```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
```

```
x = [8.0, 1, 2.5, 4, 28.0]
x_with_nan = [8.0, 1, 2.5, math.nan, 4, 28.0]
x
```

```
[8.0, 1, 2.5, 4, 28.0]
```

```
x_with_nan
```

```
[8.0, 1, 2.5, nan, 4, 28.0]
```

```
math.isnan(np.nan), np.isnan(math.nan)
```

```
(True, True)
```

```
y, y_with_nan = np.array(x), np.array(x_with_nan)
z, z_with_nan = pd.Series(x), pd.Series(x_with_nan)
y
```

```
array([ 8. ,  1. ,  2.5,  4. , 28. ])
```

```
y_with_nan
```

```
array([ 8. ,  1. ,  2.5,  nan,  4. , 28. ])
```

```
z
```

```
0     8.0
1     1.0
2     2.5
3     4.0
4    28.0
dtype: float64
```

```
z_with_nan
```

```
0     8.0
1     1.0
2     2.5
3     NaN
4     4.0
5    28.0
dtype: float64
```

```
mean_ = sum(x) / len(x)
mean_
```

```
8.7
```

```
mean_ = statistics.mean(x)
mean_
```

```
8.7
```

```
mean_ = statistics.fmean(x)
mean_
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-13-64d722b4ddb4> in <module>()
----> 1 mean_ = statistics.fmean(x)
      2 mean_

AttributeError: module 'statistics' has no attribute 'fmean'
```

SEARCH STACK OVERFLOW

```
mean_ = statistics.mean(x_with_nan)
mean_
```

```
nan
```

```
mean_ = np.mean(y)
>>> mean_
```

```
8.7
```

```
mean_ = y.mean()
mean_
```

```
8.7
```

```
np.mean(y_with_nan)
```

```
nan
```

```
y_with_nan.mean()
```

```
nan
```

```
np.nanmean(y_with_nan)
```

```
8.7
```

```
mean_ = z.mean()
mean_
```

```
8.7
```

```
z_with_nan.mean()
```

```
8.7
```

```
0.2 * 2 + 0.5 * 4 + 0.3 * 8
```

```
4.8
```

```
x = [8.0, 1, 2.5, 4, 28.0]
w = [0.1, 0.2, 0.3, 0.25, 0.15]
wmean = sum(w[i] * x[i] for i in range(len(x))) / sum(w)
wmean
```

```
6.95
```

```
wmean = sum(x_ * w_ for (x_, w_) in zip(x, w)) / sum(w)
wmean
```

```
6.95
```

```
y, z, w = np.array(x), pd.Series(x), np.array(w)
wmean = np.average(y, weights=w)
wmean
```

```
6.95
```

```
wmean = np.average(z, weights=w)
wmean
```

```
6.95
```

```
(w * y).sum() / w.sum()
```

```
6.95
```

```
w = np.array([0.1, 0.2, 0.3, 0.0, 0.2, 0.1])
(w * y_with_nan).sum() / w.sum()
```

```
nan
```

```
np.average(y_with_nan, weights=w)
```

```
nan
```

```
np.average(z_with_nan, weights=w)
```

```
nan
```

```
hmean = len(x) / sum(1 / item for item in x)
hmean
```

```
2.7613412228796843
```

```
hmean = statistics.harmonic_mean(x)
hmean
```

```
2.7613412228796843
```

```
statistics.harmonic_mean(x_with_nan)
```

```
nan
```

```
statistics.harmonic_mean([1, 0, 2])
```

```
0
```

```
scipy.stats.hmean(y)
```

```
2.7613412228796843
```

```
scipy.stats.hmean(z)
```

```
2.7613412228796843
```

```
gmean = 1
for item in x:
    gmean *= item

gmean **= 1 / len(x)
gmean
```

```
4.677885674856041
```

```
scipy.stats.gmean(y)
```

```
4.67788567485604
```

```
scipy.stats.gmean(z)
```

```
4.67788567485604
```

Median

```
n = len(x)
if n % 2:
  median_ = sorted(x)[round(0.5*(n-1))]
else:
     x_ord, index = sorted(x), round(0.5 * n)
     median_ = 0.5 * (x_ord[index-1] + x_ord[index])
median_
```

```
4
```

```
median_ = statistics.median(x)
median_
```

```
4
```

```
median_ = statistics.median(x[:-1])
median_
```

```
3.25
```

```
statistics.median_low(x[:-1])
```

```
2.5
```

```
statistics.median_high(x[:-1])
```

```
4
```

```
statistics.median(x_with_nan)
```

```
6.0
```

```
statistics.median_low(x_with_nan)
```

```
4
```

```
statistics.median_high(x_with_nan)
```

```
8.0
```

```
median_ = np.median(y)
median_
```

```
4.0
```

```
median_ = np.median(y[:-1])
```

```
median_
```

```
3.25
```

```
np.nanmedian(y_with_nan)
```

```
4.0
```

```
np.nanmedian(y_with_nan[:-1])
```

```
3.25
```

```
z.median()
```

```
4.0
```

```
z_with_nan.median()
```

```
4.0
```

