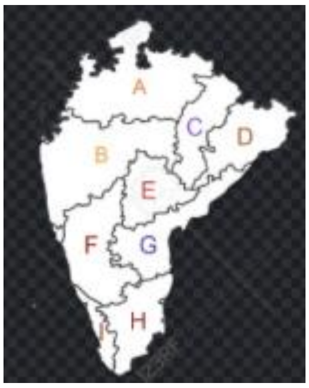
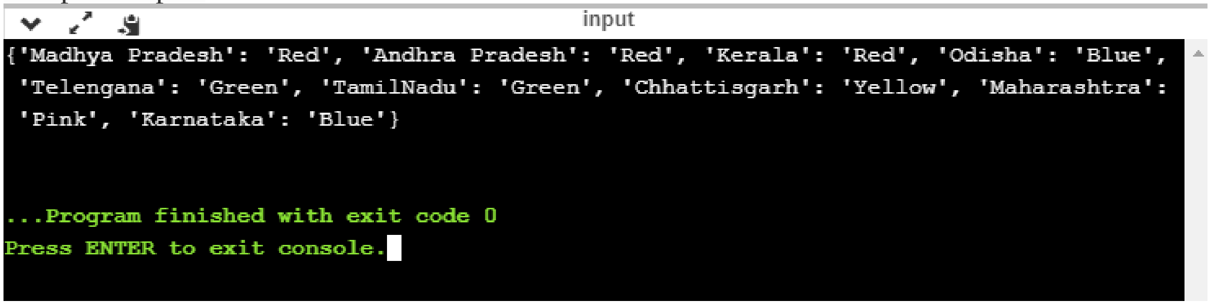
INLAB

1. You have to color a map with different colors where no two neighboring regions can have the same color. Assume that you have 5 colors red, blue, green, yellow and pink.Write an efficient python code to color the regions in the following map.

