Exercise_4_solutions

January 5, 2019

```
In [0]: import numpy as np
        import csv
        import matplotlib.pyplot as plt
        import collections
```

1.

```
In [0]: np.zeros(10)
```

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

2.

```
In [0]: np.ones(10)
```

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

3.

```
In [0]: np.full(10, 5, dtype=np.double)
```

```
Out[0]: array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

4.

```
In [0]: np.arange(10, 51, dtype=np.int)
```

```
Out[0]: array([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])
```

5.

```
In [0]: np.arange(10, 51, 2, dtype=np.int)
```

```
Out[0]: array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
               44, 46, 48, 50])
```

6.

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

```
Out[0]: array([[0, 1, 2],
               [3, 4, 5],
               [6, 7, 8]])
```

7.

```
In [0]: np.identity(3)

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

8.

```
In [0]: np.random.rand(1)

Out[0]: array([0.23328093])
```

9.

```
In [0]: np.random.randn(25)

Out[0]: array([-1.31253936, -0.7117603 , -1.02274287,  0.70450243,  1.66631903,
                0.93469688,  0.23288701, -0.48979583,  0.29036785,  1.7981496 ,
                0.00385936, -1.02732271,  1.55236709, -0.12808617, -0.33605099,
               -0.5480573 ,  0.12178067,  0.42221609,  0.24362337, -0.28516367,
               -3.84640444, -0.71157125, -0.8391718 ,  1.25489077, -1.46899959])
```

10.

```
In [0]: np.linspace(0.01, 1, 100).reshape((10, 10))

Out[0]: 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.  ]])
```

11.

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

Out[0]: 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.        ])
```

12

```
In [0]: mat = np.arange(1,26).reshape(5,5)
        mat

Out[0]: array([[ 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]])
```

12 a)

```
In [0]: mat[2:, 1:]

Out[0]: array([[12, 13, 14, 15],
               [17, 18, 19, 20],
               [22, 23, 24, 25]])
```

12 b)

```
In [0]: mat[3,4]

Out[0]: 20
```

12 c)

```
In [0]: mat[:3, 1:2]

Out[0]: array([[ 2],
               [ 7],
               [12]])
```

12 d)

```
In [0]: mat[4]

Out[0]: array([21, 22, 23, 24, 25])
```

12 e)

```
In [0]: mat[3:]

Out[0]: array([[16, 17, 18, 19, 20],
               [21, 22, 23, 24, 25]])
```

13.

```
In [0]: np.sum(mat)

Out[0]: 325
```

14.

```
In [0]: np.std(mat)
```

```
Out[0]: 7.211102550927978
```

15.

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

```
Out[0]: array([55, 60, 65, 70, 75])
```

16.

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

```
Out[0]: array([11., 12., 13., 14., 15.])
```

17.

```
In [0]: np.median(mat, axis=1)
```

```
Out[0]: array([ 3.,  8., 13., 18., 23.])
```

18.

```
In [0]: np.average(mat, axis=1)
```

```
Out[0]: array([ 3.,  8., 13., 18., 23.])
```

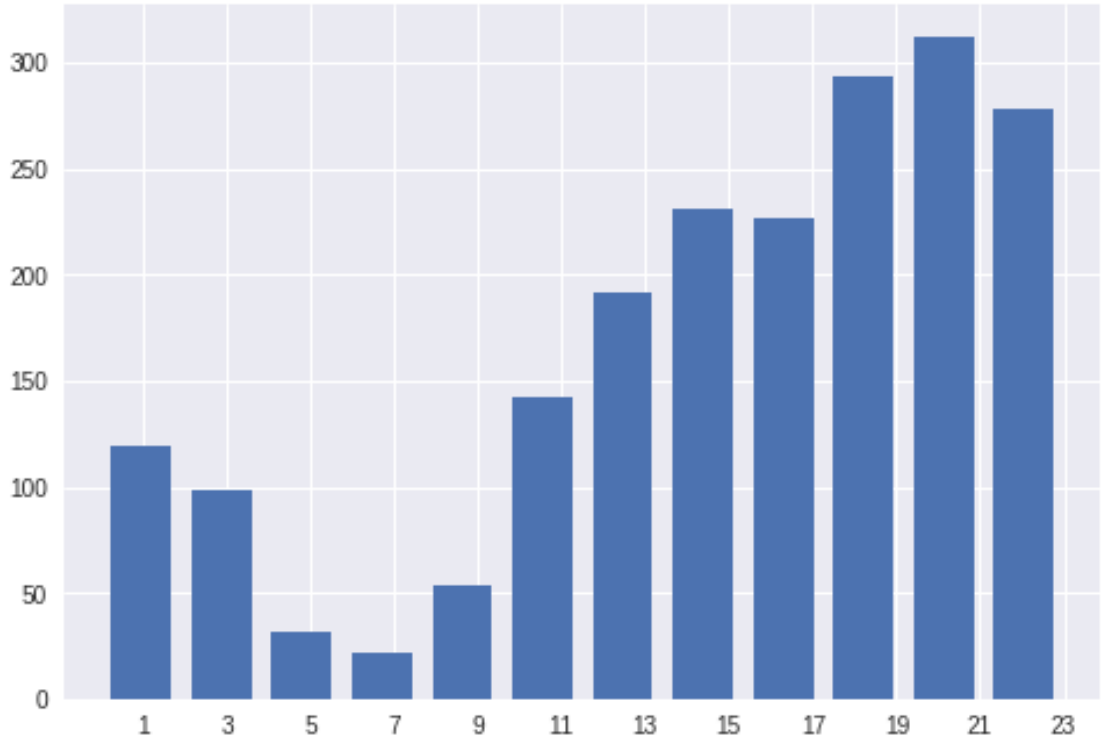
19.

