

DIGITAL IMAGE PROCESSING

Lab Manual

[Spring 2018]

|  |  |
| --- | --- |
| Student Name: *Ayaz Qureshi* |  |
| Student Id: *9910* |  |
|  |  |

|  |  |
| --- | --- |
| Submitted To: *Dr. Noman Islam* |  |
| Instructor: *Dr. Noman Islam* |  |

**LIST OF EXPERIMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Date** | **Experiment** |  |
| **1** | \_\_/\_\_/\_ | To setup the environment and familiarize with Python |  |
| **2** | \_\_/\_\_/\_ | To study and implement algorithms in Python |  |
| **3** | \_\_/\_\_/\_ | To study and understand NumPy library |  |
| **4** | \_\_/\_\_/\_ | To study and implement pandas library |  |
| **5** | \_\_/\_\_/\_ | To install OpenCV and study basics of Open CV |  |
| **6** | \_\_/\_\_/\_ | To study and implement basic image processing operations in OpenCV |  |
| **7** | \_\_/\_\_/\_ | To study and implement advanced image processing operations in OpenCV |  |
| **8** | \_\_/\_\_/\_ | To study and implement convolutional neural network using OpenCV |  |

**Lab 1: To setup the environment and familiarize with Python**

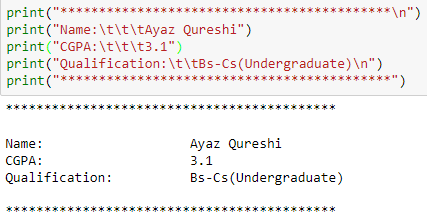
The objective of this lab is to set up the Python environment and get some familiarity with the language.

To set up the environment, follow the steps below:

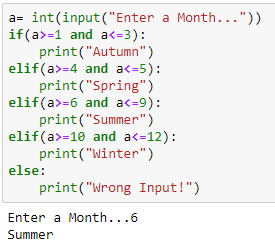
1. Download and install Anaconda. Anaconda is the leading open data science platform powered by Python
2. Download and install PyCharm. PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language.

**Lab Tasks (Solution):**

1. **Write a small program in Python to print your CV.**

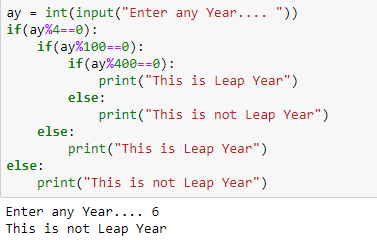


1. **Write a program that takes the month (1…12) as input. Print whether the season is summer, winter, spring or autumn depending upon the input month.**

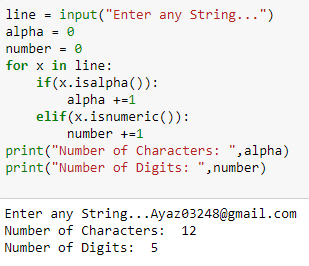


1. **To determine whether a year is a leap year, follow these steps:**
   1. **If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.**
   2. **If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.**
   3. **If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.**
   4. **The year is a leap year (it has 366 days).**
   5. **The year is not a leap year (it has 365 days).**

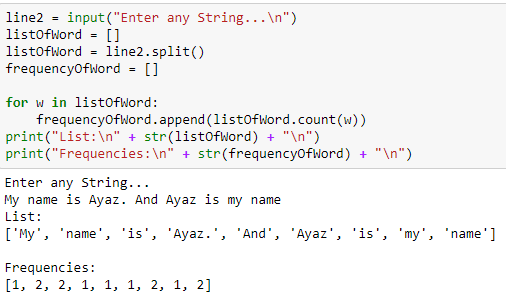
**Write a program to input an year as integer. Using if…else, determines whether the input is a leap year or not.**



1. **Write a program that takes a line as input and finds the number of letters and digits in the input**



1. **Write a program that takes a sentence as input. Compute the frequency of each words and prints them.**



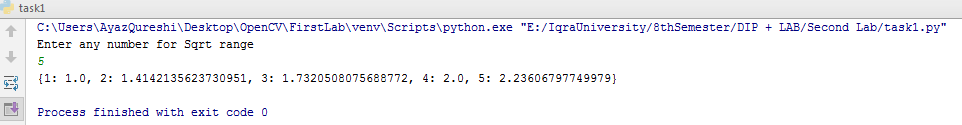
**Lab 2: To study and implement basic algorithms in Python**

In this lab, we will familiarize ourselves with functions, classes and other advanced constructs of python.

**Lab Tasks (Solution):**

1. **Write a program to generate a dictionary that contains (i,sqrt(i)), where *i* is an integer between 1 and n. *n* is a number input by the user.**

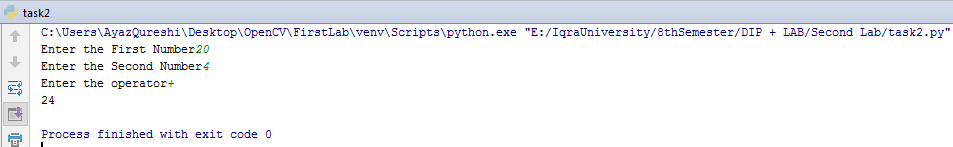
**from** math **import** sqrt  
x = int(input(**'Enter any number for Sqrt range\n'**))  
dic = {}  
**for** i **in** range(1,x+1):  
 dic[i] = sqrt(i)  
print(dic)  
***Output: -***



1. **Write a simple calculator program using functions add, sub, mul and div. The program should accepts two numbers and an operator and calls the corresponding function to perform the operation.**

x = int(input(**'Enter the First Number'**))  
y = int(input(**'Enter the Second Number'**))  
op = input(**'Enter the operator'**)  
  
**def** add(x, y):  
 Answer = x + y  
 **return** Answer  
  
**def** sub(x, y):  
 Answer = x - y  
 **return** Answer  
  
**def** mul(x, y):  
 Answer = x \* y  
 **return** Answer  
  
**def** div(x, y):  
 Answer = x / y  
 **return** Answer  
  
**if** (op==**"+"**):  
  
 print(add(x,y))  
**if** (op==**"-"**):  
  
 print( sub(x,y))  
**if** (op==**"\*"**):  
  
 print(mul(x,y))  
**if** (op==**"/"**):  
  
 print(div(x,y))

**Output: -**



1. **Write a function that generates a list with values that are square of number between 1 and 20.**

list = []  
**def** sqr(x):  
 ans = x\*x  
 **return** ans  
  
**for** i **in** range(1,21):  
 list.append(sqr(i))  
 print(i,**" square is "**,sqr(i))

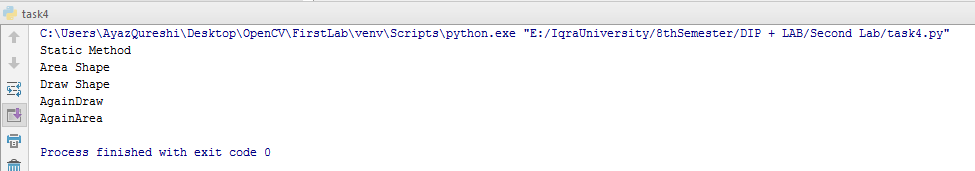
***Output: -***



1. **Define a class named Shape with static method printType. Define methods draw() and area(). Now define two class Rectangle and Triangle. Rectangle has two attributes length and width. The Triangle class has attributes a, b and c. Override the two methods of shape class. Demonstrate the functionality of class by creating its objects.**

**class** Shape():  
 @staticmethod  
 **def** printType():  
 print(**"Static Method"**)  
  
 **def** Draw(self):  
 print(**"Draw Shape"**)  
  
 **def** Area(self):  
 print(**"Area Shape"**)  
  
**class** Rectangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.width =12  
 self.height=10  
  
  
  
