**Problem #5**

**import** numpy **as** np  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **'''  
 This will create a 3x3 numpy array object like so:  
   
 1 2 3  
 4 5 6  
 7 8 9  
   
 '''** a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
  
  
 **'''  
 This will get the third row of values   
 [get list @ index 2, get all values in this list]   
 so b = [4,5,6]  
 '''** b = a[2, :]  
  
  
  
 **'''  
 This will flatten the array a to a 1d array with all of the values  
 so c = [1,2,3,4,5,6,7,8,9]  
 '''** c = a.reshape(-1)  
  
  
 **'''  
 this will create a 5x1 vector of random floating point numbers randomly sampled from  
 a normal distrobution  
 '''** f = np.random.randn(5, 1)  
  
  
 **'''  
 f>0 will create a boolean array where every element is compared   
 to 0 those that are greater will have true at that index those that are less will  
 have false. A numpy array can be indexed by booleans where any index that  
 is true in the boolean array will have a coresponding value appear in the  
 resultant array. So g will contain any values in f that mee the condition   
 f>0  
 '''** g = f[f > 0]  
  
  
 **'''  
 zeros will create a vector of zeros with length 10 the addition will add .5 to   
 all values in the vector so the result will be a vector containing 10 0.5 floats  
 '''** x = np.zeros(10) + 0.5  
  
  
 **'''  
 this will create a vector of ones whose size will be the length of another vector x so in  
 this case 10. It will then perform scalar multiplication on the vector. The result will  
 be a vector of length 10 containing all 0.5 floats  
 '''** y = 0.5 \* np.ones(len(x))  
  
  
 **'''  
 This will perform vector addition between x and y the result will be a vector of length 10  
 containing all 1s.  
 '''** z = x + y  
  
  
 **'''  
 this will create an array that spans the range 1 to 100 [exclusive]   
 with a default step size of 1 so [1,2,3,..99].  
 '''** a = np.arange(1, 100)  
  
 **'''  
 The slice notation is as follows list[<start>:<stop>:<step>]  
 By default this will span the entire list if no args are given  
 the step of minus 1 will step backwards through the list reversing it  
   
 refecerenced:  
 https://stackoverflow.com/questions/31633635/what-is-the-meaning-of-inta-1-in-python#:~:text=The%20notation%20that%20is%20used,stop\_index%3E%2C%20%5D  
 '''** b = a[::-1]  
  
  
 **'''  
 given 1 argument this function will create a vector with range 0 to n-1 and randomly permute   
 [shuffle] all of the values in the generated array. If given an array it would shuffle it. so  
 in this case we create a vector [0,1,2,..9] and randomly permute it  
 '''** c = np.random.permutation(10)  
  
  
  
 **5B** *#1* y = np.array([1, 2, 3, 4, 5, 6])  
 z = y.reshape(3,2)  
  
 *#2* x = np.max(z)  
 r,c = np.where(z==x)  
 r = r[0]  
 c = c[0]  
  
 *#3* v = np.array([1, 8, 8, 2, 1, 3, 9, 8])  
 x = np.count\_nonzero(v == 1)  
  
 *#4* n = 4  
 DiceRolls = np.random.randint(1,7,size = n)

**5c**

**import** numpy **as** np  
**import** matplotlib.pylab **as** plt  
  
  
**def** createFile(path,DataSource):  
 np.savetxt(path, DataSource, fmt=**'%d'**)  
  
**def** getFreq(list):  
 list = list.reshape(-1).tolist()  
 freqDict = {}  
 **for** x **in** list:  
 freqDict[x] = list.count(x)  
 **return** freqDict  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
  
