

# NUMPY BASICS

## NUMERICAL PYTHON

Prepared by:

Gyro A. Madrona

Electronics Engineer

L12-numpy-basics.ipynb

Python 3.13.0

notebook > L12-numpy-basics.ipynb > NumPy Basics > #!pip install numpy --upgrade

+ Code + Markdown ▶ Run All ⏮ Restart ⏻ Clear All Outputs 🔍 View data 📄 Jupyter Variables 📖 Outline ...

## NumPy Basics

Data Analyst: Gyro A. Madrona

Department: Electrical Engineering

Install and update NumPy

Python

```
1 #!pip install numpy --upgrade
```

Install and update scipy

Python

```
1 #!pip install scipy --upgrade
```

Python

```
1 import numpy as np
2 from scipy import stats
```

## 1D Array

Python

```
1 # Creating 1-dimensional array
2 array_a = np.array([1,2,3])
3 array_a
```

Open 'array\_a' in Data Wrangler

Python

```
1 # Size of an array
2 np.shape(array_a)
```

Python

```
1 array_b = np.array([4,5,6])
2 array_b
```

Open 'array\_b' in Data Wrangler

## 2D Array

Python

```
1 # Creating 2-dimensional array
2 my_array = np.array([[1,2,3],[4,5,6]])
3 my_array
```

Open 'my\_array' in Data Wrangler

Python

```
1 # Size of an array
2 np.shape(my_array)
```

Python

```
1 # Transpose of an array
2 t_array = my_array.T
3 t_array
```

Open 't\_array' in Data Wrangler

Python

```
1 np.shape(t_array)
```

## Measures of Central Tendency

reference: L3-Measures of Central Tendency

### Fruit Price List

```
1 # Fruit price list
2 fruits = np.array([120,60,85,150,200])
3 fruits
```

Open 'fruits' in Data Wrangler

Python

```
1 # Mean of fruit prices
2 fruits_mean = np.mean(fruits)
3 fruits_mean
```

Open

Python

```
1 # Median of fruit prices
2 fruits_median = np.median(fruits)
3 fruits_median
```

Open

Python

```
1 # Sort fruit prices
2 fruits_sorted = np.sort(fruits)
3 fruits_sorted
```

Open 'fruits\_sorted' in Data Wrangler

Python

```
1 # Mode of fruit prices
2 stats.mode(fruits)
```

Open

Python

### Voltage Response

```
1 # Voltage response data
2 voltage = np.array([
3     [1,2,3,4,5,6,7,8],
4     [12,5,9.1,3.3,24,18.5,15.2,np.nan],
5     [2.8,4.5,6,9,11.7,14.8,17.3,20]
6 ])
7 voltage
```

Open 'voltage' in Data Wrangler

Python

```
1 # Size of array
2 np.shape(voltage)
```

Open

Python

```
1 # Mean of voltage data
2 voltage_mean = np.mean(voltage,axis=1)
3 voltage_mean
```

Open 'voltage\_mean' in Data Wrangler

Python

```
1 # Mean of voltage data ignoring any NaN values
2 voltage_mean = np.nanmean(voltage,axis=1)
3 voltage_mean
```

Open 'voltage\_mean' in Data Wrangler

Python

```
1 # Transpose voltage data
2 voltage = voltage.T
3 voltage
```

Open 'voltage' in Data Wrangler

Python

```
1 # Size of array
2 np.shape(voltage)
```

Open

Python

```
1 # Mean of voltage data ignoring any NaN values
2 voltage_mean = np.nanmean(voltage,axis=0)
3 voltage_mean
```

Open 'voltage\_mean' in Data Wrangler

Python

```
1 # Median of voltage data ignoring any NaN values
2 voltage_median = np.nanmedian(voltage,axis=0)
3 voltage_median
```

Open 'voltage\_median' in Data Wrangler

Python



# NUMPY BASICS

## NUMERICAL PYTHON

Prepared by:

**Gyro A. Madrona**

Electronics Engineer

```
1 # Sort voltage data
2 voltage_sorted = np.sort(voltage,axis=0)
3 voltage_sorted
```

Open 'voltage\_sorted' in Data Wrangler

Python

```
1 # Mode of voltage data
2 voltage_mode = stats.mode(voltage,axis=0)
3 voltage_mode
```

Python

## Measures of Variability

reference: L4-Measures of Variability

### Exam Performance

```
1 # Exam performance data
2 grade = np.array([3.5,6.7,7,7.4,7.8,8.2,8.5,8.8,9,9.1,9.4,9.8])
3 grade
```