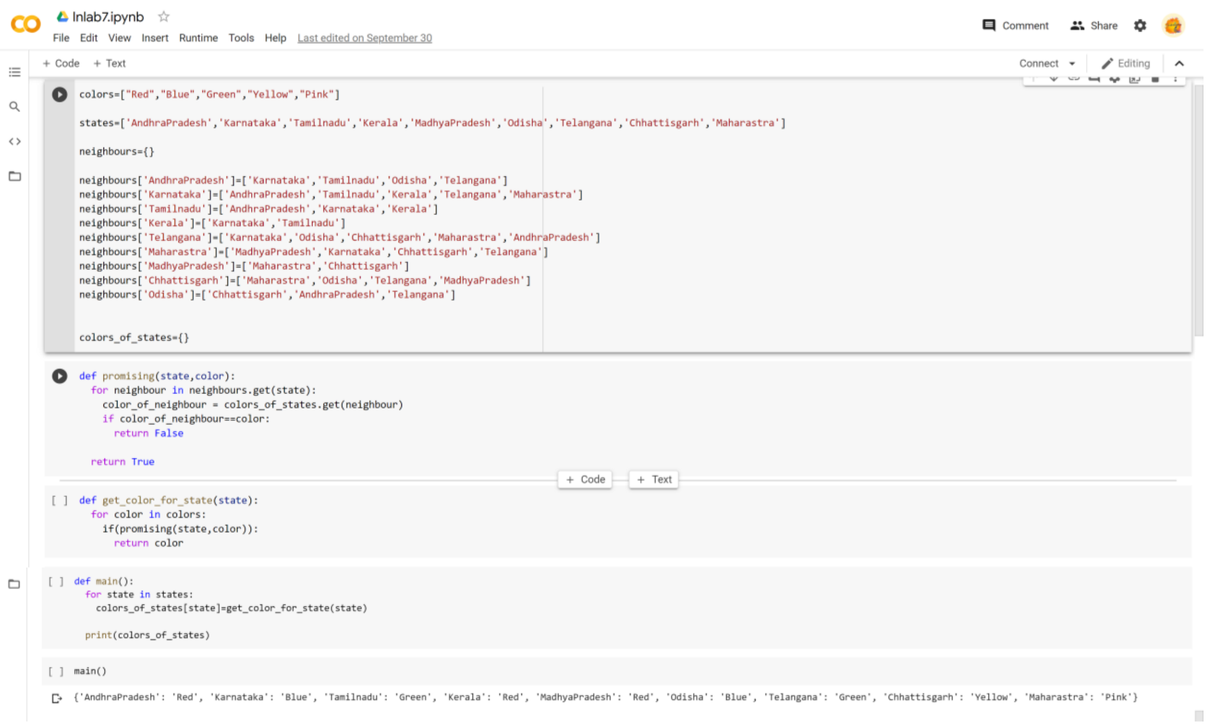


Sample Output:



```
{'Madhya Pradesh': 'Red', 'Andhra Pradesh': 'Red', 'Kerala': 'Red', 'Odisha': 'Blue',
 'Telengana': 'Green', 'TamilNadu': 'Green', 'Chhattisgarh': 'Yellow', 'Maharashtra':
 'Pink', 'Karnataka': 'Blue'}


...Program finished with exit code 0
Press ENTER to exit console.
```

Code:


Inlab7.ipynb
File Edit View Insert Runtime Tools Help Last edited on September 30
Comment Share
+ Code + Text
Connect Editing

```python
colors=["Red","Blue","Green","Yellow","Pink"]

states=['AndhraPradesh','Karnataka','Tamilnadu','Kerala','MadhyaPradesh','Odisha','Telangana','Chhattisgarh','Maharastra']

neighbours={}

neighbours['AndhraPradesh']=['Karnataka','Tamilnadu','Odisha','Telangana']
neighbours['Karnataka']=['AndhraPradesh','Tamilnadu','Kerala','Telangana','Maharastra']
neighbours['Tamilnadu']=['AndhraPradesh','Karnataka','Kerala']
neighbours['Kerala']=['Karnataka','Tamilnadu']
neighbours['Telangana']=['Karnataka','Odisha','Chhattisgarh','Maharastra','AndhraPradesh']
neighbours['Maharastra']=['MadhyaPradesh','Karnataka','Chhattisgarh','Telangana']
neighbours['MadhyaPradesh']=['Maharastra','Chhattisgarh']
neighbours['Chhattisgarh']=['Maharastra','Odisha','Telangana','MadhyaPradesh']
neighbours['Odisha']=['Chhattisgarh','AndhraPradesh','Telangana']


colors_of_states={}
```

```python
def promising(state,color):
  for neighbour in neighbours.get(state):
    color_of_neighbour = colors_of_states.get(neighbour)
    if color_of_neighbour==color:
      return False

  return True
```

```python
def get_color_for_state(state):
  for color in colors:
    if(promising(state,color)):
      return color
```

```python
def main():
  for state in states:
    colors_of_states[state]=get_color_for_state(state)

  print(colors_of_states)
```

```python
main()
```

```
{'AndhraPradesh': 'Red', 'Karnataka': 'Blue', 'Tamilnadu': 'Green', 'Kerala': 'Red', 'MadhyaPradesh': 'Red', 'Odisha': 'Blue', 'Telangana': 'Green', 'Chhattisgarh': 'Yellow', 'Maharastra': 'Pink'}
```