```
In [0]: def get_occurances_hours(filename):
          occurances_hours = []
          with open(filename, 'r') as file:
            reader = csv.reader(file, delimiter=',')
            time_index = next(reader).index('Time Occurred')
            for row in reader:
              hour = int(row[time_index][:2])+1 if row[time_index][2:] != '00' else int(row[ti
              hour %= 24
              occurances_hours.append(hour)
          return occurances_hours

        occurances_hours = get_occurances_hours('Crime_Data_from_2010_small.csv')

        plt.hist(occurances_hours, rwidth=0.75, bins=12)
        plt.xticks(list(range(1,24,2)))
```

```
Out[0]: ([<matplotlib.axis.XTick at 0x7f23145b7940>,
          <matplotlib.axis.XTick at 0x7f23146623c8>,
          <matplotlib.axis.XTick at 0x7f23145cf128>,
          <matplotlib.axis.XTick at 0x7f2314589dd8>,
          <matplotlib.axis.XTick at 0x7f2314593518>,
          <matplotlib.axis.XTick at 0x7f2314593c88>,
          <matplotlib.axis.XTick at 0x7f2314598438>,
          <matplotlib.axis.XTick at 0x7f2314598ba8>,
          <matplotlib.axis.XTick at 0x7f231459e358>,
          <matplotlib.axis.XTick at 0x7f231459eac8>,
          <matplotlib.axis.XTick at 0x7f23145a1278>,
          <matplotlib.axis.XTick at 0x7f23145a19e8>],
         <a list of 12 Text xticklabel objects>)
```



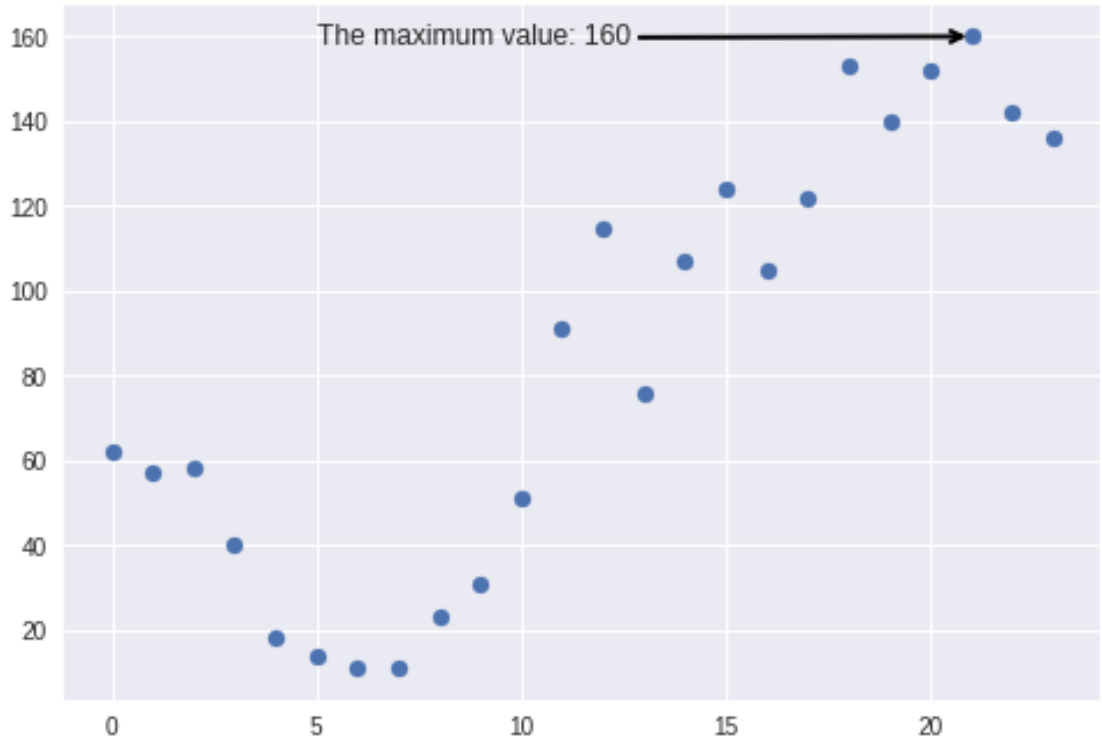
20.

