

DIGITAL IMAGE PROCESSING

Lab Manual

[Fall/ spring 2018]

|  |  |
| --- | --- |
| Student Name: M.HUNAIN |  |
| Student Id: 10222 |  |
|  |  |

|  |  |
| --- | --- |
| Prepared By: *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 machine learning algorithm using OpenCV  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:**

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

print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**)  
print(**"Name:\t\t\t M.HUNAIN"**)  
print(**"CGPA:\t\t\t 3.45"**)  
print(**"Qualification:\t\t Bscs\n"**)  
print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: M.HUNAIN

CGPA: 3.45

Qualification: Bscs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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.

m= int(input(**"Enter a Month..."**))  
**if**(m>=1 **and** m<=3):  
 print(**"Autumn"**)  
**elif**(m>=4 **and** m<=5):  
 print(**"Spring"**)  
**elif**(m>=6 **and** m<=9):  
 print(**"Summer"**)  
**elif**(m>=10 **and** m<=12):  
 print(**"Winter"**)  
**else**:  
 print(**"Wrong Input!"**)

Enter a Month...4

Spring

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.

ly = int(input(**"Enter any Year.... "**))  
**if**(ly%4==0):  
 **if**(ly%100==0):  
 **if**(ly%400==0):  
 print(**"This is Leap Year"**)  
 **else**:  
 print(**"This is not Leap Year"**)  
 **else**:  
 print(**"This is Leap Year"**)  
**else**:  
 print(**"This is not Leap Year"**)

Enter any Year.... 6

This is not Leap Year

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

line = input(**"Enter any String..."**)  
alpha = 0  
number = 0  
**for** x **in** line:  
 **if**(x.isalpha()):  
 alpha +=1  
 **elif**(x.isnumeric()):  
 number +=1  
print(**"Number of Characters: "**,alpha)  
print(**"Number of Digits: "**,number)

Enter any String...M.HUNAIN 10388

Number of Characters: 11

Number of Digits: 5

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

line2 = input(**"Enter any String..."**)  
listOfWord = []  
listOfWord = line2.split()  
frequencyOfWord = []  
  
**for** w **in** listOfWord:  
 frequencyOfWord.append(listOfWord.count(w))  
print(**"List:\n"** + str(listOfWord) + **"\n"**)  
print(**"Frequencies:\n"** + str(frequencyOfWord) + **"\n"**)

Enter any String...M.HUNAIN 10388

List:

['3.45', 'name', '10388']

Frequencies:

[1, 1, 1]

**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:**

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 a number'**))  
dic = {}  
**for** i **in** range(1,x+1):  
 dic[i] = sqrt(i)  
print(dic)

Enter a number4

{1: 1.0, 2: 1.4142135623730951, 3: 1.7320508075688772, 4: 2.0}

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.

*#Q2 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))

Enter the First Number7

Enter the Second Number4

Enter the operator+

11

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))

1 square is 1

2 square is 4

3 square is 9

4 square is 16

5 square is 25

6 square is 36

7 square is 49

8 square is 64

9 square is 81

10 square is 100

11 square is 121

12 square is 144

13 square is 169

14 square is 196

15 square is 225

16 square is 256

17 square is 289

18 square is 324

19 square is 361

20 square is 400

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"**)  
 **def** area(self):  
 print(**"area"**)  
  
**class** rectangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.width = 23  
 self.length =34  
  
  
**class** triangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.a = 2  
 self.b = 3  
 self.c = 4  
 **def** draw(self):  
 print(**"draw again2"**)  
 **def** area(self):  
 print(**"area again2"**)  
  
s = Shape()  
s.printType()  
t = triangle()  
r = rectangle()  
s.area()  
s.draw()  
r.draw()  
t.draw()

static method

area

draw

draw

draw again2

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

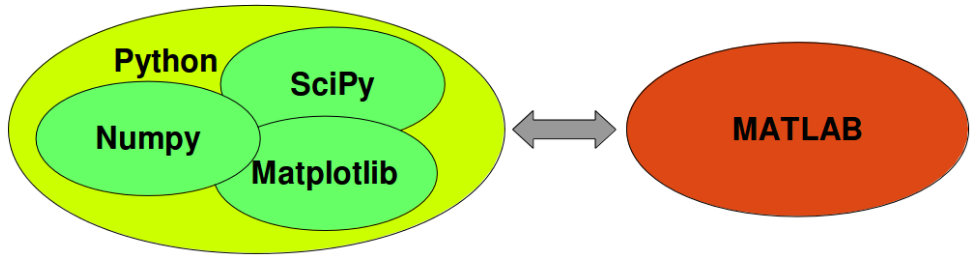
task5 = input(**"Enter any String:"**)  
**def** rreverse(task5):  
 **if** task5 == **""**:  
 **return ""  
 else**:  
 **return** rreverse(task5[1:])+task5[0]  
print(**"Answer: "**,rreverse(task5))

Enter any String:abc

Answer: cb

**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:**

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

a. Import the "numpy" library as "np".

In [ ]:

import numpy as np

​

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

In [ ]:

a = np.zeros((2,3,4))

print(a)

[[[ 0. 0. 0. 0.] [ 0. 0. 0. 0.] [ 0. 0. 0. 0.]] [[ 0. 0. 0. 0.] [ 0. 0. 0. 0.] [ 0. 0. 0. 0.]]]

​

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

In [ ]:

a = np.ones((2,3,4))

print(a)

:

[[[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]]

[[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]]]

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

In [ ]:

a = np.arange(0,1000)

print(a)

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999]

​

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"

In [ ]:

a = np.array([2,3.2,5.5,-6.4,-2.2,2.4])

print(a)

[ 2. 3.2 5.5 -6.4 -2.2 2.4]

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

In [ ]:

print(a[1])

3.2

​

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

In [ ]:

print(a[1:4])

[ 3.2 5.5 -6.4]

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]]

In [ ]:

b = np.array([[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]])

print(b)

[[ 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]

In [ ]:

print(b[:,3])

print(b[1:4,0:4])

print(b[1:,2])

[ -6.4 0.1 21. ]

[[ 1. 22. 4. 0.1]

[ 3. 1. 2.1 21. ]]

[ 4. 2.1]

​

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

In [ ]:

arr = np.array([range(0,4),range(10,14)])

print(arr)

print("Shape: ",arr.shape)

print("size: ",arr.size)

​[[ 0 1 2 3] [10 11 12 13]] Shape: (2, 4) size: 8

In [ ]:

print("Maximum Value: ",np.max(arr))

​Maximum Value: 13

In [ ]:

print("Minimum Value: ",np.min(arr))

​Minimum Value: 0

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.

In [ ]:

arr = arr.reshape((2,2,2))

print(np.transpose(arr))

print(arr.astype(float))

​[[[ 0 10]

[ 2 12]]

[[ 1 11]

[ 3 13]]]

[[[ 0. 1.]

[ 2. 3.]]

[[ 10. 11.]

[ 12. 13.]]]