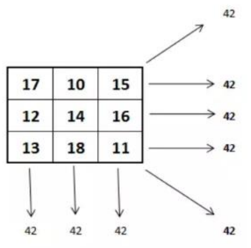
Code:

```
colors=["Red","Blue","Green","Yellow","Pink"]

states=['AndhraPradesh','Karnataka','Tamilnadu','Kerala','MadhyaPradesh
','Odisha','Telangana','Chhattisgarh','Maharastra']

neighbours={}

neighbours['AndhraPradesh']=['Karnataka','Tamilnadu','Odisha','Telangan
a']
neighbours['Karnataka']=['AndhraPradesh','Tamilnadu','Kerala','Telangan
a','Maharastra']
neighbours['Tamilnadu']=['AndhraPradesh','Karnataka','Kerala']
neighbours['Kerala']=['Karnataka','Tamilnadu']
neighbours['Telangana']=['Karnataka','Odisha','Chhattisgarh','Maharastr
a','AndhraPradesh']
neighbours['Maharastra']=['MadhyaPradesh','Karnataka','Chhattisgarh','T
elangana']
neighbours['MadhyaPradesh']=['Maharastra','Chhattisgarh']
neighbours['Chhattisgarh']=['Maharastra','Odisha','Telangana','MadhyaPr
adesh']
neighbours['Odisha']=['Chhattisgarh','AndhraPradesh','Telangana']


colors_of_states={}
def promising(state,color):
  for neighbour in neighbours.get(state):
    color_of_neighbour = colors_of_states.get(neighbour)
    if color_of_neighbour==color:
      return False
  return True

def get_color_for_state(state):
  for color in colors:
    if(promising(state,color)):
      return color

def main():
  for state in states:
    colors_of_states[state]=get_color_for_state(state)

  print(colors_of_states)

main()
```

POSTLAB

1. Solve the following problem using Constraint Satisfaction Problems (CSP):

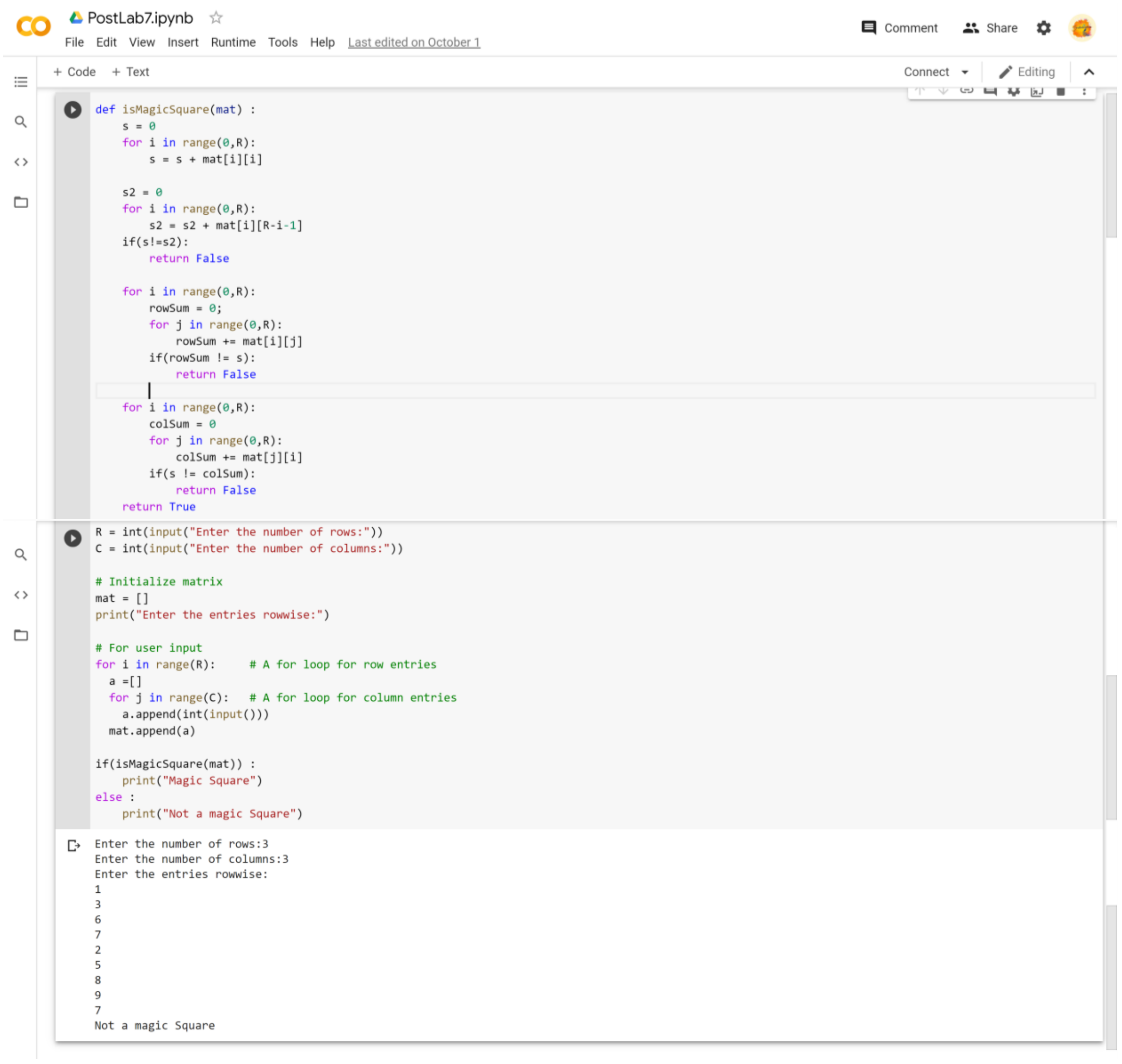
Test Case 1: Magic Square ([[10,11,12], [13, 14, 15], [16, 17, 18]])
False, this is not a magic square. The numbers in the rows/columns/diagonals do not add up to the same value. Let's try another list of lists.



