Numpy Lesson2 - Statistics

https://docs.scipy.org/doc/numpy/reference/routines.statistics.html (https://docs.scipy.org/doc/numpy/reference/routines.statistics.html)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
np.random.seed(123)
x = np.random.randint(100, size=10)
x
```

Out[2]:
```
array([66, 92, 98, 17, 83, 57, 86, 97, 96, 47])
```

In [3]:
```
np.mean(x)
```

Out[3]:
```
73.9
```

In [4]:
```
np.amin(x), np.amax(x)
```

Out[4]:
```
(17, 98)
```

Range

In [5]:
```
# Range of values (max - min)
# ptp - peak to peak

np.ptp(x)
```

Out[5]:
```
81
```

In [6]:
```
# 2-D
np.random.seed(123)
x2 = np.random.randint(100, size=(5,2))
x2
```

Out[6]:
```
array([[66, 92],
       [98, 17],
       [83, 57],
       [86, 97],
       [96, 47]])
```

In [7]:
```
np.amin(x2, axis=0), np.amax(x2, axis=0)
```

Out[7]:
```
(array([66, 17]), array([98, 97]))
```

In [8]:
```
np.ptp(x2, axis=0)
```

Out[8]:
```
array([32, 80])
```

In [9]:
```
np.amin(x2, axis=1), np.amax(x2, axis=1)
```

Out[9]:
```
(array([66, 17, 57, 86, 47]), array([92, 98, 83, 97, 96]))
```

In [10]:
```
np.ptp(x2, axis=1)
```

Out[10]:
```
array([26, 81, 26, 11, 49])
```

Quantiles

In [11]:
```
np.random.seed(123)
x = np.random.randint(100, size=10)
x
```

Out[11]:
```
array([66, 92, 98, 17, 83, 57, 86, 97, 96, 47])
```

In [12]:
```
np.median(x)
```

Out[12]:
```
84.5
```

In [13]:
```
np.quantile(x, 0.5)
```

Out[13]:
```
84.5
```

In [14]:
```
np.sort(x)
```

Out[14]:
```
array([17, 47, 57, 66, 83, 86, 92, 96, 97, 98])
```

In [15]:
```
np.quantile(x, [0, 0.25, 0.5, 0.75, 1])
```

Out[15]:
```
array([17.  , 59.25, 84.5 , 95.  , 98.  ])
```

In [16]:
```
np.quantile(x, [0, 0.25, 0.5, 0.75, 1], interpolation = 'nearest')
```

Out[16]:
```
array([17, 57, 83, 96, 98])
```

In [17]:
```
# 2-D
x2
```

Out[17]:
```
array([[66, 92],
       [98, 17],
       [83, 57],
       [86, 97],
       [96, 47]])
```

In [18]:
```
np.quantile(x2, 0.5, axis=0)
```

Out[18]:
```
array([86., 57.])
```

Percentiles

In [19]:
```
np.random.seed(123)
x = np.random.randint(100, size=10)
x
```

Out[19]:
```
array([66, 92, 98, 17, 83, 57, 86, 97, 96, 47])
```

In [20]:
```
np.sort(x)
```

Out[20]:
```
array([17, 47, 57, 66, 83, 86, 92, 96, 97, 98])
```

In [21]:
```
np.percentile(x, 50)
```

Out[21]:
```
84.5
```

In [22]:
```
np.percentile(x, [0, 25, 50, 75, 100])
```

Out[22]:
```
array([17.  , 59.25, 84.5 , 95.  , 98.  ])
```

In [23]:
```
np.percentile(x, [0, 25, 50, 75, 100], interpolation = 'nearest')
```

Out[23]:
```
array([17, 57, 83, 96, 98])
```

In [24]:
```
# 2-D
x2
```

Out[24]:
```
array([[66, 92],
       [98, 17],
       [83, 57],
       [86, 97],
       [96, 47]])
```

In [25]:
```
np.percentile(x2, 50, axis=0)
```

Out[25]:
```
array([86., 57.])
```

Histograms

In [26]:
```
# np.histogram  - Compute the histogram of a set of data

