will return False
- 5.In place of all continue statements we will return True

```
from IPython.display import clear_output
Positions = list(['G', 'G', 'G', '-', 'B', 'B', 'B'])
print("##### Always enter an integer less than 7 #####")
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

while True:
    pos = input("Press q to quit else \nEnter position of piece:")
    if pos == 'q':
        print("YOU LOSE!!")
        break
    pos = int(pos)

    if pos < 0 or 6 < pos:
        print("Invalid move")
        continue

    if Positions[pos] == '-':
        print("Invalid move")
        continue

    if Positions[pos] == 'G':
        if (pos + 1) <= 6 and Positions[pos + 1] == '-':
```

```

        pass
    elif (pos + 2) <= 6 and Positions[pos + 2] == '-' and
Positions[pos + 1] == 'B':
        pass
    else:
        print("Invalid move")
        break
if Positions[pos] == 'B':
    if (pos - 1) >= 0 and Positions[pos - 1] == '-':
        pass
    elif (pos - 2) >= 0 and Positions[pos - 2] == '-' and
Positions[pos - 1] == 'G':
        pass
    else:
        print("Invalid move")

pos2 = 0
if Positions[pos] == 'G':
    if Positions[pos + 1] == '-':
        pos2 = (pos+1)
    elif Positions[pos + 2] == '-':
        pos2 = (pos+2)
if Positions[pos] == 'B':
    if Positions[pos - 1] == '-':
        pos2 = (pos-1)
    elif Positions[pos - 2] == '-':
        pos2 = (pos-2)
Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

clear_output()
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

if Positions == ['B','B','B','-','G','G','G']:
    print("Congratulations.! YOU WIN 🎉")
    break

```

make_a_move()'

1. Pass the selected position to this function.
2. Copy the code snippet that calculates position2 and also swaps the selected position with position2.

```
def make_a_move(pos):
    pos2 = 0
    if Positions[pos] == 'G':
        if Positions[pos + 1] == '-':
            pos2 = (pos+1)
        elif Positions[pos + 2] == '-':
            pos2 = (pos+2)
    elif Positions[pos] == 'B':
        if Positions[pos - 1] == '-':
            pos2 = (pos-1)
        elif Positions[pos - 2] == '-':
            pos2 = (pos-2)

    Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]
```

Inside the while loop we have removed validating conditions and also make a move calculation. Instead of that write the following condition.

If value returned by i_invalid() if 'True'

then print 'invalid move'

Else

call make_a_move() function

Write the whole code by making given changes.

```
from IPython.display import clear_output
Positions = list(['G','G','G','-','B','B','B'])
print("##### Always enter an integer less than 7 #####")
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)
```

```

def is_Invalid(pos):
    if pos<0 or 6<pos:
        return True
    if Positions[pos] == '-':
        return True

    if Positions[pos] == 'G':
        if (pos + 1) <= 6 and Positions[pos + 1] == '-':
            return False
        elif (pos + 2) <= 6 and Positions[pos + 2] == '-' and
Positions[pos + 1] == 'B':
            return False
        else:
            return True

    if Positions[pos] == 'B':
        if (pos - 1) >= 0 and Positions[pos - 1] == '-':
            return False
        elif (pos - 2) >= 0 and Positions[pos - 2] == '-' and
Positions[pos - 1] == 'G':
            return False
        else:
            return True

def make_a_move(pos):
    pos2 = 0
    if Positions[pos] == 'G':
        if Positions[pos + 1] == '-':
            pos2 = (pos+1)
        elif Positions[pos + 2] == '-':
            pos2 = (pos+2)
    elif Positions[pos] == 'B':
        if Positions[pos - 1] == '-':
            pos2 = (pos-1)
        elif Positions[pos - 2] == '-':
            pos2 = (pos-2)

    Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

while(True):

```

```

pos = input("Press q to quit else \nEnter position of piece:")
if pos == 'q':
    print("YOU LOSE!!")
    break
pos = int(pos)
if is_Invalid(pos):
    print("Invalid move")
    continue
else:
    make_a_move(pos)

clear_output()
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)
if Positions == ['B','B','B','-','G','G','G']:
    print("Congratulations.! YOU WIN 🎉")
    break

```

Now the game is complete.

If you want to make it look more stylish then replace with emoji.