**class** triangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.a=2  
 self.b=22  
 self.c=33  
  
 **def** Draw(self):  
 print(**"AgainDraw"**)  
  
 **def** Area(self):  
 print(**"AgainArea"**)  
  
  
S = Shape()  
S.printType()  
S.Area()  
S.Draw()  
T=triangle()  
T.Draw()  
T.Area()  
R = Rectangle()

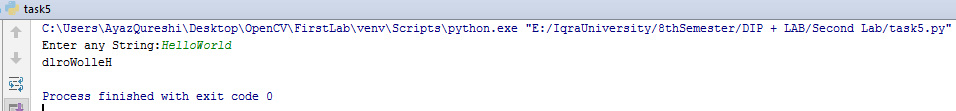
***Output: -***



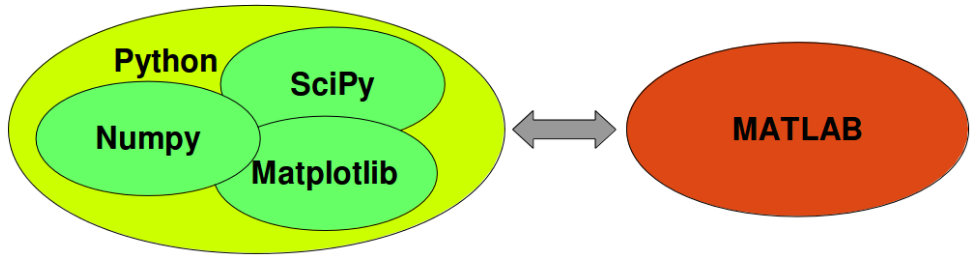
1. **Using recursion, write a program to calculate the reverse of a string.**

task5 = input(**"Enter any String:"**)  
  
**def** rreverse(task5):  
 **if** task5 == **""**:  
 **return ""**;  
 **else**:  
 **return** rreverse(task5[1:]) + task5[0]  
  
print(rreverse(task5))

***Output: -***

**Lab 3: To study and understand numpy library**

In this lab, we are going to explore numpy. NumPy is an acronym for "Numeric Python" or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines.



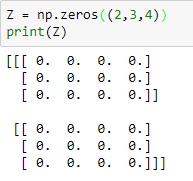
**Lab Task(Solution):**

Open the Python Notebook provided with this lab and perform the tasks.

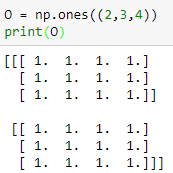
1. **Import the "numpy" library as "np".**



**b. Create an array of shape (2, 3, 4) of zeros.**

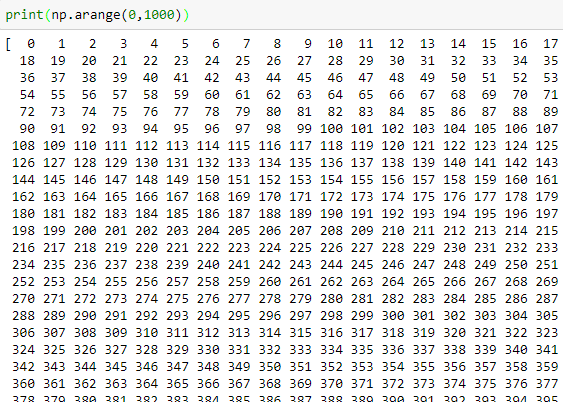


**c. Create an array of shape (2, 3, 4) of ones**



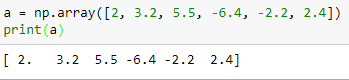
**d. Create an array with values 0 to 999 using the "np.arange" function**

**​**



**e. Create an array from the list [2, 3.2, 5.5, -6.4, -2.2, 2.4] and assign it to the variable "a"**

**​**



**f. Do you know what a[1] will equal? Print it to see**

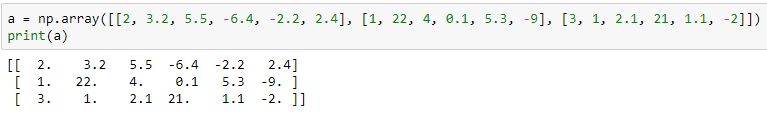


**g. Try printing a[1:4] to see what that equals**

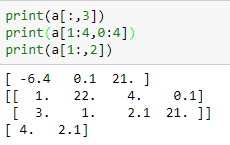
**​**



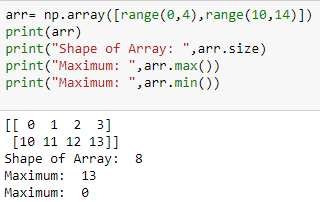
**h. Create a 2-D array from the following list and assign it to the variable "a": [[2, 3.2, 5.5, -6.4, -2.2, 2.4], [1, 22, 4, 0.1, 5.3, -9], [3, 1, 2.1, 21, 1.1, -2]]**



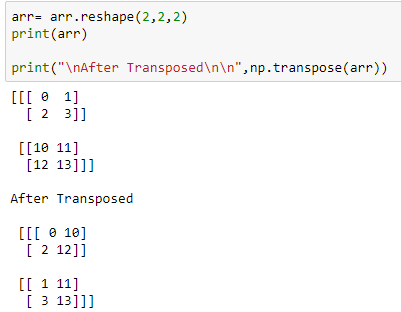
**i. Can you guess what the following slices are equal to? Print them to check your understanding. a[:, 3] a[1:4, 0:4] a[1:, 2]**



**j. Create a 2-D array of shape (2, 4) containing two lists (range(4), range(10, 14)) and assign it to the variable "arr".Print the shape of the array. Print the size of the array. Print the maximum and minimum of the array**



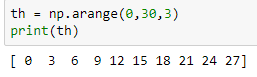
**k. Continue to use the array "arr" as defined above.Print the array re-shaped to (2, 2, 2).Print the array transposed.Print the array flattened to a single dimension. Print the array converted to floats.**



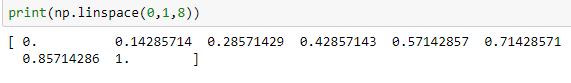
**l. Create an an array counting from 1 to 20 inclusive**



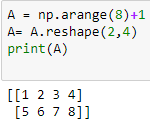
**m. The array of multiples of 3 greater than 0 and less than 30**



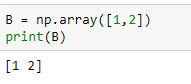
**n. The array of 8 equally spaced floats x where 0 ≤ x ≤ 1**



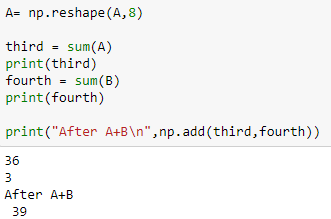
**o. Use np.arange and reshape to create the array A = [[1 2 3 4] [5 6 7 8]]**



**p. Use np.array to create the array B = [1 2]**



**q. Use broadcasting to add B to A to create the final array A + B**



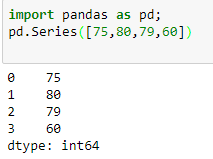
**Lab 4: To study and implement pandas library**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

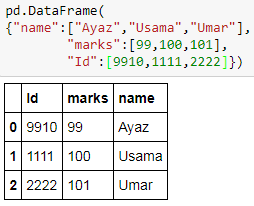
**Lab Task(Solution):**

Open the Python Notebook provided with this lab and perform the tasks.