np.random.seed(123)
x = np.random.randint(100, size=10)
x
```

Out[26]:
```
array([66, 92, 98, 17, 83, 57, 86, 97, 96, 47])
```

In [27]:
```
np.sort(x)
```

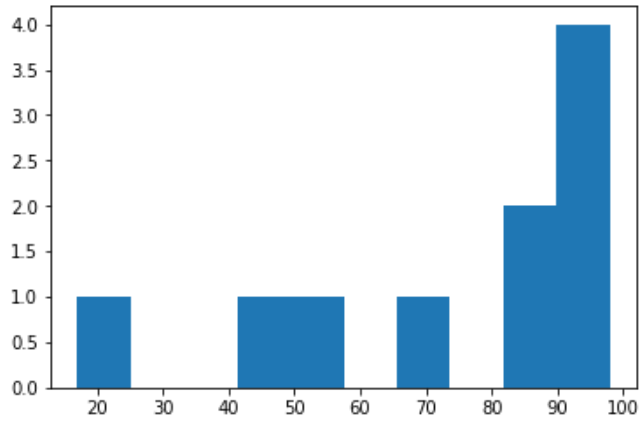
Out[27]:
```
array([17, 47, 57, 66, 83, 86, 92, 96, 97, 98])
```

In [28]:
```
np.histogram(x)
```

Out[28]:
```
(array([1, 0, 0, 1, 1, 0, 1, 0, 2, 4]),
 array([17. , 25.1, 33.2, 41.3, 49.4, 57.5, 65.6, 73.7, 81.8, 89.9, 98. ]))
```

In [29]:
```
plt.hist(x)
```

Out[29]:
```
(array([1., 0., 0., 1., 1., 0., 1., 0., 2., 4.]),
 array([17. , 25.1, 33.2, 41.3, 49.4, 57.5, 65.6, 73.7, 81.8, 89.9, 98. ]),
 <a list of 10 Patch objects>)
```



In [30]:
```
np.histogram(x, bins=3)
```

Out[30]:
```
(array([1, 3, 6]), array([17., 44., 71., 98.]))
```

In [31]:
```
counts, bin_edges = np.histogram(x, bins=3)
```

In [32]:
```
counts
```

Out[32]:
```
array([1, 3, 6])
```

In [33]:
```
bin_edges
```

Out[33]:
```
array([17., 44., 71., 98.])
```

In [34]:
```
bin_edges.size - 1
```

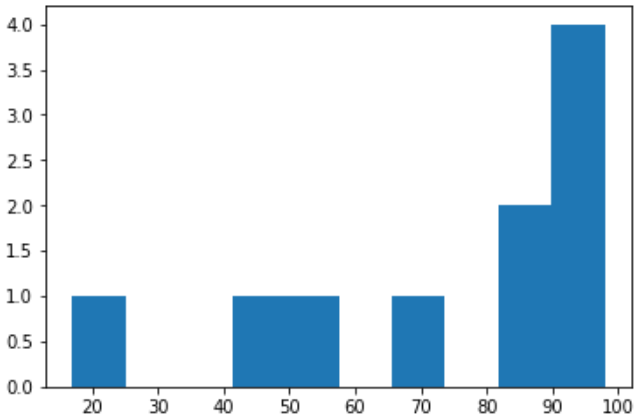
Out[34]:
```
3
```

In [35]:
```
for i in range(bin_edges.size - 1):
    print(bin_edges[i], '-', bin_edges[i+1], '=', counts[i])
```

