

# Applied Data Science with Python for Beginners

Lecture 3 - 16 December 2021



Clean data






# Clean data

Data comes often in an untidy form, therefore some data cleaning is necessary

Name	Town
Clara	Frankfurt a.M.
Sarah	Frankfurt am Main
John	Berlin



```
df1['Town'] = df1['Town'].str.replace(  
    'a.M.',  
    'am Main')
```



# Clean data

Data comes often in an untidy form, therefore some data cleaning is necessary

Name	Subject
Clara	Physics
Sarah	physics
John	Math



```
df['Subject'] = df['Subject'].str.lower()
```



# Fill missing values

Name	Score
Clara	10
Sarah	5
John	NaN



```
df['Score'] = df['Score'].fillna(0)
```

# Fill missing values

Name	Score
Clara	10
Sarah	5
John	NaN → 0

```
df['Score'] = df['Score'].fillna(0)
```





# Fill missing values

Name	Score
Clara	10
Sarah	5
John	NaN → 7.5

```
df['Score'] = df['Score'].fillna(df.mean())
```



# Drop missing values

Name	Score
Clara	10
Sarah	5
John	NaN

```
df = df.dropna()
```





# Working with dates





# QUIZ

What kind of data type is this: "27-03-2021" ?

- a) integer   b) float   c) string   d) date

# Transform string to datetime object



```
df['Birthday'] = pd.to_datetime(df['Birthday'])
```

Name	Birthday
Clara	"10/10/1995"
Sarah	"01/10/1999"
John	"03/05/2001"

# Transform string to datetime object



```
df['Birthday'] = pd.to_datetime(df['Birthday'], format="%d/%m/%y")
```

Name	Birthday
Clara	"10/10/1995"
Sarah	"01/10/1999"
John	"03/05/2001"

# Transform string to datetime object



```
df['Birthday'] = pd.to_datetime(df['Birthday'], format="%m-%d-%y")
```

Name	Birthday
Clara	"10-10-1995"
Sarah	"10-01-1999"
John	"05-03-2001"

# Transform string to datetime object



```
df['day'] = df['Birthday'].dt.day  
df['weekday'] = df['Birthday'].dt.weekday  
df['month'] = df['Birthday'].dt.month
```

Name	Birthday	day	weekday	month
Clara	"10-10-1995"	10	1	10
Sarah	"10-01-1999"	1	4	10
John	"05-03-2001"	3	3	5

# Set date as index

date	City	Temperature
"2021-04-20"	'Frankfurt'	10
"2021-04-21"	'Frankfurt'	11
"2021-04-22"	'Frankfurt'	12

```
df = df.set_index(['date'])
```



# Select timespans

When the index is in datetime format, you can access the data in the following way:

date	City	Temperature
"2021-04-20"	'Frankfurt'	10
"2021-04-21"	'Frankfurt'	11
"2021-04-22"	'Frankfurt'	12

```
df['2021'] # all data from 2021  
df['2021-04':'2021-05']  
df['2021':]
```





# Exercise 1

```
df['date'] = pd.to_datetime(df['date'])
df = df.set_index(['date'])
df['2021-04':'2021-05']
df[df['col_name'] == 'some condition']

plt.plot(df['col_name']['2000'])
```





# pandas has a lot of useful functionality

- The official documentation of pandas has more than 3000 pages
- work with missing values
  - fill with mean value
  - interpolate between values
  - fill with last value
- Read data from various sources
  - Excel, CSV, SQL, Stata, SPSS, SAS, HTML Tables from websites
- Windowing functions
  - Moving average,...

# Advanced plots with Seaborn





# seaborn - library

- statistical data visualization tools
- based on matplotlib
- makes it easy to create more sophisticated plots
- best to visualize relations between columns of a dataset



```
import seaborn as sns
```