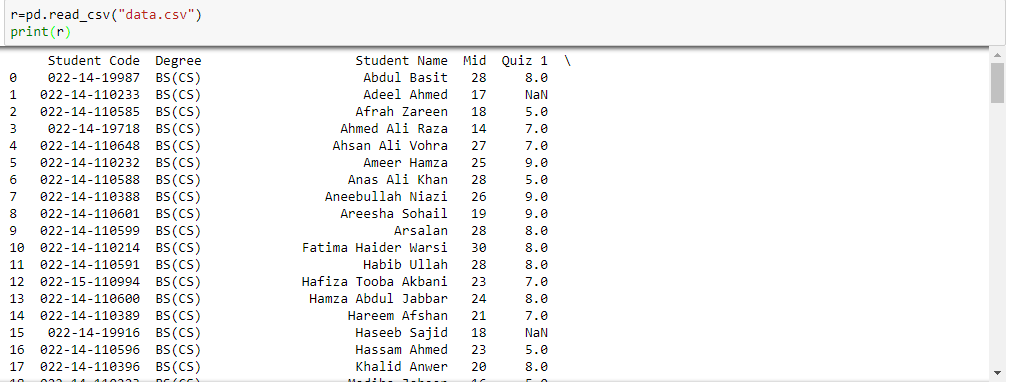
**1. Create a data series with marks of students : 75, 80, 79, 60**



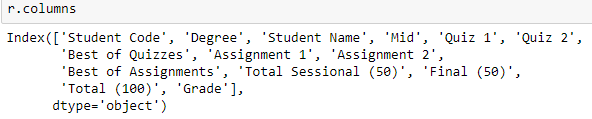
**2. Create a data frame with name of students, id and marks**



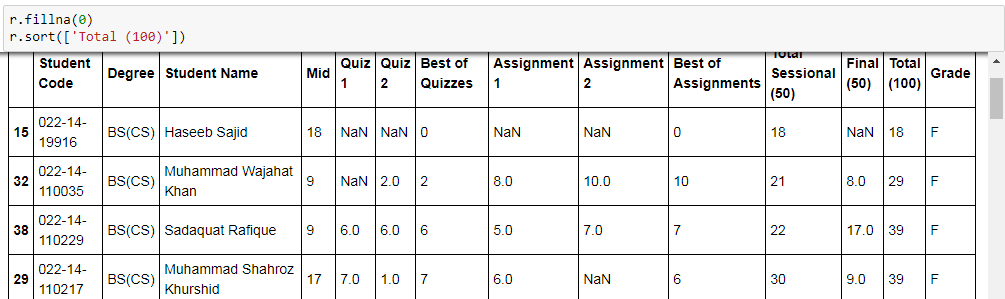
**3. Now read the file 'data.csv' in panda**



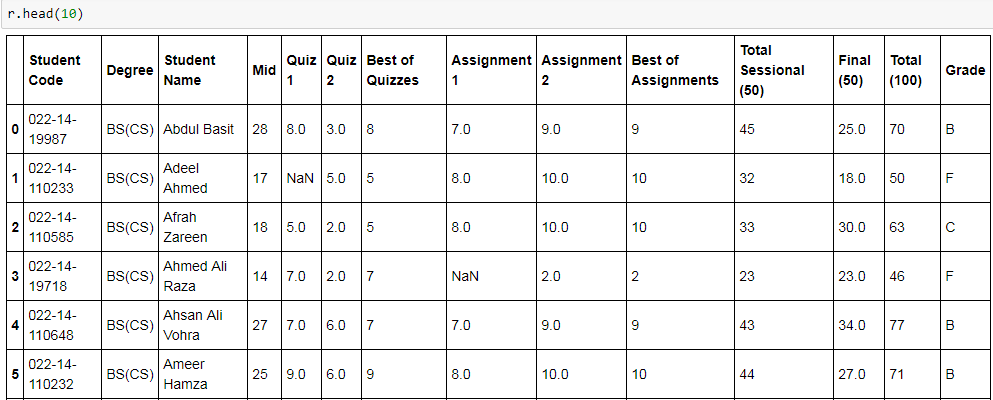
**4. What are the columns in the dataframe?**



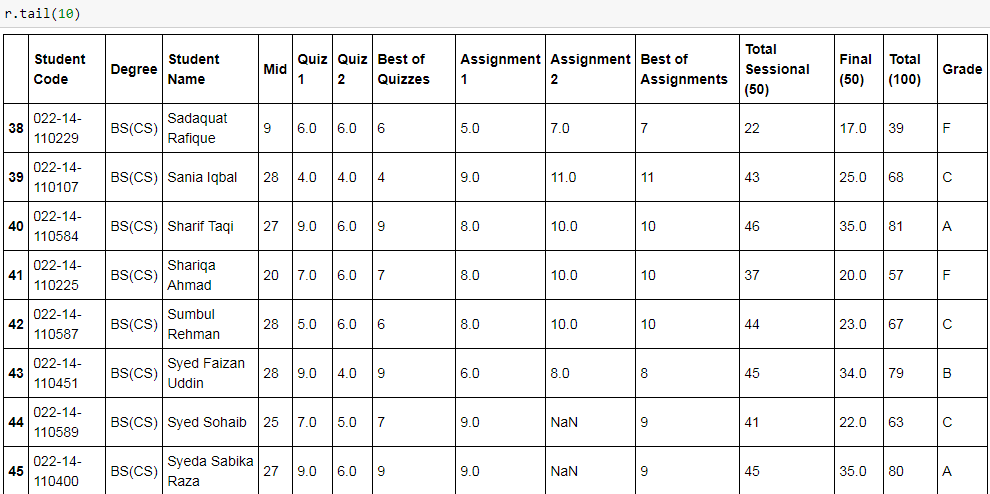
**5. Sort the data based on Marks obtained. Fill all the 'na' cells with 0**



