# Lec-07 in Python

Dr. Sushil Sharma

1. NumPy
2. Matplotlib
3. Pandas

https://tinyurl.com/tut-07

# NumPy (Numerical Python)

Open source Python library that's widely used across all disciplines of science.

The NumPy library contains multidimensional array data structures, such as the homogeneous, N-dimensional ndarray, and a large library of functions that operate efficiently on these data structures.

**Why use Numpy (& not lists)?**

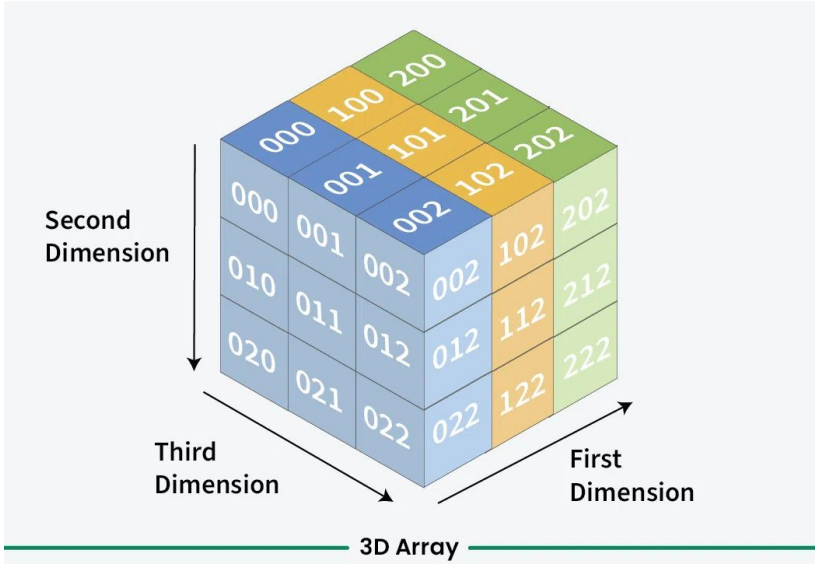
- improve speed,
- reduce memory consumption, and
- offer a high-level syntax for performing a variety of common processing tasks

| | | | |
|---|---|---|---|
| 1 | 5 | 2 | 0 |
| 8 | 3 | 6 | 1 |
| 1 | 7 | 2 | 9 |

# What is an “array”?

| | | | |
|---|---|---|---|
| 1 | 5 | 2 | 0 |

In computer programming, an array is a structure for storing and retrieving data.




Restrictions on `ndarray`:

- All elements of the array must be of the same type of data.
- Once created, the total size of the array can’t change.
- The shape must be “rectangular”, not “jagged”; e.g., each row of a two-dimensional array must have the same number of columns.

# Basics

```
import numpy as np

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

print(arr)

print(np.__version__)

print(type(arr))
```

```
import numpy as np

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

print(arr[2] + arr[3])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
```

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, -1])
```

# Slicing *[start:stop:step]*

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[4:])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[:4])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[-3:-1])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[::2])
```

```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 1:4])
```

# Two Cute Functions

- linspace allows you to specify the number of steps
- arange allows you to specify the size of the steps

`np.linspace(start, stop, num, ...)`

where:

- **start**: The starting value of the sequence
- **stop**: The end value of the sequence
- **num**: the number of values to generate

`np.arange(start, stop, step, ...)`

where:

- **start**: The starting value of the sequence
- **stop**: The end value of the sequence
- **step**: The spacing between values

# Q1 - Marks Matrix

You have marks of 3 students in 4 subjects:

```
[[80, 70, 90, 60],
 [75, 85, 95, 65],
 [60, 70, 80, 90]]
```

Using NumPy:

1. Store this in an array
2. Print the marks of the 2nd student
3. Print all marks of subject 3
4. Print the highest mark

# Q2 - Slice Without Loop

From this array:

```
[10, 20, 30, 40, 50, 60, 70]
```

Extract only the even-positioned values using slicing.

