Website: IvLabs **Follow Us:** Facebook, Instagram, YouTube

TechnoSeason 2022

TechnoSeason is the annual recruitment drive for prospective Summer Interns of IvLabs, VNIT, Nagpur.

General Instructions:

- This task consists of two topics: Python and OpenCV.
- The points you'll get will be judged based on your implementation, the logic used etc.
- While you may take help from the internet for the errors that you are facing or understanding any algorithm, directly copying solutions is strictly not allowed.
- Few tasks have some bonus points. This will be awarded if someone finds a unique way to solve the problem or does something out of the box.
- A partially completed task is also acceptable. Points will be awarded according to the extent of completion.
- After completion of tasks, create a ZIP folder with all the files and upload them on this google form **TechnoSeason Submission**.
- The last date of submission is 24th March before 11:59pm . The final decision lies with the event coordinators only and is not subject to change.

Python Task

General Instructions:

- This part consists of 4 questions along with one Bonus part(included in 4th question).
- You can submit either *.ipynb* files (Jupyter Notebooks, Google Colab file *not the link*) or python *.py* files (in a ZIP folder). You are free to use multiple files per project.
- The questions are levelled from easy to hard and have respective points.
- Do not hardcode the inputs, make it user-driven.
- Add comments to your code so that we(and you) can understand your code.

1. Magical Pursuit

Difficulty Level: Easy

Points: 15

Problem Statement:

What If: The story of Harry Potter and the Philosopher's Stone went differently?

Fast forward to the first Christmas at Hogwarts. Hermione left for her Christmas holidays. You find Ron and Harry playing chess, you meet them and tell them about Fluffy and what it's hiding. That very night of Christmas, Harry finds his invisibility cloak and you plan to go see what Fluffy is guarding. You sneakily move through the small door that Fluffy is guarding and reach a cave. Moving into the cave you find that there is an angel standing, who greets you. She gives you the following scenario to solve to find the Philosopher's Stone: only one of you can go through the selected door. She provides you with the whole scenario.

The scenario goes like this:

1) There are three closed doors, labelled #1, #2 and #3. The angel randomly selects one of the three doors and puts the Philosopher's Stone behind it. The other two doors hide nothing.
2) You, who doesn't know where the stone is, select one of the three doors. This door has not opened yet.
3) Now the angel chooses one of the three doors and opens it. The door that she opens

   (a) does not hide the prize, and

   (b) is not the door that you selected.

   There may be one or two such doors. If there are two, she randomly selects one or the other.

4) There are now two closed doors, the one you selected in step 2, and one that you didn't select. You decide whether to keep their original choice, or to switch to the other closed door.
5) You win if the door you selected in step 4 is the same as the one Angel put the philosopher's stone behind in step 1.

Your strategy is given by your choices in steps 2 and 4. Write a program to determine the success rate of your strategy by simulating the scenario 100 times and calculating the percentage of the times you acquire the Philosopher's Stone. Determine the success rate if:
Harry were to choose door #1 in step 2, and always sticks with door #1 in step 4 and,
Ron were to choose door #1 in step 2, and always switches to the other closed door in step 4.

2. SOS game

Difficulty Level: Moderately Hard

Points: 30

Problem Statement:

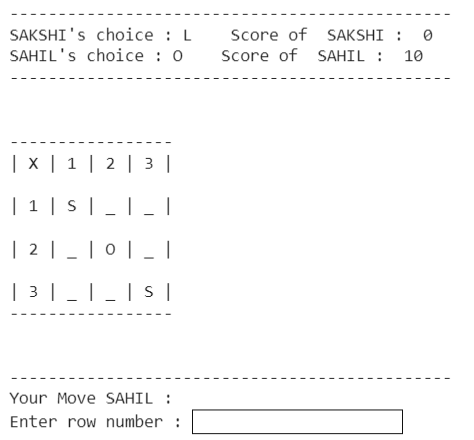
You have to implement a 2 player SOS game in a N×N board using python. This game is a modified form of the standard SOS game. Size of board will be given as input by the player prior to the game

The game will have the flow as:

1. Each player will have to choose a distinct character.(say 1st player- **"L"**, 2nd player- **"O").**
2. At the start of a player's turn, the player chooses either their distinct character or "S" as the character they want to play.
3. They then select a location on the board to place their character on. Goal of both the players is to make the maximum **S_S** combination, where the character in the middle denotes the player's character (viz. for the 1st player making **"SLS"** combination and for the 2nd player making **SOS**).
4. Combinations can be in row, column or in diagonal
5. If any player by chance makes the opponent's combination then a point will be rewarded to the opponent.
6. The game will continue until the complete board is filled up.
7. At last the one with the maximum score/combination wins the game.

```
-----------------------------------------------
SAKSHI's choice : L    Score of  SAKSHI :  0
SAHIL's choice : O    Score of  SAHIL :  0
-----------------------------------------------


-----------------
| X | 1 | 2 | 3 |

| 1 | _ | _ | _ |

| 2 | _ | _ | _ |

| 3 | _ | _ | _ |
-----------------


-----------------------------------------------
Your Move SAKSHI :
Enter row number :
```

```
-----------------------------------------------
SAKSHI's choice : O    Score of  SAKSHI :  0
SAHIL's choice : L    Score of  SAHIL :  0
-----------------------------------------------


-----------------
| X | 1 | 2 | 3 |

| 1 | _ | _ | _ |

| 2 | _ | _ | _ |

| 3 | _ | _ | _ |
-----------------


-----------------------------------------------
Your Move SAKSHI :
Enter row number : 1
Enter column number :
```



...and so on until the board is filled

Note: You need not create the exact replica of the above example. Above given images are just to explain the flow.

3. Single Symbol square

Difficulty Level: Hard

Points: 40

Bonus: 10

Problem Statement:

Given an integer N, your task is to find an NxN layout of X's and O's such that no axis-aligned square (2x2 or larger) within the grid has the same symbol at each of its four corners.

For instance, given N = 5, the following would **not** be a valid output:

O O O X X
X X O O O
X O X O X
O X O O X
X O X X O

because there's a 3x3 square whose four corners are all X's:

. . . . .
. . . . .
X . X . .
. . . . .
X . X . .

That is, if any four cells(elements) of the grid are at equal distances from each other(square), they must not be either all X's or all O's.

Example input

5

Example output

O O O X X
X X O O O
O O X O X
O X O O X
X O X X O

Run time

To receive full credit for this task, your code must run and give the correct output for all values of
N <= 6.

Note- In case your code does not run or give the correct output for N=6, partial credit will be given based on the highest value of N for which it works.

Bonus Question

Note- This is completely optional.

If your code runs correctly until N=6, you will receive full credit for it. However if you wish to score some additional bonus marks, try to write a code that gives the correct output even for N>6.

OpenCV Task

General Instructions:

- This part consists of 3 questions along with bonus parts for the first 2 questions.
- We expect you to explore the concepts of these questions on your own as that is how the questions are designed.
- You will need to explore a bit on the internet to understand the concepts, but you strictly should not copy any code from the internet directly.
- Take care of boundary conditions, e.g. if the input should be a number but the user entered a letter, throw an error stating 'Incorrect Input'. Make use of the 'try except' block for efficient error handling.
- You should submit the code as either .py or .ipynb files (don't submit .c or .html files). You can also submit a Google Colab link, although make sure to give access to everyone before submitting.
- Keep your code clean and well commented. Any assumptions you make, specify in the comments.
- You can refer internet for any and all types of resources but copying the code

1. Paint Tool

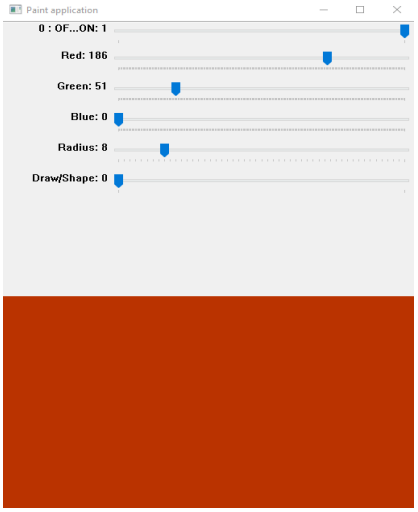
Difficulty Level: Easy

Points: 15

Bonus: 10

In your first task of OpenCV and NumPy, you have to create a basic paint application.
The following are the bare minimum features you will have to add to score full marks.

- Basic Doodling with mouse.
- Colour change with the help of track bars.



Trackbars **Canvas for doodling**

Bonus(optional):

Bonus marks will be rewarded on the basis of extra features which you can add as per your wish.

