Excercise_4_solutions

May 30, 2019

1 Lecture 4

1st problem

Import NumPy as np Create an array of 10 zeros

```
In [1]: import numpy as np

        A = np.zeros(10)
        print(A)
```

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

Create an array of 10 ones

```
In [2]: B = np.ones(10)
        print(B)
```

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

Create an array of 10 fives

```
In [3]: C = np.full(10,5)
        print(C)
```

```
[5 5 5 5 5 5 5 5 5 5]
```

Create an array of integers from 10 to 50

```
In [4]: D = np.arange(10,51)
        print(D)
```

```
[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]
```

Create an array of all the even integers from 10 to 50

```
In [5]: E = np.arange(10,51, 2)
        print(E)
```

```
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50]
```

Create a 3x3 matrix with values ranging from 0 to 8

```
In [6]: F = np.arange(0,9)
        F = F.reshape(3,3)

        print(F)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Create a 3x3 identity matrix

```
In [7]: G = np.eye(3)
        G
```

```
Out[7]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

```
In [8]: np.random.rand(1,1)
```

```
Out[8]: array([[0.89605952]])
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

```
In [9]: np.random.randn(5,5)
```

```
Out[9]: array([[-1.26576065e+00, -7.27847335e-01,  5.22464976e-01,
                 1.16966001e-03, -4.39218374e-01],
               [ 5.84044808e-01,  4.09579685e-01,  6.96341765e-01,
                 1.64860859e+00,  4.20648502e-01],
               [-2.07047780e+00, -7.94693560e-01, -6.39643332e-01,
                -1.43119062e-02, -7.32220641e-01],
               [-6.52339162e-01, -2.70365197e-01, -3.84922845e-01,
                -2.79626544e+00,  4.24524528e-02],
               [-1.05070801e-01,  1.21722278e+00,  1.67578216e+00,
                 1.30547129e-01, -1.92222386e-01]])
```

Create the following matrix:

```
In [10]: np.arange(0.01,1.01,0.01).reshape(10,10)
```

```
Out[10]: array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
                [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
                [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
                [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
                [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
                [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
                [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
                [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
                [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
                [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

Create an array of 20 linearly spaced points between 0 and 1:

```
In [11]: np.linspace(0.,1.,20)
```

```
Out[11]: array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
                0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
                0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
                0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

2 Numpy Indexing and Selection

Now you will be given a few matrices, and be asked to replicate the resulting matrix outputs:

```
In [12]: mat = np.arange(1,26).reshape(5,5)
```

WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T BE ABLE TO SEE THE OUTPUT ANY MORE

```
In [13]: print(mat[2:5,1:5])
```

```
[[12 13 14 15]
 [17 18 19 20]
 [22 23 24 25]]
```

```
In [14]: print(mat[:3, 1])
```

```
[ 2  7 12]
```

```
In [15]: print(mat[3,4])
```

```
20
```

```
In [16]: print(mat[4,:])
```

```
[21 22 23 24 25]
```

```
In [17]: print(mat[3:5,:])
```

```
[[16 17 18 19 20]
 [21 22 23 24 25]]
```

Get the sum of all the values in mat