# Look into a penguins dataset

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

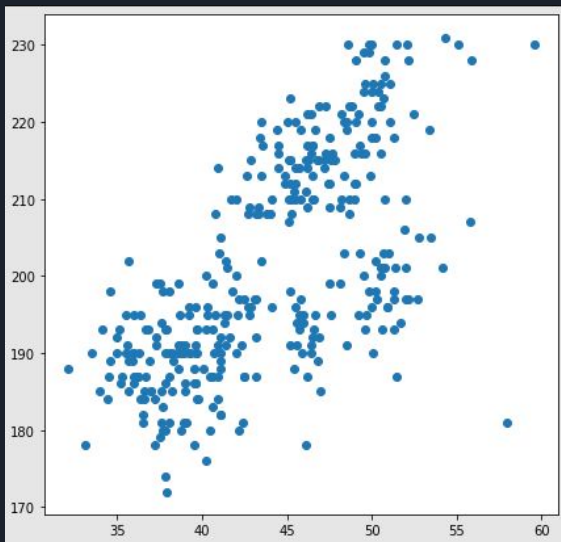
# Look into a penguins dataset

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female

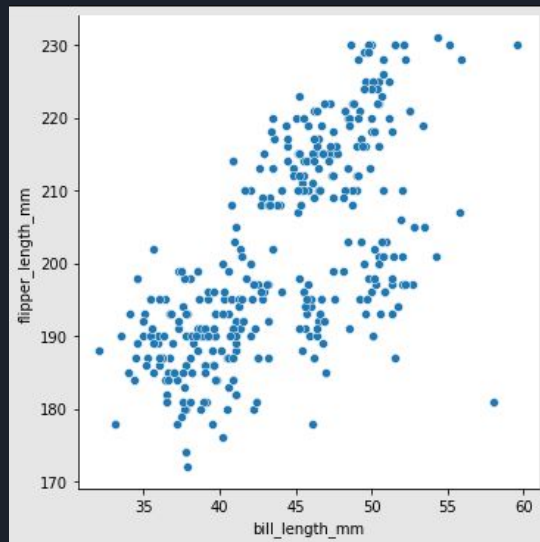


# Matplotlib vs Seaborn

```
plt.scatter(df['bill_length_mm'],df['flipper_length_mm'])
```



```
sns.relplot(df['bill_length_mm'],df['flipper_length_mm'])
```

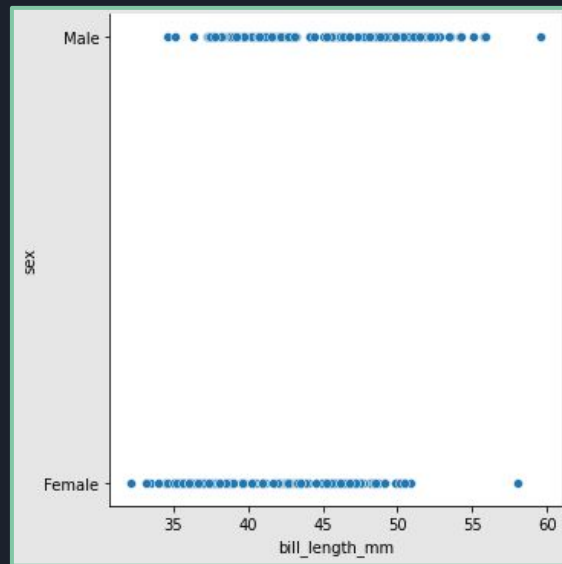


# Matplotlib vs Seaborn

```
plt.scatter(df['bill_length_mm'], df[sex])
```

```
sns.relplot(df['bill_length_mm'], df[sex])
```