 *#1* path = **'inputP5A.npy'** DataSource = np.random.randint(1, 255, (100, 100))  
 *#createFile(path,DataSource)* A = np.loadtxt(**'inputP5A.npy'**, dtype=int)  
 f = getFreq(A)  
 valueFreqList = sorted(f.items(), key=**lambda** x: x[1], reverse=**True**)  
 print(**"Ordered Intensity List"**)  
 print(valueFreqList)  
 *#valueFreqList = sorted(f.items())* x, y = zip(\*valueFreqList) *# unpack a list of pairs into two tuples  
 #plt.plot(y)* fig, ax = plt.subplots()  
 scatter = ax.scatter(x , y, c=y)  
 ax.legend(\*scatter.legend\_elements(),title=**'Frequency of value'**,bbox\_to\_anchor=(.9, .6),loc=**'center left'**)  
 plt.title(**'sorted frequncy scatter'**)  
 plt.xlabel(**'value'**)  
 plt.ylabel(**'frequncy'**)  
 plt.show()  
 *#fig1.savefig('Frequncy\_VS\_value\_5\_c\_1.png')*

*A close up of a map

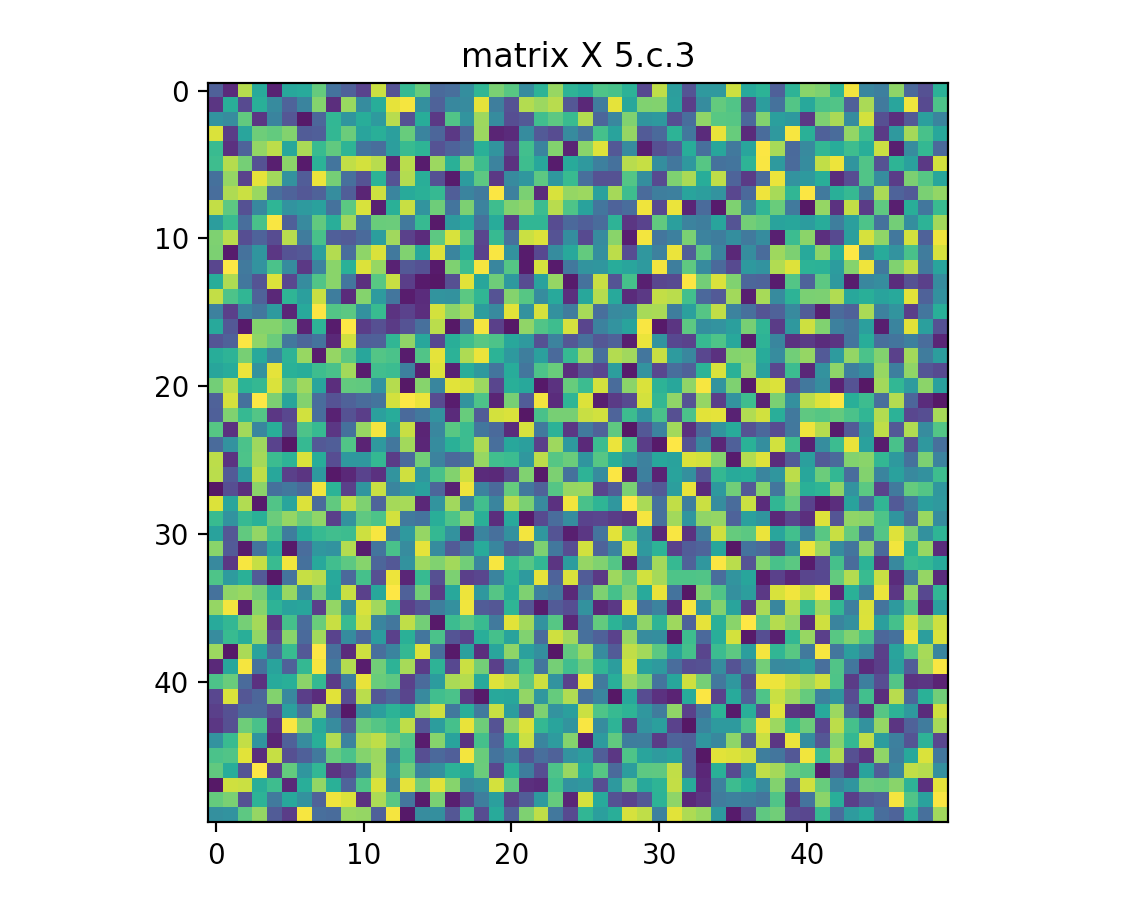
Description automatically generated*

*[(79, 57), (176, 53), (189, 53), (154, 53), (80, 53), (59, 52), (113, 52), (62, 51), (184, 50), (151, 50), (145, 50), (51, 50), (205, 50), (4, 49), (167, 49), (115, 49), (207, 49), (153, 49), (149, 49), (123, 49), (172, 49), (216, 48), (60, 48), (173, 48), (17, 48), (253, 48), (175, 48), (182, 48), (225, 47), (47, 47), (185, 47), (73, 47), (14, 47), (140, 46), (103, 46), (183, 46), (66, 46), (93, 46), (155, 46), (199, 46), (34, 46), (70, 46), (69, 46), (234, 45), (43, 45), (132, 45), (148, 45), (157, 45), (6, 45), (46, 45), (195, 45), (196, 44), (203, 44), (136, 44), (220, 44), (95, 44), (49, 44), (5, 44), (221, 44), (112, 44), (20, 44), (229, 44), (107, 44), (121, 44), (16, 44), (201, 44), (238, 44), (187, 44), (186, 43), (118, 43), (161, 43), (30, 43), (126, 43), (179, 43), (54, 43), (76, 43), (83, 43), (122, 43), (143, 42), (96, 42), (230, 42), (78, 42), (165, 42), (37, 42), (38, 