pandas

Data Academy

Pandas

TECH TALENT ACADEMY

pandas

Session Content

What is Pandas?

Series/Dataframes

Importing Data

Data Selection & Manipulation

TTA

pandas

What is Pandas?

Software library for use with Python

Ideal for working with datasets

Library facilitates data manipulation, visualisation and analysis

Created by software developer Wes McKinney in 2008

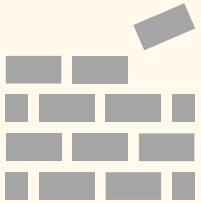
TTA

Short video about the author: https://www.youtube.com/watch?v=pqMu18dmEdM

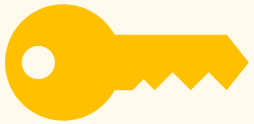
pandas

Why use Pandas?

You can import, analyse and visualise data easier



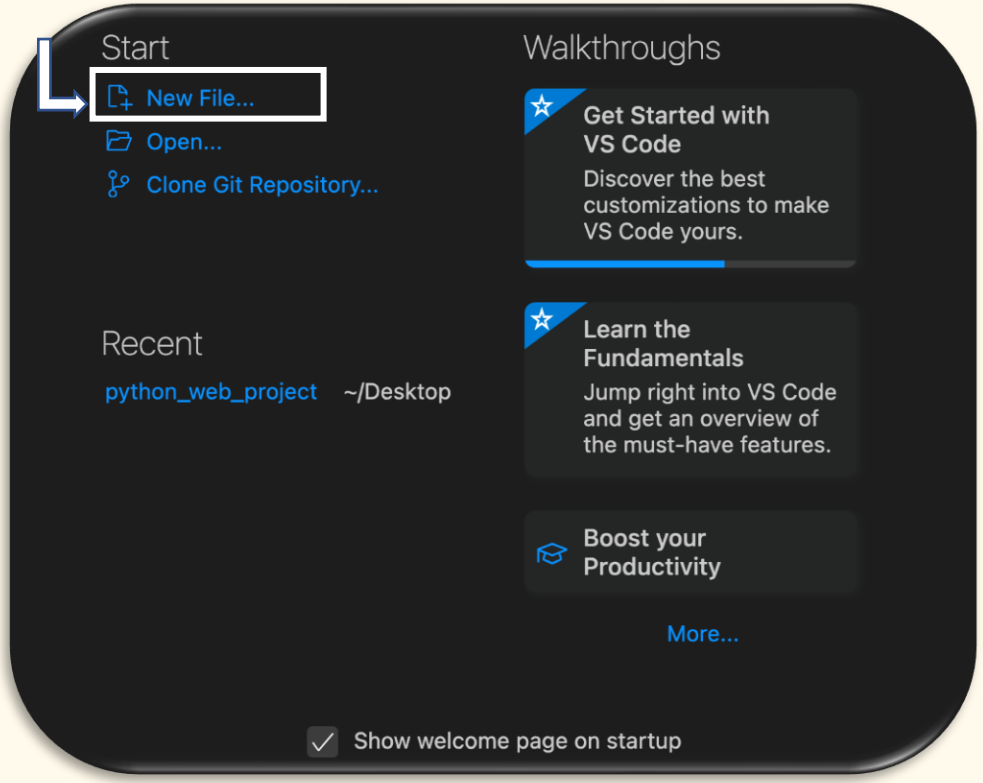
Builds on packages such as NumPy

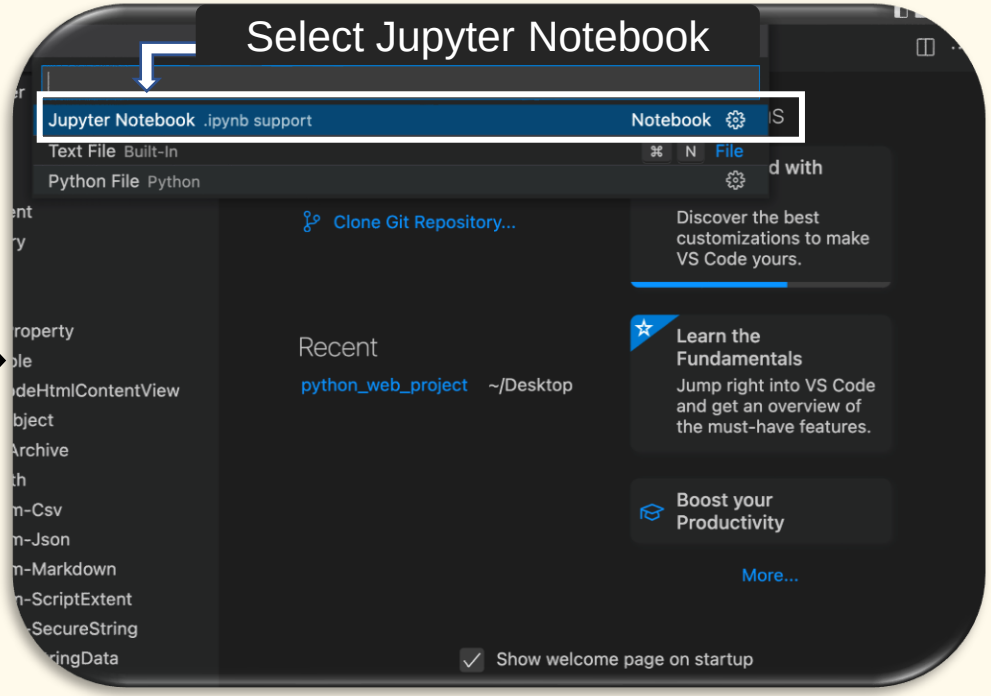


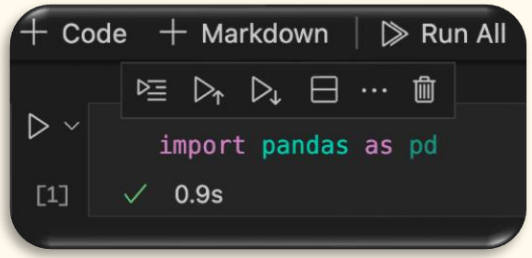
Key concepts of Pandas are indexing and dataframes

TTA

Jupyter Notebook setup on VS Code







TTA

If you get an error message that the pandas library is not installed, run the following command from the PowerShell:
pip install pandas

VS Code documentation: https://code.visualstudio.com/docs/datascience/jupyter-notebooks

pandas

Pandas data structure

Two types of pandas data structure:

Series

(1D like array)

rows index

	Country
0	United Kingdom
1	France