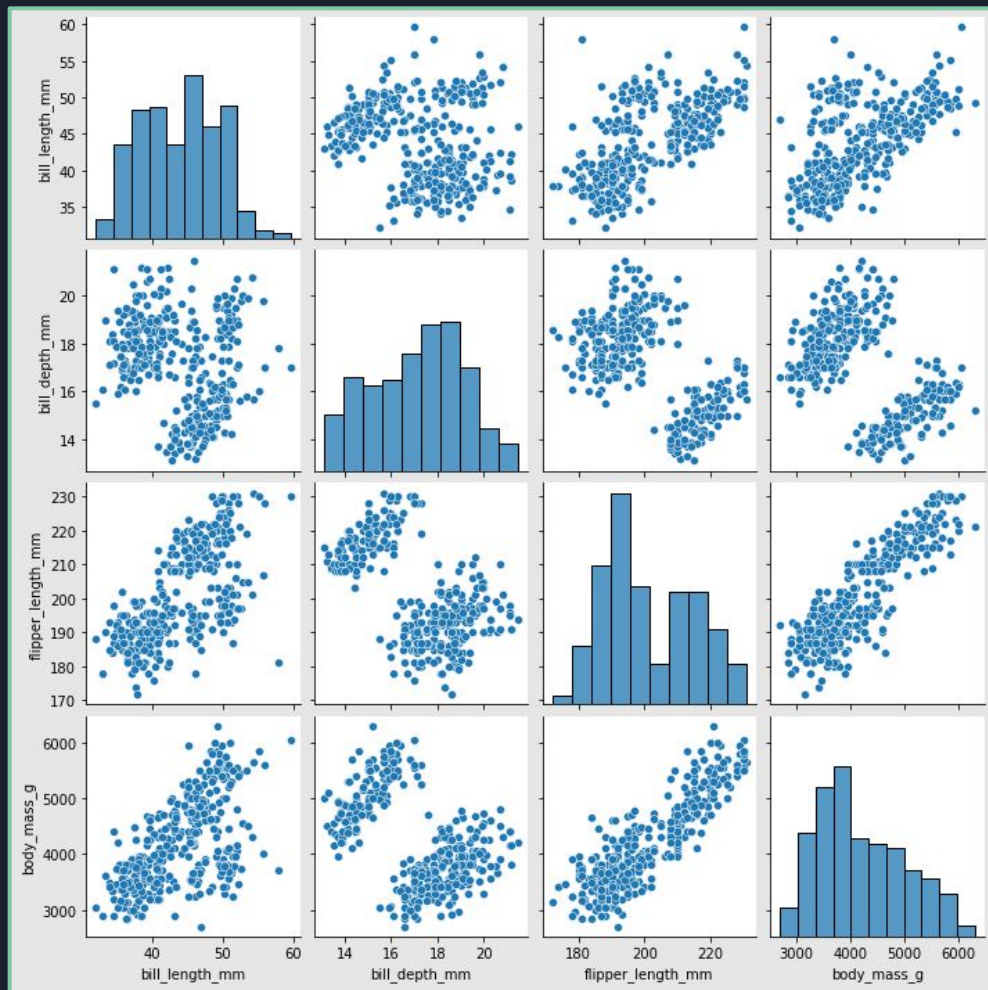
`TypeError`





# Pairplot

```
sns.pairplot(df)
```





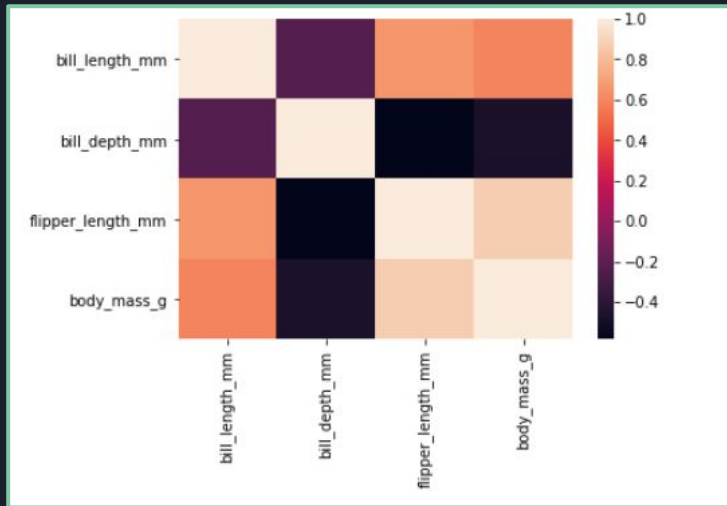
# Visualize correlations

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
bill_length_mm	1.000000	-0.235053	0.656181	0.595110
bill_depth_mm	-0.235053	1.000000	-0.583851	-0.471916
flipper_length_mm	0.656181	-0.583851	1.000000	0.871202
body_mass_g	0.595110	-0.471916	0.871202	1.000000

```
df.corr()
```



# Visualize correlations



```
sns.heatmap(df.corr())
```



## Exercise 2

```
sns.pairplot(df)  
sns.pairplot(df, hue='col_name')
```





# Matplotlib & Seaborn Gallery

- Documentation contains example plots
- Each plot comes with the corresponding code
- <https://matplotlib.org/stable/gallery/index.html>
- <https://seaborn.pydata.org/examples/index.html>

# Intro to modelling



# Linear Regression



Source: <https://avantecture.com/p/phoenixsee/>

How expensive is this house?

Based on information like:

- amount of rooms
- location
- size



# Housing regression

Old collected data

House	Location	Rooms	Price
A	Berlin	6	500k €
B	Frankfurt	8	600k €
C	Berlin	7	300k €

New data

House	Location	Rooms	Price
D	Frankfurt	8	?
E	Frankfurt	5	?
F	Berlin	4	?