42), (124, 42), (68, 42), (147, 42), (158, 42), (61, 42), (235, 41), (110, 41), (241, 41), (237, 41), (92, 41), (208, 41), (105, 41), (1, 41), (242, 41), (248, 41), (210, 41), (142, 41), (114, 41), (111, 41), (226, 40), (85, 40), (215, 40), (2, 40), (88, 40), (25, 40), (116, 40), (109, 40), (139, 40), (204, 40), (191, 40), (202, 40), (152, 40), (219, 40), (194, 40), (31, 40), (106, 39), (102, 39), (236, 39), (97, 39), (249, 39), (8, 39), (254, 39), (23, 39), (223, 39), (214, 39), (42, 39), (198, 39), (75, 39), (141, 39), (188, 39), (44, 39), (180, 39), (67, 39), (162, 39), (146, 39), (129, 39), (53, 39), (12, 38), (137, 38), (98, 38), (156, 38), (213, 38), (134, 38), (168, 38), (84, 38), (56, 38), (159, 38), (169, 38), (10, 38), (127, 38), (41, 37), (81, 37), (39, 37), (82, 37), (135, 37), (227, 37), (163, 37), (212, 37), (181, 37), (90, 37), (57, 37), (108, 37), (144, 37), (250, 37), (104, 36), (206, 36), (32, 36), (74, 36), (164, 36), (217, 36), (63, 36), (45, 36), (77, 36), (36, 36), (72, 36), (48, 36), (177, 36), (224, 36), (120, 35), (22, 35), (21, 35), (58, 35), (125, 35), (99, 35), (244, 35), (101, 35), (13, 35), (243, 35), (52, 34), (87, 34), (251, 34), (138, 34), (252, 34), (131, 34), (174, 34), (40, 34), (71, 34), (86, 34), (18, 34), (231, 34), (222, 34), (128, 34), (11, 34), (94, 33), (24, 33), (240, 33), (119, 33), (160, 33), (133, 33), (28, 33), (232, 33), (170, 33), (150, 33), (117, 33), (245, 33), (190, 32), (27, 32), (26, 32), (65, 32), (166, 32), (197, 32), (15, 32), (239, 31), (19, 31), (200, 31), (130, 31), (89, 31), (192, 31), (247, 31), (35, 31), (209, 31), (218, 31), (233, 30), (7, 30), (33, 30), (3, 30), (91, 30), (9, 30), (178, 29), (55, 29), (29, 29), (228, 29), (100, 28), (211, 28), (50, 28), (64, 27), (246, 27), (193, 25), (171, 24)]*