Open 'grade' in Data Wrangler

Python

```
1 # Maximum grade of exam data
2 grade_max = np.max(grade)
3 grade_max
```

Python

```
1 # Minimum grade of exam data
2 grade_min = np.min(grade)
3 grade_min
```

Python

```
1 # Range of exam data
2 grade_range = grade_max - grade_min
3 grade_range
```

Python

```
1 # First quartile (Q1) of exam data
2 grade_q1 = np.percentile(grade,25) # 25%
3 grade_q1
```

Python

```
1 # Second quartile (Q2) of exam data
2 grade_q3 = np.percentile(grade,75) # 75%
3 grade_q3
```

Python

```
1 # Interquartile range (IQR) of exam data
2 grade_iqr = grade_q3 - grade_q1
3 grade_iqr
```

Python

```
1 # Population variance
2 grade_var = np.var(grade)
3 grade_var
```

Python

```
1 # Sample variance
2 grade_var = np.var(grade,ddof=1)
3 grade_var
```

Python

```
1 # Population standard deviation
2 grade_std = np.std(grade)
3 grade_std
```

Python

```
1 # Sample standard deviation
2 grade_std = np.std(grade,ddof=1)
3 grade_std
```

Python

## Ice Cream Price List

```
1 price = np.array([
2     [3.5,4,3.75,4.25,3.9,4.1,3.6,4.5,3.8,4.15],
3     [203,232,217.5,246.5,226.2,237.8,208.8,261,220.4,240.7]
4 ])
5 price = price.T
6 price
```

Open 'price' in Data Wrangler

Python

```
1 # Mean of USD
2 usd_mean = np.mean(price,axis=0)[0] # [0] 1st column
3 usd_mean
```

Open

Python

```
1 # Standard deviation of USD
2 usd_std = np.std(price,ddof=1,axis=0)[0] # [0] 1st column
3 usd_std
```

Open

Python

```
1 # USD Coefficient of variation
2 usd_cv = usd_std/usd_mean
3 usd_cv
```

Open

Python

```
1 # Mean of PHP
2 php_mean = np.mean(price,axis=0)[1] # [0] 2nd column
3 php_mean
```

Open

Python

```
1 # Standard deviation of PHP
2 php_std = np.std(price,ddof=1,axis=0)[1] # [1] 2nd column
3 php_std
```

Open

Python

```
1 # PHP Coefficient of variation
2 php_cv = php_std/php_mean
3 php_cv
```

Open

Python

## Pooled Standard Deviation

```
1 battery = np.array([
2     ['A','A','A','A','A','B','B','B','B','B','B','C','C','C','C','C'],
3     [12.5,12.8,12.7,13.3,12.6,13.5,14.1,13.9,14.3,13.7,11.8,11.9,12.1,12.2,11.6]
4 ])
5 battery = battery.T
6 battery
```

Open 'battery' in Data Wrangler

Python

```
1 # Extract rows where the 1st column is 'A'
2 model_a = battery[battery[:,0]=='A']
3 model_a
```

Open 'model\_a' in Data Wrangler

Python

```
1 # Extract rows where the 1st column is 'B'
2 model_b = battery[battery[:,0]=='B']
3 model_b
```

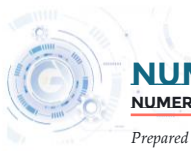
Open 'model\_b' in Data Wrangler

Python

```
1 # Extract rows where the 1st column is 'C'
2 model_c = battery[battery[:,0]=='C']
3 model_c
```

Open 'model\_c' in Data Wrangler

Python



## NUMPY BASICS

### NUMERICAL PYTHON

Prepared by:

**Gyro A. Madrona**

Electronics Engineer

```
1 # Convert string to float
2 voltage_a = model_a[:,1].astype(float)
3 voltage_b = model_b[:,1].astype(float)
4 voltage_c = model_c[:,1].astype(float)
```

Python

```
1 # Average variance
2 a_var = np.var(voltage_a, ddof=1)
3 b_var = np.var(voltage_b, ddof=1)
4 c_var = np.var(voltage_c, ddof=1)
5
6 ave_var = np.mean([a_var, b_var, c_var])
7 ave_var
```

Python

```
1 # Pooled standard deviation is the square root of average variance
2 pooled_std = np.sqrt(ave_var)
3 pooled_std
```

Python