# Matplotlib - A Visualization Tool

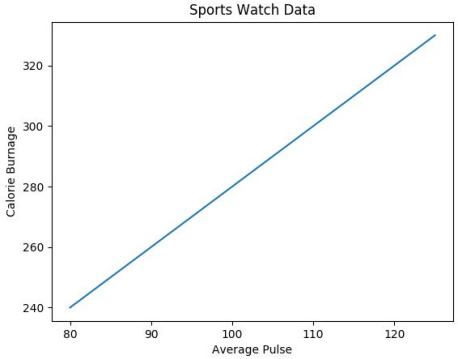
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

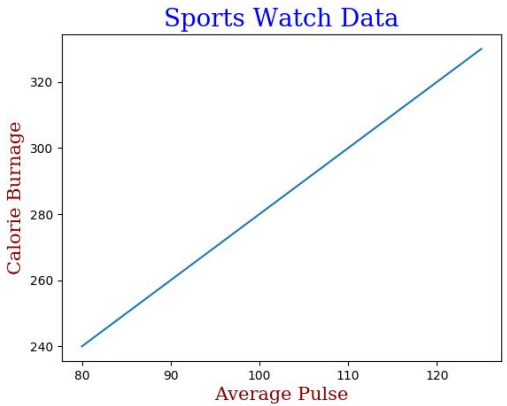
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o')
plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120,
125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310,
320, 330])

plt.plot(x, y)
#plt.plot(x, y+1) #=> Try this

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.show()
```



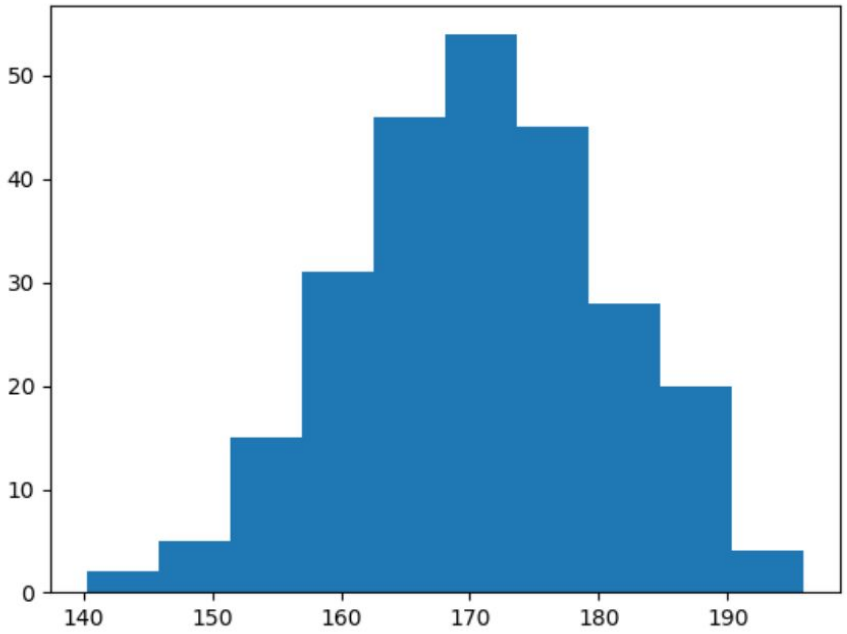

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.array([80, 85, 90, 95, 100, 105, 110, 115,
120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310,
320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 =
{'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

plt.plot(x, y)
plt.show()
```

# Subplots in Matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([1,2,3,4])
temp = np.array([22, 24, 19, 23])
sales = np.array([200, 300, 250, 400])

plt.figure(figsize=(5,10))

plt.subplot(2, 1, 1)
plt.plot(x, temp)
plt.title("Temperature")

plt.subplot(2, 1, 2)
plt.plot(x, sales)
plt.title("Sales")

plt.tight_layout()
plt.show()
```

**Histogram** - graph showing the number of observations within each given interval.

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



# Q3 - Temperature Plot

You have daily temperatures for a week:

```
[22, 24, 19, 23, 25, 21, 20]
```

Plot:

- Day number on X-axis
- Temperature on Y-axis
- Title: "Weekly Temperature"

# Q4 - Compare Two Students

Marks of two students:

```
A = [70, 75, 80, 85]

B = [60, 65, 78, 88]
```

**Try** plot both on the same graph so you can visually compare performance.

# Q5 - Debug This Graph

This code gives a wrong graph. Why?

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

y = np.array([10, 20, 30])

plt.plot(x, y)
```

Fix it and explain.