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

In [ ]:

a = np. arange(20)+1

print(a)

​[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

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

In [ ]:

c = np.arange(9)+1

c = c\*3

print(c)

​[ 3 6 9 12 15 18 21 24 27]

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

In [ ]:

d = np.linspace(0,1,8)

print(d)

​[ 0. 0.14285714 0.28571429 0.42857143 0.57142857 0.71428571

0.85714286 1. ]

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

In [ ]:

a = np.arange(8)+1

A = a.reshape(2,4)

print(A)

​[[1 2 3 4] [5 6 7 8]]

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

In [ ]:

B = np.array([[1],[2]])

print(B)

​[[1] [2]]

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

In [ ]:

a = np.arange(8)+1

A = a.reshape(2,4)

print(A)

​[[1 2 3 4] [5 6 7 8]]

In [ ]:

print(np.add([A],[B]))

​[[[ 2 3 4 5]

[ 7 8 9 10]]]

In [ ]:

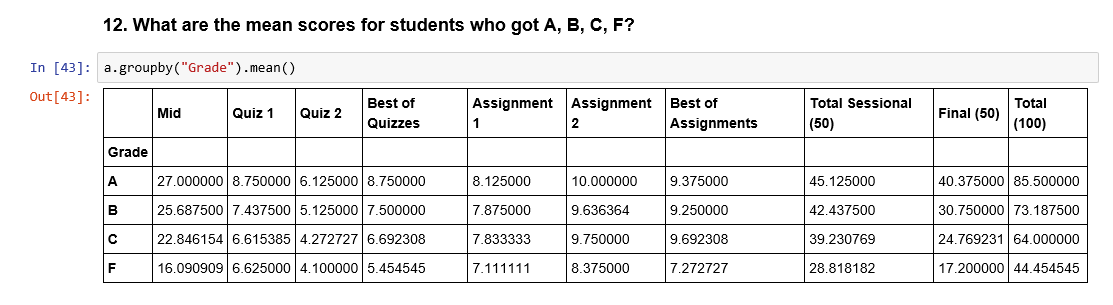
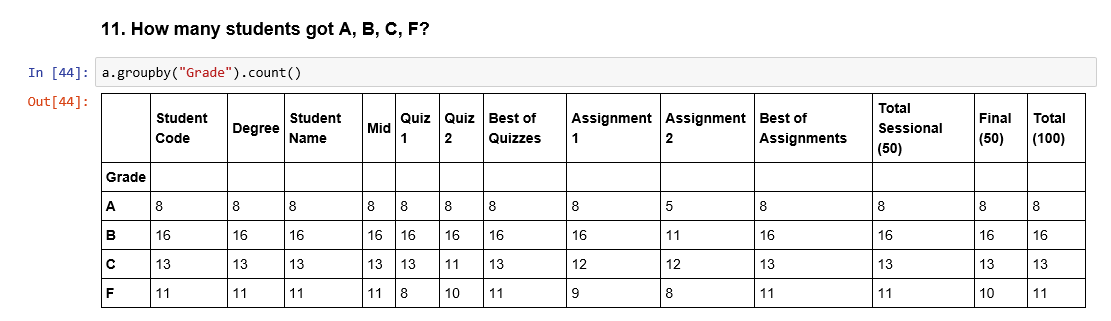
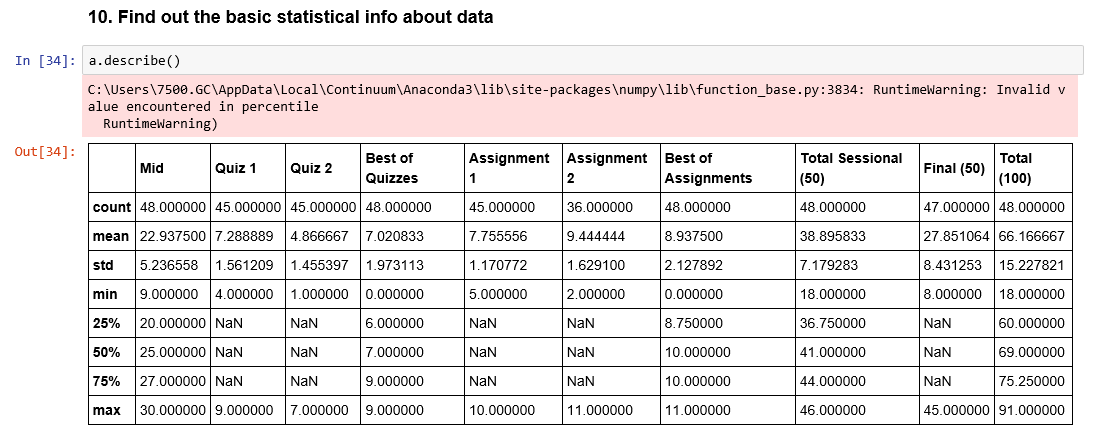
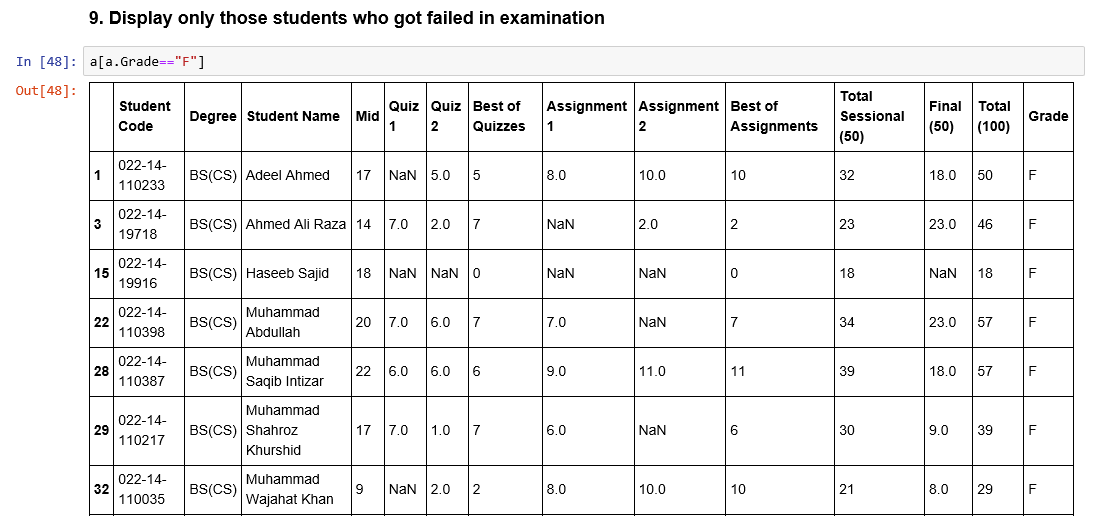
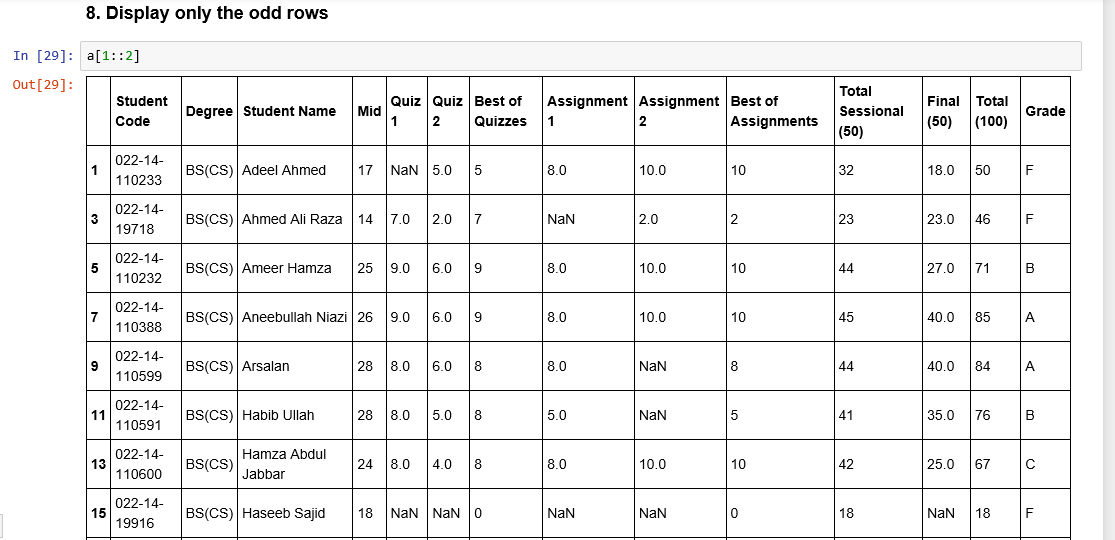
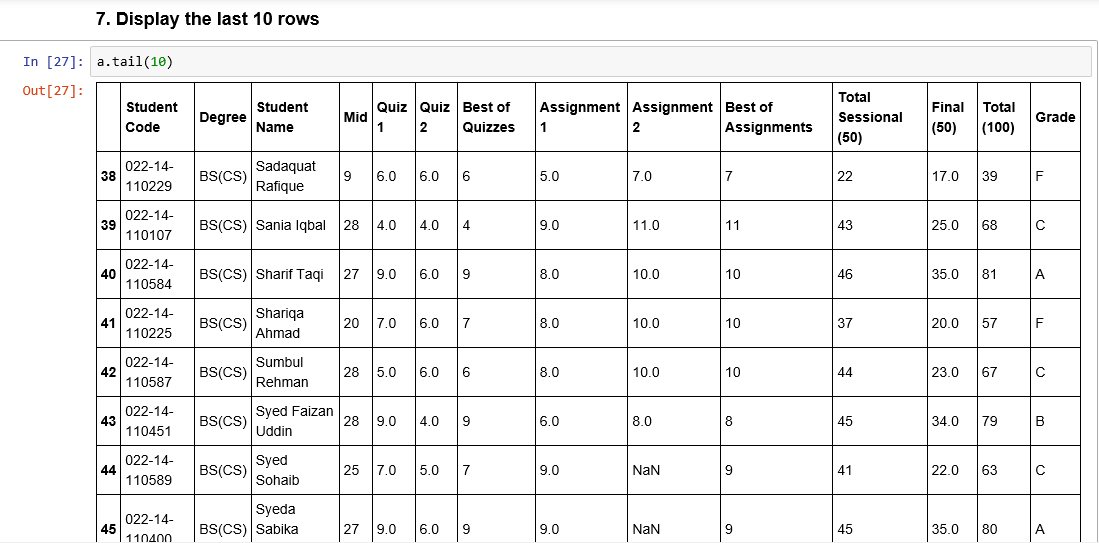
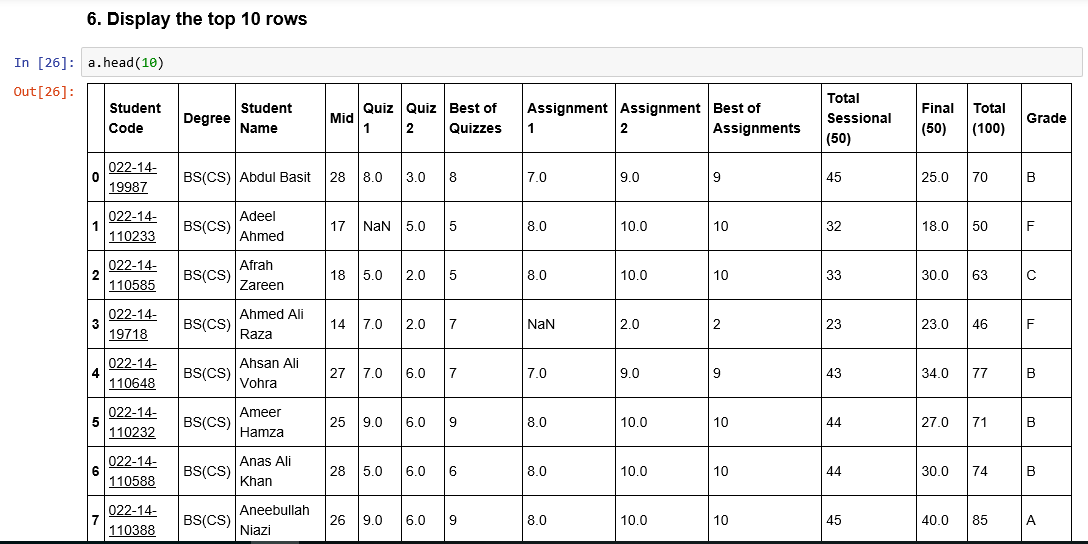
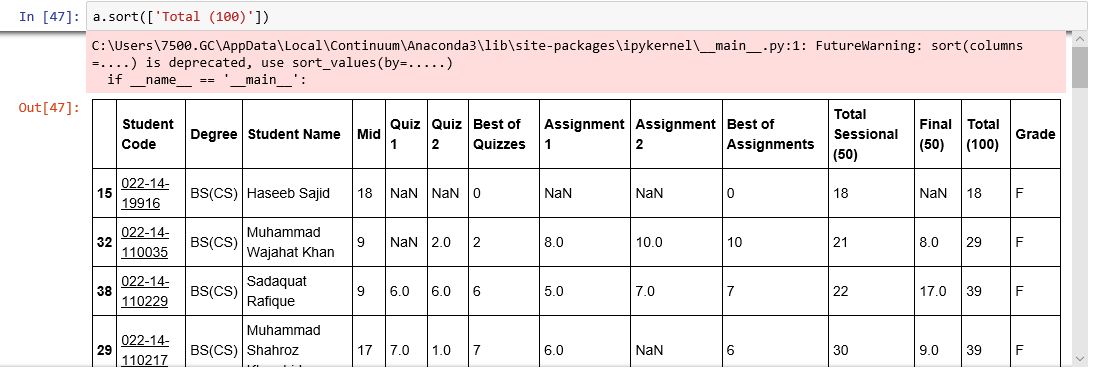
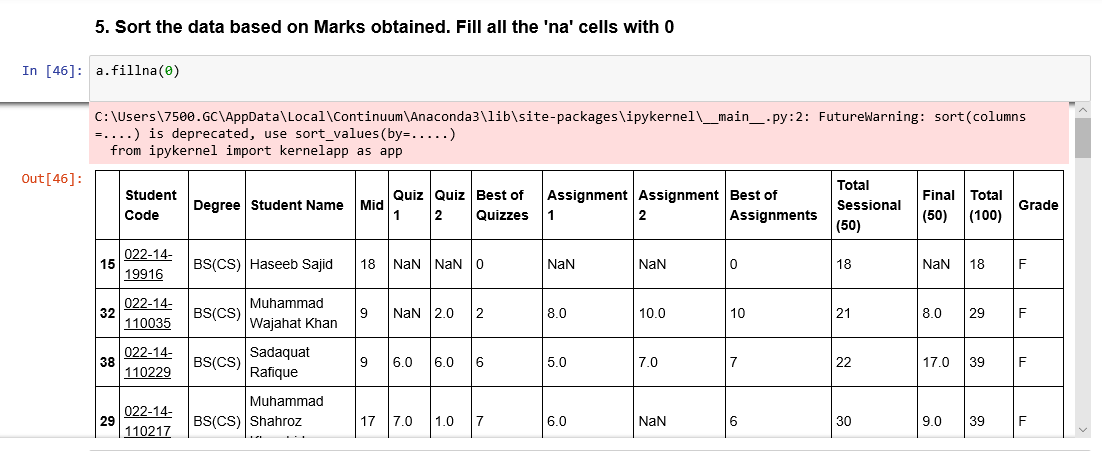
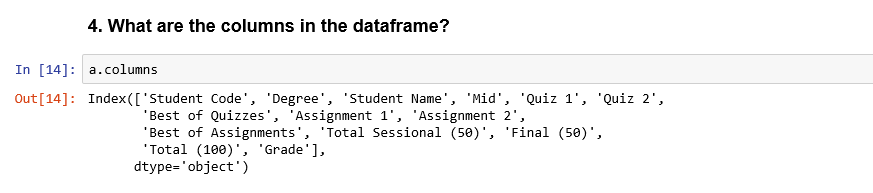
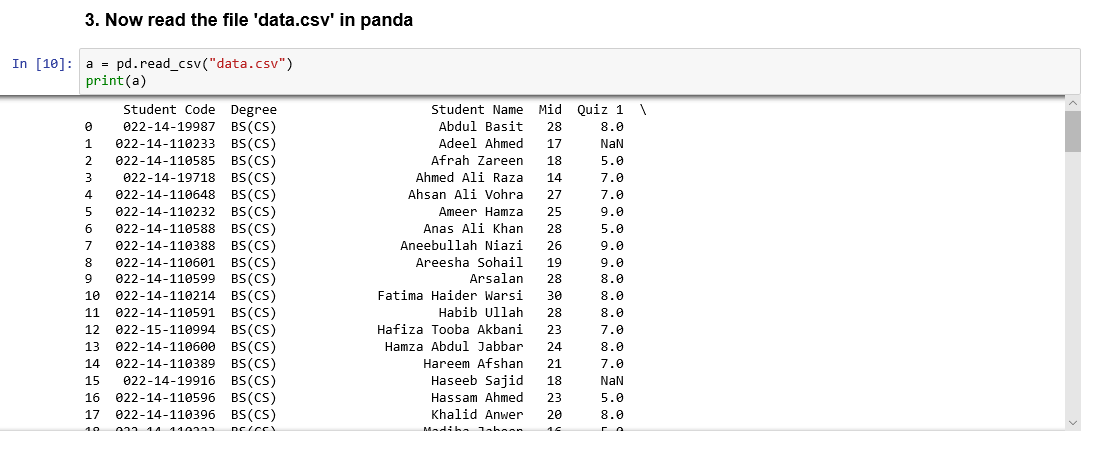
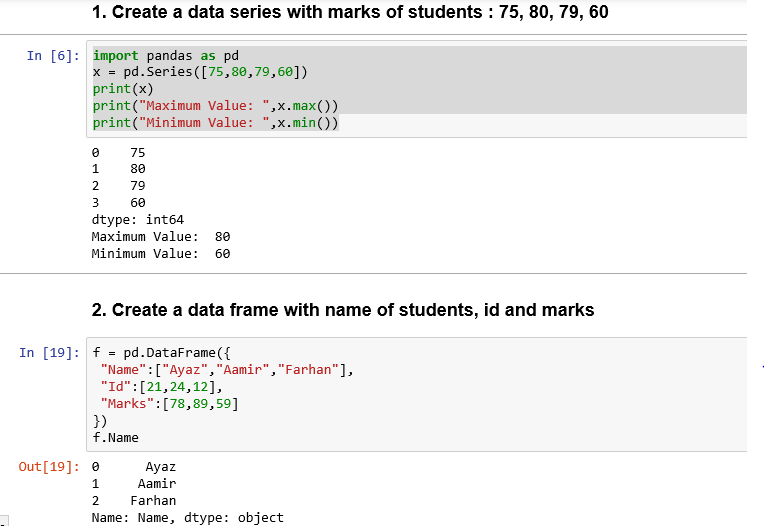
**Lab 4: To study and implement pandas library**