2. Image Editor

Difficulty Level: Moderately Hard

Points: 30

Bonus: 5

Problem Statement:

Your next task is to implement the numpy and opencv libraries to create a Full fledged photo editor from scratch.
The program must execute functions as given below :

- Crop
- Blending
- Rotation
- Brightness manipulation

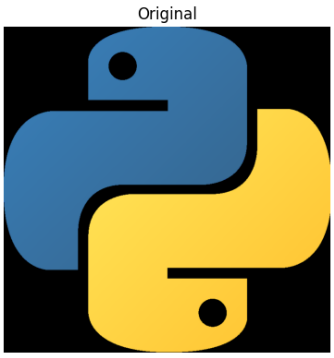
Note:

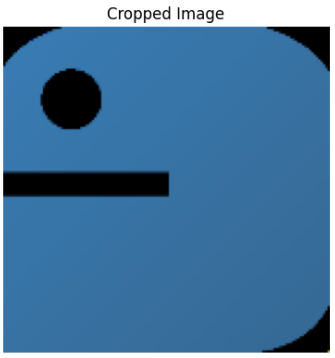
- **Use matplotlib only for reading and displaying the image.**
- **Do not use any in-built functions from OpenCV or any other standard library for any of the given tasks**
- **You have to implement all these from scratch using NumPy.**
- **Compile everything in an organised menu-driven format (GUI Not Needed).**

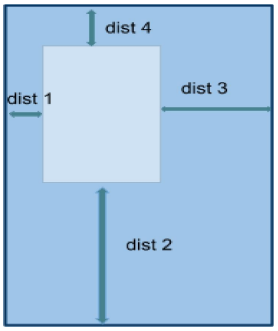
1. **def crop(image, dist1, dist2, dist3, dist 4):**

   To crop the given image and where dist1, dist2, dist3, dist 4 are the distances from left edge, bottom edge, right edge and top edge respectively in pixels.

   Eg.
   dist1 = 0; dist2 = 0; dist3 = 156; dist4 = 156







2. **def blending(img1,img2, weight1,weight2, gamma):**

   Image blending basically adds image addition, but different weights are given to images so that it gives a feeling of blending or transparency. Images are added as per the equation below:
   $\mathbf{g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x) + \gamma}$

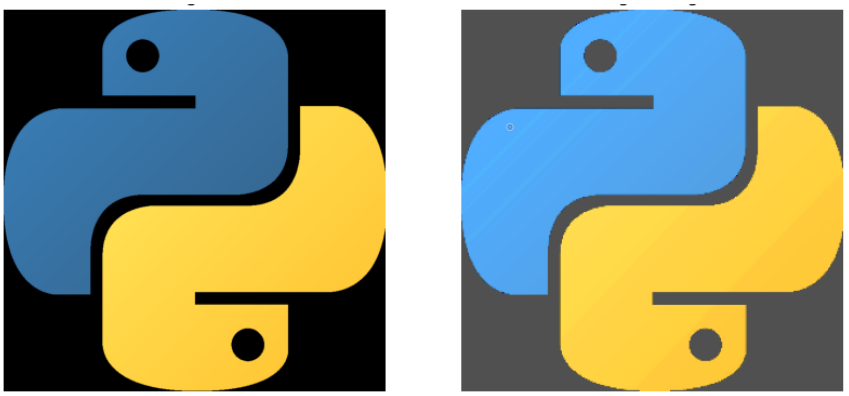
   By varying α from 0→1, you can perform a cool transition between one image to another.

   Eg.
   weight1 = 0.7 ; weight2 = 1 - 0.7 = 0.3

3. **def bright_manipulate(image, brightness%):**

   To adjust the brightness of the image, we change the values of all pixels of the image by a constant.

   Eg.

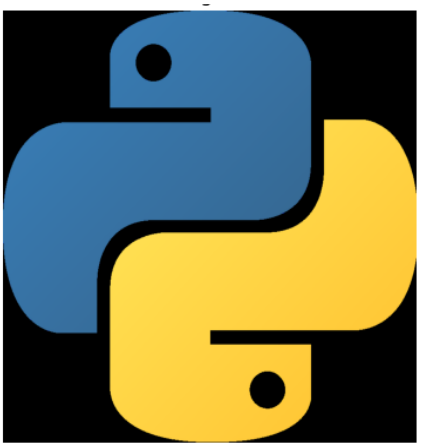


Input Image **Output Image**

4. **def rotate(image, direction, angle):**

   To rotate the image where direction = ‘clockwise’ or ‘anticlockwise’ and angle = ‘90’ or ‘180’. The option for the same should be given to the user. (eg: direction=clockwise , angle= -90)

   Eg.
   direction = clockwise; angle = 90



Input Image

Output Image

Bonus : Define a function rotate_at_angle(image, direction, angle), which rotates the image at an angle in the given direction.

