NUMPY BASICS

NUMERICAL PYTHON

Prepared by:
Gyro A. Madrona
Electronics Engineer

L12-numpy-basics.ipynb

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

+ Code + Markdown | Run All | Restart | Clear All Outputs | View data | Jupyter Variables | Outline | ... | Python 3.13.0

NumPy Basics

Data Analyst: Gyro A. Madrona
Department: Electrical Engineering

Install and update NumPy

```python
#%pip install numpy --upgrade
```

Python

Install and update scipy

```python
#%pip install scipy --upgrade
```

Python

```python
import numpy as np
from scipy import stats
```

Python

1D Array

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

Open 'array_a' in Data Wrangler

Python

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

Python

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

Open 'array_b' in Data Wrangler

Python

2D Array

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

Open 'my_array' in Data Wrangler

Python

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

Python

```python
# Transpose of an array
t_array = my_array.T
t_array
```

Open 't_array' in Data Wrangler

Python

```python
np.shape(t_array)
```

Python

NUMPY BASICS

NUMERICAL PYTHON

Prepared by:
Gyro A. Madrona
Electronics Engineer

Measures of Central Tendency

reference: L3-Measures of Central Tendency

Fruit Price List

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

Open 'fruits' in Data Wrangler — Python

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

Python

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

Python

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

Open 'fruits_sorted' in Data Wrangler — Python

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

Python

Voltage Response

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

Open 'voltage' in Data Wrangler — Python

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

Python

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

Open 'voltage_mean' in Data Wrangler — Python

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

Open 'voltage_mean' in Data Wrangler — Python

```
# Transpose voltage data
voltage = voltage.T
voltage
```

Open 'voltage' in Data Wrangler — Python

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

Python

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

Open 'voltage_mean' in Data Wrangler — Python

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

Open 'voltage_median' in Data Wrangler — Python

NUMPY BASICS

NUMERICAL PYTHON

Prepared by:

Gyro A. Madrona

Electronics Engineer

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

Open 'voltage_sorted' in Data Wrangler — Python

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

Python

Measures of Variability

reference: L4-Measures of Variability

Exam Performance

```
# Exam performance data
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])
grade
```

Open 'grade' in Data Wrangler — Python

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

Python

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

Python

```
# Range of exam data
grade_range = grade_max - grade_min
grade_range
```

Python

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

Python

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

Python

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

Python

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

Python

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

Python

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

Python

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

Python

NUMPY BASICS

NUMERICAL PYTHON

Prepared by:
Gyro A. Madrona
Electronics Engineer

Ice Cream Price List

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

Open 'price' in Data Wrangler — Python

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

Python

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

Python

```
# USD Coefficient of variation
usd_cv = usd_std/usd_mean
usd_cv
```

Python

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

Python

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

Python

```
# PHP Coefficient of variation
php_cv = php_std/php_mean
php_cv
```

Python

Pooled Standard Deviation

```
battery = np.array([
    ['A','A','A','A','A','B','B','B','B','B','C','C','C','C','C'],
    [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]
])
battery = battery.T
battery
```

Open 'battery' in Data Wrangler — Python

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

Open 'model_a' in Data Wrangler — Python

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

Open 'model_b' in Data Wrangler — Python

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

Open 'model_c' in Data Wrangler — Python

NUMPY BASICS

NUMERICAL PYTHON

Prepared by:

Gyro A. Madrona

Electronics Engineer

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

[ ] Python

```python
# Average variance
a_var = np.var(voltage_a, ddof=1)
b_var = np.var(voltage_b, ddof=1)
c_var = np.var(voltage_c, ddof=1)

ave_var = np.mean([a_var,b_var,c_var])
ave_var
```

[ ] Python

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

[ ] Python