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:**

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

​



**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:**

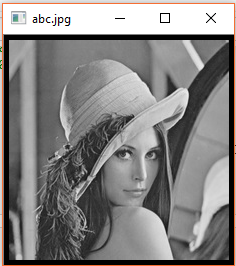
1. Consider the following image of model Lena. Load the image using Open CV and show on screen



**import** cv2  
img = cv2.imread(**"abc.jpg"**)  
cv2.imshow(**"abc.jpg"**, img)  
cv2.waitKey(0)

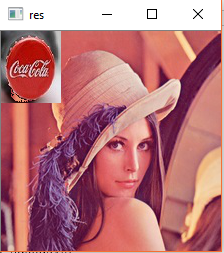
1. Create a border around the image

**import** cv2  
**import** numpy **as** np  
image = cv2.imread(**'abc.jpg'**,0)  
constant= cv2.copyMakeBorder(image,5,5,5,5,cv2.BORDER\_CONSTANT)  
  
cv2.imshow(**'abc.jpg'**,constant)  
cv2.waitKey(0)  
cv2.destroyAllWindows()



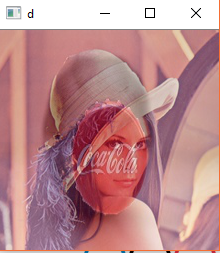
1. Create a copy of the face and paste it to the top right position

**import** cv2  
img1 = cv2.imread(**"abc.jpg"**)  
img2 = cv2.imread(**"cocacola.jpg"**)  
  
rows,cols,channels = img2.shape  
roi = img1[0:rows, 0:cols]  
  
img2gray = cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)  
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH\_BINARY)  
mask\_inv = cv2.bitwise\_not(mask)  
  
img1\_bg = cv2.bitwise\_and(roi,roi,mask = mask\_inv)  
  
img2\_fg = cv2.bitwise\_and(img2,img2, mask = mask)  
  
dst = cv2.add(img1\_bg,img2\_fg)  
img1[0:rows, 0:cols] = dst  
  
cv2.imshow(**'res'**,img1)  
cv2.waitKey(0)



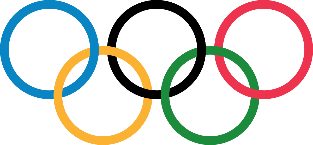
1. Consider the following Pepsi logo. Blend it over the Lena’s image

**import** cv2  
img = cv2.imread(**"abc.jpg"**)  
img1 = cv2.imread(**"coca.jpg"**)  
d = cv2.addWeighted(img,0.6,img1,0.4,0)  
cv2.imshow(**'d'**, d)  
cv2.waitKey(0)  
*#blending*



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.
2. Draw the following Olympic circles using Open CV.