# Housing regression

Old collected data

House	Location	Rooms	Price
A	Berlin	6	500k €
B	Frankfurt	8	600k €
C	Berlin	7	300k €

Features

New data

House	Location	Rooms	Price
D	Frankfurt	8	?
E	Frankfurt	5	?
F	Berlin	4	?



# Housing regression

Old collected data

House	Location	Rooms	Price
A	Berlin	6	500k €
B	Frankfurt	8	600k €
C	Berlin	7	300k €

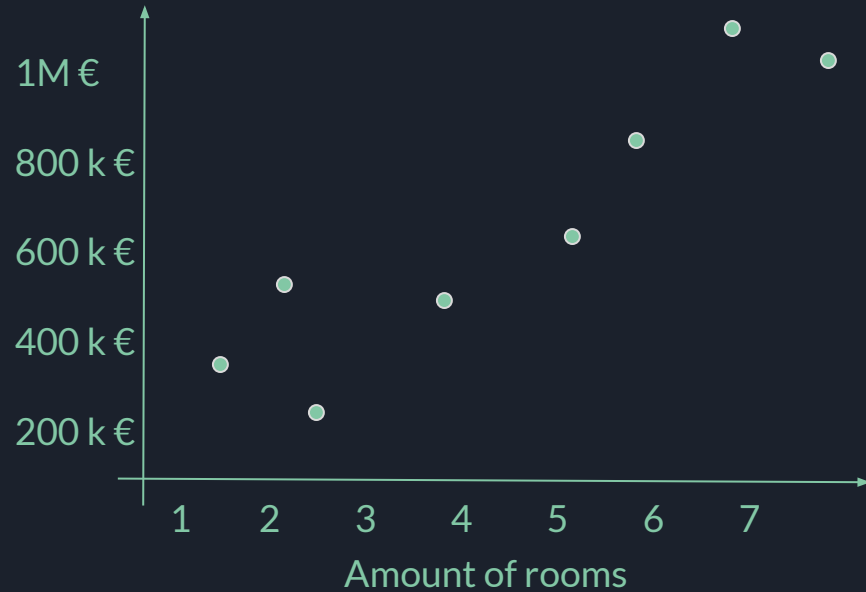
Features

Target column

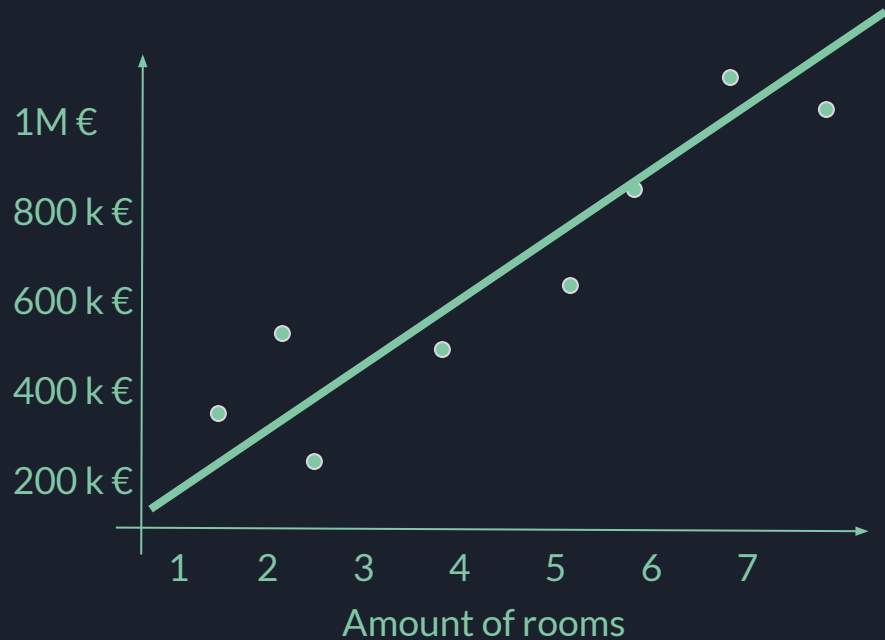
New data

House	Location	Rooms	Price
D	Frankfurt	8	?
E	Frankfurt	5	?
F	Berlin	4	?