Eg.
direction = anticlockwise; angle = 45

Input image

Output image

3. 3x3 Picture Puzzle

Difficulty Level: Hard

Points: 45

Problem Statement:

You are supposed to make a 3x3 puzzle from a given picture and then shuffle those individual blocks.
You may index the blocks after shuffling as

```
1 2 3        1 4 7
4 5 6  OR    2 5 8
7 8 9        3 6 9
```

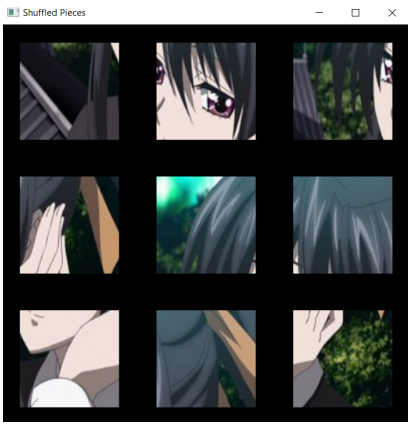
Then you need to stack these blocks one by one on a different board(which can be a white/black background image or any other way which suffices the task).
To avoid confusion you can use the same indexing on the other board as well.
Player is required to give two indexes as inputs:

- The 1st index denotes the block to be moved.
- The 2nd index tells the place on the other board where this block needs to be stacked.

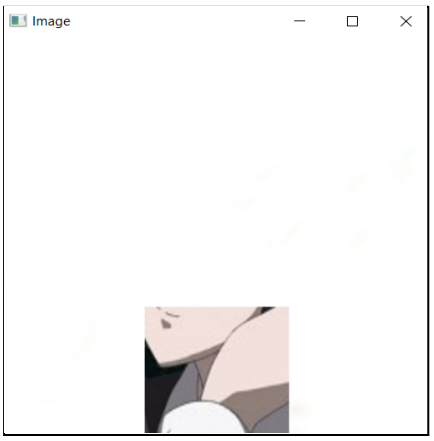
Game will continue until the original image is formed on the other board.
There are no restrictions on movement of shuffled puzzle boxes. One can place any shuffled puzzle box at any position from 1 to 9 on the board.

Original image:

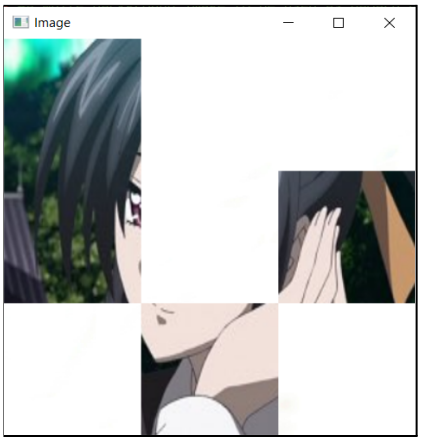
Shuffled pieces:



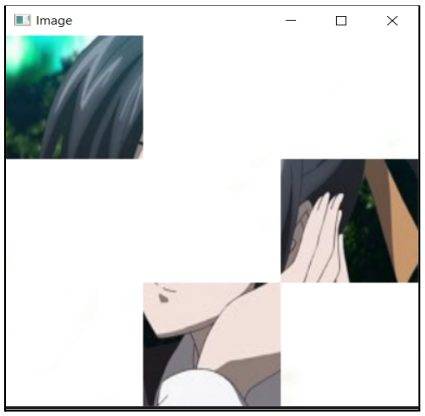
index1=7 index2=8 :



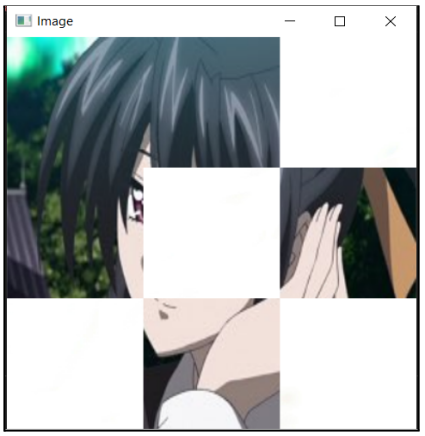
index1=3 index2=4 :



index1=5 index2=1 :



index1=6 index2=2 :



index1=8 index2=3 :

index1=1 index2=7 :