Python meeting
01: Initiation

15th April 2026

The Python Meeting will start soon

COURSE PYTHON SCHEDULE

01: 20250415, 10am: Initiation – Case-study: P02_15: Above average prices

- Set-up: Python, Visual Studio Code
- Project creation: Folder, .py, Virtual Environment, CMD
- Libraries: Import, requirements file
- Naming conventions
- Basic objects: Dataframe (table), Series (column), Variable, Function, Group_by
- Basic functions: Import, Rename, Filter, Join, Create columns, Group, Aggregate, Export

02: 20250422, 9am: Melt, split, for – Case-studies: D01: Trial balance, P01_19: Supplier debtors, O01_19: Customer creditors

- Using melt, split
- For loop

03: 20250429, 9am: Web-download – Case-study: P01_10: Suppliers in OFAC

- Request for downloading sanctions
- IO for encoding
- Regex for comparison
- Split, explode for row generation
- Levenshtein distance scoring

04: 20250506, 9am: JSON, expressions, rolling Window – Case-study: P02_19: Split POs

- Import JSON, Expression object for adding labels,
- cum_sum() for rolling window

05: 20250513, 9AM: zip, dynamic fields – Case-study: P02_03: PO whilst blocked

- Zip to group lists together
- Using dynamic fields in a for loop

09:00 – 09:05

COURSE PYTHON SCHEDULE

06: 20250520, 10am: Cumulative sum, dates – Case-study: P02_18: PO slow rotation

- cum_sum for calculating future or past inventory movements

07: 20250527, 9am: Isolation forest for unusual transactions – Case-study: Unusual bank transfers

- Using isolation forest library to score for unusual transactions

08: 20250603: 9am: SpaCy – Case-study: P01_11/ O01_11: Suppliers/ customers that are people

- Using SpaCy NLP object to categorize names

09: 20250610: 9am: Contract review – Case-study: Scoring of contracts

- Using Gemini model to check for paragraphs of similar meaning

10: 20250617: 9am: Open AI API – Case-study: Generating audit reports

- Using API to OpenAI to generate text for audit reports

11: 20250624: 10am: Regex and Sklearn matching – Case-study: HR03_12/ H403_13: Unusual T&E

- Using Regex and Sklearn to check for unexpected travel and expenses

12: 20250701: 9am: OCR: Image recognition – Case-study: HR03_14: Expenses for others