**import** cv2  
**import** numpy **as** np  
image = cv2.imread(**'abc.jpg'**,0)  
img = np.ones((461,650,3))  
img = cv2.circle(img,(197,180),73,(255,0,0),7)  
img = cv2.circle(img,(355,180),73,(0,0,0),7)  
img = cv2.circle(img,(512,180),73,(0,0,255),7)  
img = cv2.circle(img,(275,275),73,(0,255,255),7)  
img = cv2.circle(img,(435,275),73,(0,255,0),7)  
cv2.imshow(**'abc.jpg'**,img)  
cv2.waitKey(0)



**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:**

* 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

import cv2

image = cv2.imread("lena.png")

cv2.imshow("original", image)

r = 100.0 / image.shape[1]

dim = (100, int(image.shape[0] \* r))

resized = cv2.resize(image, dim, interpolation=cv2.INTER\_AREA)

cv2.imshow("resized", resized)

(h, w) = image.shape[:2]

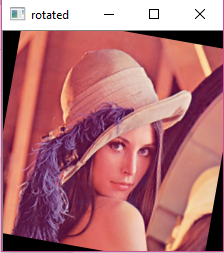
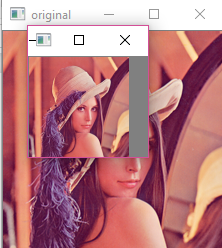
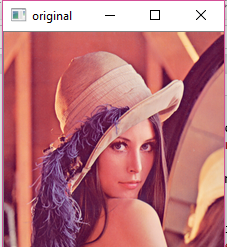
center = (w / 2, h / 2)

M = cv2.getRotationMatrix2D(center, 350, 1.0)

rotated = cv2.warpAffine(image, M, (w, h))

cv2.imshow("rotated", rotated)

cv2.waitKey(0)



* 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

import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

img = cv.imread('Hello.jpg',0)

ret,thresh1 = cv.threshold(img,127,255,cv.THRESH\_BINARY)

ret,thresh2 = cv.threshold(img,127,255,cv.THRESH\_BINARY\_INV)

ret,thresh3 = cv.threshold(img,127,255,cv.THRESH\_TRUNC)

ret,thresh4 = cv.threshold(img,127,255,cv.THRESH\_TOZERO)

ret,thresh5 = cv.threshold(img,127,255,cv.THRESH\_TOZERO\_INV)

titles = ['Original Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']

images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

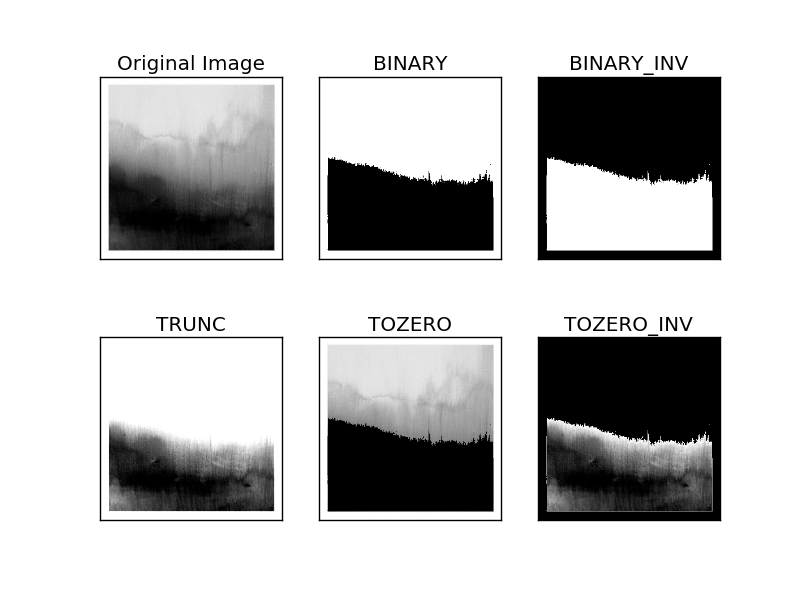
for i in range(6):

plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()



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

import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

img = cv.imread('Hello.jpg',0)

ret,thresh1 = cv.threshold(img,127,255,cv.THRESH\_BINARY)

ret,thresh2 = cv.threshold(img,127,255,cv.THRESH\_BINARY\_INV)

ret,thresh3 = cv.threshold(img,127,255,cv.THRESH\_TRUNC)

ret,thresh4 = cv.threshold(img,127,255,cv.THRESH\_TOZERO)

ret,thresh5 = cv.threshold(img,127,255,cv.THRESH\_TOZERO\_INV)

titles = ['Original Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']

images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

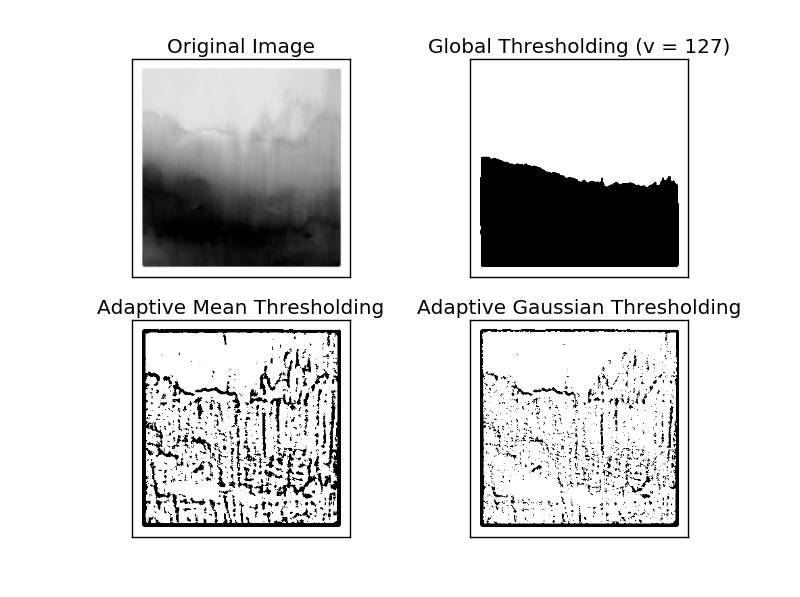
for i in range(6):

plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()



import cv2 as cv

import numpy as np

from matplotlib import pyplot as plt

img = cv.imread('Hello.jpg',0)

# global thresholding

ret1,th1 = cv.threshold(img,127,255,cv.THRESH\_BINARY)

# Otsu's thresholding

ret2,th2 = cv.threshold(img,0,255,cv.THRESH\_BINARY+cv.THRESH\_OTSU)

# Otsu's thresholding after Gaussian filtering

blur = cv.GaussianBlur(img,(5,5),0)

ret3,th3 = cv.threshold(blur,0,255,cv.THRESH\_BINARY+cv.THRESH\_OTSU)

# plot all the images and their histograms

images = [img, 0, th1,

img, 0, th2,

blur, 0, th3]

titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',

'Original Noisy Image','Histogram',"Otsu's Thresholding",

'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

for i in range(3):

plt.subplot(3,3,i\*3+1),plt.imshow(images[i\*3],'gray')

plt.title(titles[i\*3]), plt.xticks([]), plt.yticks([])

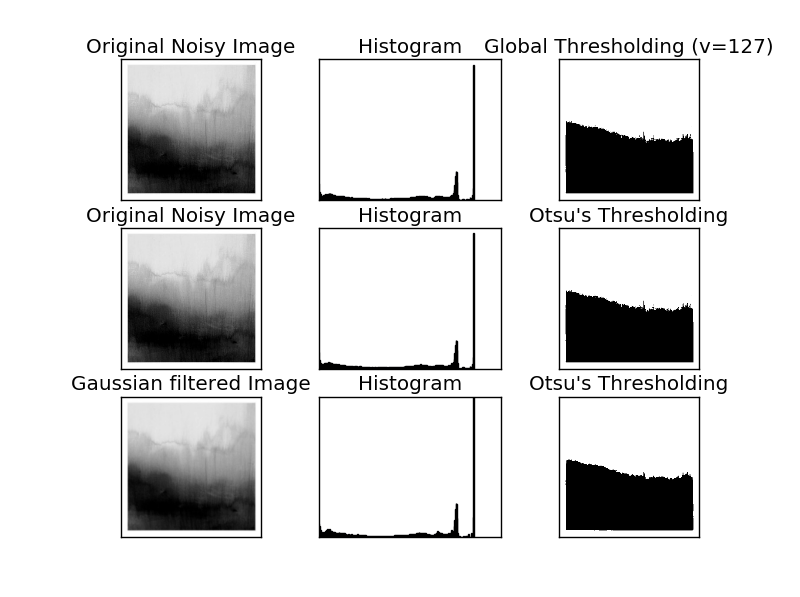
plt.subplot(3,3,i\*3+2),plt.hist(images[i\*3].ravel(),256)

plt.title(titles[i\*3+1]), plt.xticks([]), plt.yticks([])

plt.subplot(3,3,i\*3+3),plt.imshow(images[i\*3+2],'gray')

plt.title(titles[i\*3+2]), plt.xticks([]), plt.yticks([])

plt.show()



