NumPy - Statistical Functions

NumPy has quite a few useful statistical functions for finding minimum, maximum, percentile standard deviation and variance, etc. from the given elements in the array. The functions are explained as follows –

numpy.amin() and numpy.amax()

These functions return the minimum and the maximum from the elements in the given array along the specified axis.

Example

Live Demo

```
import numpy as np

a = np.array([[3,7,5],[8,4,3],[2,4,9]])


print 'Our array is:'

print a

print '\n'


print 'Applying amin() function:'

print np.amin(a,1)

print '\n'


print 'Applying amin() function again:'

print np.amin(a,0)

print '\n'


print 'Applying amax() function:'

print np.amax(a)
```

```
print '\n'



print 'Applying amax() function again:'

print np.amax(a, axis = 0)
```

It will produce the following output –

```
Our array is:
[[3 7 5]
[8 4 3]
[2 4 9]]

Applying amin() function:
[3 3 2]

Applying amin() function again:
[2 4 3]

Applying amax() function:
9

Applying amax() function again:
[8 7 9]
```

numpy.ptp()

The **numpy.ptp()** function returns the range (maximum-minimum) of values along an axis.

Live Demo

```
import numpy as np

a = np.array([[3,7,5],[8,4,3],[2,4,9]])



print 'Our array is:'

print a

print '\n'



print 'Applying ptp() function:'

print np.ptp(a)

print '\n'
```

```
print 'Applying ptp() function along axis 1:'

print np.ptp(a, axis = 1)

print '\n'


print 'Applying ptp() function along axis 0:'

print np.ptp(a, axis = 0)
```

It will produce the following output –

```
Our array is:
[[3 7 5]
[8 4 3]
[2 4 9]]

Applying ptp() function:
7

Applying ptp() function along axis 1:
[4 5 7]

Applying ptp() function along axis 0:
[6 3 6]
```

numpy.percentile()

Percentile (or a centile) is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations fall. The function **numpy.percentile()** takes the following arguments.

```
numpy.percentile(a, q, axis)
```

Where,

Sr.No.	Argument & Description
1	**a** Input array
2	**q** The percentile to compute must be between 0-100

3	**axis** The axis along which the percentile is to be calculated

Example

Live Demo

```
import numpy as np

a = np.array([[30,40,70],[80,20,10],[50,90,60]])


print 'Our array is:'

print a

print '\n'


print 'Applying percentile() function:'

print np.percentile(a,50)

print '\n'


print 'Applying percentile() function along axis 1:'

print np.percentile(a,50, axis = 1)

print '\n'


print 'Applying percentile() function along axis 0:'

print np.percentile(a,50, axis = 0)
```

It will produce the following output –

```
Our array is:
[[30 40 70]
 [80 20 10]
 [50 90 60]]

Applying percentile() function:
50.0

Applying percentile() function along axis 1:
```

```
[ 40. 20. 60.]

Applying percentile() function along axis 0:
[ 50. 40. 60.]
```

numpy.median()

Median is defined as the value separating the higher half of a data sample from the lower half. The **numpy.median()** function is used as shown in the following program.

Example

Live Demo

```
import numpy as np

a = np.array([[30,65,70],[80,95,10],[50,90,60]])


print 'Our array is:'

print a

print '\n'


print 'Applying median() function:'

print np.median(a)

print '\n'


print 'Applying median() function along axis 0:'

print np.median(a, axis = 0)

print '\n'


print 'Applying median() function along axis 1:'

print np.median(a, axis = 1)
```

It will produce the following output –

```
Our array is:
[[30 65 70]
```

```
 [80 95 10]
 [50 90 60]]

Applying median() function:
65.0

Applying median() function along axis 0:
[ 50. 90. 60.]

Applying median() function along axis 1:
[ 65. 80. 60.]
```

numpy.mean()

Arithmetic mean is the sum of elements along an axis divided by the number of elements. The **numpy.mean()** function returns the arithmetic mean of elements in the array. If the axis is mentioned, it is calculated along it.

Example

Live Demo

```
import numpy as np

a = np.array([[1,2,3],[3,4,5],[4,5,6]])


print 'Our array is:'

print a

print '\n'


print 'Applying mean() function:'

print np.mean(a)

print '\n'


print 'Applying mean() function along axis 0:'

print np.mean(a, axis = 0)

print '\n'


print 'Applying mean() function along axis 1:'
```

```
print np.mean(a, axis = 1)
```

It will produce the following output –

```
Our array is:
[[1 2 3]
 [3 4 5]
 [4 5 6]]

Applying mean() function:
3.66666666667

Applying mean() function along axis 0:
[ 2.66666667 3.66666667 4.66666667]

Applying mean() function along axis 1:
[ 2. 4. 5.]
```

numpy.average()

Weighted average is an average resulting from the multiplication of each component by a factor reflecting its importance. The **numpy.average()**function computes the weighted average of elements in an array according to their respective weight given in another array. The function can have an axis parameter. If the axis is not specified, the array is flattened.

Considering an array [1,2,3,4] and corresponding weights [4,3,2,1], the weighted average is calculated by adding the product of the corresponding elements and dividing the sum by the sum of weights.

Weighted average = (1*4+2*3+3*2+4*1)/(4+3+2+1)

Example

Live Demo

```
import numpy as np

a = np.array([1,2,3,4])



print 'Our array is:'

print a

print '\n'
```

```
print 'Applying average() function:'

print np.average(a)

print '\n'


# this is same as mean when weight is not specified

wts = np.array([4,3,2,1])


print 'Applying average() function again:'

print np.average(a,weights = wts)

print '\n'


# Returns the sum of weights, if the returned parameter is set to True.

print 'Sum of weights'

print np.average([1,2,3, 4],weights = [4,3,2,1], returned = True)
```

It will produce the following output –

```
Our array is:
[1 2 3 4]

Applying average() function:
2.5

Applying average() function again:
2.0

Sum of weights
(2.0, 10.0)
```

In a multi-dimensional array, the axis for computation can be specified.

Example

Live Demo

```
import numpy as np

a = np.arange(6).reshape(3,2)


print 'Our array is:'
```

```
print a

print '\n'



print 'Modified array:'

wt = np.array([3,5])

print np.average(a, axis = 1, weights = wt)

print '\n'



print 'Modified array:'

print np.average(a, axis = 1, weights = wt, returned = True)
```

It will produce the following output –

```
Our array is:
[[0 1]
 [2 3]
 [4 5]]

Modified array:
[ 0.625 2.625 4.625]

Modified array:
(array([ 0.625, 2.625, 4.625]), array([ 8., 8., 8.]))
```

Standard Deviation

Standard deviation is the square root of the average of squared deviations from mean. The formula for standard deviation is as follows –

```
std = sqrt(mean(abs(x - x.mean())**2))
```

If the array is [1, 2, 3, 4], then its mean is 2.5. Hence the squared deviations are [2.25, 0.25, 0.25, 2.25] and the square root of its mean divided by 4, i.e., sqrt (5/4) is 1.1180339887498949.

Example

Live Demo

```
import numpy as np

print np.std([1,2,3,4])
```

It will produce the following output −

```
1.1180339887498949
```

Variance

Variance is the average of squared deviations, i.e., **mean(abs(x - x.mean())**2)**. In other words, the standard deviation is the square root of variance.

Example

Live Demo

```
import numpy as np

print np.var([1,2,3,4])
```

It will produce the following output −

```
1.25
```