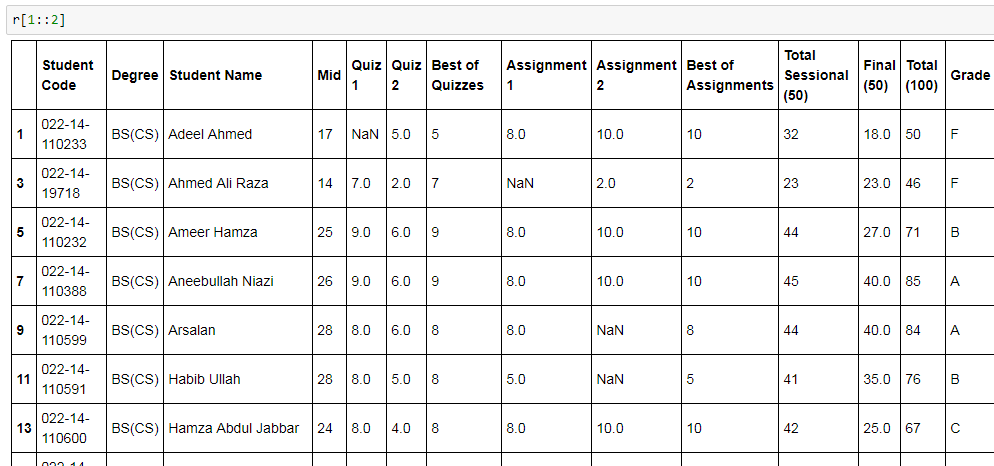
**6. Display the top 10 rows**



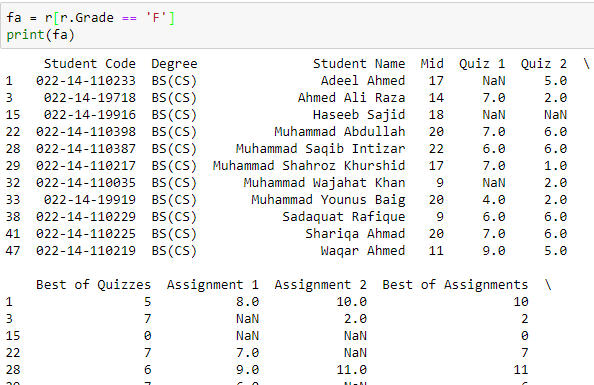
**7. Display the last 10 rows**



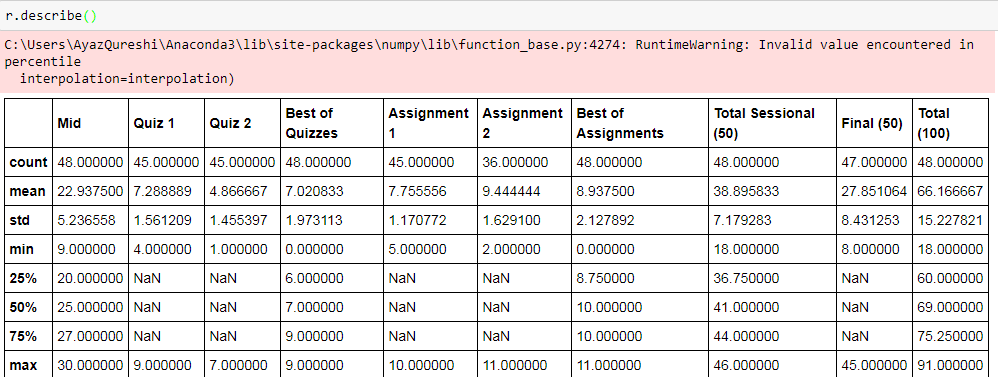
**8. Display only the odd rows**



**9. Display only those students who got failed in examination**

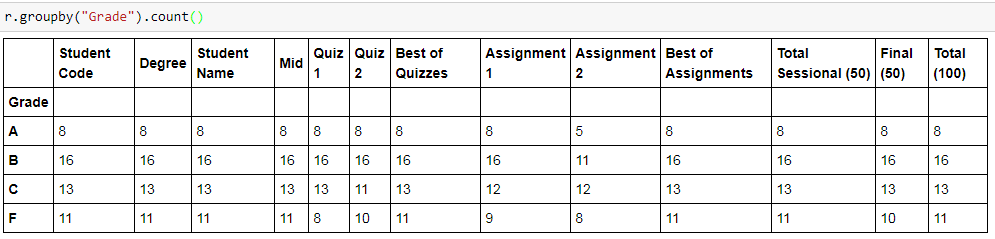


**10. Find out the basic statistical info about data**

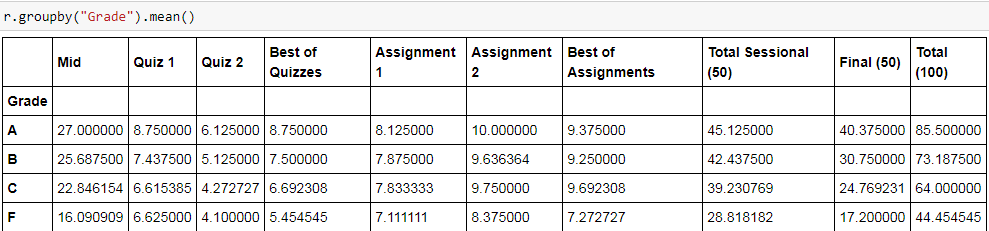


**11. How many students got A, B, C, F?**

**​**



**12. What are the mean scores for students who got A, B, C, F?**



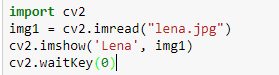
**Lab 5: To install OpenCV and study basics of Open CV**

In this lab, we will study how can images can be loaded, drawing operations can be performed on an image using Open CV library.

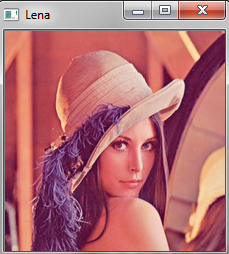
**Lab Tasks(Solution):**

1. **Consider the following image of model Lena. Load the image using Open CV and show on screen**





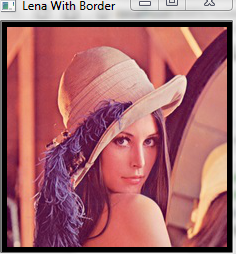
***Output: -***



1. **Create a border around the image**



***Output: -***

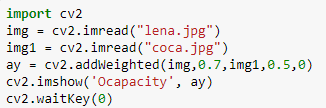


1. **Create a copy of the face and paste it. to the top right position**

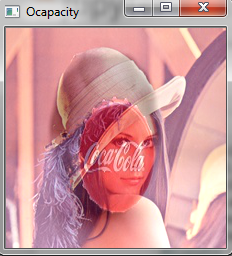


1. **Consider the following Pepsi logo. Blend it over the Lena’s image**

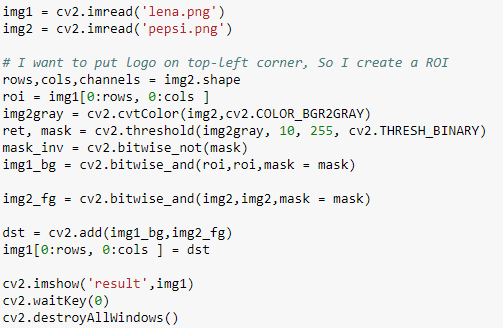




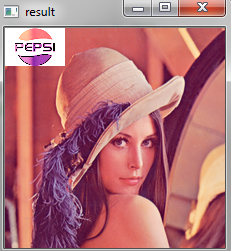
***Output: -***



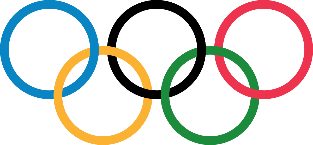
1. **Using bitwise AND, OR and NOT operators, paste the image of Pepsi on Lena’s image. The background of Pepsi logo should not be pasted over, but only ROI will be pasted.**

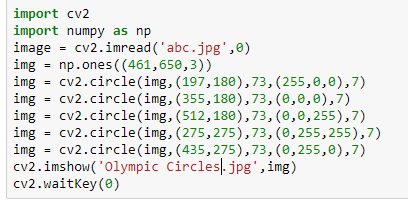


***Output: -***

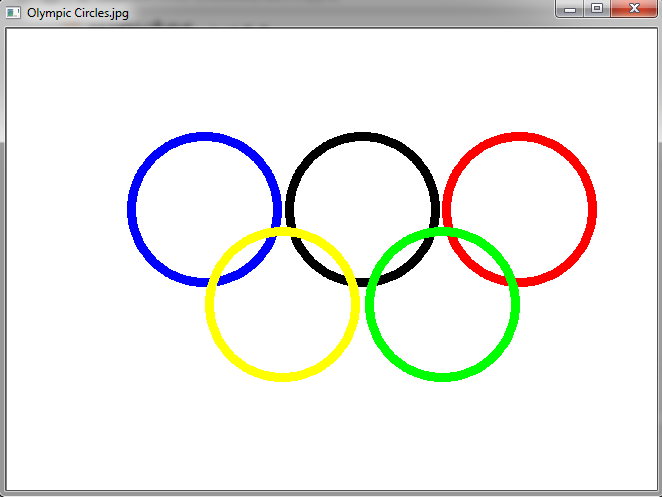


1. **Draw the following Olympic circles using Open CV.**





***Output: -***

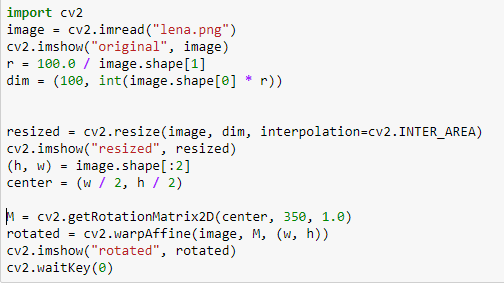


**Lab 6: To study and implement basic image processing operations in OpenCV**

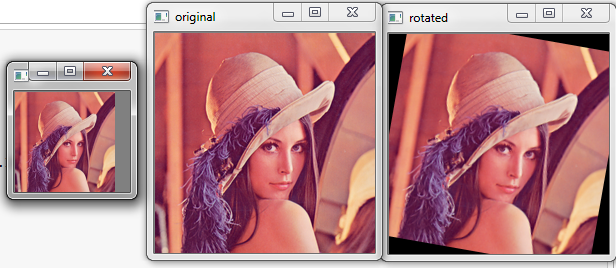
In this lab, we will study basic image processing operations in Python such as transformation, morphological operations, and edge detection.

