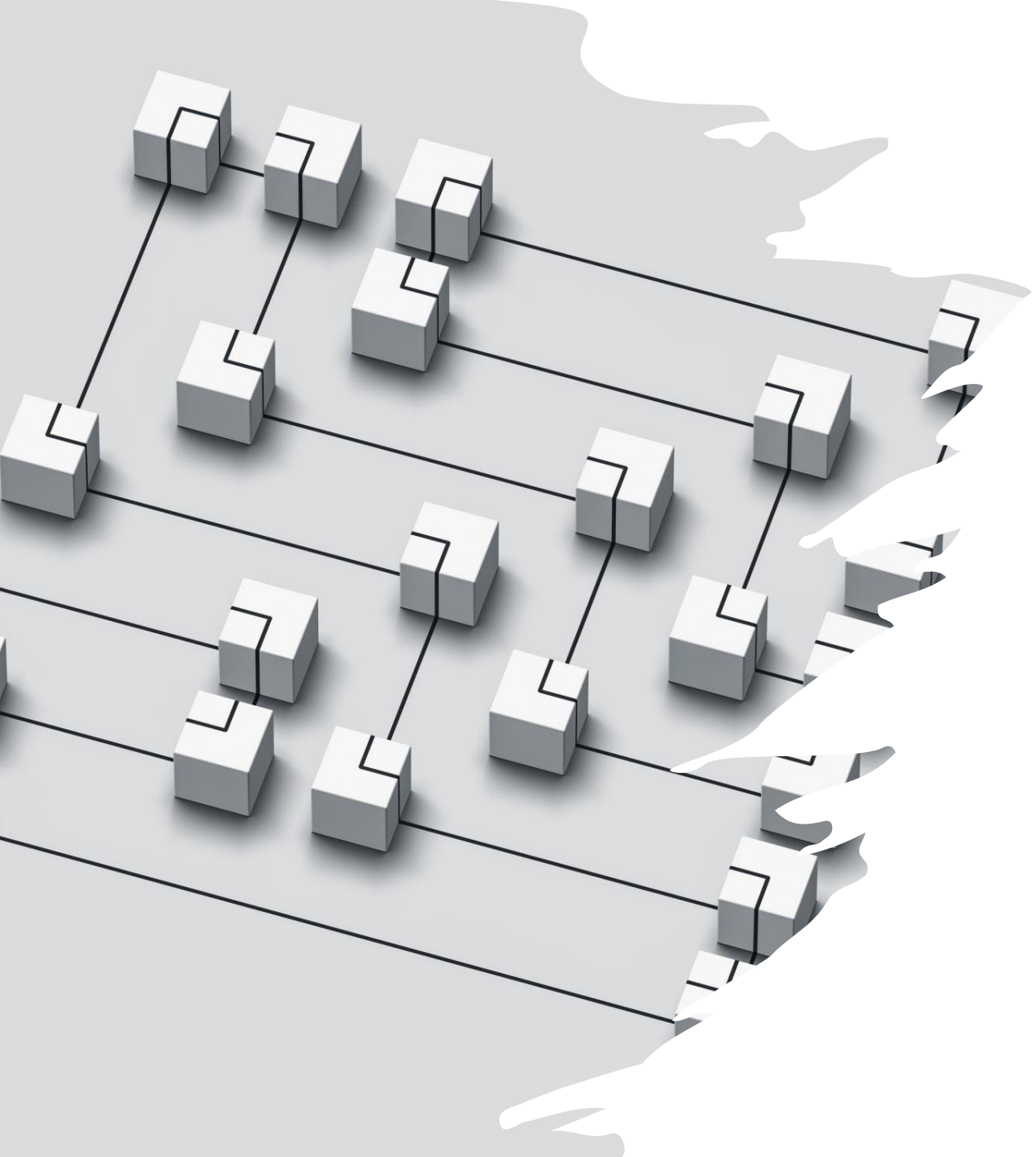
DATA SCIENCE INTERNSHIP

Session 2

HDLC TECHNOLOGIES



Create a DataFrame with Pandas

- A data frame is a structured representation of data.
- Let's define a data frame with 3 columns and 5 rows with fictional numbers:

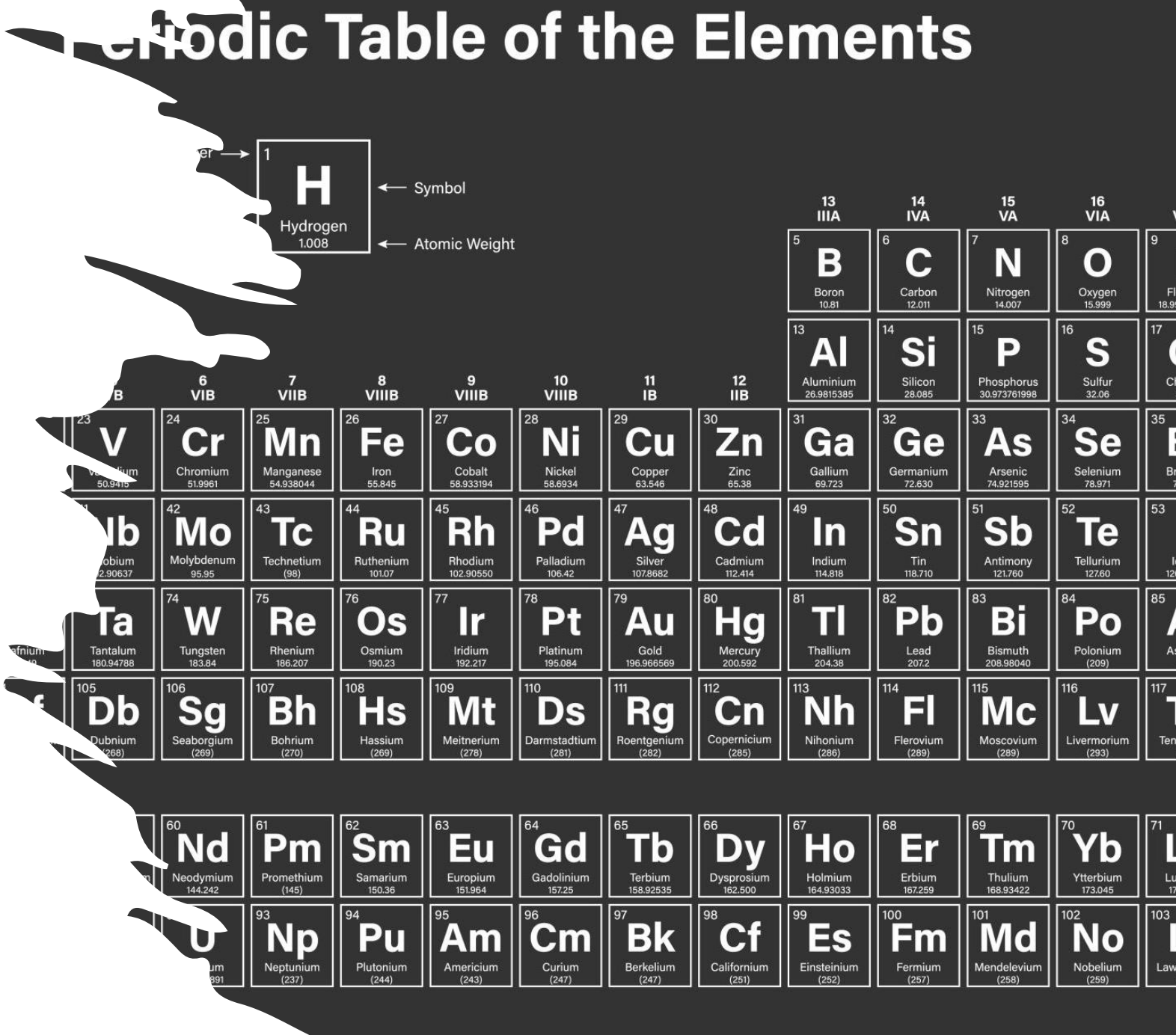
```
import pandas as pd
d = {'col1': [1, 2, 3, 4, 7], 'col2':[4, 5, 6, 9, 5], 'col3': [7, 8, 12, 1, 11]}
df = pd.DataFrame(data=d)
print(df)
```

Output

		col1	col2	col3
1. row	0	1	4	7
2. row	1	2	5	8
3. row	2	3	6	12
4. row	3	4	9	1
5. row	4	7	5	11

Count the number of columns:

- `count_column = df.shape[1]`
  `print(count_column)`



Count the number of rows:

- ```
  count_row = df.shape[0]
  print(count_row)
  ```
  ```

Data Science Functions

- The Sports Watch Data Set

Duration	Average_Pulse	Max_Pulse	Calorie_Burnage	Hours_Work	Hours_Sleep
30	80	120	240	10	7
30	85	120	250	10	7
45	90	130	260	8	7
45	95	130	270	8	7
45	100	140	280	0	7
60	105	140	290	7	8
60	110	145	300	7	8
60	115	145	310	8	8
75	120	150	320	0	8
75	125	150	330	8	8

- The data set above consists of 6 variables, each with 10 observations:
- **Duration** - How long lasted the training session in minutes?
- **Average_Pulse** - What was the average pulse of the training session? This is measured by beats per minute
- **Max_Pulse** - What was the max pulse of the training session?
- **Calorie_Burnage** - How much calories were burnt on the training session?
- **Hours_Work** - How many hours did we work at our job before the training session?
- **Hours_Sleep** - How much did we sleep the night before the training session?

The max() function

- ```
  Average_pulse_max =
  max(80, 85, 90, 95, 100, 105, 110, 115, 120, 12
  5)

  print (Average_pulse_max)
  ```
  ```

The min() function

- ```
  Average_pulse_min =
  min(80, 85, 90, 95, 100, 105, 110, 115, 120, 12
  5)

  print (Average_pulse_min)
  ```
  ```

The mean() function

```
import numpy as np

Calorie_burnage
= [240, 250, 260, 270, 280, 290, 300, 310, 320,
 330]

Average_calorie_burnage
= np.mean(Calorie_burnage)

print(Average_calorie_burnage)
```

Data Science - Data Preparation

- Before analyzing data, a Data Scientist must extract the data, and make it clean and valuable.
- Extract and Read Data With Pandas
- Before data can be analyzed, it must be imported/extracted.