# Linear Regression

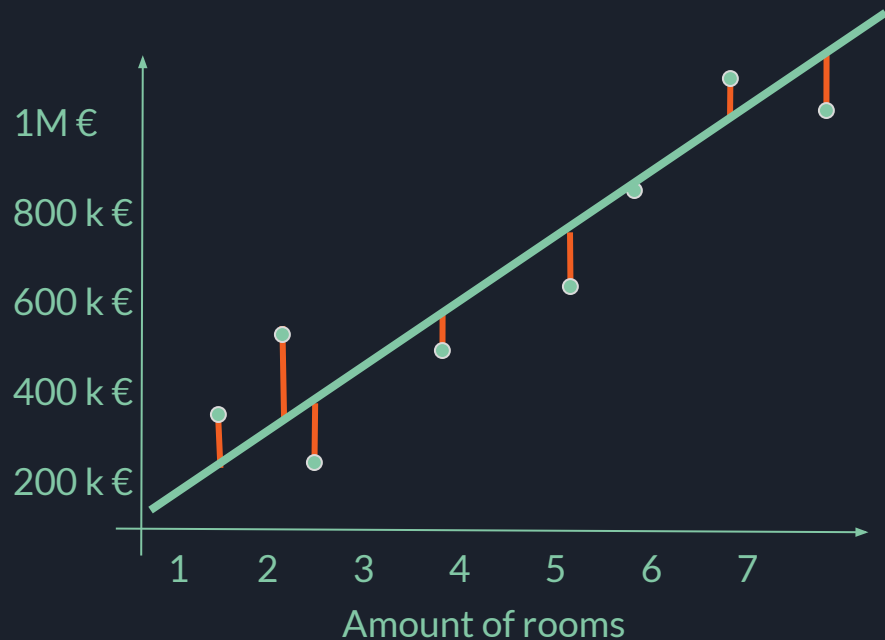


# Linear Regression



$$price = a \cdot rooms + b$$

# Linear Regression



$$price = a \cdot rooms + b$$

$$RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^T (y_{i,predicted} - y_{i,true})^2}$$



# Linear Regression

House	Bedrooms	Rooms	Price
A	1	6	500k €
B	3	8	600k €
C	2	7	300k €

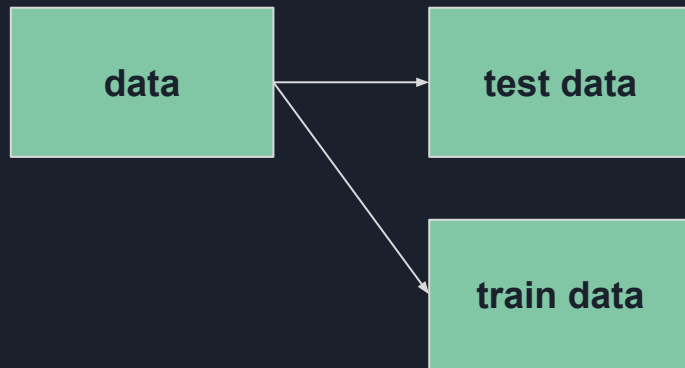
$$price = a_1 \cdot rooms + a_2 \cdot size + a_3 \cdot bedrooms + b$$



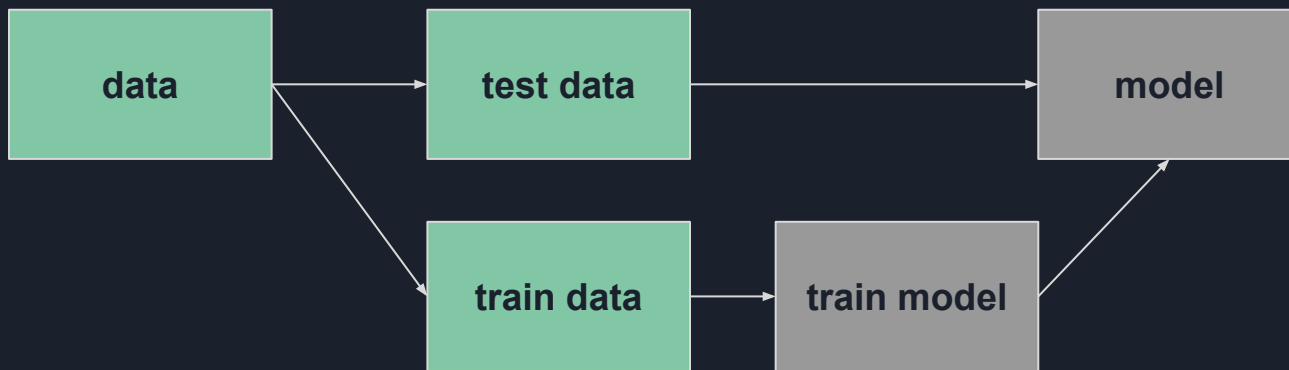
# How good is your model?