**Lab Tasks(Solution):**

* **Consider the image provided in previous lab. Transform the image as follows: resize to twice of the original size, translated 30 pixels horizontally and 50 pixels vertically, rotated by 45o clockwise**



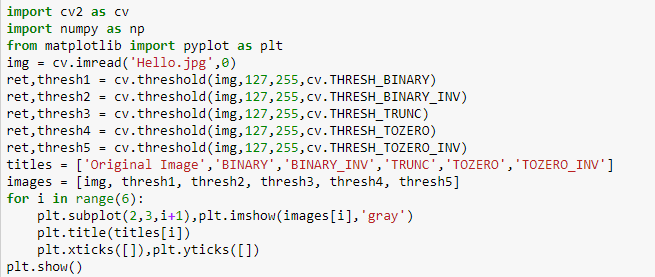
***Output: -***



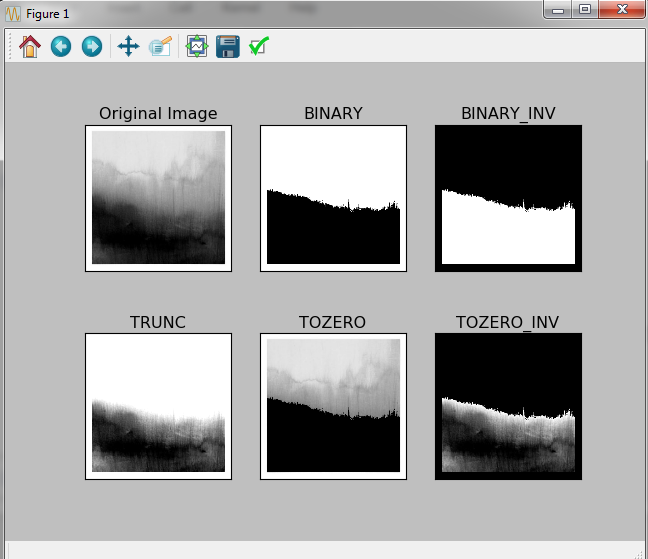
* **Consider the following image:**



* **Perform the following thresholding on the image: cv2.THRESH\_BINARY, cv2.THRESH\_BINARY\_INV, cv2.THRESH\_TRUNC, cv2.THRESH\_TOZERO, cv2.THRESH\_TOZERO\_INV. Provide your narration on the behavior of various types of thresholding.**

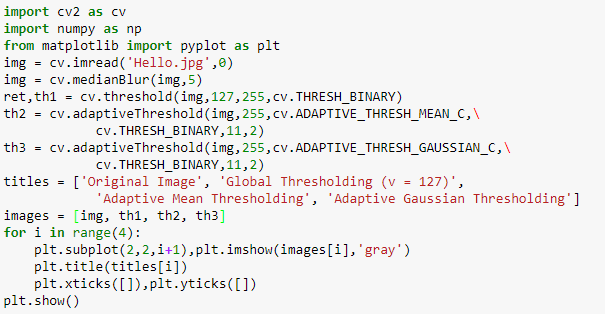


***Output: -***

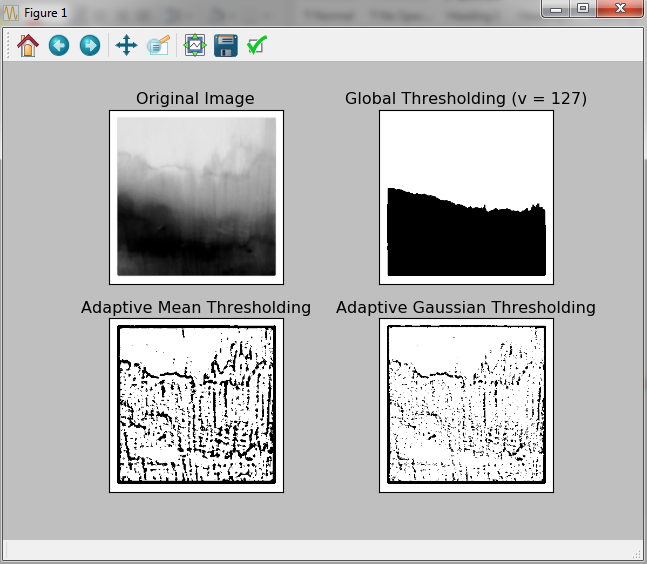


* **Now apply adaptive thresholding and Otsu binarization. Do you see any improvement in the result? Explain your answer with proper reason.**

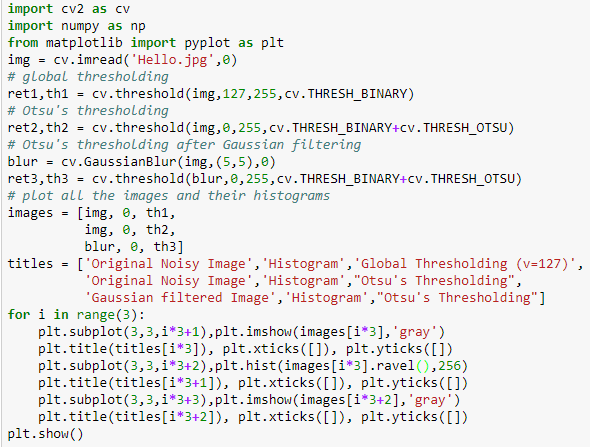
***Adaptive Thresholding***



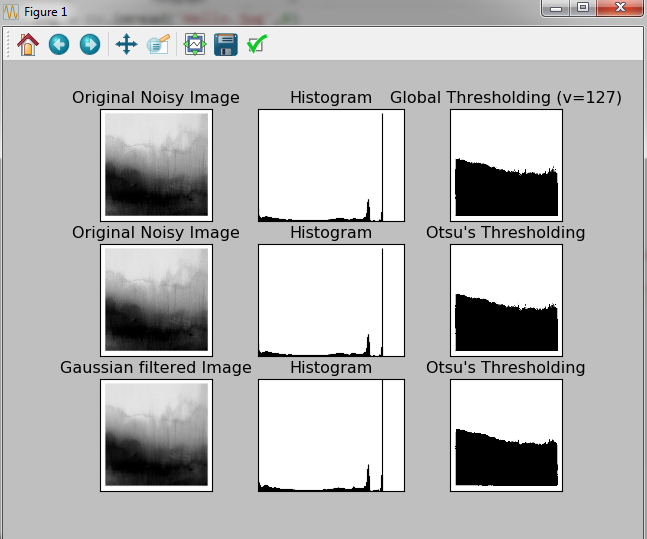
***Output: -***



***Now, Otsu binarization***



***Output: -***



***Reason which one is the Best?***

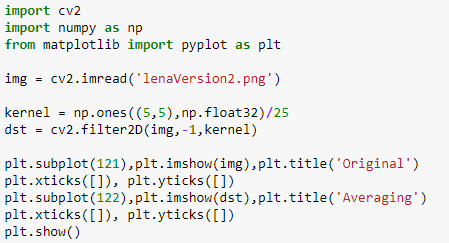
In adaptive, Images breaks into segmentation and apply thresholding on each segment separately, it’s better than global threshold because global threshold not be good in all condition where image has different lighting conditions in different areas. So that why adaptive is more better than global if we talking about **Otsu thresholding** then in global, we use arbitrary value for threshold value. So, we know a value we selected is good or not? Answer is trial and error method. But consider **bimodal image**. **Otsu Thresholding** it automatically calculates a threshold value from image histogram for a bimodal image.