```
In [18]: sum_mat = mat.sum()
         print(sum_mat)
```

```
325
```

Get the standard deviation of the values in mat

```
In [19]: standard_dev_mat = mat.std()
         print(standard_dev_mat)
```

```
7.211102550927978
```

Get the sum of all the columns in mat

```
In [20]: mat.sum(axis = 0)
```

```
Out[20]: array([55, 60, 65, 70, 75])
```

Find median values in all columns

```
In [21]: b = np.median(mat, axis = 0)
         print(b)
```

```
[11. 12. 13. 14. 15.]
```

Find average values in all columns

```
In [22]: c = np.mean(mat, axis = 0)
         print(c)
```

```
[11. 12. 13. 14. 15.]
```

Find median values in all rows

```
In [23]: d = np.median(mat, axis = 1)
         print(d)
```

```
[ 3.  8. 13. 18. 23.]
```

Find average values in all rows

```
In [24]: e = np.mean(mat, axis = 1)
         print(e)
```

```
[ 3.  8. 13. 18. 23.]
```

3 Matplotlib exercise

Read the data attached in ISOD about the crimes in Los Angeles from 2010. And generate a histogram analyzing the number of crimes committed along the day. Show the hour of day distribution of crimes using two kinds of plots: histiogram and a scatter plot. (Caution! For scatter plot you will have to calculate number of crime occurrences in each hour.)