How does the model perform on unseen data?

Split data in a train (70%) and test (30%) set to evaluate your model

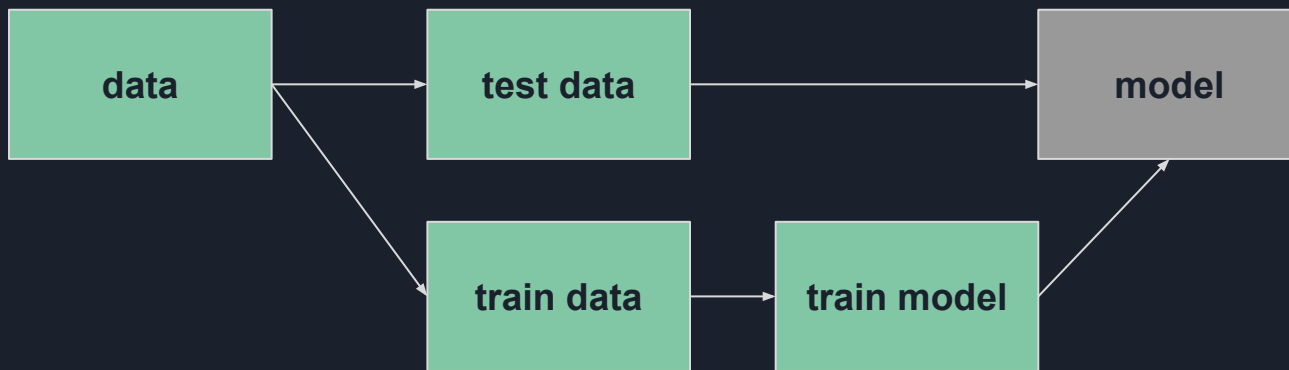


# Typical workflow



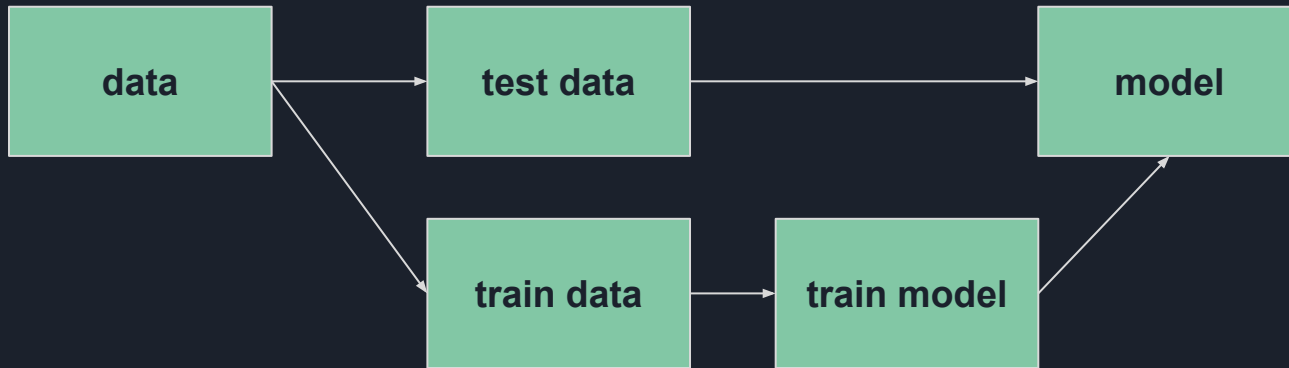


# Typical workflow





# Typical workflow



# Linear Regression with Python



```
import pandas as pd # data
from sklearn.model_selection import train_test_split # split data
from sklearn.linear_model import LinearRegression # create model
from sklearn.metrics import mean_squared_error # test model
```

# Linear Regression with Python



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('/content/housing.csv')
X = df[['total_rooms', 'households']]
y_target = df['median_house_value']
```

# Linear Regression with Python



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('/content/housing.csv')
X = df[['total_rooms', 'households']]
y_target = df['median_house_value']

X_train, X_test, y_train, y_test = train_test_split(X, y_target, test_size=0.3)
model = LinearRegression()
model.fit(X_train, y_train)
```