* **Consider the following image:**

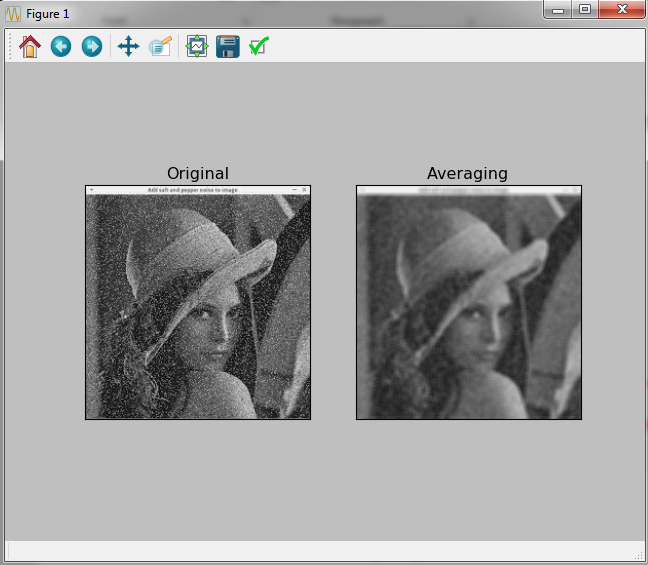


* **Apply various filters such as averaging, Gaussian and median. Which one gives better result? Explain your answer with proper reason**

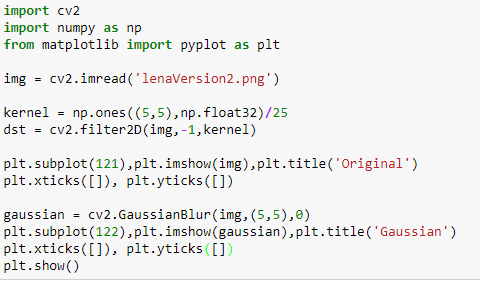
***Averaging Filters***



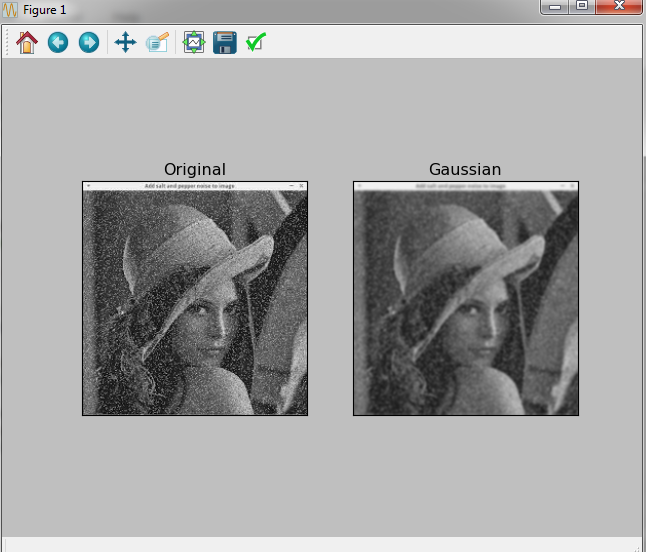
***Output: -***



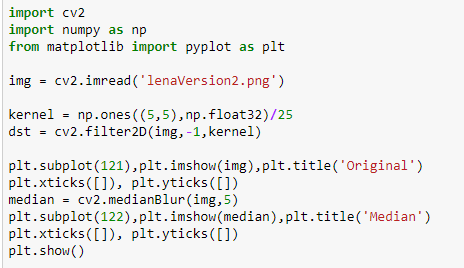
***Gaussian Filter: -***



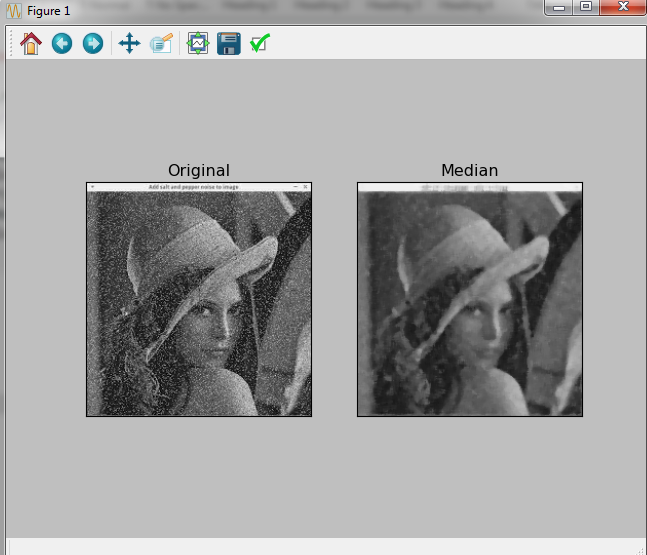
***Output: -***



***Median Filers: -***



***Output: -***



***Which one Is better?***

**Averaging** is done by convolving the image with normalized box filter. It simply takes the average of all the pixels under kernel area and replaces the central element with this average. This is done by the function **cv2.blur()** or **cv2.boxFilter()**.

In ***Gaussian******Filter****,* is highly effective in removing Gaussian noise from the image.

*In* ***Median Filter****,* this is highly effective in removing salt-and-pepper noise.

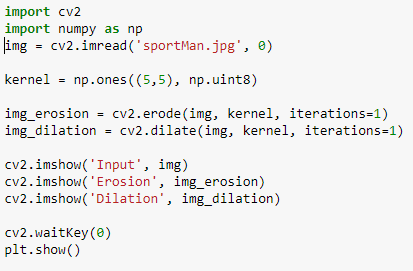
*Gaussian Filter is better result because of remove gaussian and more closed to the Original image.*

* **Consider the following image:**

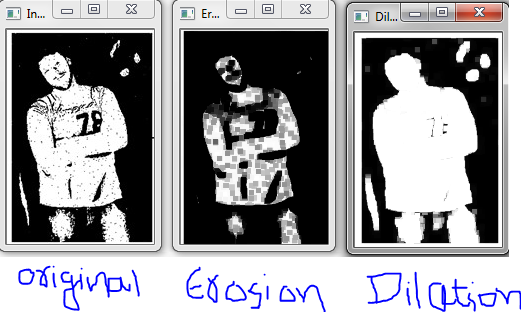


* **Applying erosion, dilation, opening and closing on the image. Explain the behavior of the operators**

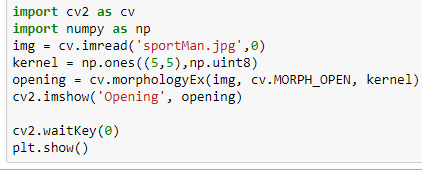
***Erosion and Dilation: -***



***Output: -***



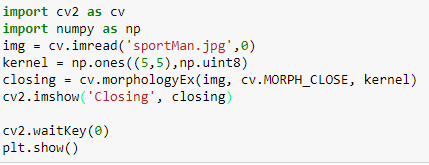
***Opening Filters: -***



***Output: -***



***Closing Filter: -***



***Output: -***



***Erosion: -***  is just like soil erosion only, it erodes away the boundaries of foreground object. so, what it does? The kernel slides through the image (as in 2D convolution).

***Dilation:-*** It is just opposite of erosion. Here, a pixel element is '1' if atleast one pixel under the kernel is '1'. So, it increases the white region in the image or size of foreground object increases.

***Opening:*** - Opening is just another name of **erosion followed by dilation**. It is useful in removing noise.

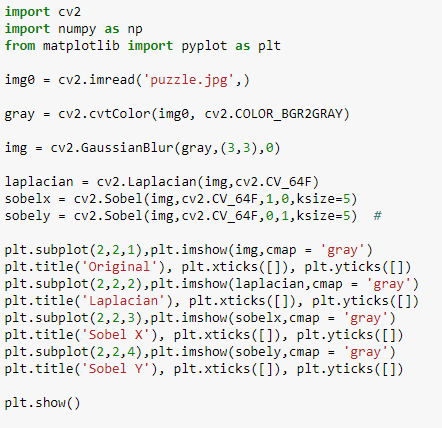
***Closing:*** - Closing is reverse of Opening, **Dilation followed by Erosion**. It is useful in closing small holes inside the foreground objects, or small black points on the object.