```
In [0]: cnt = collections.Counter(occurances_hours)
        all_common = cnt.most_common()
        most_common = cnt.most_common(1)[0]

        x = [c[0] for c in all_common]
        y = [c[1] for c in all_common]

        plt.scatter(x,y)
        plt.annotate(f'The maximum value: {max(y)}', xytext=(5,158),xy=most_common, arrowprops=
```

Out[0]: Text(5,158,'The maximum value: 160')



21.

```
In [0]: # funkcja concatenate nie modyfikuje macierzy przekazanych jako argumenty
        # kopi naley natomias wykona w przypadku macierzy b, gdy reshape jest metod inplace

        def gaussian(A,b):
          reshape_b = np.copy(b).reshape((A.shape[0], 1))

          return np.concatenate((A, reshape_b), axis=1)

        def eliminate(Ae):
          for i in range(Ae.shape[0]-1):
            # jeli warto diagonalna wiersza wynosi 0 zostanie on zamieniony z innym wierszem j
            if (Ae[i,i] == 0):
              for j in range(i+1, Ae.shape[0]):
                if (Ae[j,j] != 0):
                  # Ae[i, i:], Ae[j, i:] = Ae[j, i:], Ae[i, i:] niepoprawny wynik - czym moe t
                  tmp = Ae[i, i:]
                  Ae[i, i:] = Ae[j, i:]
                  Ae[j, i:] = tmp
                  break;
```

```
            for j in range(i+1, Ae.shape[0]):
              rate = Ae[j,i]/Ae[i,i]
              Ae[j, i:] = Ae[j, i:] - rate*Ae[i, i:]

          return Ae

        def back(A, b):
          x = np.zeros(A.shape[0])
          for i in range(A.shape[0]-1, -1, -1):
            x[i] = (b[i] - np.dot(A[i,i+1:],x[i+1:]))/A[i,i]
          return x

        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'Matrix A:\n{A}')

        x = np.ones(A.shape[0])
        print(f'Matrix x:\n{x}')
        b = A @ x.T
        print(f'Matrix b:\n{b}')
        Ae = gaussian(A, b)
        print(f'Matrix Ae:\n{Ae}')
        print(f'Matrix Ae after elimination:\n{eliminate(Ae)}')
        print(f'Matrix Ae after elimination part A:\n{Ae[:,:-1]}')
        print(f'Matrix Ae after elimination part b:\n{Ae[:,-1:]}')

        x = back(Ae[:,:-1], Ae[:,-1:])
        print(f'Solution:\n{x}')
```

```
Matrix A:
[[ 1.  2.  3.  4.]
 [ 5.  6. 88. -3.]
 [ 9. 10. 11. -3.]
 [13. 14. 15. 16.]]
Matrix x:
[1. 1. 1. 1.]
Matrix b:
[10. 96. 27. 58.]
Matrix Ae:
[[ 1.  2.  3.  4. 10.]
 [ 5.  6. 88. -3. 96.]
 [ 9. 10. 11. -3. 27.]
 [13. 14. 15. 16. 58.]]
Matrix Ae after elimination:
[[   1.     2.     3.     4.    10. ]
```

```
 [   0.    -4.    73.   -23.    46. ]
 [   0.     0.  -162.     7.  -155. ]
 [   0.     0.     0.    22.5   22.5]]
Matrix Ae after elimination part A:
[[   1.     2.     3.     4. ]
 [   0.    -4.    73.   -23. ]
 [   0.     0.  -162.     7. ]
 [   0.     0.     0.    22.5]]
Matrix Ae after elimination part b:
[[  10. ]
 [  46. ]
 [-155. ]
 [  22.5]]
Solution:
[1. 1. 1. 1.]
```