```
In [25]: import pandas as pd

         dataset = pd.read_csv('Crime_Data_from_2010_small.csv')
         dataset.head()
```

Out[25]:

	DR Number	Date Reported	Date Occurred	Time Occurred	Area ID	Area Name	\
0	1208575	03/14/2013	03/11/2013	1800	12	77th Street	
1	102005556	01/25/2010	01/22/2010	2300	20	Olympic	
2	418	03/19/2013	03/18/2013	2030	18	Southeast	
3	101822289	11/11/2010	11/10/2010	1800	18	Southeast	
4	42104479	01/11/2014	01/04/2014	2300	21	Topanga	

	Reporting District	Crime Code	Crime Code Description	\
0	1241	626	INTIMATE PARTNER - SIMPLE ASSAULT	
1	2071	510	VEHICLE - STOLEN	
2	1823	510	VEHICLE - STOLEN	
3	1803	510	VEHICLE - STOLEN	
4	2133	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	

	MO Codes	...	\
0	0416 0446 1243 2000	...	
1	NaN	...	
2	NaN	...	
3	NaN	...	
4	0329	...	

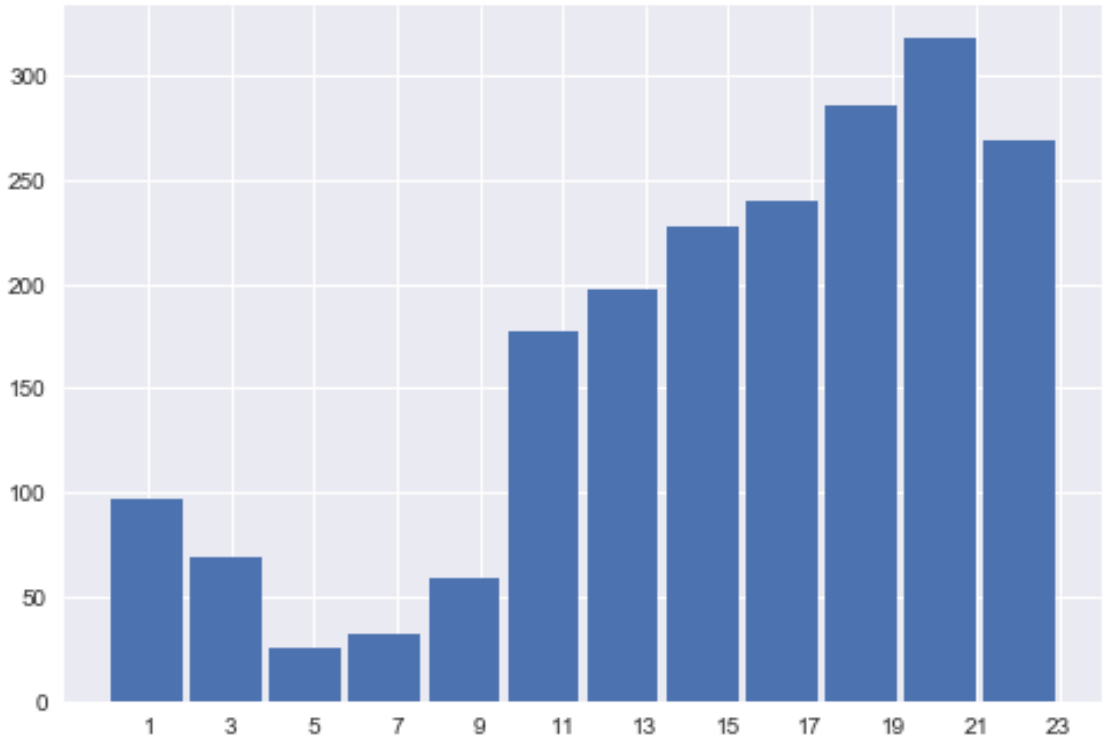
	Weapon Description	Status Code	\
0	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	AO	
1	NaN	IC	
2	NaN	IC	

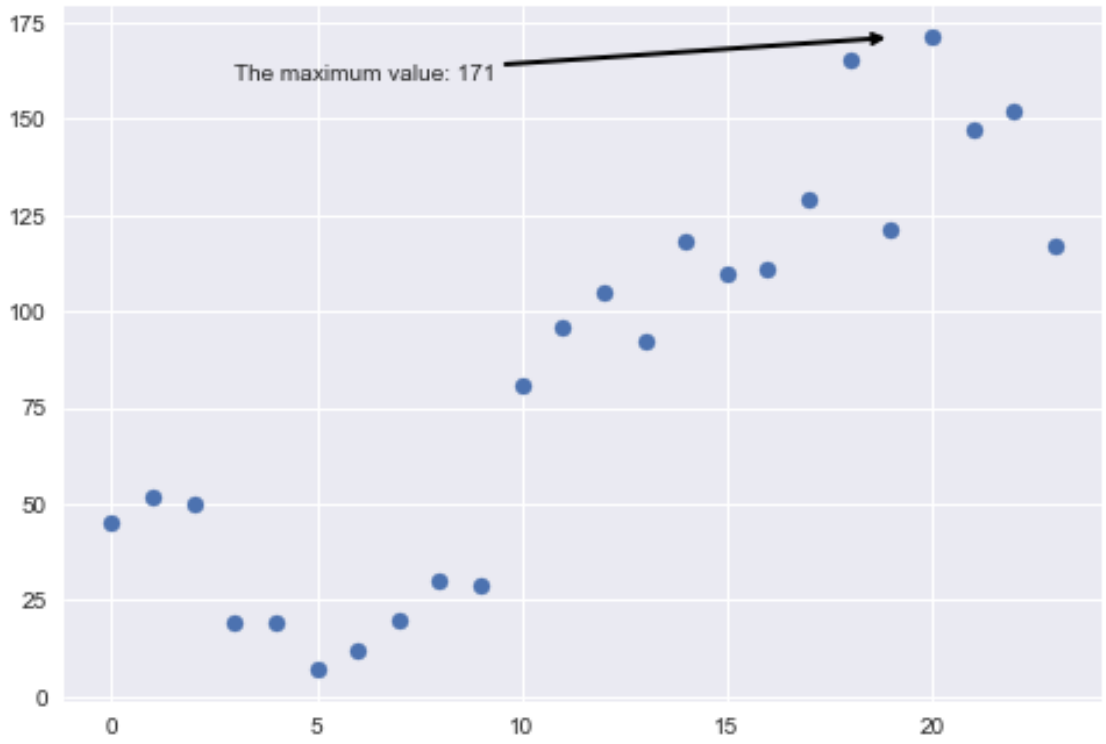
```
3                                                NaN          IC
4                                                NaN          IC

  Status Description  Crime Code 1 Crime Code 2  Crime Code 3 Crime Code 4  \
0        Adult Other           626          NaN           NaN          NaN
1        Invest Cont           510          NaN           NaN          NaN
2        Invest Cont           510          NaN           NaN          NaN
3        Invest Cont           510          NaN           NaN          NaN
4        Invest Cont           745          NaN           NaN          NaN