* Consider the following image:



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

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('lenaVersion2.png')

kernel = np.ones((5,5),np.float32)/25

dst = cv2.filter2D(img,-1,kernel)

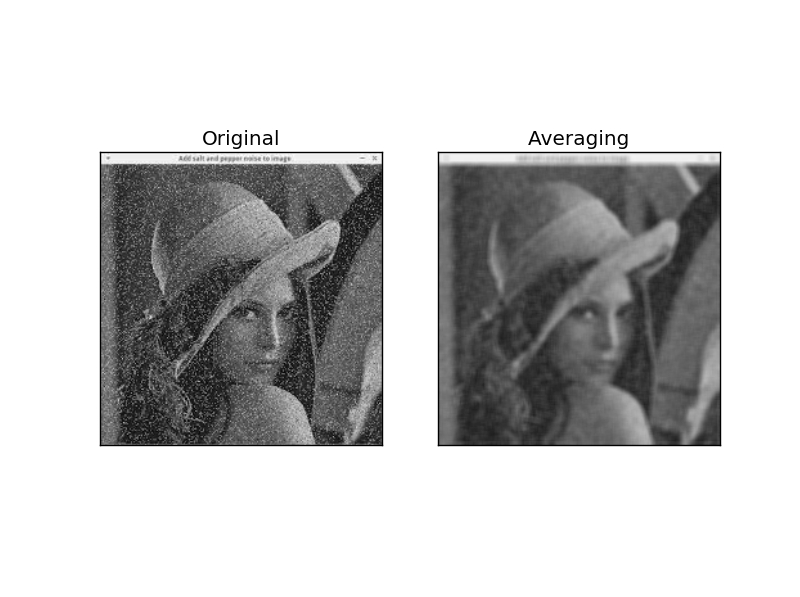
plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(dst),plt.title('Averaging')

plt.xticks([]), plt.yticks([])

plt.show()

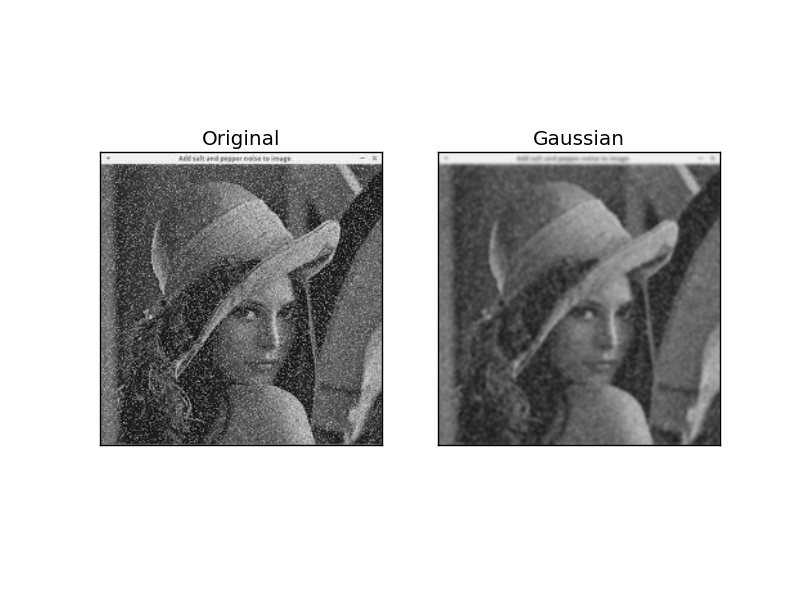


gaussian = cv2.GaussianBlur(img,(5,5),0)

plt.subplot(122),plt.imshow(gaussian),plt.title('Gaussian')

plt.xticks([]), plt.yticks([])

plt.show()



median = cv2.medianBlur(img,5)

plt.subplot(122),plt.imshow(median),plt.title('Median')

plt.xticks([]), plt.yticks([])

plt.show()



* Consider the following image:



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

import cv2

import numpy as np

img = cv2.imread('sportMan.jpg', 0)

kernel = np.ones((5,5), np.uint8)

img\_erosion = cv2.erode(img, kernel, iterations=1)

img\_dilation = cv2.dilate(img, kernel, iterations=1)

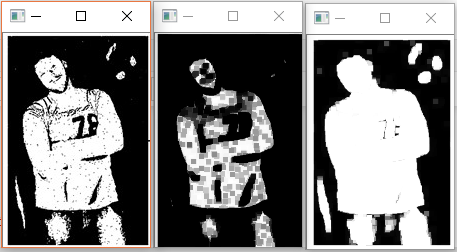
cv2.imshow('Input', img)

cv2.imshow('Erosion', img\_erosion)

cv2.imshow('Dilation', img\_dilation)

cv2.waitKey(0)

plt.show()



import cv2 as cv

import numpy as np

img = cv.imread('sportMan.jpg',0)

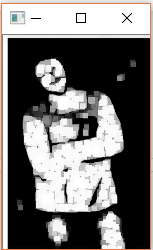
kernel = np.ones((5,5),np.uint8)

opening = cv.morphologyEx(img, cv.MORPH\_OPEN, kernel)

cv2.imshow('Opening', opening)

cv2.waitKey(0)

plt.show()



import cv2 as cv

import numpy as np

img = cv.imread('sportMan.jpg',0)

kernel = np.ones((5,5),np.uint8)

closing = cv.morphologyEx(img, cv.MORPH\_CLOSE, kernel)

cv2.imshow('Closing', closing)

cv2.waitKey(0)

plt.show()



* Consider the following image:



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

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('puzzle.jpg',0)

edges = cv2.Canny(img,100,200)

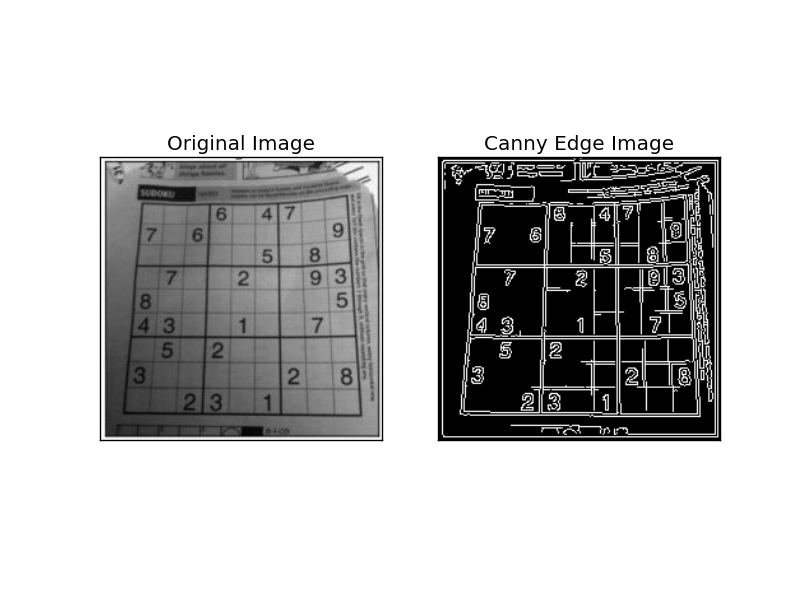
plt.subplot(121),plt.imshow(img,cmap = 'gray')

plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(edges,cmap = 'gray')

plt.title('Canny Edge Image'), plt.xticks([]), plt.yticks([])

plt.show()



import cv2

import numpy as np

from matplotlib import pyplot as plt

img0 = cv2.imread('puzzle.jpg',)

gray = cv2.cvtColor(img0, cv2.COLOR\_BGR2GRAY)

img = cv2.GaussianBlur(gray,(3,3),0)

laplacian = cv2.Laplacian(img,cv2.CV\_64F)

