

COURSE: Introduction to Python (IBM Course)

Image Steganography and Snakes and Ladder Game

[Varad Gandhi- I014

Rohit Bhatia-I006

Devang Tyagi-I060]

Image
Steganography

Software/IDE/Interactive notebook: Google Collab

Algorithm and Program:

```
import cv2
import numpy as np
import types
from google.colab.patches import cv2_imshow

def messageToBinary(message):
    if type(message) == str:
        return ".join([ format(ord(i), "08b") for i in message ])
    elif type(message) == bytes or type(message) == np.ndarray:
        return [ format(i, "08b") for i in message ]
    elif type(message) == int or type(message) == np.uint8:
        return format(message, "08b")
    else:
        raise TypeError("Input type not supported")

def hideData(image, secret_message):

    # calculate the maximum bytes to encode
    n_bytes = image.shape[0] * image.shape[1] * 3 // 8
    print("Maximum bytes to encode:", n_bytes)

    #Check if the number of bytes to encode is less than the maximum bytes in the image
    if len(secret_message) > n_bytes:
        raise ValueError("Error encountered insufficient bytes, need bigger image or less data !!")
```

```
secret_message += "#####" # you can use any string as the delimiter
```

```
data_index = 0
```

```
# convert input data to binary format using messageToBinary() function
```

```
binary_secret_msg = messageToBinary(secret_message)
```

```
data_len = len(binary_secret_msg) #Find the length of data that needs to be hidden  
for values in image:
```

```
    for pixel in values:
```

```
        # convert RGB values to binary format
```

```
        r, g, b = messageToBinary(pixel)
```

```
        # modify the least significant bit only if there is still data to store
```

```
        if data_index < data_len:
```

```
            # hide the data into least significant bit of red pixel
```

```
            pixel[0] = int(r[:-1] + binary_secret_msg[data_index], 2)
```

```
            data_index += 1
```

```
        if data_index < data_len:
```

```
            # hide the data into least significant bit of green pixel
```

```
            pixel[1] = int(g[:-1] + binary_secret_msg[data_index], 2)
```

```
            data_index += 1
```

```
        if data_index < data_len:
```

```
            # hide the data into least significant bit of blue pixel
```

```
            pixel[2] = int(b[:-1] + binary_secret_msg[data_index], 2)
```

```
            data_index += 1
```

```
        # if data is encoded, just break out of the loop
```

```
        if data_index >= data_len:
```

```
            break
```

```
return image
```

```
def showData(image):
```

```
    binary_data = ""
```

```
    for values in image:
```

```
        for pixel in values:
```

```
            r, g, b = messageToBinary(pixel) #convert the red,green and blue values into binary format
```

```
            binary_data += r[-1] #extracting data from the least significant bit of red pixel
```

```
            binary_data += g[-1] #extracting data from the least significant bit of red pixel
```

```
            binary_data += b[-1] #extracting data from the least significant bit of red pixel
```

```
    # split by 8-bits
```

```
    all_bytes = [ binary_data[i: i+8] for i in range(0, len(binary_data), 8) ]
```

```

# convert from bits to characters
decoded_data = ""
for byte in all_bytes:
    decoded_data += chr(int(byte, 2))
    if decoded_data[-5:] == "#####": #check if we have reached the delimiter which is "#####"
        break
#print(decoded_data)
return decoded_data[:-5]

def encode_text():
    image_name = input("Enter image name(with extension): ")
    image = cv2.imread(image_name) # Read the input image using OpenCV-Python.
    #It is a library of Python bindings designed to solve computer vision problems.

    #details of the image
    print("The shape of the image is: ",image.shape) #check the shape of image to calculate the number of bytes in it
    print("The original image is as shown below: ")
    resized_image = cv2.resize(image, (500, 500)) #resize the image as per your requirement
    cv2.imshow(resized_image) #display the image

    data = input("Enter data to be encoded : ")
    if (len(data) == 0):
        raise ValueError('Data is empty')

    filename = input("Enter the name of new encoded image(with extension): ")
    encoded_image = hideData(image, data) # call the hideData function to hide the secret message into the selected image
    cv2.imwrite(filename, encoded_image)

def decode_text():
    # read the image that contains the hidden image
    image_name = input("Enter the name of the steganographed image that you want to decode (with extension) :")
    image = cv2.imread(image_name) #read the image using cv2.imread()

    print("The Steganographed image is as shown below: ")
    resized_image = cv2.resize(image, (500, 500)) #resize the original image as per your requirement
    cv2.imshow(resized_image) #display the Steganographed image

```

```

text = showData(image)
return text

def Steganography():
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")
    )
    userinput = int(a)
    if (userinput == 1):
        print("\nEncoding....")
        encode_text()

    elif (userinput == 2):
        print("\nDecoding....")
        print("Decoded message is " + decode_text())
    else:
        raise Exception("Enter correct input")

Steganography() #encode image

```

Applications:

1. Image Steganography is used to hide very important message in the form of audio or pictures.
2. Enhancing the secrecy of encrypted data.
3. It is used in media database system.