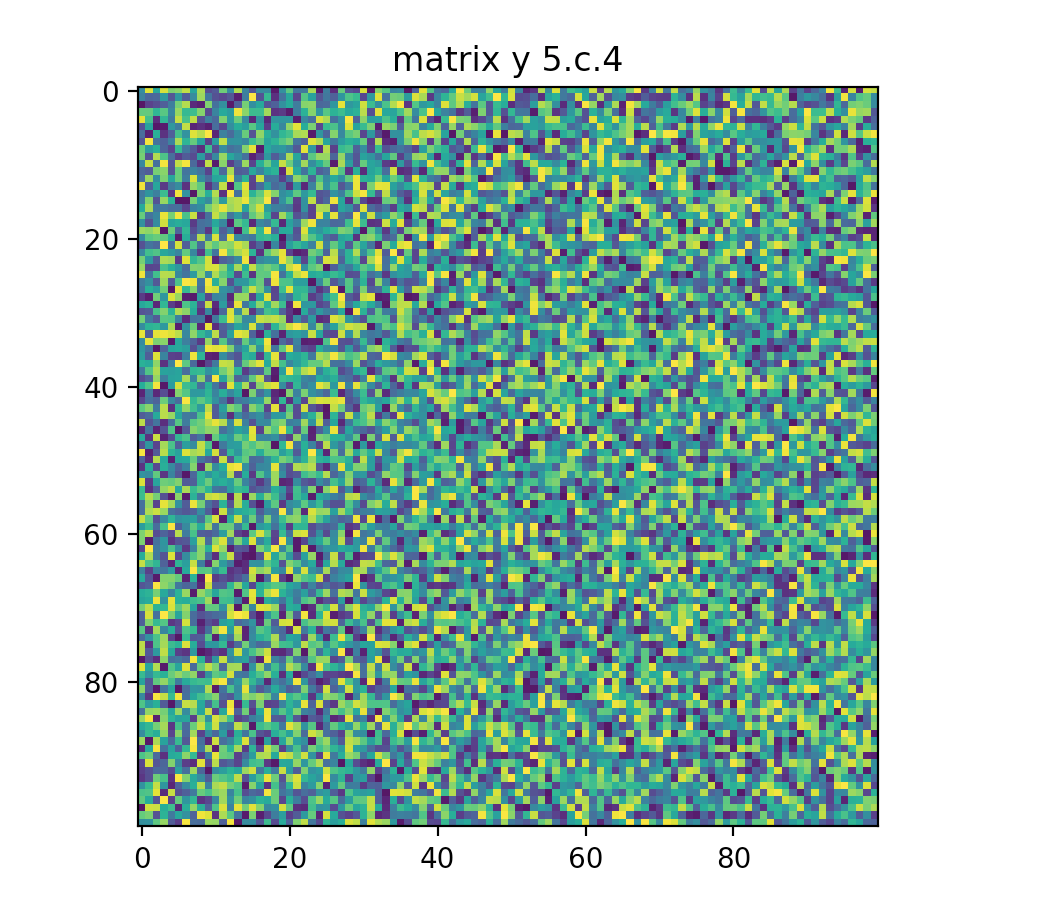
*#2* plt.hist(x, density=**False**, bins=20)  
 plt.title(**'image value histogram 20 buckets'**)  
 plt.xlabel(**'value'**)  
 plt.ylabel(**'frequncy'**)  
 plt.show()  
 *#fig2.savefig('Value\_Histogram\_5\_c\_2.png')*

*A screenshot of a cell phone

Description automatically generated  
  
  
 #3* x = [M **for** SubA **in** np.split(A,2, axis = 0) **for** M **in** np.split(SubA,2, axis = 1)][2]  
 plt.matplotlib.pyplot.imshow(x,interpolation=**'none'**)  
 plt.title(**'matrix X 5.c.3'**)  
 plt.show()  
 *#fig3.savefig('Left\_Quad\_5\_c\_3.png')* path = **'outputP5X.npy'** createFile(path,x)