- Using OCR python library to read travel and expense receipts

TODAY'S SCHEDULE

01

Ensure you are set-up with Python & VSC

→ Get into python in 5 minutes

02

Case-study

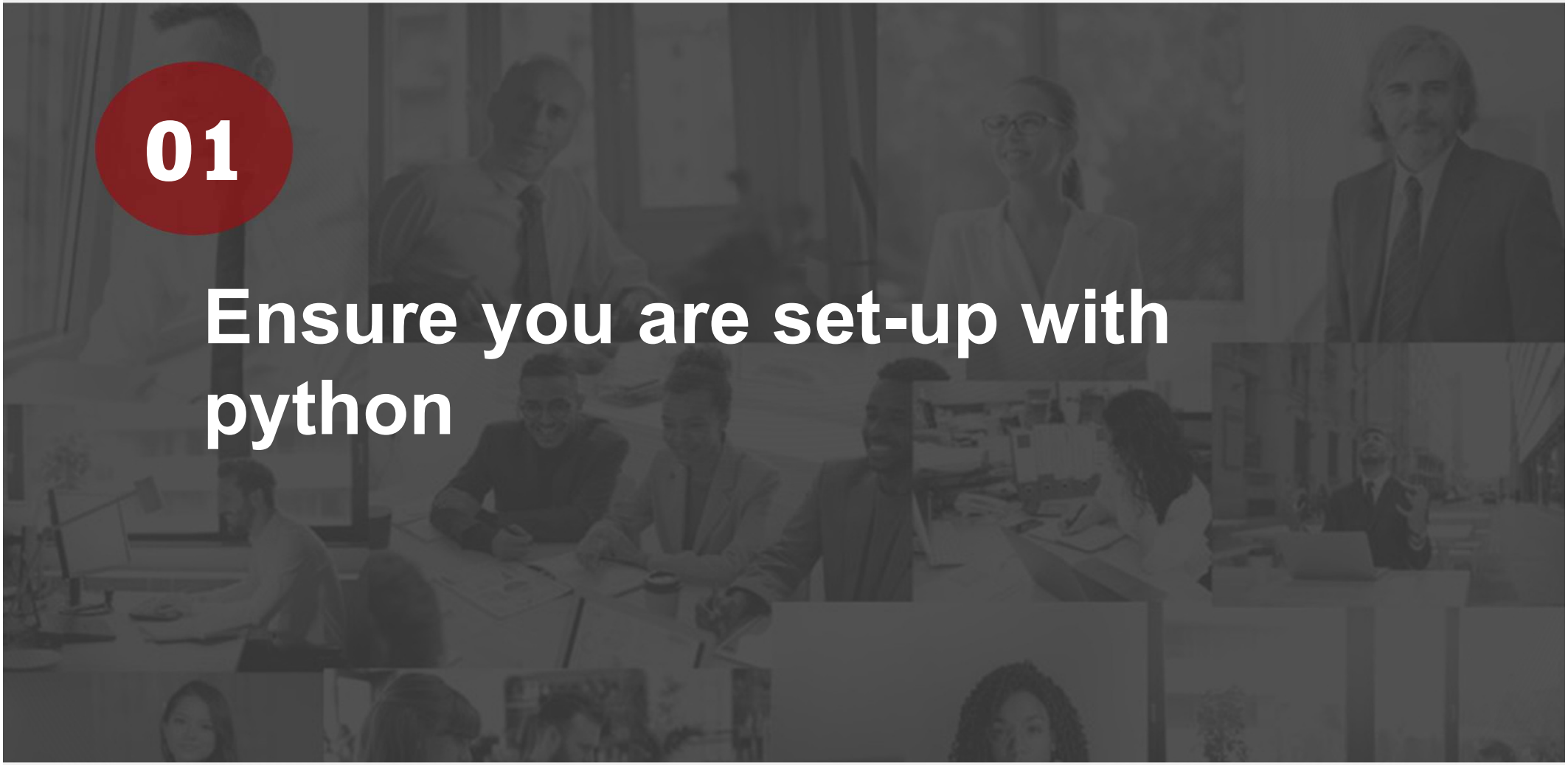
→ Purchase order price higher than average

03

Python structures

Visualizing python structures

09:05 – 09:10



09:05 – 09:10

Pre-requisites

1. You have python installed on your machine
2. You have Visual Studio Code installed on your machine

We will now follow these instructions that you received by e-mail

03_INSTALL_YOU
_PYTHON_ENVIRO

You might need to open PowerShell as administrator and do:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

01: Install python and VSC on your machine

02: Run a hello-world script

09:05 – 09:10

02

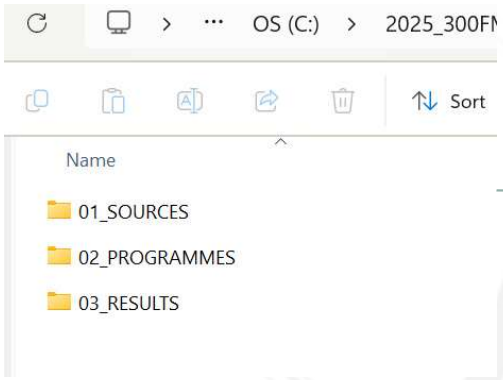
Basic functions
Case study: purchase orders with price > average price

Note: normally you would have to add currency conversion step

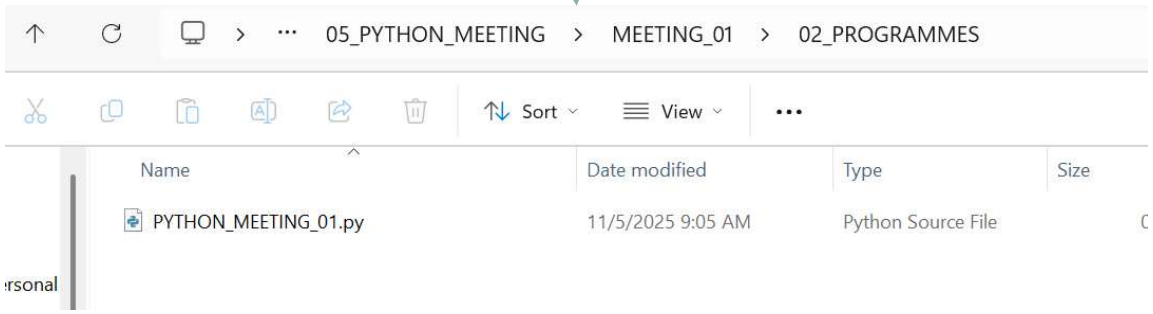
09:10 – 09:15

Basic functions – .venv and library import -> your own requirements.txt

Set-up your folders:

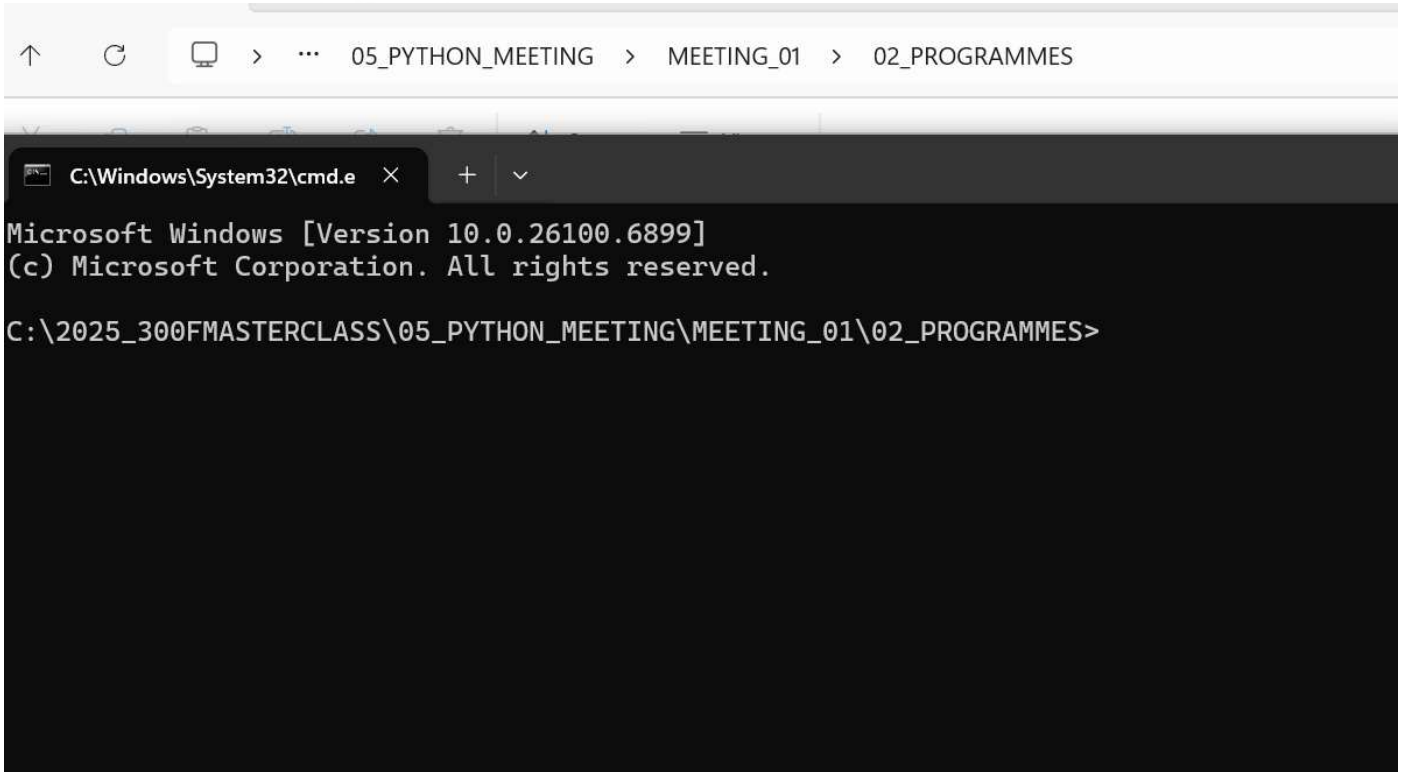


Put .py empty text file in Programmes folder



Basic functions – .venv and library import -> your own requirements.txt

Open visual studio code in the right place -> type CMD in Windows folder



Basic functions – .venv and library import -> your own requirements.txt

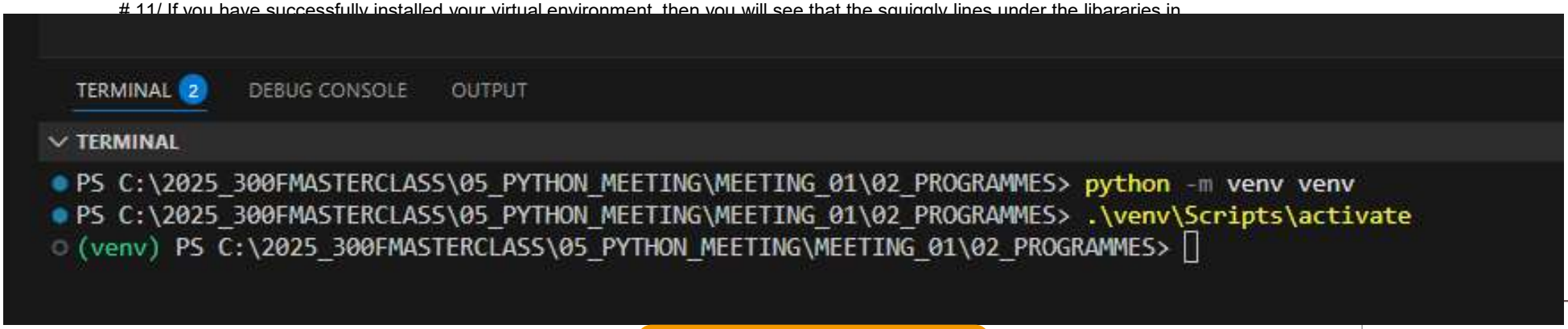
.venv environment

Copy this text at the top of the python script – as a reminder:

```
# Before running this script, create a virtual environemnt for your project, activate the virtual environment and then install the requirements to your virtual environment
# 1/ Go to the 02_PROGRAMMES folder and type CMD in the Windows search bar
# 2/ Then, in the black box that opens, type code . (code space dot)
# 3/ Visual studio code will then open in the correct location
# 3/ Open the terminal, in Visual Studio Code: (View-> Terminal or CTRL+SHIFT+`)
# 4/ In the terminal make sure that you are in the 02_PROGRAMMES folder (if you are not do cd .. to go back up a directory or cd <directoryName> to go to a directory)
# 5/ Then in the terminal type python -m venv venv
# 6/ Then select the new environment that you just made, as the