Simulation results :

Encoding :

```
def Steganography():  
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")  
    userInput = int(a)  
    if (userInput == 1):  
        print("\nEncoding....")  
        encode_text()  
  
    elif (userInput == 2):  
        print("\nDecoding....")  
        print("Decoded message is " + decode_text())  
    else:  
        raise Exception("Enter correct input")  
  
Steganography() #encode image
```

Image Steganography
1. Encode the data
2. Decode the data
Your input is: 1

Encoding....
Enter image name(with extension): jetplane.tif
The shape of the image is: (512, 512, 3)
The original image is as shown below:



Enter data to be encoded : hi there
Enter the name of new encoded image(with extension): jetplanenew1.tif
Maximum bytes to encode: 98304

Decoding:

```
def Steganography():  
    a = input("Image Steganography \n 1. Encode the data \n 2. Decode the data \n Your input is: ")  
    userInput = int(a)  
    if (userInput == 1):  
        print("\nEncoding....")  
        encode_text()  
  
    elif (userInput == 2):  
        print("\nDecoding....")  
        print("Decoded message is " + decode_text())  
    else:  
        raise Exception("Enter correct input")  
  
Steganography() #decode image
```

Image Steganography
1. Encode the data
2. Decode the data
Your input is: 2

Decoding....
Enter the name of the steganographed image that you want to decode (with extension) :jetplanenew1.tif
The Steganographed image is as shown below:



Decoded message is hi there

Conclusion:

Image steganography has a lot of applications in data hiding and secrecy. It has some similar characteristics as Cryptography but it is distinguished on the basis of its working i.e Cryptography makes the data unreadable but Steganography just hides the data which is given as input.

Snakes and Ladder game

Software/IDE/Interactive notebook:

we have used pycharm with python3.8

interpreter.

Algorithm and Program :

```
import time
import random
import sys
SLEEP_BETWEEN_ACTIONS = 1
MAX_VAL = 100
DICE_FACE = 6

snakes = {
    8: 4,
    18: 1,
    26: 10,
    39: 5,
    51: 6,
    54: 36,
    56: 1,
    60: 23,
    75: 28,
    83: 45,
    85: 59,
    90: 48,
    92: 25,
    97: 87,
    99: 63
}

ladders = {
    3: 20,
    6: 14,
    11: 28,
    15: 34,
    17: 74,
    22: 37,
    38: 59,
    49: 67,
    57: 76,
    61: 78,
    73: 86,
```

```

81: 98,
88: 91
}

player_turn_text = [
    "Your turn.",
    "Go.",
    "Please proceed.",
    "Lets win this.",
    "Are you ready?",
    "",
]

snake_bite = [
    "boohoo",
    "bummer",
    "snake bite",
    "oh no",
    "dang"
]

ladder_jump = [
    "woohoo",
    "woww",
    "nailed it",
    "oh my God...",
    "yaayyy"
]

def welcome_msg():
    msg = """
    Welcome to Snake and Ladder Game.

    Rules:
    1. Initially both the players are at starting position i.e. 0.
       Take it in turns to roll the dice.
       Move forward the number of spaces shown on the dice.
    2. If you lands at the bottom of a ladder, you can move up to the top
of the ladder.
    3. If you lands on the head of a snake, you must slide down to the
bottom of the snake.
    4. The first player to get to the FINAL position is the winner.
    5. Hit enter to roll the dice.

    """
    print(msg)

def get_player_names():
    player1_name = None
    while not player1_name:
        player1_name = input("Please enter a valid name for first player:

```



```

").strip()

    player2_name = None
    while not player2_name:
        player2_name = input("Please enter a valid name for second player:
").strip()

    print("\nMatch will be played between '" + player1_name + "' and '" +
player2_name + "'\n")
    return player1_name, player2_name

def get_dice_value():
    time.sleep(SLEEP_BETWEEN_ACTIONS)
    dice_value = random.randint(1, DICE_FACE)
    print("Its a " + str(dice_value))
    return dice_value

def got_snake_bite(old_value, current_value, player_name):
    print("\n" + random.choice(snake_bite).upper() + " ~~~~~>")
    print("\n" + player_name + " got a snake bite. Down from " +
str(old_value) + " to " + str(current_value))

def got_ladder_jump(old_value, current_value, player_name):
    print("\n" + random.choice(ladder_jump).upper() + " #####")
    print("\n" + player_name + " climbed the ladder from " + str(old_value) +
" to " + str(current_value))

def snake_ladder(player_name, current_value, dice_value):
    time.sleep(SLEEP_BETWEEN_ACTIONS)
    old_value = current_value
    current_value = current_value + dice_value

    if current_value > MAX_VAL:
        print("You need " + str(MAX_VAL - old_value) + " to win this game.
Keep trying.")
        return old_value

    print("\n" + player_name + " moved from " + str(old_value) + " to " +
str(current_value))
    if current_value in snakes:
        final_value = snakes.get(current_value)
        got_snake_bite(current_value, final_value, player_name)

    elif current_value in ladders:
        final_value = ladders.get(current_value)
        got_ladder_jump(current_value, final_value, player_name)

    else:
        final_value = current_value

