Programming Tools in Artificial Intelligence
Course Code: (CT5132/CT5148)
Assignment 3

	Student Name	Student ID	GitHub Username
1	Moiz Meyaji	19233048	MoizSM
2	Waseem Shariff	19233839	waseemshariff126

Tasks:

Task 1: 6f8cd79b.json

Task 2: ba26e723.json

Task 3: 9dfd6313.json

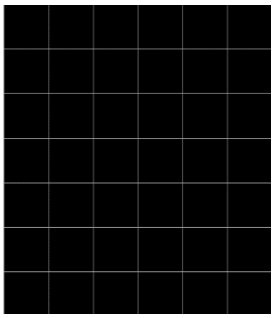
Task 4: 794b24be.json

Task 5: 6cdd2623.json

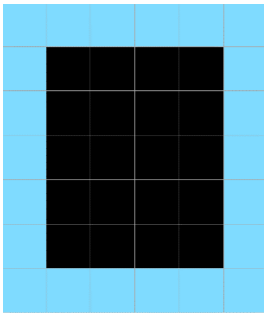
GitHub Link: https://github.com/MoizSM/ARC

Task 1: 6f8cd79b.json

Input:



Output:



Pattern Logic:

From the above representation of the images we can observe that, the input image is a blank grid and the expected output is the grid with blue color at its first and last row and column. We have to find a logic such that, any given input with blank grid, outputs a grid with a color at its boundaries.

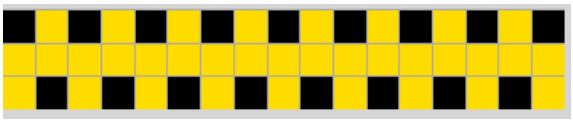
Code Logic:

We have defined a function **solve,** which consists of the logic of this problem. In the solve function we first take the input of all the training cases and all the test cases. We then solve each case by displaying their specific outputs. Using for loop we first took the rows of the grid, then we colored the first and last row of the grid. Now for the first and last column of the grid, we use nested for loop, where we get the elements of each row separately. Then we assign the colors to the first and last element of each row.

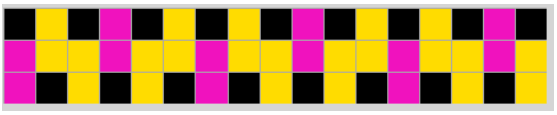
- The function solve() produces the correct output for all the test and training cases.

Task 2: ba26e723.json

Input:



Output:



Pattern Logic:

As we can see from the above representations that the input grid contains yellow (4) colored elements in a pattern above. The output grid obtained replaces the yellow colored elements with pink (6) color elements for every j+3 (where j is the column number) elements starting from the first column. The shape of the pattern remains the same.

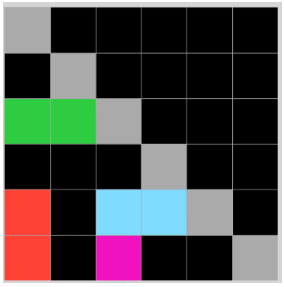
Code Logic:

In the code for this task, the solve function parses the json file and runs a for-loop on all the training and testing input grids. We the use a while-loop to index through the column number (j) and a for-loop to index the row number (i). The row elements are colored pink (6) for every j + 3 iterations.

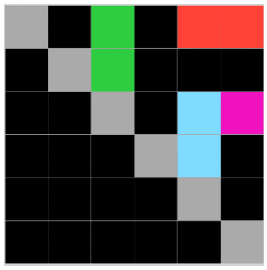
- The function solve() produces the correct output for all the training and testing cases.

Task 3: 9dfd6313.json

Input:



Output:



Pattern Logic:

As we can see from the above representations that the input grid contains a random collection of colored elements scattered throughout the grid with grey (5) occupying the left diagonal. The resultant grid produces an output that transforms the input grid counterclockwise by 90° and flips it up-down.

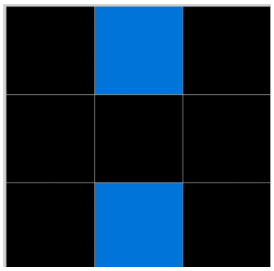
Code Logic:

In the code for this task, the solve function parses the json file and runs a for-loop for all training and testing input grids. It then transposes the grid by 90° and flips it up-down using the NumPy functions. It then prints the output grid for all the training and test cases.

