**Array Aggregation Functions**

import numpy as np

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

print("a :",a)

sum=np.sum(a)

print("sum :",sum)

product=np.prod(a)

print("product :",product)

mean=np.mean(a)

print("mean :",mean)

standard\_deviation=np.std(a)

print("standard\_deviation :",standard\_deviation)

variance=np.var(a)

print("variance :",variance)

minimum=np.min(a)

print("minimum value :",minimum)

maximum=np.max(a)

print("maximum value :",maximum)

minimum\_index=np.argmin(a)

print("minimum index :",minimum\_index)

maximum\_index=np.argmax(a)

print("maximum-index :",maximum\_index)

median=np.median(a)

print("median :",median)

Output

a : [1 2 3 4 5]

sum : 15

product : 120

mean : 3.0

standard\_deviation : 1.4142135623730951

variance : 2.0

minimum value : 1

maximum value : 5

minimum index : 0

maximum-index : 4

median : 3.0

**Vectorized Operations**

**import** numpy **as** np  
print(np.sum(np.arange(10000)))  
print(2 \* np.array([2, 3, 4]))  
print(np.array([10.2, 3.5, -0.9]) - np.array([8.2, 3.5, 6.5]))  
print(np.dot(np.array([1, -3, 4]), np.array([2, 0, 1])))

oputput:

49995000

[4 6 8]

[ 2. 0. -7.4]

6

**Use Map, Filter, Reduce and Lambda**

*#lamdafunction*square = **lambda** n: n \*\* 2  
print(square(5))  
*#map function***def** square(n):  
 **return** n \*\* 2  
squares = map(square, range(1, 10, 2))  
squares  
print(list(squares))  
*#filter function*nums = [1, 34, 23, 56, 89, 44, 92]  
odds = list(filter(**lambda** x: x % 2 != 0, nums))  
print(odds)  
*#reduce function***from** functools **import** reduce  
nums = [1, 2, 3, 4, 5]  
summ = reduce(**lambda** x, y: x + y, nums)  
print(summ)

Output

25

[1, 9, 25, 49, 81]

[1, 23, 89]

15