                                    Address Cross Street             Location
0  6300    BRYNHURST                    AV          NaN  (33.9829, -118.3338)
1                                  VAN NESS         15TH  (34.0454, -118.3157)
2   200 E  104TH                        ST          NaN   (33.942, -118.2717)
3                                     88TH         WALL  (33.9572, -118.2717)
4  7200    CIRRUS                       WY          NaN  (34.2009, -118.6369)

[5 rows x 26 columns]
```

3.1 Implement a backward substitution and find a solution

In [28]:
```
import csv
import matplotlib.pyplot as plt


with open('Crime_Data_from_2010_small.csv') as f:
  reader = csv.reader(f, delimiter=',')
  for row in reader:
    occurences_hours = [row[3] for row in reader][1:]
    occurences_hours = [int(t) // 100 for t in occurences_hours]

plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.hist(occurences_hours, rwidth=0.9, bins=12)
ax.set_xticks(range(1, 24, 2))

plt.show()


import collections


occurences_hours = collections.Counter(occurences_hours).items()
x, y = list(map(lambda a: a[0], occurences_hours)), list(map(lambda a: a[1], occurenc
fig, ax = plt.subplots()
ax.scatter(x, y)
ax.annotate(f'The maximum value: {max(y)}', xytext=(3,160), xy=(19, 171),
            arrowprops=dict(arrowstyle="->",lw=2))
```

```
plt.show()
```


300
250
200
150
100
50
0
1
3
5
7
9
11
13
15
17
19
21
23


The maximum value: 171
175
150
125
100
75
50
25
0
0
5
10
15
20

In [27]:
```
import numpy as np

def gaussian(A, b):
  A_aug = np.column_stack((A, b))
  for i in range(len(A_aug) - 1):
    row = A_aug[i]
    sub = [r - row * (r[i] / row[i]) for r in A_aug[i+1:]]
    A_aug = np.concatenate((A_aug[:i + 1], sub))
  return A_aug

def back(A, b):
  xs = []
  for a in A[::-1]:
    a, bl, b = a[::-1], b[-1], b[:-1]

    r = bl - sum(c * x for c, x in zip(a, xs))
    xs.append(r / a[len(xs)])

  return xs


A = np.arange(1, 17, dtype=np.float64).reshape(4,4)
A[1,2] = 88
A[1,3] = -3
A[2,3] = -3
print(f'A = {A}')

x = np.ones(A.shape[0])
print(f'Original x = {x}')
b = A  @ x.T
print(f'Right hand side for testing: b = {b}')


Ae = gaussian(A, b)
print(f'Check if A was unchanged ')
print(f'Eliminated augmented matrix:\n {Ae}')
print(f'Eliminated augmented matrix A part: {Ae[:,:-1]}')
print(f'Eliminated augmented matrix b part: {Ae[:,Ae.shape[1]-1]}')

# Find solution
x = back(Ae[:,:-1], Ae[:,Ae.shape[1]-1])
print(f'Solution: {x}')
```

```
A = [[ 1.  2.  3.  4.]
 [ 5.  6. 88. -3.]
```

```
 [ 9. 10. 11. -3.]
 [13. 14. 15. 16.]]
Original x = [1. 1. 1. 1.]
Right hand side for testing: b = [10. 96. 27. 58.]
Check if A was unchanged
Eliminated augmented matrix:
 [[   1.     2.     3.     4.    10. ]
 [   0.    -4.    73.   -23.    46. ]
 [   0.     0.  -162.     7.  -155. ]
 [   0.     0.     0.    22.5   22.5]]
Eliminated augmented matrix A part: [[   1.     2.     3.     4. ]
 [   0.    -4.    73.   -23. ]
 [   0.     0.  -162.     7. ]
 [   0.     0.     0.    22.5]]
Eliminated augmented matrix b part: [  10.    46.  -155.    22.5]
Solution: [1.0, 1.0, 1.0, 1.0]
```