# Linear Regression with Python



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('/content/housing.csv')
X = df[['total_rooms', 'households']]
y_target = df['median_house_value']

X_train, X_test, y_train, y_test = train_test_split(X, y_target, test_size=0.3)
model = LinearRegression()
model.fit(X_train, y_train)

y_predict = model.predict(X_test)
rmse = mean_squared_error(y_test, y_predict, squared=False)
```

# Exercise 3



# Linear Regression with Python



```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import median_absolute_error

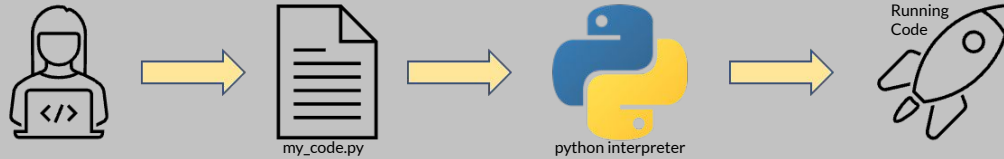
rmse = mean_squared_error(y_test, y_predict, squared=False)
r2 = r2_score(y_test, y_predict)
mae = median_absolute_error(y_test, y_predict)
```



# Install and Run Python locally

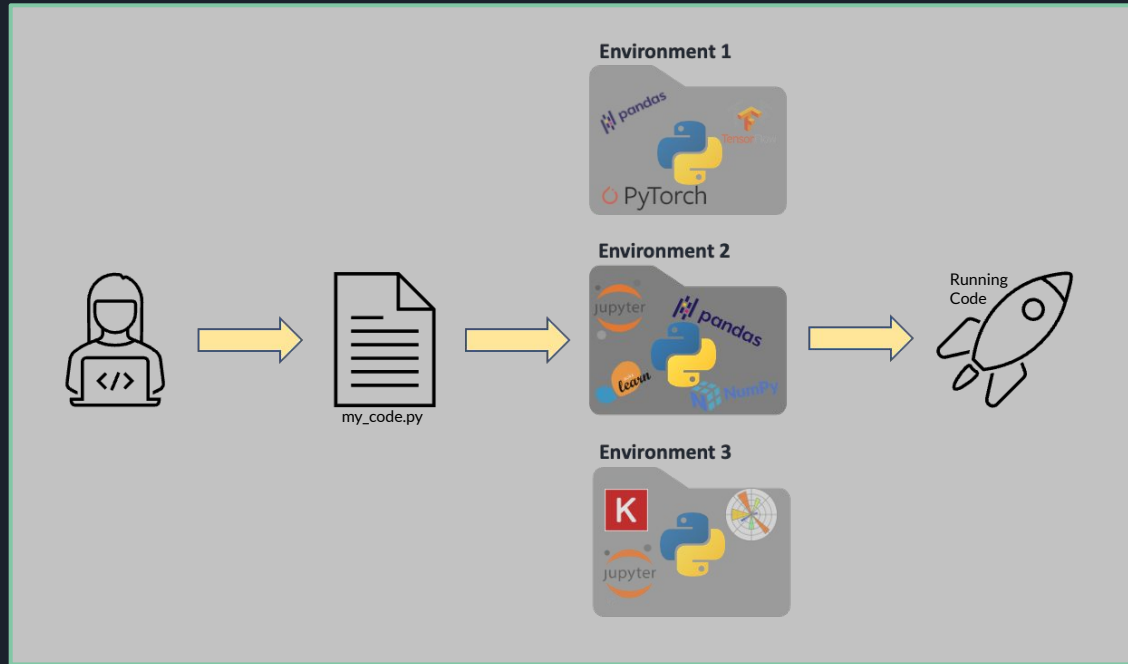


# Code Execution



- **Download Python:**  
<https://www.python.org/>

# Environments



- To install packages for a particular project
- Dependencies are installed in different directories



# Anaconda distribution



ANACONDA®

- Python interpreter
- Jupyter Notebooks (similar to Colab, but local)
- Manage environments and packages
- Runs on Windows, Mac and Linux

Installation:

<https://docs.anaconda.com/anaconda/install/>

Getting started:

<https://docs.anaconda.com/anaconda/user-guide/getting-started/>



# Pycharm



- Python IDE
- Free Educational License for Pro Version
- Many Alternatives: VS Code, Atom, ...

Installation:

<https://www.jetbrains.com/community/education/#students>