Replace:

'G' = '🐸' = '\U0001F438'

'B' = '🐙' = '\U0001F42C'

'-' = '🍁' = '\U0001F341'

```

from IPython.display import clear_output
Positions =
list(['\U0001F438', '\U0001F438', '\U0001F438', '\U0001F341', '\U0001F42C', '\U
0001F42C', '\U0001F42C'])
print("##### Always enter an integer less than 7 #####")
print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
print(Positions)

def is_Invalid(pos):
    if pos<0 or 6<pos:
        return True
    if Positions[pos] == '\U0001F341':

```



```

        return True

    if Positions[pos] == '\U0001F438':
        if (pos + 1) <= 6 and Positions[pos + 1] == '\U0001F341':
            return False
        elif (pos + 2) <= 6 and Positions[pos + 2] == '\U0001F341' and
Positions[pos + 1] == '\U0001F42C':
            return False
        else:
            return True

    if Positions[pos] == '\U0001F42C':
        if (pos - 1) >= 0 and Positions[pos - 1] == '\U0001F341':
            return False
        elif (pos - 2) >= 0 and Positions[pos - 2] == '\U0001F341' and
Positions[pos - 1] == '\U0001F438':
            return False
        else:
            return True

def make_a_move(pos):
    pos2 = 0
    if Positions[pos] == '\U0001F438':
        if Positions[pos + 1] == '\U0001F341':
            pos2 = (pos+1)
        elif Positions[pos + 2] == '\U0001F341':
            pos2 = (pos+2)
    elif Positions[pos] == '\U0001F42C':
        if Positions[pos - 1] == '\U0001F341':
            pos2 = (pos-1)
        elif Positions[pos - 2] == '\U0001F341':
            pos2 = (pos-2)

    Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

while(True):
    print("Press q to quit else \nEnter position of piece:", end='')
    pos = input()
    if pos == 'q':
        print("YOU LOSE!!")

```

```

        break
    pos = int(pos)
    if is_Invalid(pos):
        print("Invalid move")
        continue
    else:
        make_a_move(pos)

    clear_output()
    print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
    print(Positions)

    if Positions ==
['\U0001F42C', '\U0001F42C', '\U0001F42C', '\U0001F341', '\U0001F438', '\U0001F
438', '\U0001F438']:
        print("Congratulations.! YOU WIN 🎉")
        break

```

How to Play

Step 1: Display

The game begins by displaying the initial set of frogs on the left and right sides.

Step 2: Input

Enter the position of the frog you want to move. For example, entering position 2 will look like this:

```

[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
['G', 'G', 'G', '-', 'B', 'B', 'B']

```

```

[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
['🐸', '🐸', '🐸', '🍁', '🐸', '🐸', '🐸']

```

Step 3: Rules and Validation

Ensure that the entered position is between 0 and 6 (inclusive) or 'q' to quit the game.

The entered position cannot be the position of an empty leaf ('🍁').

Check if the selected frog can perform the move based on the game rules.

Step 4: Make a Move

Make the appropriate move by updating the game display. The frog can move to an empty space or leap over another frog.

Step 5: Winning Condition

Check if the two sets of frogs have switched positions. If so, the player wins the game.

Game Flow

Display the initial game state.

Enter the position of the frog to move.

Validate the input according to the rules.

If the input is valid, make the move and update the display.

Check if the player has won.

Repeat steps 2-5 until the player quits or wins.

Using Emoji

To make the game more visually appealing, emoji representations are used:

If you want to make it look more stylish then replace with emoji.

Replace:

'G' = '🐸' = '\U0001F438'

'B' = '🐸' = '\U0001F42C'

'.' = '🍁' = '\U0001F341'

'🐸' for green frogs

'🐸' for brown frogs

'🍁' for empty spaces

Example Gameplay



```
Assignment-2 frog-leap game.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

if Positions[pos + 1] == '\U0001F341':
    pos2 = (pos+1)
elif Positions[pos + 2] == '\U0001F341':
    pos2 = (pos+2)
elif Positions[pos] == '\U0001F42C':
    if Positions[pos - 1] == '\U0001F341':
        pos2 = (pos-1)
    elif Positions[pos - 2] == '\U0001F341':
        pos2 = (pos-2)

Positions[pos], Positions[pos2] = Positions[pos2], Positions[pos]

while(True):
    print("Press q to quit else \nEnter position of piece:", end='')
    pos = input()
    if pos == 'q':
        print("YOU LOSE!!")
        break
    pos = int(pos)
    if is_invalid(pos):
        print("Invalid move")
        continue
    else:
        make_a_move(pos)

    clear_output()
    print("[ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]")
    print(Positions)

    if Positions == ['\U0001F42C', '\U0001F42C', '\U0001F42C', '\U0001F341', '\U0001F438', '\U0001F438', '\U0001F438']:
        print("Congratulations! YOU WIN 🎉")
        break

##### Always enter an integer less than 7 #####
[ 🐸 , 🐸 , 🐸 , 🍁 , 🐸 , 🐸 , 🐸 ]
Press q to quit else
Enter position of piece:

```

[0 , 1 , 2 , 3 , 4 , 5 , 6]
['🐸', '🐸', '🐸', '🍁', '🌀', '🌀', '🌀']

Press q to quit else
Enter position of piece:2

Press q to quit else
Enter position of piece:4

Press q to quit else
Enter position of piece:5

Press q to quit else
Enter position of piece:3

Press q to quit else
Enter position of piece:1

Press q to quit else
Enter position of piece:0

Press q to quit else
Enter position of piece:2

Press q to quit else
Enter position of piece:4

Press q to quit else
Enter position of piece:6

Press q to quit else
Enter position of piece:5

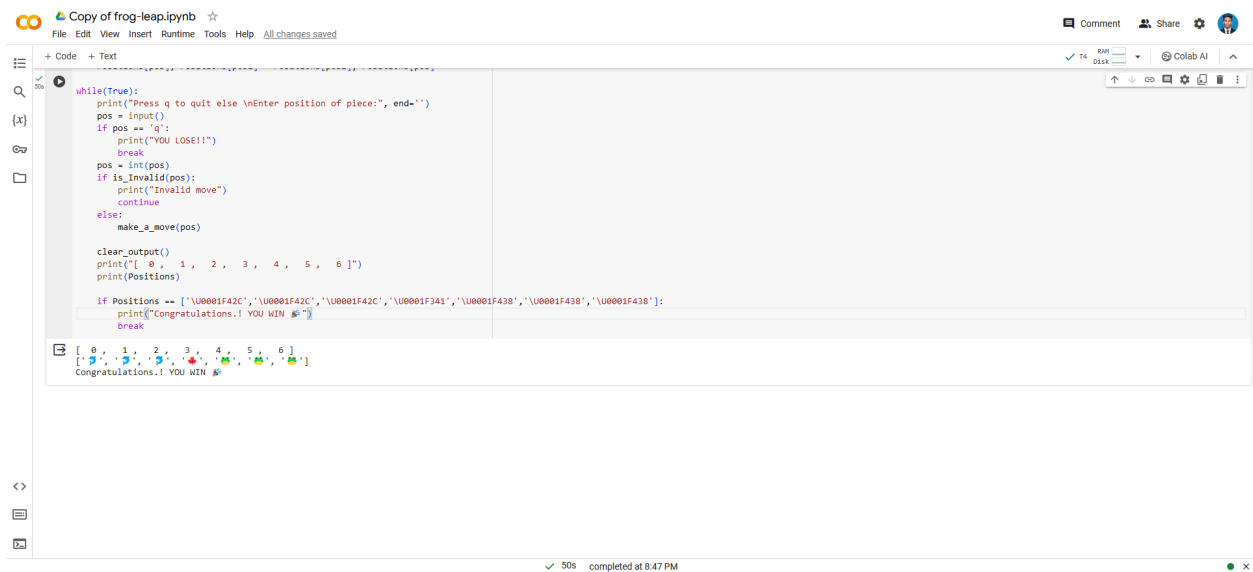
Press q to quit else
Enter position of piece:3

Press q to quit else
Enter position of piece:1

Press q to quit else
Enter position of piece:2

Press q to quit else
Enter position of piece:4

Press q to quit else
Enter position of piece:3



```
Copy of frog-leap.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
while(True):
    print("Press q to quit else \nEnter position of piece:", end='')
    pos = input()
    if pos == 'q':
        print("YOU LOSE!!")
        break
    pos = int(pos)
    if is_invalid(pos):
        print("Invalid move")
        continue
    else:
        make_a_move(pos)

    clear_output()
    print([ 0, 1, 2, 3, 4, 5, 6 ])
    print(Positions)

    if Positions == ['\U0001F42C', '\U0001F42C', '\U0001F42C', '\U0001F341', '\U0001F438', '\U0001F438', '\U0001F438']:
        print("Congratulations.! YOU WIN 🎉")
        break

[ 0, 1, 2, 3, 4, 5, 6 ]
[ 🐸, 🐸, 🐸, 🍁, 🐸, 🐸, 🐸 ]
Congratulations.! YOU WIN 🎉
50s completed at 8:47 PM
```

[0, 1, 2, 3, 4, 5, 6]
[🐸, 🐸, 🐸, 🍁, 🐸, 🐸, 🐸]

Congratulations.! YOU WIN 🎉

Note

To enhance the display further, you can use emoji representations for better aesthetics.

Enjoy playing the Frog Leap game