```
17.0 - 44.0 = 1
44.0 - 71.0 = 3
71.0 - 98.0 = 6
```

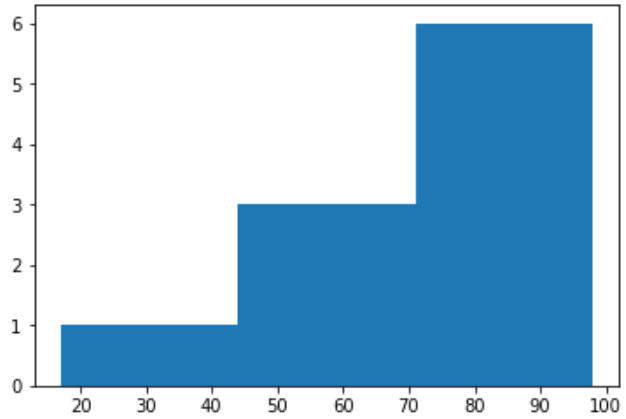
In [36]:
```
plt.hist(x)
```

Out[36]:
```
(array([1., 0., 0., 1., 1., 0., 1., 0., 2., 4.]),
 array([17. , 25.1, 33.2, 41.3, 49.4, 57.5, 65.6, 73.7, 81.8, 89.9, 98. ]),
 <a list of 10 Patch objects>)
```



In [37]:
```
plt.hist(x, bins=3)
```

Out[37]:
```
(array([1., 3., 6.]), array([17., 44., 71., 98.]), <a list of 3 Patch objects>)
```



In [38]:
```
# densities
```

In [39]:
```
x
```

Out[39]:
```
array([66, 92, 98, 17, 83, 57, 86, 97, 96, 47])
```

In [40]:
```
densities, bin_edges = np.histogram(x, bins = 3, density = True)
```

In [41]:
```
densities
```

Out[41]:
```
array([0.0037037 , 0.01111111, 0.02222222])
```

In [42]:
```
bin_edges
```

Out[42]:
```
array([17., 44., 71., 98.])
```

In [43]:
```
np.diff(bin_edges)
```

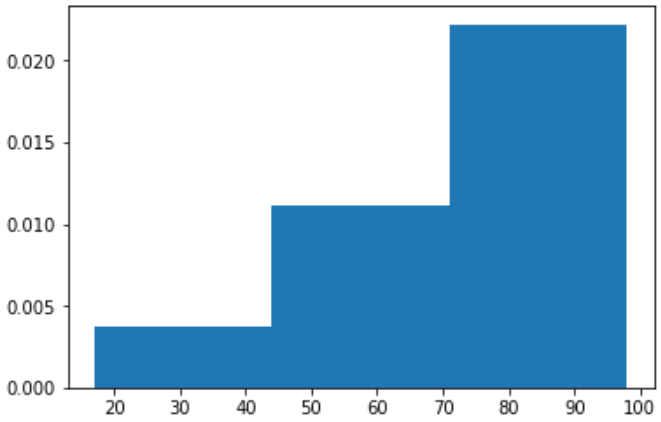
Out[43]:
```
array([27., 27., 27.])
```

In [44]:
```
np.sum(densities * np.diff(bin_edges))
```

Out[44]:
```
1.0
```

In [45]:
```
plt.hist(x, bins = 3, density = True)
```

Out[45]:
```
(array([0.0037037 , 0.01111111, 0.02222222]),
 array([17., 44., 71., 98.]),
 <a list of 3 Patch objects>)
```



Number of occurrences

np.bincount

- Count number of occurrences of each value in array of non-negative ints

In [46]:
```
np.random.seed(567)
x = np.random.randint(100, size=10)
x
```

Out[46]:
```
array([67, 22, 44,  5,  9, 16, 44, 62, 33, 74])
```

In [47]:
```
x2 = x.repeat(2)
x2
```

Out[47]:
```
array([67, 67, 22, 22, 44, 44,  5,  5,  9,  9, 16, 16, 44, 44, 62, 62, 33,
       33, 74, 74])
```

In [48]:
```
np.bincount(x2)
```

Out[48]:
```
array([0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
       0, 2, 0, 0, 0, 0, 0, 0, 2])
```

In [49]:
```
np.bincount(x2).size
```

Out[49]:
```
75
```

In [50]:
```
np.amax(x2) + 1
```

Out[50]:
```
75
```

In [51]:
```
result = np.bincount(x2)
result[result != 0]
```

Out[51]:
```
array([2, 2, 2, 2, 2, 4, 2, 2, 2])
```

In [ ]:
```
```

Digitize

np.digitize

- Return the indices of the bins to which each value in input array belongs

In [52]:
```
np.random.seed(167)
x = np.random.randint(100, size=10)
x[-1] = 25
x
```

Out[52]:
```
array([65, 10, 59,  3, 71, 37, 94, 56,  4, 25])
```

In [53]:
```
bins = np.linspace(0, 100, 5)
bins
```

Out[53]:
```
array([  0.,  25.,  50.,  75., 100.])
```

In [54]:
```
indices = np.digitize(x, bins)
indices
```

Out[54]:
```
array([3, 1, 3, 1, 3, 2, 4, 3, 1, 2])
```

In [55]:
```
for i in range(x.size):
    print('{:5.2f} <= {:3d} < {:5.2f}'.format(\
                                    bins[indices[i] - 1], \
                                    x[i], \
                                    bins[indices[i]]))
```

```
50.00 <=  65 < 75.00
 0.00 <=  10 < 25.00
50.00 <=  59 < 75.00
 0.00 <=   3 < 25.00
50.00 <=  71 < 75.00
25.00 <=  37 < 50.00
75.00 <=  94 < 100.00
50.00 <=  56 < 75.00
 0.00 <=   4 < 25.00
25.00 <=  25 < 50.00
```

Case Study - Bin Smoothing

In [56]:
```
x
```

Out[56]:
```
array([65, 10, 59,  3, 71, 37, 94, 56,  4, 25])
```

In [57]:
```
for i in range(1, bins.size):
    print(x[indices == i])
```

```
[10  3  4]
[37 25]
[65 59 71 56]
[94]
```

In [58]:
```
bin_means = [x[indices == i].mean() for i in range(1, bins.size)]
bin_means
```

Out[58]:
```
[5.666666666666667, 31.0, 62.75, 94.0]
```

In [59]:
```
x_smoothed = np.copy(x)
x_smoothed
```

Out[59]:
```
array([65, 10, 59,  3, 71, 37, 94, 56,  4, 25])
```

In [60]:
```
for i in range(1, bins.size):
    x_smoothed[indices == i] = bin_means[i-1]

x_smoothed
```

Out[60]:
```
array([62,  5, 62,  5, 62, 31, 94, 62,  5, 31])
```

In [61]:
```
 x.dtype.name
```

Out[61]:
```
'int64'
```

In [62]:
```
x_smoothed = (np.copy(x)).astype(float)
x_smoothed
```

Out[62]:
```
array([65., 10., 59.,  3., 71., 37., 94., 56.,  4., 25.])
```

In [63]:
```
for i in range(1, bins.size):
    x_smoothed[indices == i] = bin_means[i-1]

x_smoothed
```

Out[63]:
```
array([62.75      ,  5.66666667, 62.75      ,  5.66666667, 62.75      ,
       31.        , 94.        , 62.75      ,  5.66666667, 31.        ])
```

In [ ]:
```

```