interpreter, by right-clicking on environment in the bottom right hand corner of Visual Studio Code and entering the address to the folder where the python.exe is found on your machine
#    for example:
C:\2025_300FMASTERCLASS\02_PRESENTATION\01_LIVE_SESSION_01\PYTHON_EXAMPLE\02_PROGRAMMES\venv\Scripts
# 7/ With your new virtual environment selected as the interpreter, enter the following in the terminal: .\venv\Scripts\activate
# 8/ After you do that you should see that the folder address in the terminal then has (venv) in front of it in green.
# 9/ After that, you can activate the requirements file, and the lirbaries that are required for this python script will go inside
#    that virtual environment. Do import the requirements into your envrionment, you can run this in the terminal: pip install -r requirements.txt
# 10/ Then wait for about 5 minutes while it installs those librarires into the venv of this project.
# 11/ If you have successfully installed your virtual environment, then you will see that the squiggly lines under the libararies in
```

Tips: View-> word wrap

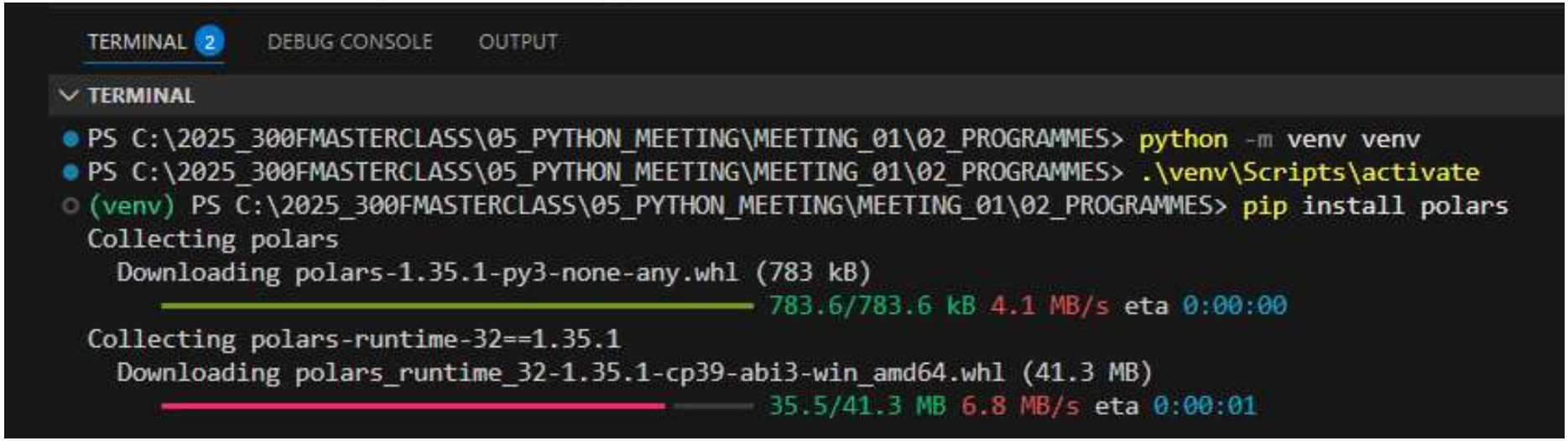
Tips: Answer Yes if it asks you to use that environment