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

sobely = cv2.Sobel(img,cv2.CV\_64F,0,1,ksize=5)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

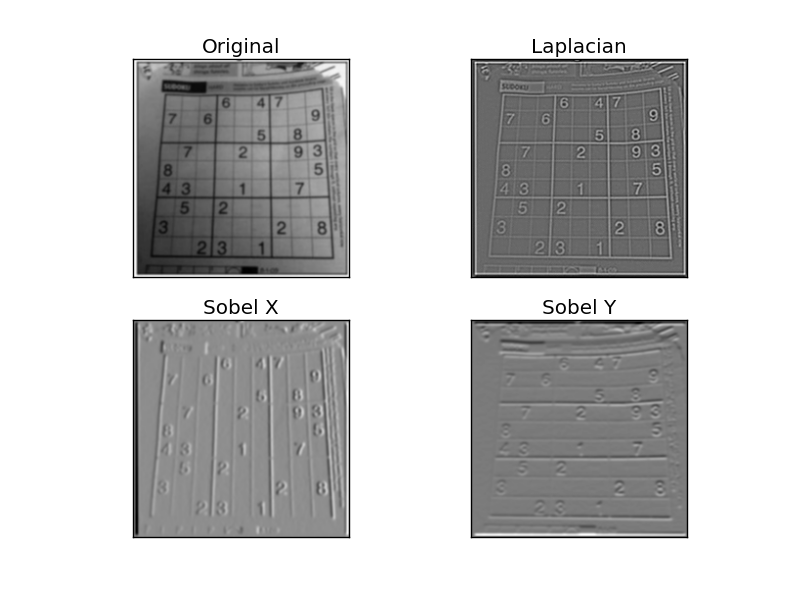
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()



**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

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('afridi.png',0)

f = np.fft.fft2(img)

fshift = np.fft.fftshift(f)

magnitude\_spectrum = 20\*np.log(np.abs(fshift))

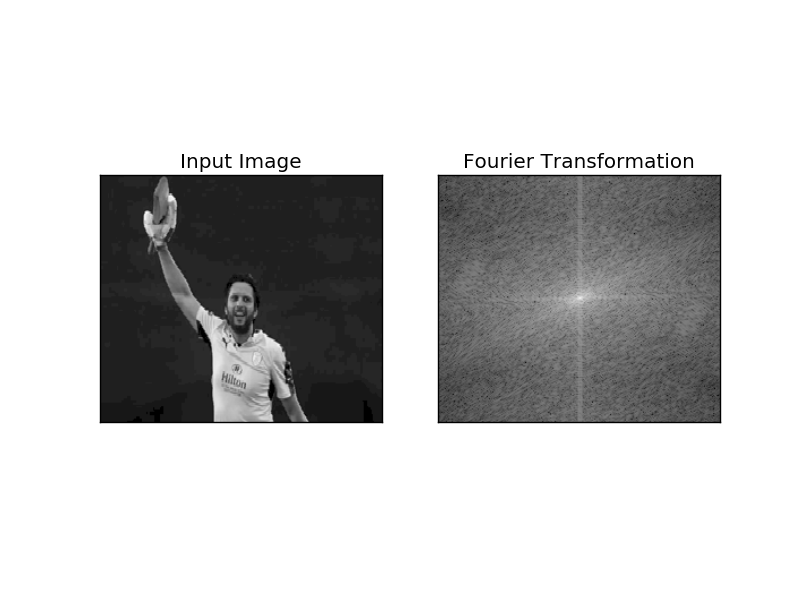
plt.subplot(121),plt.imshow(img, cmap = 'gray')

plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(magnitude\_spectrum, cmap = 'gray')

plt.title('Fourier Transformation'), plt.xticks([]), plt.yticks([])

plt.show()



* 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?

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread("afridi.png",0)

lap = cv2.Laplacian(img,cv2.CV\_64F)

sobelx = cv2.Sobel(img,cv2.CV\_64F,1,0,ksize=5)

plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2),plt.imshow(lap,cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(sobelx,cmap = 'gray')

f = np.fft.fft2(img)

fshift = np.fft.fftshift(f)

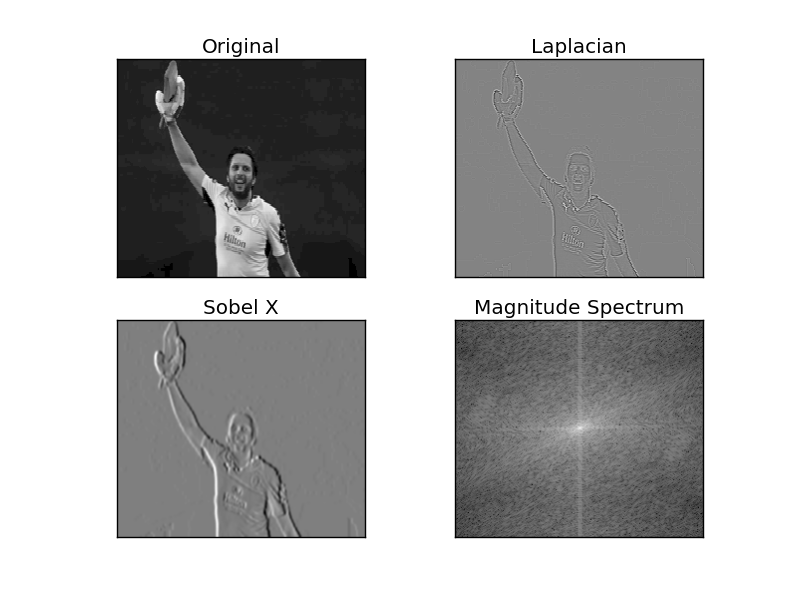
magnitude\_spectrum = 20\*np.log(np.abs(fshift))

plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4),plt.imshow(magnitude\_spectrum, cmap = 'gray')

plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])

plt.show()



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



import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('afridi.png',0)

img2 = img.copy()

template = cv2.imread('afridihead.png',0)

w, h = template.shape[::-1]

# All the 6 methods for comparison in a list

methods = ['cv2.TM\_CCOEFF']

for meth in methods:

img = img2.copy()

method = eval(meth)

res = cv2.matchTemplate(img,template,method)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

top\_left = min\_loc

else:

top\_left = max\_loc

bottom\_right = (top\_left[0] + w, top\_left[1] + h)

cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

plt.subplot(121),plt.imshow(res,cmap = 'gray')