# Q6 - Kinematics

You are given:

```
speed = [10, 20, 30, 40]

time = [1, 2, 3, 4]

distance = [10, 40, 90, 160]
```

Plot:

- Speed vs Time in the first subplot
- Distance vs Time in the second subplot

# Pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

```
import pandas as pd

print(pd.__version__)
```

# Series - Column in a table

It is a one-dimensional array holding data of any type.

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

# DataFrames - A 2D Array => Table with rows & columns

```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

```python
#refer to the row index:
print(df.loc[0])
```

```python
#use a list of indexes:
print(df.loc[[0, 1]])
```

```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data,
         index = ["day1", "day2", "day3"])

print(df)
```

```python
#refer to the named index:
print(df.loc["day2"])
```

Dataframe = Table
Series = Column of a Table

## Let’s try these Special Attributes & Methods

1. `df.shape` => Shape of the dataframe => Number of Rows & Columns
2. `df.columns` => Shows the column names
3. `df.dtypes` => Shows the data types of each column
4. `df.describe()` => Statistical Summary
5. `df.tail()` => Last 5 Rows
6. `df.sort_values("col_name", ascending=False)`

```
df = pd.read_csv('sample_data/california_housing_train.csv')
```

# Read CSV

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

```
import pandas as pd

pd.options.display.max_rows = 9999

df = pd.read_csv('data.csv')

print(df)
```

The number of rows returned is defined in Pandas option settings.

You can check your system's maximum rows with the `pd.options.display.max_rows` statement.

# Read JSON

JSON = Python Dictionary

JSON objects have the same format as Python dictionaries.

```
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

```
import pandas as pd

data = {
  "Duration":{
    "0":60,
    "1":60,
    "2":60,
    "3":45,
    "4":45,
    "5":60
  },
  "Pulse":{
    "0":110,
    "1":117,
    "2":103,
    "3":109,
    "4":117,
    "5":102
  },
```

```
  "Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    "3":175,
    "4":148,
    "5":127
  },
  "Calories":{
    "0":409,
    "1":479,
    "2":340,
    "3":282,
    "4":406,
    "5":300
  }
}

df = pd.DataFrame(data)
```

# Analyzing DataFrames

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(10))
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

# Data Cleaning = Fixing bad data in your data set

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

# Let’s Create the Dataset First

**Let’s go to the Colab, & run it together**

```
import pandas as pd

data = {
    "Duration": [60, 60, None, 45, 200, 60, 45, 300],
    "Calories": [420, 380, 390, None, 450, 380, 390, 450],
    "Date": ["2024-01-01", "2024-01-02", "2024/01/03", "2024-01-04",
             "2024-01-05", "2024-01-02", "2024-01-03", "invalid"]
}
df = pd.DataFrame(data)
df
```

# 1. Empty Cells

```
import pandas as pd

df = pd.read_csv('data.csv')

new_df = df.dropna()

print(new_df.to_string())
```

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna({"Calories": 130}, inplace=True)
```

```
import pandas as pd

df = pd.read_csv('data.csv')

df.dropna(inplace = True)

print(df.to_string())
```

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna(130, inplace = True)
```

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].median()

df.fillna({"Calories": x}, inplace=True)
```

Mean = the average value (the sum of all values divided by number of values).

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mean()

df.fillna({"Calories": x}, inplace=True)
```

Median = the value in the middle, after you have sorted all values ascending.

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mode()[0]

df.fillna({"Calories": x}, inplace=True)
```

Mode = the value that appears most frequently.

# 2. Wrong Format

```
import pandas as pd

df = pd.read_csv('data.csv')

df['Date'] = pd.to_datetime(df['Date'], format='mixed')

print(df.to_string())
```

```
df.dropna(subset=['Date'], inplace = True)
```

# 3. Wrong Data

```
df.loc[7, 'Duration'] = 45
```

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
```

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

# 4. Removing Duplicates

Returns `True` for every row that is a duplicate, otherwise `False`:

```
print(df.duplicated())
```

Remove all duplicates:

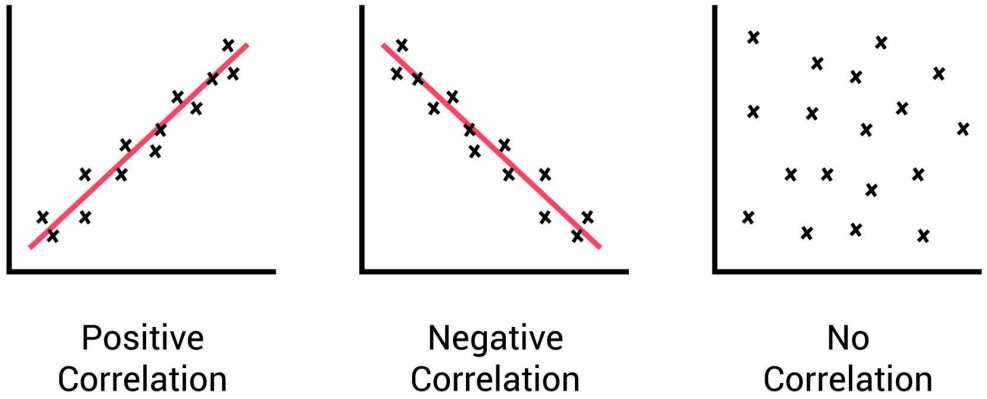
```
df.drop_duplicates(inplace = True)
```

# Correlations - relationship between each column in your data set using df.corr() method.

Python | data.csv

```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df.corr())
```

```
             Duration     Pulse  Maxpulse  Calories
Duration     1.000000 -0.155408  0.009403  0.922721
Pulse       -0.155408  1.000000  0.786535  0.025120
Maxpulse     0.009403  0.786535  1.000000  0.203814
Calories     0.922721  0.025120  0.203814  1.000000
```



Positive Correlation

Negative Correlation

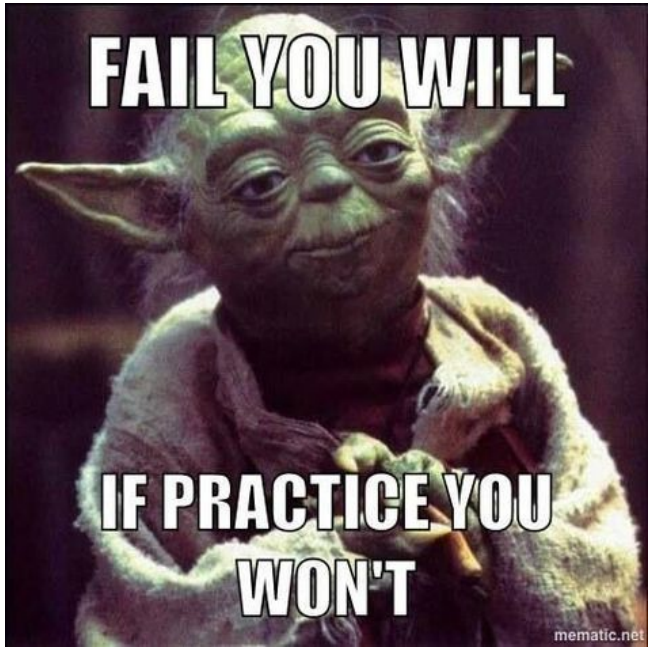
No Correlation

# Plotting in Pandas

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot()

plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```

df = pd.read_csv("/content/sample_data/california_housing_train.csv")




- `longitude`
- `latitude`
- `housing_median_age`
- `total_rooms`
- `total_bedrooms`
- `population`
- `households`
- `median_income`
- `median_house_value`

Let’s Practice Together, Some things we will learn along the way

1. Print the shape of the dataset
2. Display column names
3. Show data types
4. Print the last 5 rows
5. Show a statistical summary

6. Print only median_income and median_house_value
7. Print the row at index 100
8. Print houses with median_income > 5
9. Count how many rows have housing_median_age > 30

10. Find the average house value

11. Find the maximum & minimum median income

12. Find the top 5 most expensive houses

13. Find the correlation between income and house value

14. Check for missing values

15. Verify if there are duplicate rows

16. Create a new column:

rooms_per_household = total_rooms / households

17. Plot Median Income vs Median House Value
18. Plot a histogram of Median House Value
19. Plot latitude vs longitude (scatter)
20. Create 2 subplots:
    - Income vs House Value
    - Population vs House Value

# Thank You