- ```
  import pandas as pd

  health_data =
  pd.read_csv("data.csv")

  print(health_data)
  ```

- ```
  print(health_data.head())
  ```

Data Cleaning

- Look at the imported data. As you can see, the data are "dirty" with wrongly or unregistered values:
- There are some blank fields
- Average pulse of 9 000 is not possible
- 9 000 will be treated as non-numeric, because of the space separator
- One observation of max pulse is denoted as "AF", which does not make sense
- So, we must clean the data in order to perform the analysis.

	Duration	Average_Pulse	Max_Pulse	Calorie_Burnage	Hours_Work	Hours_Sleep
0	30.0	80	120	240.0	10.0	7.0
1	45.0	85	120	250.0	10.0	7.0
2	45.0	90	130	260.0	8.0	7.0
3	60.0	95	130	270.0	8.0	7.0
4	60.0	100	140	280.0	0.0	7.0
5	NaN	NaN	NaN	NaN	NaN	NaN
6	60.0	105	140	290.0	7.0	8.0
7	60.0	110	145	300.0	7.0	8.0
8	45.0	NaN	AF	NaN	8.0	8.0
9	45.0	115	145	310.0	8.0	8.0
10	60.0	120	150	320.0	0.0	8.0
11	60.0	9 000	130	NaN	NaN	8.0
12	45.0	125	150	330.0	8.0	8.0

Remove Blank Rows

- `health_data.dropna(axis=0,inplace=True)`

```
print(health_data)
```

	Duration	Average_Pulse	Max_Pulse	Calorie_Burnage	Hours_Work	Hours_Sleep
0	30.0	80	120	240.0	10.0	7.0
1	45.0	85	120	250.0	10.0	7.0
2	45.0	90	130	260.0	8.0	7.0
3	60.0	95	130	270.0	8.0	7.0
4	60.0	100	140	280.0	0.0	7.0
6	60.0	105	140	290.0	7.0	8.0
7	60.0	110	145	300.0	7.0	8.0
9	45.0	115	145	310.0	8.0	8.0
10	60.0	120	150	320.0	0.0	8.0
12	45.0	125	150	330.0	8.0	8.0

Data Categories

- To analyze data, we also need to know the types of data we are dealing with.
- Data can be split into three main categories:

1. **Numerical** - Contains numerical values. Can be divided into two categories:
    1. Discrete: Numbers are counted as "whole". Example: You cannot have trained 2.5 sessions, it is either 2 or 3
    2. Continuous: Numbers can be of infinite precision. For example, you can sleep for 7 hours, 30 minutes and 20 seconds, or 7.533 hours
2. **Categorical** - Contains values that cannot be measured up against each other. Example: A color or a type of training
3. **Ordinal** - Contains categorical data that can be measured up against each other. Example: School grades where A is better than B and so on

- By knowing the type of your data, you will be able to know what technique to use when analyzing them.

Data Types

- `print(health_data.info())`

We see that this data set has two different types of data:

- Float64
- Object

We cannot use objects to calculate and perform analysis here. We must convert the type object to float64 (float64 is a number with a decimal in Python).

```
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Duration         12 non-null     float64
 1   Average_Pulse    11 non-null     object
 2   Max_Pulse        12 non-null     object
 3   Calorie_Burnage  10 non-null     float64
 4   Hours_Work       11 non-null     float64
 5   Hours_Sleep      12 non-null     float64
dtypes: float64(4), object(2)
```

Convert object into float

```
health_data["Average_Pulse"] =
health_data['Average_Pulse'].astype(float
)
health_data["Max_Pulse"]
= health_data["Max_Pulse"].astype(float)
print (health_data.info())
```

```
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Duration         10 non-null     float64
 1   Average_Pulse    10 non-null     float64
 2   Max_Pulse        10 non-null     float64
 3   Calorie_Burnage  10 non-null     float64
 4   Hours_Work       10 non-null     float64
 5   Hours_Sleep      10 non-null     float64
dtypes: float64(6)
```

Analyze the Data

- `print(health_data.describe())`

	Duration	Average_Pulse	Max_Pulse	Calorie_Burnage	Hours_Work	Hours_Sleep
Count	10.0	10.0	10.0	10.0	10.0	10.0
Mean	51.0	102.5	137.0	285.0	6.6	7.5
Std	10.49	15.4	11.35	30.28	3.63	0.53
Min	30.0	80.0	120.0	240.0	0.0	7.0
25%	45.0	91.25	130.0	262.5	7.0	7.0
50%	52.5	102.5	140.0	285.0	8.0	7.5
75%	60.0	113.75	145.0	307.5	8.0	8.0
Max	60.0	125.0	150.0	330.0	10.0	8.0

- **Count** - Counts the number of observations
- **Mean** - The average value
- **Std** - Standard deviation
- **Min** - The lowest value
- **25%**, **50%** and **75%** are percentiles.
- **Max** - The highest value