*#4* Y = A - np.mean(A)  
 plt.matplotlib.pyplot.imshow(Y, interpolation=**'none'**)  
 plt.title(**'matrix y 5.c.4'**)  
 plt.show()  
 *#fig4.savefig('Average\_Diffed\_5\_c\_4.png')* path = **'outputP5Y.npy'** createFile(path,Y)



*#5*Z = np.ones((100,100))\*255  
Z[A <= np.mean(A) ] = 0  
Z = [[red,0,0] **for** row **in** Z **for** red **in** row]  
Z = np.array(Z, dtype=np.uint8).reshape((100, 100, 3))  
plt.matplotlib.pyplot.imshow(Z, interpolation=**'none'**)  
plt.title(**'matrix z 5.c.5'**)  
plt.show()  
plt.matplotlib.pyplot.imshow(Z, interpolation=**'none'**)  
plt.title(**'matrix z 5.c.5'**)  
plt.savefig(**'outputP5Z.png'**)

A picture containing rug

Description automatically generated  
  
  
  
  
 *#####references* **'''  
 https://www.geeksforgeeks.org/counting-the-frequencies-in-a-list-using-dictionary-in-python/  
 https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-values/37266356  
 https://stackoverflow.com/questions/12811981/slicing-python-matrix-into-quadrants  
 '''**

**Problem #6**

**import** cv2  
**import** math  
**import** numpy **as** np  
  
  
**class** Kernal:  
  
 **def** \_\_init\_\_(self,N,sigma):  
 self.N = N  
 self.sigma = sigma  
 self.frontBit = 1 / (sigma \*\* 2 \* 2 \* math.pi)  
 self.k = np.zeros((N, N))  
 self.idxMin = int(N / 2)  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 x, y = self.getXYfromIDX(i, j)  
 self.k[i, j] = self.frontBit \* math.exp(-(x \*\* 2 + y \*\* 2) / (2 \* sigma \*\* 2))  
  
  
  
 **def** getXYfromIDX(self,i,j):  
 x = i-self.idxMin  
 y = j-self.idxMin  
 **return** x, y  
  
  
**class** Image:  
  
 paddingOffset = **None  
  
 def** \_\_init\_\_(self,fpath):  
 self.image = cv2.imread(fpath, 0) *#np.array([[1,2,3],[4,5,6],[7,8,9]])#* self.height = len(self.image)  
 self.width = len(self.image[0])  
  
 **def** setPadding(self,width):  
  
 *#from numpy docs @ https://numpy.org/doc/stable/reference/generated/numpy.pad.html* **def** pad\_with(vector, pad\_width, iaxis, kwargs):  
 pad\_value = kwargs.get(**'padder'**, 10)  
 vector[:pad\_width[0]] = pad\_value  
 vector[-pad\_width[1]:] = pad\_value  
  
 padded = np.pad(self.image, width, pad\_with, padder=0)  
 self.paddingOffset = width  
 self.padded = padded  
  
  
  
  
 **def** convolve(self,k):  
 *#thankfully kernal is symetric so no flipy bois* **for** r **in** range(self.height):  
 print(r)  
 **for** c **in** range(self.width):  
 cellVal= 0  
 **for** y **in** range (k.N):  
 **for** x **in** range(k.N):  
 cellVal += self.padded[r+x,c+y] \* k.k[x,y]  
 self.image[r,c] = cellVal  
 *# print("plz")  
 # print(self.image)***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 fpath = **'inputP6.jpg'** I1 = Image(fpath)  
 kern = Kernal(5, 1.414)  
 I1.setPadding(kern.idxMin)  
 cv2.imshow(**"preconvolve"**, I1.image)  
 I1.convolve(kern)  
 cv2.imshow(**"postconvolve3"**, I1.image)  
  
  
 cv2.waitKey(0)

A person sitting in front of a building

Description automatically generated