Mode

```
u = [2, 3, 2, 8, 12]
mode_ = max((u.count(item), item) for item in set(u))[1]
mode_
```

```
2
```

```
statistics.mode([2, math.nan, 2])
```

```
2
```

```
statistics.mode([2, math.nan, 0, math.nan, 5])
```

```
nan
```

```
u, v = np.array(u), np.array(v)
mode_ = scipy.stats.mode(u)
mode_
```

```
ModeResult(mode=array([2]), count=array([2]))
```

```
mode_ = scipy.stats.mode(v)
mode_
```

```
ModeResult(mode=array([12]), count=array([3]))
```

```
mode_.mode
```

```
array([12])
```

```
mode_.count
```

```
array([3])
```

```
u, v, w = pd.Series(u), pd.Series(v), pd.Series([2, 2, math.nan])
u.mode()
```

```
0    2
dtype: int64
```

```
v.mode()
```

```
0    12
1    15
dtype: int64
```

```
w.mode()
```

```
0    2.0
dtype: float64
```

Variance

```
n = len(x)
mean_ = sum(x) / n
var_ = sum((item - mean_)**2 for item in x) / (n - 1)
var_
```

```
123.19999999999999
```

```
var_ = statistics.variance(x)
var_
```

```
123.2
```

```
statistics.variance(x_with_nan)
```

```
nan
```

```
var_ = np.var(y, ddof=1)
var_
```

```
123.19999999999999
```

```
var_ = y.var(ddof=1)
var_
```

```
var_
```

```
123.19999999999999
```

```
np.var(y_with_nan, ddof=1)
```

```
nan
```

```
y_with_nan.var(ddof=1)
```

```
nan
```

```
np.nanvar(y_with_nan, ddof=1)
```

```
123.19999999999999
```

```
z.var(ddof=1)
```

```
123.19999999999999
```

```
z_with_nan.var(ddof=1)
```

```
123.19999999999999
```

Standard Deviation

```
std_ = var_ ** 0.5
std_
```

```
11.099549540409285
```

```
std_ = statistics.stdev(x)
std_
```

```
11.099549540409287
```

```
np.std(y, ddof=1)
```

```
11.099549540409285
```

```
y.std(ddof=1)
```

```
11.099549540409285
```

```
np.std(y_with_nan, ddof=1)
```

```
nan
```

```
y_with_nan.std(ddof=1)
```

```
nan
```

```
np.nanstd(y_with_nan, ddof=1)
```

```
11.099549540409285
```

```
z.std(ddof=1)
```

```
11.099549540409285
```

```
z_with_nan.std(ddof=1)
```

```
11.099549540409285
```

Correlation

```
x = list(range(-10, 11))
y = [0, 2, 2, 2, 2, 3, 3, 6, 7, 4, 7, 6, 6, 9, 4, 5, 5, 10, 11, 12, 14]
x_, y_ = np.array(x), np.array(y)
x__, y__ = pd.Series(x_), pd.Series(y_)
```

Covariance

```
n = len(x)
mean_x, mean_y = sum(x) / n, sum(y) / n
cov_xy = (sum((x[k] - mean_x) * (y[k] - mean_y) for k in range(n))
          / (n - 1))
cov_xy
```

```
19.95
```

```
cov_matrix = np.cov(x_, y_)
cov_matrix
```

```
array([[38.5       , 19.95      ],
       [19.95      , 13.91428571]])
```

```
x_.var(ddof=1)
```

```
38.5
```

```
y_.var(ddof=1)
```

```
13.914285714285711
```

```
cov_xy = cov_matrix[0, 1]
cov_xy
```

```
19.95
```

```
cov_xy = cov_matrix[1, 0]
cov_xy
```

```
19.95
```

```
cov_xy = x__.cov(y__)
cov_xy
```

```
19.95
```

```
cov_xy = y__.cov(x__)
cov_xy
```

```
19.95
```

Correlation Coefficient

```
var_x = sum((item - mean_x)**2 for item in x) / (n - 1)
var_y = sum((item - mean_y)**2 for item in y) / (n - 1)
std_x, std_y = var_x ** 0.5, var_y ** 0.5
r = cov_xy / (std_x * std_y)
r
```

```
0.861950005631606
```

```
r, p = scipy.stats.pearsonr(x_, y_)
r
```

```
0.8619500056316061
```

```
p
```

```
5.122760847201135e-07
```

```
corr_matrix = np.corrcoef(x_, y_)
corr_matrix
```

```
array([[1.        , 0.86195001],
       [0.86195001, 1.        ]])
```

```
r = corr_matrix[0, 1]
r
```

```
0.8619500056316061
```

```
r = corr_matrix[1, 0]
```

```
r
```

```
0.861950005631606
```

```
scipy.stats.linregress(x_, y_)
```

```
LinregressResult(slope=0.5181818181818181, intercept=5.714285714285714,
rvalue=0.861950005631606, pvalue=5.122760847201164e-07, stderr=0.06992387660074979,
intercept_stderr=0.4234100995002589)
```

```
result = scipy.stats.linregress(x_, y_)
r = result.rvalue
r
```

```
0.861950005631606
```

```
r = x__.corr(y__)
r
```

```
0.8619500056316061
```

```
r = y__.corr(x__)
r
```

```
0.861950005631606
```

✓ 0s completed at 2:06 PM