* **Consider the following image:**

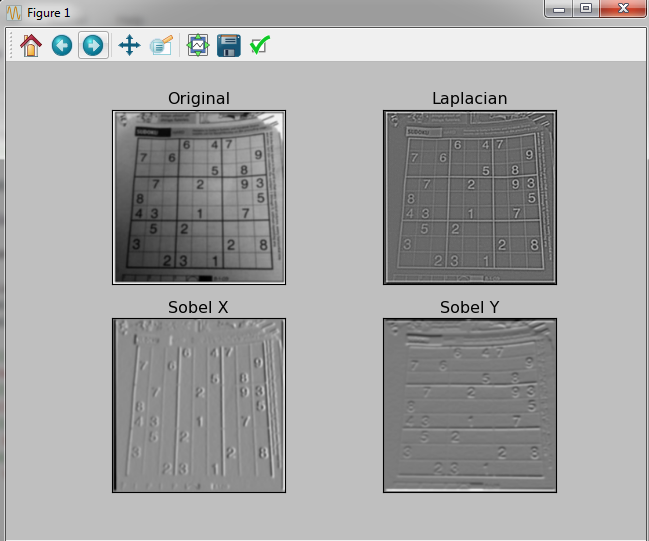


* **Apply different derivative operators such as Sobel, Laplacian and Canny edge detection on the image.**

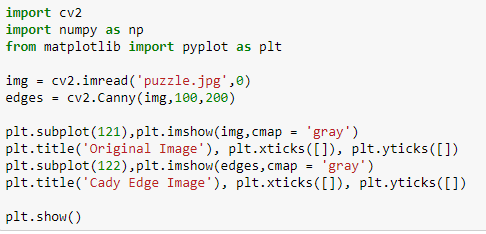
***Laplacian And Sobel: -***



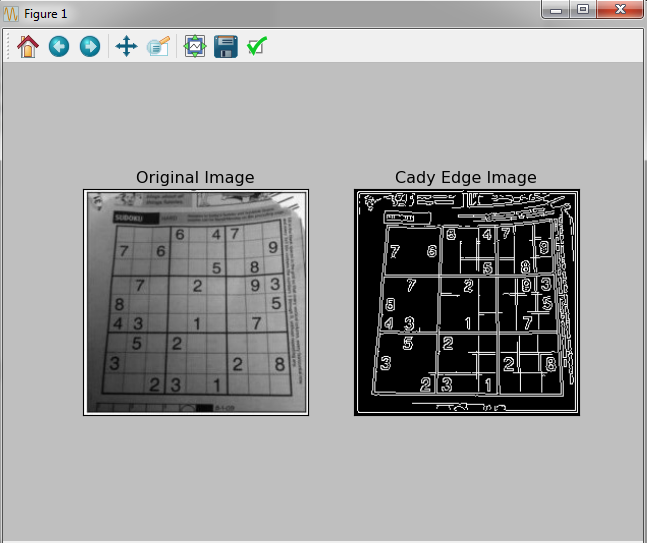
***Output: -***



***Candy Edge Detection: -***



**Output: -**



**Lab 7: To study and implement advanced image processing operations in OpenCV**

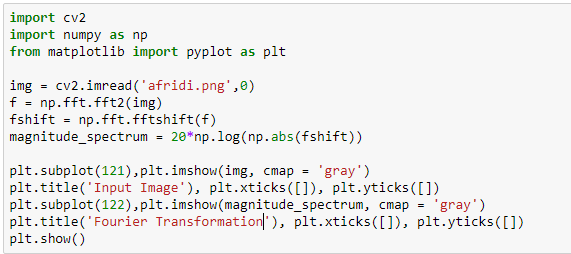
In this lab, we will study advanced image processing operations in OpenCV such as finding template in an image,

Lab Tasks:

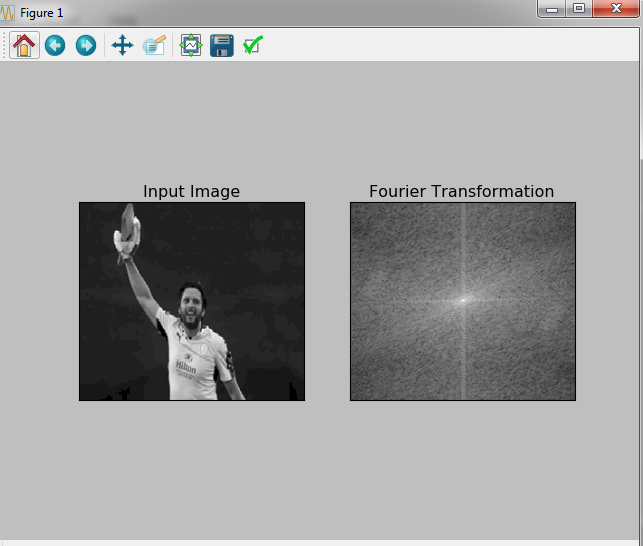
* **Consider the following image:**



* **Calculate the Fourier transform of the image and plot**



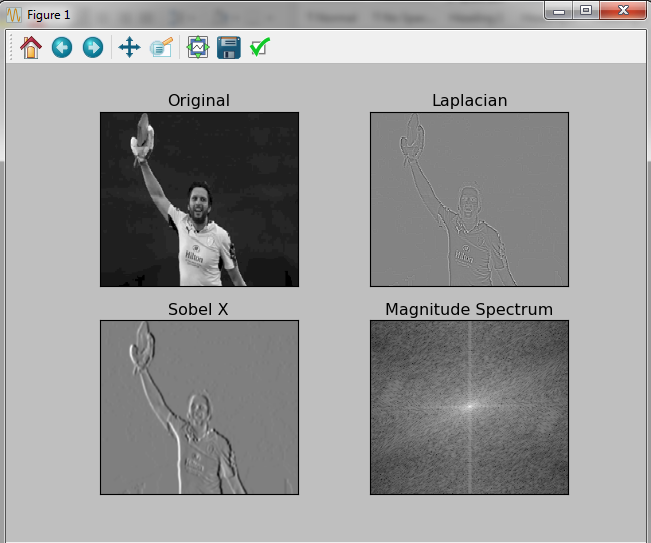
***Output: -***



* Now apply Laplacian, Gaussian and Sobel operator on the image. Now calculate the Fourier transform of the processed image and plot it. What behavior do you see?

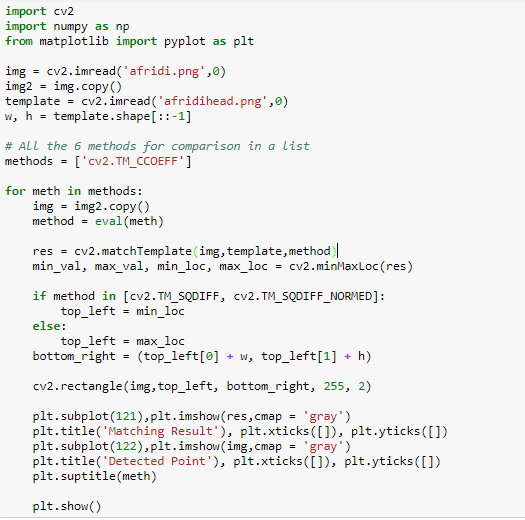


***Output: -***

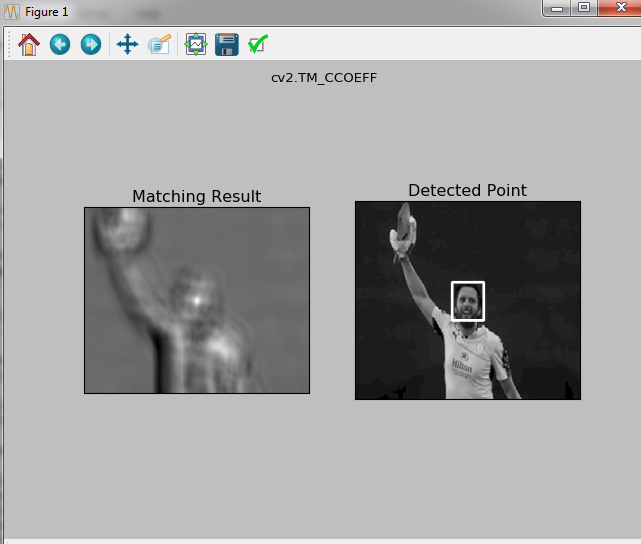


* **Find the position of head of the player in the image using the following template:**