2	Mexico
3	Canada

Command to create a Serie: **pd.Series( )**

Dataframe

(2D like array or more)

columns index

rows index

	0	1
	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154

Command to create a dataframe: **pd.DataFrame( )**

TTA

pandas

Creating a pandas Series

From a list [ ]:

1-Create a list:

list_countries= ['United Kingdom', 'France', 'Mexico', 'Canada']

2-Pass the list in the pandas series function:

pd.Series (data=list_countries)

output

0	United Kingdom
1	France
2	Mexico
3	Canada

From a dictionary { }:

1-Create a dictionary:

dic_pop= {'United Kingdom':68521968,
'France':65273511,
'Mexico':128932753,
'Canada':37742154}

2-Pass the dictionary in the pandas series function:

pd.Series (dic_pop)

output

United Kingdom	68521968
France	65273511
Mexico	128932753
Canada	37742154

TTA

pandas

Creating a pandas dataframe

Import the numpy and pandas libraries:
import numpy as np
import pandas as pd

From a numpy array:

1-Create a numpy array:

numpy_array= np.array ([[5, 26, 3, 14], [31, 68, 53, 6], [2, 56, 8, 12]])

2-Create an index for the rows using a list (optional):

row_index=['A', 'B', 'C']

3-Create an index for the columns using a list (optional):

column_index= ['C0', 'C1','C2','C3']