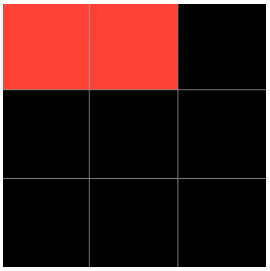
- The function solve() produces the correct output for all the train and test cases.

Task 4: 794b24be.json

Input:



Output:



Pattern Logic:

From the above representation of the images we can observe that, the input grid consists of 'n' number of scattered blue (8) colored elements in a grid. The output grid produces red (2) colored elements in a sequence of line. However, the if n < = 3, then output elements will be within the first row, else if n >3, then the 4^{th} element will be in the second row and second column.

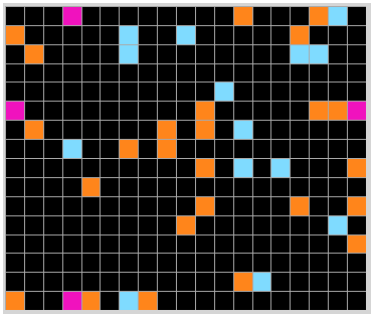
Code Logic:

The solve function parses the json file and runs a for-loop on all the training and testing inputs. We are storing the total number of blue elements of the grid in **count.** We are then using an **if** condition to color the elements of the first row of the output grid red (2) for the count value.

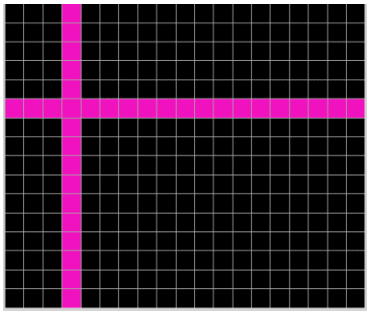
- The function solve() produces the correct output for all the train and test cases.

Task 5: 6cdd2623.json

Input:



Output:



Pattern Logic:

As we can see from the above representation of the grids that the input grid contains randomly colored elements scattered around the grid. For any row or column in the input grid that has the same first and last color, the entire column or row is colored with that element In the output grid.

Code Logic:

In the code, the solve function contains one for-loop to check the columns and one for-loop that checks the rows. The for-loops check weather the first and last element are the same for any column or row. It then colors the entire row or column with that color if the condition satisfies. The final for-loop colors the rest of the grid black.

- The function solve() produces the correct output for all the training and testing cases.

Summary

The tasks given in the ARC Corpus by Chollet have provided us with the opportunity to implement critical thinking and logical reasoning skills. It was ecstatic to create our own algorithms to solve different tasks from the corpus.

The libraries we used for our solutions were:

- **sys:** It's a module in the standard library that we imported to utilize the json file that is passed as a parameter in the command line while executing the python program.
- **json:** We imported the json module to parse the json file that is read by the python program
- **NumPy:** We imported the NumPy library to convert the json grids into a NumPy array. This helped us utilize the NumPy features to work with he grids to solve the task.

Python helps to implement easily readable code and supports large number of libraries. All our solutions were implemented using the libraries mentioned above. The tasks above provided different challenges that were based on patterns of colors in a grid. Each task had different outputs based on the challenge. Our algorithms for every task mainly consisted of **for-loops and while-loops** to create solutions for the challenge.

Contributions

1) **Moiz Meyaji –** I worked on creating the solution for task 2 (25d8a9c8.json) and task 3 (9dfd6313.json). I also collaborated to solve task 4 (794b24be.json) and task 5 (6cdd2623.json) with Waseem. We created and implemented the logic of the solution. I wrote the requirements of my tasks and the code logic of task 4 and 5 along with the summary that concludes our work along with the Python features/libraries that were used.

2) **Waseem Shariff-** I worked on creating the solution for task 1 (6f8cd79b.json). I also collaborated to solve task 4 (794b24be.json) and task 5 (6cdd2623.json) with Moiz. We created and implemented the logic for the solution. I wrote the requirements for my task and the pattern logic for task 4 in the report. Lastly, I changed the READ.md file with all the required information to reflect the repository.