```

```

    return final_value

def check_win(player_name, position):
    time.sleep(SLEEP_BETWEEN_ACTIONS)
    if MAX_VAL == position:
        print("\n\n\nThats it.\n\n" + player_name + " won the game.")
        print("Congratulations " + player_name)
        print("\nThank you for playing the game. Please visit
https://www.pythoncircle.com\n\n")
        sys.exit(1)

def start():
    welcome_msg()
    time.sleep(SLEEP_BETWEEN_ACTIONS)
    player1_name, player2_name = get_player_names()
    time.sleep(SLEEP_BETWEEN_ACTIONS)

    player1_current_position = 0
    player2_current_position = 0

    while True:
        time.sleep(SLEEP_BETWEEN_ACTIONS)
        input_1 = input("\n" + player1_name + ": " +
random.choice(player_turn_text) + " Hit the enter to roll dice: ")
        print("\nRolling dice...")
        dice_value = get_dice_value()
        time.sleep(SLEEP_BETWEEN_ACTIONS)
        print(player1_name + " moving....")
        player1_current_position = snake_ladder(player1_name,
player1_current_position, dice_value)

        check_win(player1_name, player1_current_position)

        input_2 = input("\n" + player2_name + ": " +
random.choice(player_turn_text) + " Hit the enter to roll dice: ")
        print("\nRolling dice...")
        dice_value = get_dice_value()
        time.sleep(SLEEP_BETWEEN_ACTIONS)
        print(player2_name + " moving....")
        player2_current_position = snake_ladder(player2_name,
player2_current_position, dice_value)

        check_win(player2_name, player2_current_position)

if __name__ == "__main__":
    start()

```

Applications:

Snakes and Ladders is an ancient Indian board game regarded today as a worldwide classic. It is played between two or more players on a gameboard having numbered, gridded squares. A number of “ladders” and snakes are pictured on the board each connecting two specific board squares. The object of the game is to navigate one’s game piece according to die rolls, from start to the finish, helped or hindered by ladders and snakes respectively.

In this coronavirus era, these short games are really interesting to spend the time and enjoy with friends.

Simulation results

```
C:\Users\Dell\AppData\Local\Programs\Python\Python38\python.exe "C:/Users/Dell/PycharmProjects/projects/snake and ladder.py"

Welcome to Snake and Ladder Game.

Rules:
1. Initially both the players are at starting position i.e. 0.
   Take it in turns to roll the dice.
   Move forward the number of spaces shown on the dice.
2. If you lands at the bottom of a ladder, you can move up to the top of the ladder.
3. If you lands on the head of a snake, you must slide down to the bottom of the snake.
4. The first player to get to the FINAL position is the winner.
5. Hit enter to roll the dice.

Please enter a valid name for first player: Rohit
Please enter a valid name for second player: Varad

Match will be played between 'Rohit' and 'Varad'

Rohit: Are you ready? Hit the enter to roll dice:

Rolling dice...
Its a 6
Rohit moving....

Rohit moved from 0 to 6

YAAYYY #####

Rohit climbed the ladder from 6 to 14

Varad: Hit the enter to roll dice:
```

Varad: Hit the enter to roll dice:

Rolling dice...
Its a 2
Varad moving....

Varad moved from 0 to 2

Rohit: Go. Hit the enter to roll dice:

Rolling dice...
Its a 3
Rohit moving....

Rohit moved from 14 to 17

NAILED IT #####

Rohit climbed the ladder from 17 to 74

Varad: Lets win this. Hit the enter to roll dice:

Rolling dice...
Its a 5
Varad moving....

Varad moved from 2 to 7

Rohit: Please proceed. Hit the enter to roll dice:

Rolling dice...
Its a 6
Rohit moving....

Rohit moved from 74 to 80

Rolling dice...
Its a 6
Rohit moving....

Rohit moved from 74 to 80

Varad: Lets win this. Hit the enter to roll dice:

Rolling dice...
Its a 6
Varad moving....

Varad moved from 7 to 13

Rohit: Lets win this. Hit the enter to roll dice:

Rolling dice...
Its a 2
Rohit moving....

Rohit moved from 80 to 82

Varad: Are you ready? Hit the enter to roll dice:

Rolling dice...
Its a 2
Varad moving....

Varad moved from 13 to 15

YAAYYY #####

Varad climbed the ladder from 15 to 34

Rohit: Your turn. Hit the enter to roll dice: |

```
Rohit moved from 82 to 86

Varad: Please proceed. Hit the enter to roll dice:

Rolling dice...
Its a 5
Varad moving....

Varad moved from 34 to 39

BOOHOO ~~~~~>

Varad got a snake bite. Down from 39 to 5

Rohit: Your turn. Hit the enter to roll dice:

Rolling dice...
Its a 6
Rohit moving....

Rohit moved from 86 to 92

DANG ~~~~~>

Rohit got a snake bite. Down from 92 to 25

Varad: Lets win this. Hit the enter to roll dice:

Rolling dice...
Its a 2
Varad moving....

Varad moved from 5 to 7

Rohit: Please proceed. Hit the enter to roll dice:
```

Conclusion

We have implemented snakes and ladders game using dictionaries and user defined functions.