4-Pass the different variables in the pandas dataframe function:

pd.DataFrame (**data**=numpy_array, **index**=row_index, **columns**=column_index)

Parameters used and others can be found in the pandas documentation:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

See code on next slide

TTA

pandas

Creating a pandas dataframe

In VS Code:

```python
#Library importation:
import pandas as pd
import numpy as np
```

✓ 0.2s Python

```python
#Creating a pandas dataframe from a numpy array:
numpy_array= np.array ([ [5, 26, 3, 14], [31, 68, 53, 6], [2, 56, 8, 12] ])
row_index=["A", "B", "C"]
column_index= ['C0', 'C1','C2','C3']
pd.DataFrame (data=numpy_array, index=row_index, columns=column_index)
```

✓ 0.3s Python

output

	C0	C1	C2	C3
A	5	26	3	14
B	31	68	53	6
C	2	56	8	12

Parameters: used in functions/methods (data, index, columns etc.)

Attributes: gives information about the data

Methods: functions that transform the data

TTA

pandas

Data importation

1-Get the file path of your datafile (census.csv) and store it in a variable you name:
path_datafile= 'path of the file/census.csv'

2- Create a new variable to store your dataframe and pass the variable that contains your file path in the following command:
dataframe= pd.read_csv (path_datafile)

3-Call the variable name to display the dataset
dataframe

Commands to read a CSV or Excel file:
pd.read_csv()
pd.read_excel()

TTA

pandas

Data importation (cont.)

In VS Code:

```python
path_datafile='/Users/pc/Desktop/datacensus.csv'
dataframe=pd.read_csv(path_datafile)
dataframe
```

✓ 0.4s Python

output

	Country	Population
0	United Kingdom	68521968.0
1	France	65273511.0
2	Mexico	128932753.0
3	Canada	37742154.0
4	Peru	NaN

Missing value:
Not A Number

TTA

pandas

Data exploration

.head() method

Explore the 5 first rows:
dataframe.head()

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

.tail() method

Explore the 5 last rows:
dataframe.tail()

23	Nigeria	206139589
24	Japan	126476461
25	Poland	37846611
26	Malaysia	32365999
27	Laos	7275560

The attribute **shape** give the total numbers of rows and columns:
dataframe.shape

output

(27, 2)

27 rows and 2 columns

Try:
dataframe.head (10)
dataframe.tail (22)

TTA

pandas

Data exploration cont.

Get a quick summary of the dataframe with the **.info() method** (i.e. # of columns and rows, data type, missing values #): **dataframe.info()**

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	NaN

output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Country     5 non-null      object
 1   Population  4 non-null      float64
dtypes: float64(1), object(1)
memory usage: 208.0+ bytes
```

missing value

TTA

pandas

Data selection (columns)

Select one column by column name using double backets [[ ]]: **dataframe[['Country']]**

The new column can be stored in a new variable: **countries=dataframe[['Country']]**

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

Select more than one column: **dataframe[['Country', 'Population']]**

It is possible to select data with one pair of [ ], but python will return a Series object not a dataframe: try **dataframe['Country']**

```
0    United Kingdom
1            France
2            Mexico
3            Canada
4              Peru
Name: Country, dtype: object
```

The method **type()** gives the data type

```
type(dataframe['Country'])
✓ 0.2s
pandas.core.series.Series
```

TTA

pandas

Data selection (rows)

.loc and .iloc commands

dataframe.loc[0, 'Country']

dataframe.iloc[2,1]

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

.loc
(primarily label based)
loc[row label, column label]

.iloc
(integer based)
iloc[row position, column position]

TTA

pandas

Data selection (rows cont.)

.loc and .iloc commands

dataframe.loc[0]
OR
dataframe.iloc[0]

dataframe.loc[2:4]

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

dataframe.iloc[2:4]
It selects the first row index to n-1

It is possible to **select/slice** a part of the dataframe using a colon or/and a comma

Try:
dataframe.iloc[0:3]
dataframe.iloc[0:3, 0:1]

TTA

pandas

Data selection (pattern)

.str.contains() method is used to search for a particular string pattern:
dataframe['Country'].str.contains('France')

1-The function evaluate each rows on the Country column for the presence of the string **'France'**. If there is no match, it returns **False**, if there is a match it returns **True**.
2-The function returns a pandas series object of Boolean values.
3-Selecting the previous command with **dataframe[]** will return all data related to the string 'France'.

1

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

2

Country
False
True
False
False
False

3

	Countries	Population
0	France	65273511

VS Code:

```
dataframe[dataframe['Country'].str.contains('France')]
```

TTA

pandas

Data manipulation

Rename a column name:

.rename(columns={'previous name': 'new name’})

dataframe.rename(columns={'Country': 'COUNTRY'})

	COUNTRY	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

Reassign a new row index:

new_index=['UK','FR','MX','CA','PR']

new_dataframe= dataframe.copy()

new_dataframe.index=new_index

	Country	Population
UK	United Kingdom	68521968
FR	France	65273511
MX	Mexico	128932753
CD	Canada	37742154
PR	Peru	32971854

To not erase the initial dataframe, we can create a copy so the change of index will affect only the copy not the original dataframe

TTA

pandas

Data manipulation

Drop one column:
dataframe.drop('Population', axis=1)

Drop one row:
dataframe.drop (4, axis=0)

default axis

	COUNTRY	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

axis=1

axis=0

	COUNTRY
0	United Kingdom
1	France
2	Mexico
3	Canada

axis=0 corresponds to the rows
axis=1 corresponds to the columns

TTA

pandas

Data manipulation

.str.replace('old value', 'new value')

dataframe['Country'].str.replace('United Kingdom', 'UK')

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

	Country	Population
0	UK	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

TTA

pandas

Handling missing data

Depending on the context of your data, you might want to replace missing values by “zero” or leave them.

.fillna() function is use to replace missing values

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	NaN

NaN: Not A Number

```
dataframe['Population']=dataframe['Population'].fillna(0)
```

can be any numerical value

TTA

pandas

Data cleaning

Checks: Remove missing values (NaN) from the dataset

Value replacement: perform the average of other values

Check info(): it will give you the count of non null values

Data uniformity: change type of data/rename variables names

Detect missing values: isnull() function

Drop columns with drop() function

Transform your numbers: absolute number

Remove outliers (can be seen when plotting the data)

TTA

lead to make proper analysis

Session Content

Using Booleans

Calculations

Combining data

JSON data and API

TTA

pandas

Data manipulation

Operations

→ Import the file datacensus2.csv

.sum() method

```
dataframe['Unemployed Persons'].sum()
✓ 0.2s
8432582
```

.mean() method

```
dataframe['Unemployed Persons'].mean()
✓ 0.3s
1686516.4
```

	Country	Population	Unemployed Persons*
0	United Kingdom	68521968	1704000
1	France	65273511	2967000
2	Mexico	128932753	2150582
3	Canada	37742154	1100000
4	Peru	32971854	511000

TTA

*Number of people registered as unemployed on Feb 2022 (might not reflect true number)

pandas

Data manipulation

Operations cont.

Create a new column and calculate the % of unemployed person:

-Code in multiple steps

1-Create a variable that contains the ratio of Unemployed persons÷Population
2-The **method round()** is use to round down the number of decimal, here to 2
3-Create a new column name that hold the new calculated ratio
4-Call the dataframe

VS Code:

```
ratio= dataframe['Unemployed Persons']/dataframe['Population']
ratio_percentage=ratio*100
ratio_percentage= round(ratio_percentage, 2)
dataframe['Unemployed person %']=ratio_percentage
dataframe
```

✓ 0.6s

	Countries	Population	Unemployed persons*	Unemployed person %
0	United Kingdom	68521968	1704000	2.49
1	France	65273511	2967000	4.55
2	Mexico	128932753	2150582	1.67
3	Canada	37742154	1100000	2.91
4	Peru	32971854	511000	1.55

Save your new dataframe on your desktop: **dataframe.to_csv('path /filename.csv')**

TTA

pandas

Data manipulation

Data filtering using boolean indexing

Use the following operators to test conditions:
<, >, ==, >=, <= , & (and), | (or), ~ (not)

Let's filter the data and keep countries with population size > 100 million: **dataframe['Population']>100000000**

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

Pandas Series output

Population
False
False
True
False
False

Encompassing the previous command with **dataframe[ ]** will return the true condition in a pandas dataframe:
dataframe[dataframe['Population']>100000000]

	Country	Population
2	Mexico	128932753

TTA

pandas

Data manipulation

Data filtering using boolean indexing cont.

Let's filter the data based on two conditions:
(dataframe['Country']=="Mexico") & (dataframe['Population']==128932753)

	Country	Population
0	United Kingdom	68521968
1	France	65273511
2	Mexico	128932753
3	Canada	37742154
4	Peru	32971854

Population
False
False
True
False
False

Encompassing the previous command with **dataframe[ ]** will return the true condition in a pandas dataframe:
dataframe[(dataframe['Country']=="Mexico") & (dataframe['Population']==128932753)]

	Country	Population
2	Mexico	128932753

TTA

pandas

Data manipulation

Grouping data with groupby()

- Import the videogames.csv file with the pandas library

dataframe.groupby("platform")["score"].mean()

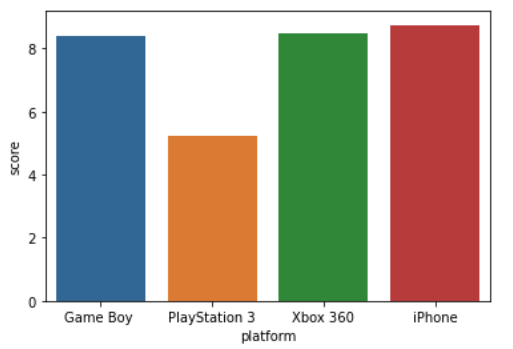
score_phrase	game_title	platform	genre	date_released	score
Great	Critter Crunch	iPhone	Puzzle	2008	8.5
Great	NHL 13	Xbox 360	Sports	2012	8.5
Amazing	Mario Tennis Power Tour	Game Boy	Sports	2005	9.0
Awful	Double Dragon: Neon	PlayStation 3	Fighting	2012	3.0
Good	Dr. Mario & Puzzle League	Game Boy	Puzzle	2005	7.8
Good	Tekken Tag Tournament 2	PlayStation 3	Fighting	2012	7.5
Amazing	Puzzle Craft	iPhone	Puzzle	2012	9.0

return a pandas series

```
platform
Game Boy        8.40
PlayStation 3   5.25
Xbox 360        8.50
iPhone          8.75
Name: score, dtype: float64
```

Use the parameter **as_index=False** in the groupby() function to return a pandas dataframe:
dataframe.groupby("platform",as_index=False)["score"].mean()

	platform	score
0	Game Boy	8.40
1	PlayStation 3	5.25
2	Xbox 360	8.50
3	iPhone	8.75



seaborn

Import visualisation library seaborn:
import seaborn as sns
sns.barplot(data=dataframe, x='platform', y='score')

TTA

pandas

Data exploration

How to Count unique values:

dataframe["genre"].value_counts()

score_phrase	game_title	platform	genre	date_released	score
Great	Critter Crunch	iPhone	Puzzle	2008	8.5
Great	NHL 13	Xbox 360	Sports	2012	8.5
Amazing	Mario Tennis Power Tour	Game Boy	Sports	2005	9.0
Awful	Double Dragon: Neon	PlayStation 3	Fighting	2012	3.0
Good	Dr. Mario & Puzzle League	Game Boy	Puzzle	2005	7.8
Good	Tekken Tag Tournament 2	PlayStation 3	Fighting	2012	7.5
Amazing	Puzzle Craft	iPhone	Puzzle	2012	9.0

```
Puzzle      3
Sports      2
Fighting    2
Name: genre, dtype: int64
```

TTA

pandas

Working with JSON data

JSON data: Text format derived from JavaScript (JavaScript Object Notation).
JSON syntax is similar to python dictionaries with {key : value} pairs, and python arrays []

Example of JSON object

Examples of JSON string

Example of JSON array