Test Case 2: Magic Square ([[17,10,15],[12,14,16],[13,18,11]])
True

• Develop a python program that satisfies below operations.

Code:


PostLab7.ipynb
File Edit View Insert Runtime Tools Help Last edited on October 1
Comment Share
+ Code + Text
Connect Editing

```
def isMagicSquare(mat) :
    s = 0
    for i in range(0,R):
        s = s + mat[i][i]

    s2 = 0
    for i in range(0,R):
        s2 = s2 + mat[i][R-i-1]
    if(s!=s2):
        return False

    for i in range(0,R):
        rowSum = 0;
        for j in range(0,R):
            rowSum += mat[i][j]
        if(rowSum != s):
            return False

    for i in range(0,R):
        colSum = 0
        for j in range(0,R):
            colSum += mat[j][i]
        if(s != colSum):
            return False
    return True
```

```
R = int(input("Enter the number of rows:"))
C = int(input("Enter the number of columns:"))

# Initialize matrix
mat = []
print("Enter the entries rowwise:")

# For user input
for i in range(R):      # A for loop for row entries
  a =[]
  for j in range(C):    # A for loop for column entries
    a.append(int(input()))
  mat.append(a)

if(isMagicSquare(mat)) :
    print("Magic Square")
else :
    print("Not a magic Square")
```

```
Enter the number of rows:3
Enter the number of columns:3
Enter the entries rowwise:
1
3
6
7
2
5
8
9
7
Not a magic Square
```

Code:

```
def isMagicSquare(mat) :
    s = 0
    for i in range(0,R):
        s = s + mat[i][i]

    s2 = 0
    for i in range(0,R):
        s2 = s2 + mat[i][R-i-1]
    if(s!=s2):
        return False

    for i in range(0,R):
        rowSum = 0;
        for j in range(0,R):
            rowSum += mat[i][j]
        if(rowSum != s):
            return False

    for i in range(0,R):
        colSum = 0
        for j in range(0,R):
            colSum += mat[j][i]
        if(s != colSum):
            return False
    return True


R = int(input("Enter the number of rows:"))
C = int(input("Enter the number of columns:"))

# Initialize matrix
mat = []
print("Enter the entries rowwise:")

# For user input
for i in range(R):      # A for loop for row entries
  a =[]
  for j in range(C):    # A for loop for column entries
    a.append(int(input()))
  mat.append(a)

if(isMagicSquare(mat)) :
    print("Magic Square")
else :
    print("Not a magic Square")
```