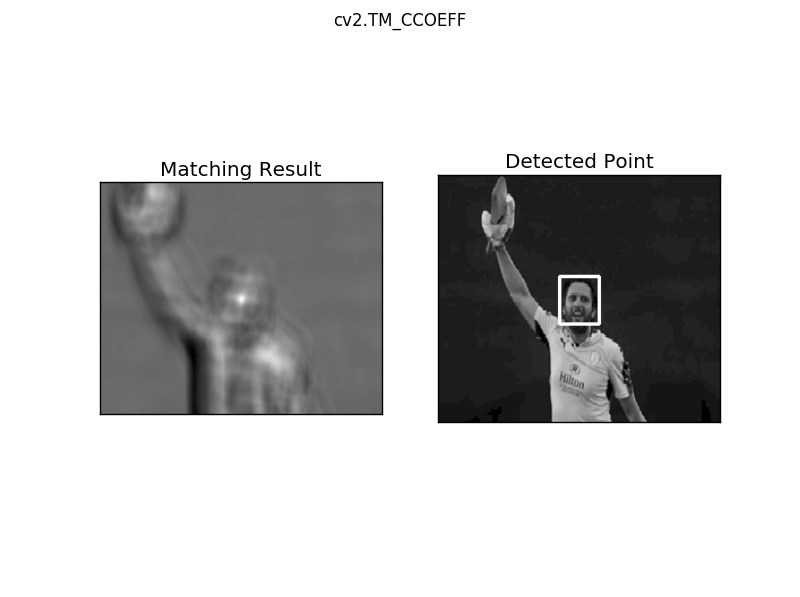
plt.title('Matching Result'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(img,cmap = 'gray')

plt.title('Detected Point'), plt.xticks([]), plt.yticks([])

plt.suptitle(meth)

plt.show()



* Consider the following image:



* Find the histogram of the image

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('flower.jpg',0)

equ = cv2.equalizeHist(img)

res = np.hstack((img,equ))

cv2.imwrite('res.png',res)

cv2.imshow("Hello",res)

cv2.waitKey()



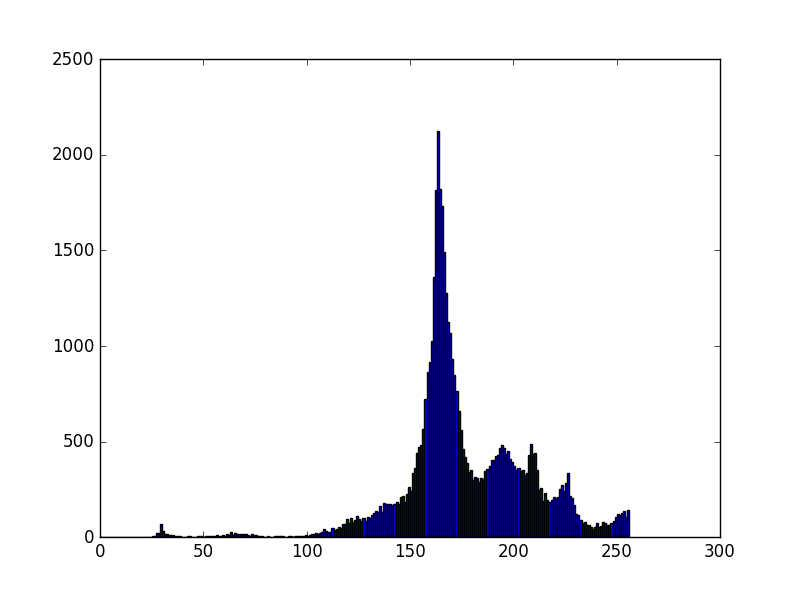
import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('flower.jpg',0)

plt.hist(img.ravel(),256,[0,256]); plt.show()



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

import numpy as np

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('digits.png')

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

# 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()

# Initiate kNN, train the data, then test it with test data for k=1

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)

# save the data

np.savez('knn\_data.npz',train=train, train\_labels=train\_labels)

# Now load the data

with np.load('knn\_data.npz') as data:

print(data.files)

train = data['train']

train\_labels = data['train\_labels']

91.76

['train\_labels', 'train']

**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)

True

91.92

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)

94.15

**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

**import** numpy **as** np  
**import** cv2

gray = cv2.imread(**'digits.png'**,0)

cells = [np.hsplit(row,100) **for** row **in** np.vsplit(gray,50)]

x = np.array(cells)

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)*

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)

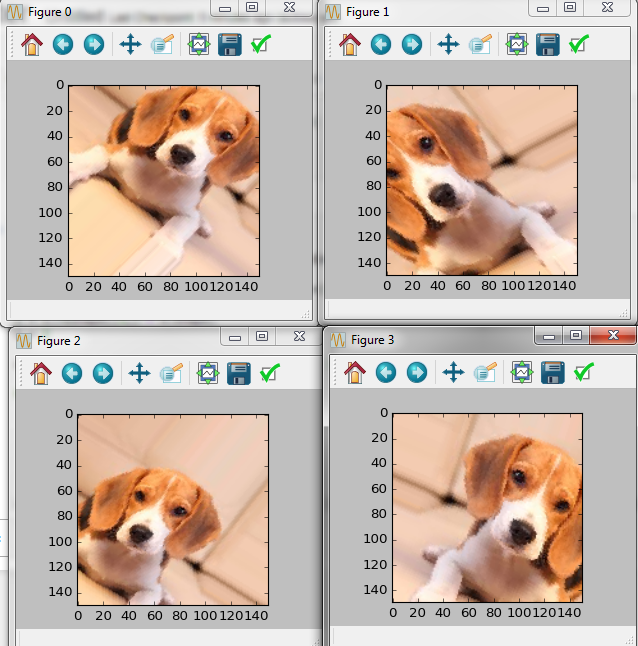
matches = result==test\_labels  
correct = np.count\_nonzero(matches)  
accuracy = correct\*100.0/result.size  
print(accuracy)

**91.76**

1. Consider the following images. Using data augmentation, generate 100 images.



**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**



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.applications **import** VGG16  
conv\_base = VGG16(weights=**'imagenet'**,  
include\_top=**False**,  
input\_shape=(150, 150, 3))

conv\_base.summary()

**from** keras **import** models  
**from** keras **import** layers  
model = models.Sequential()  
model.add(conv\_base)  
model.add(layers.Flatten())  
model.add(layers.Dense(256, activation=**'relu'**))  
model.add(layers.Dense(1, activation=**'sigmoid'**))

model.summary()

conv\_base.trainable = **True**set\_trainable = **False  
for** layer **in** conv\_base.layers:  
 **if** layer.name == **'block5\_conv1'**:  
 set\_trainable = **True  
 if** set\_trainable:  
 layer.trainable = **True  
 else**:  
 layer.trainable = **False**

train\_dir=**'data'**validation\_dir=**'data'**

**from** keras.preprocessing.image **import** ImageDataGenerator  
**from** keras **import** optimizers  
train\_datagen = ImageDataGenerator(  
rescale=1./255,  
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'**)  
test\_datagen = ImageDataGenerator(rescale=1./255)  
train\_generator = train\_datagen.flow\_from\_directory(  
train\_dir,  
target\_size=(150, 150),  
batch\_size=20,  
class\_mode=**'binary'**)  
validation\_generator = test\_datagen.flow\_from\_directory(  
validation\_dir,  
target\_size=(150, 150),  
batch\_size=20,  
class\_mode=**'binary'**)  
model.compile(loss=**'binary\_crossentropy'**,  
optimizer=optimizers.RMSprop(lr=2e-5),  
metrics=[**'acc'**])

Found 200 images belonging to 2 classes.

Found 200 images belonging to 2 classes.

history = model.fit\_generator(  
train\_generator,  
steps\_per\_epoch=10,  
epochs=2,  
validation\_data=validation\_generator,  
validation\_steps=10)