```
1  {
2  "school": "TechTalent Academy",
3  "address": "55 Colmore Row,
   Birmingham",
4  "city": "Birmingham",
5  "founder":"Janice Rae",
6  "year": "2019"
7  ,
8  "weblinks":[
9  {
10 "link": "website",
11 "mainpage":
   "https://techtalent.academy"
12 },
13 {
14 "link": "email",
15 "email":"hello@techtalent.academy"
16 }
17 ]
18 }
```

TTA

pandas

Working with JSON data

Import JSON data from an url:

```python
url="http://api.open-notify.org/astros.json"
dataframe = pd.read_json(url)
dataframe
```

✓ 0.4s Python

output

nested JSON data needs to be flatten

	number	message	people
0	7	success	{'craft': 'ISS', 'name': 'Raja Chari'}
1	7	success	{'craft': 'ISS', 'name': 'Tom Marshburn'}
2	7	success	{'craft': 'ISS', 'name': 'Kayla Barron'}
3	7	success	{'craft': 'ISS', 'name': 'Matthias Maurer'}
4	7	success	{'craft': 'ISS', 'name': 'Oleg Artemyev'}
5	7	success	{'craft': 'ISS', 'name': 'Denis Matveev'}
6	7	success	{'craft': 'ISS', 'name': 'Sergey Korsakov'}

Current astronauts in space JSON data:
http://api.open-notify.org/astros.json

TTA

pandas

Working with JSON data

How to work with nested JSON data:

pd.json_normalize data() function will flatten the nested JSON data from the columns 'people'

```python
normalised_data=pd.json_normalize(dataframe['people'])
normalised_data
```

✓ 0.1s Python

⬇

	craft	name
0	ISS	Raja Chari
1	ISS	Tom Marshburn
2	ISS	Kayla Barron
3	ISS	Matthias Maurer
4	ISS	Oleg Artemyev
5	ISS	Denis Matveev
6	ISS	Sergey Korsakov

It is common that JSON data will be more complex and have nested dictionaries that need to be flatten

TTA

pandas

Working with JSON data

How to work with nested JSON data (cont.):

The previous flatten JSON data needs to be added to the previous JSON data.
To combine dataframes: **pd.concat()** allows the concatenation of 2 or more dataframes, and the previous column 'people' is dropped with the function drop() as we saw in session 1