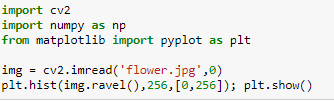
***Output: -***



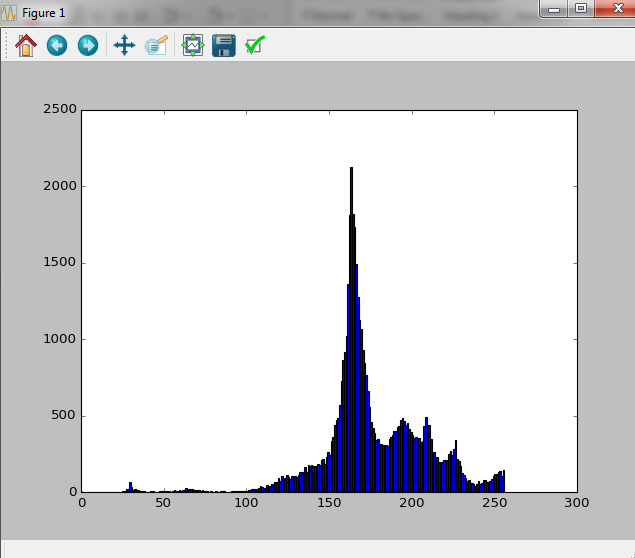
* **Consider the following image:**



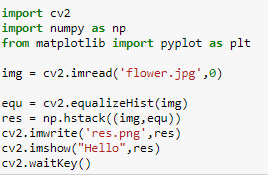
* **Find the histogram of the image**



***Output: -***



* **Apply histogram equalization on the input image. Explain the behavior of the equalization operation.**



***Output: -***



**Lab8: To study and implement machine learning algorithm using OpenCV**

In this lab, we will study various machine learning algorithms of OpenCV.

**Lab Tasks:**

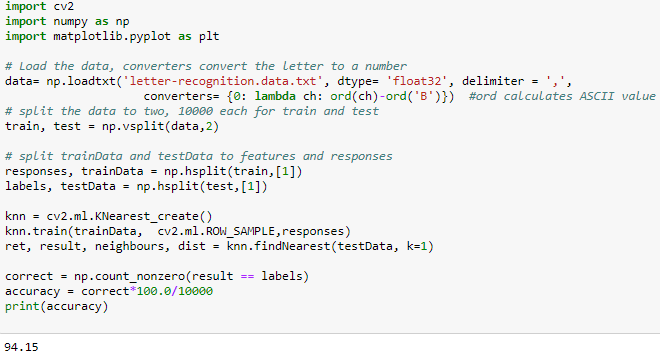
1. **Our goal is to build an application which can read the handwritten digits. For this we need some train\_data and test\_data. OpenCV comes with an image digits.png (in the folder opencv/samples/python2/data/) which has 5000 handwritten digits (500 for each digit). Each digit is a 20x20 image. So our first step is to split this image into 5000 different digits. For each digit, we flatten it into a single row with 400 pixels. That is our feature set, i.e. intensity values of all pixels. It is the simplest feature set we can create. We use first 250 samples of each digit as train\_data, and next 250 samples as test\_data.**

**import** numpy **as** np  
**import** cv2  
  
gray = cv2.imread(**'digits.png'**,0)  
  
*# Now we split the image to 5000 cells, each 20x20 size*cells = [np.hsplit(row,100) **for** row **in** np.vsplit(gray,50)]  
  
*# Make it into a Numpy array. It size will be (50,100,20,20)*x = np.array(cells)  
  
*# Now we prepare train\_data and test\_data.*train = x[:,:50].reshape(-1,400).astype(np.float32) *# Size = (2500,400)*test = x[:,50:100].reshape(-1,400).astype(np.float32) *# Size = (2500,400)  
  
# Create labels for train and test data*k = np.arange(10)  
train\_labels = np.repeat(k,250)[:,np.newaxis]  
test\_labels = train\_labels.copy()  
  
knn = cv2.ml.KNearest\_create()  
knn.train(train, cv2.ml.ROW\_SAMPLE, train\_labels)  
ret,result,neighbours,dist = knn.findNearest(test,k=5)  
  
*# Now we check the accuracy of classification  
# For that, compare the result with test\_labels and check which are wrong*matches = result==test\_labels  
correct = np.count\_nonzero(matches)  
accuracy = correct\*100.0/result.size  
print(accuracy)



1. **Next we will do the same for English alphabets, but there is a slight change in data and feature set. Here, instead of images, OpenCV comes with a data file, letter-recognition.data in opencv/samples/cpp/ folder. If you open it, you will see 20000 lines which may, on first sight, look like garbage. Actually, in each row, first column is an alphabet which is our label. Next 16 numbers following it are its different features. These features are obtained from UCI Machine Learning Repository. There are 20000 samples available, so we take first 10000 data as training samples and remaining 10000 as test samples. We should change the alphabets to ascii characters because we can’t work with alphabets directly.**

**import** cv2  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
  
*# Load the data, converters convert the letter to a number*data= np.loadtxt(**'letter-recognition.data.txt'**, dtype= **'float32'**, delimiter = **','**,  
 converters= {0: **lambda** ch: ord(ch)-ord(**'A'**)})  
*# split the data to two, 10000 each for train and test*train, test = np.vsplit(data,2)  
  
*# split trainData and testData to features and responses*responses, trainData = np.hsplit(train,[1])  
labels, testData = np.hsplit(test,[1])  
  
knn = cv2.ml.KNearest\_create()  
knn.train(trainData, cv2.ml.ROW\_SAMPLE,responses)  
ret, result, neighbours, dist = knn.findNearest(testData, k=5)  
  
correct = np.count\_nonzero(result == labels)  
accuracy = correct\*100.0/10000  
print(accuracy)



**To study and implement convolutional neural network using OpenCV**

This lab introduces convolutional neural networks, also known as convnets, a type of deep-learning model almost universally used in computer vision applications. You’ll learn to apply convnets to image-classification problems—in particular those involving small training datasets, which are the most common use case if you aren’t a large tech company

**Lab Tasks:**

1. **Using MNIST dataset, train a convolutional network for handwritten digits**
2. **Consider the following images. Using data augmentation, generate 100 images.**

1. **Using pre-trained VGG model and modifying the base layers, train the modified model with the images of previous example. Use the model to classify the cats and dogs.**

**from** keras.preprocessing **import** image  
**import** matplotlib.pyplot **as** plt  
**from** keras.preprocessing.image **import** ImageDataGenerator  
  
img = image.load\_img(**'dog.png'**, target\_size=(150, 150))  
x = image.img\_to\_array(img)  
x = x.reshape((1,) + x.shape)  
i = 0  
  
datagen = ImageDataGenerator(rotation\_range=40,width\_shift\_range=0.2,height\_shift\_range=0.2,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True,fill\_mode=**'nearest'**)  
  
**for** batch **in** datagen.flow(x, batch\_size=1):  
 plt.figure(i)  
 imgplot = plt.imshow(image.array\_to\_img(batch[0]))  
 i += 1  
 **if** i % 4 == 0:  
 plt.show()  
 **break**

***Output: -***