```
concat_data= pd.concat([dataframe, normalised_data], axis=1).drop('people', axis=1)
concat_data
```

✓ 0.1s Python

	number	message	craft	name
0	7	success	ISS	Raja Chari
1	7	success	ISS	Tom Marshburn
2	7	success	ISS	Kayla Barron
3	7	success	ISS	Matthias Maurer
4	7	success	ISS	Oleg Artemyev
5	7	success	ISS	Denis Matveev
6	7	success	ISS	Sergey Korsakov

TTA

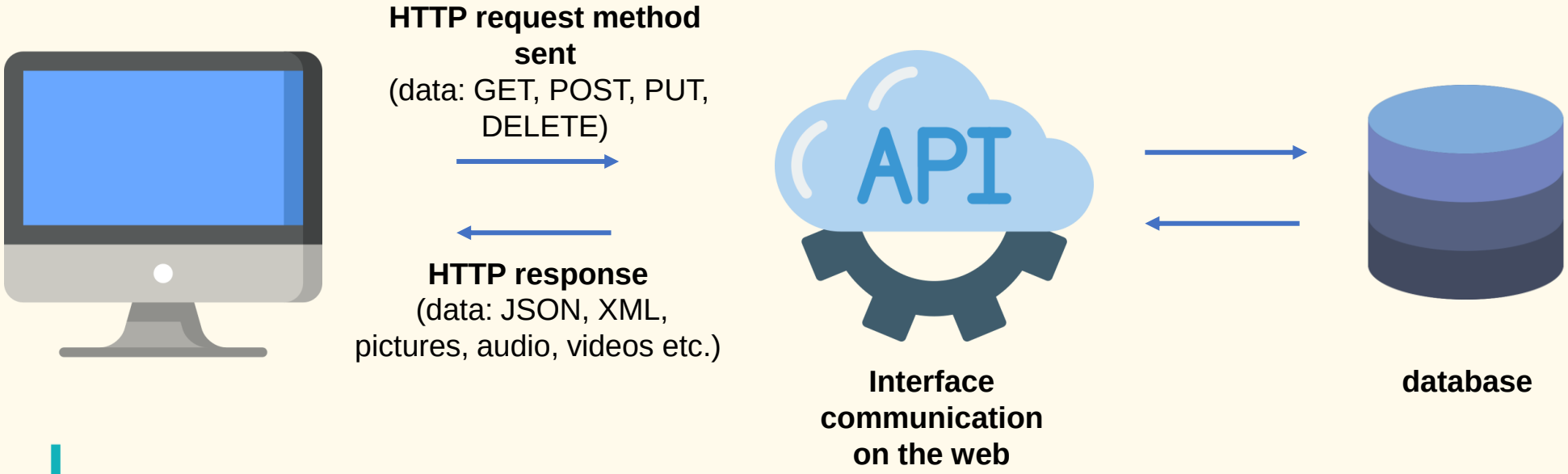
pandas

Data importation from APIs

Application Programming Interface (API):

APIs can be used to access/store/delete information/data

HTTP request methods are used to access or modify data from database via an API



TTA

pandas

Data importation from an API

First, we need to import the **request module** to make an HTTP request in VS Code:

```python
import requests
```

✓ 0.3s Python

The request method is used to send HTTP requests to get data from an url: **requests.get()**
(API calling from the following website: https://disease.sh/v3/covid-19/countries)

```python
url="https://disease.sh/v3/covid-19/countries"
API_requests= requests.get(url)
API_requests
```

✓ 0.5s Python

```
<Response [200]>
```

Output: HTTP response status.
200 means it's a successful response

TTA

Source API website: https://disease.sh/docs/

pandas

Data importation from an API (cont.)

The JSON library needs to be imported to work with JSON data.
The get request needs to be converted into text and then loaded into a JSON format:

```python
#Import json module:
import json
#Get the request into text and change it to JSON format:
data_API= API_requests.text
json_data=json.loads(data_API)
json_data
```

✓ 0.3s

output


```
[{'updated': 1651848352526,
  'country': 'Afghanistan',
  'countryInfo': {'_id': 4,
   'iso2': 'AF',
   'iso3': 'AFG',
   'lat': 33,
   'long': 65,
   'flag': 'https://disease.sh/assets/img/flags/af.png'},
  'cases': 178919,
   ...
```


We can transform the JSON data into a pandas dataframe:

```python
dataframe= pd.DataFrame(json_data)
dataframe.head()
```

✓ 0.9s

output

	updated	country	countryInfo	cases	todayCases	deaths	todayDeaths	recovered	todayRecovered	active	...
0	1651924565523	Afghanistan	{'_id': 4, 'iso2': 'AF', 'iso3': 'AFG', 'lat':...	178922	3	7684	0	161936	12	9302	...
1	1651924565506	Albania	{'_id': 8, 'iso2': 'AL', 'iso3': 'ALB', 'lat':...	275266	0	3496	0	271480	0	290	...
2	1651924565507	Algeria	{'_id': 12, 'iso2': 'DZ', 'iso3': 'DZA', 'lat'...	265791	0	6875	0	178361	0	80555	...
3	1651924565630	Andorra	{'_id': 20, 'iso2': 'AD', 'iso3': 'AND', 'lat'...	41717	0	153	0	41021	0	543	...
4	1651924565534	Angola	{'_id': 24, 'iso2': 'AO', 'iso3': 'AGO', 'lat'...	99194	0	1900	0	97149	0	145	...

5 rows × 23 columns

TTA

TECH TALENT
ACADEMY