Basic functions – install a library

pip install polars

Tip – if for some reason you don't have pip – type python -m ensurepip --upgrade



Basic functions – install a library

Get the version of polars that you just installed: pip show polars

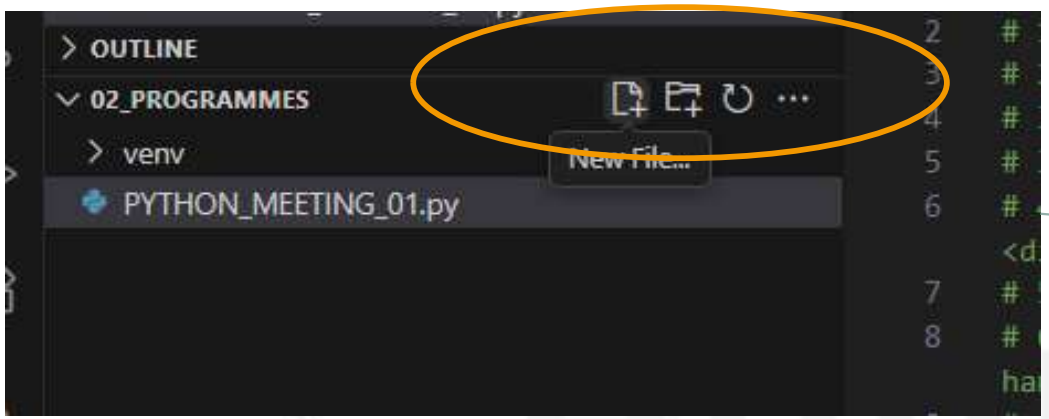
```
(venv) PS C:\2025_300FMASTERCLASS\05_PYTHON_MEETING\MEETING_01\02_PROGRAMMES> pip show polars
Name: polars
Version: 1.35.1
Summary: Blazingly fast DataFrame library
Home-page:
Author:
Author-email: Ritchie Vink <ritchie46@gmail.com>
License: Copyright (c) 2025 Ritchie Vink
Some portions Copyright (c) 2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

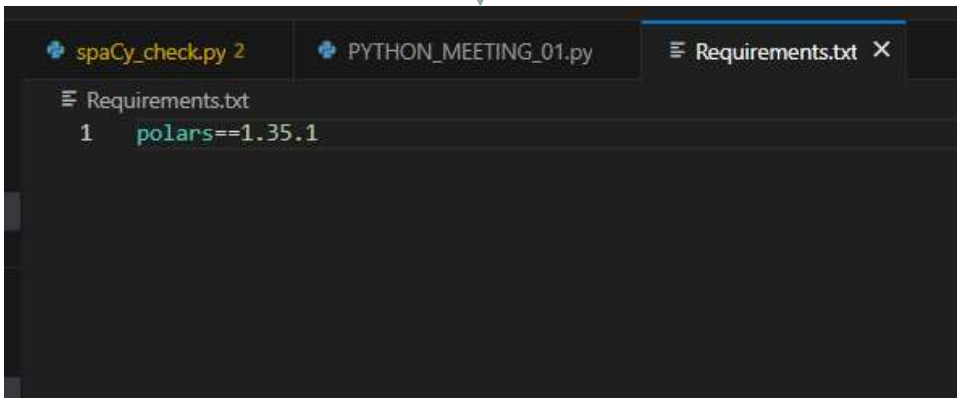
Tip – use this button to make terminal bigger and then you can reduce it again

Basic functions – install a library

Create requirements file

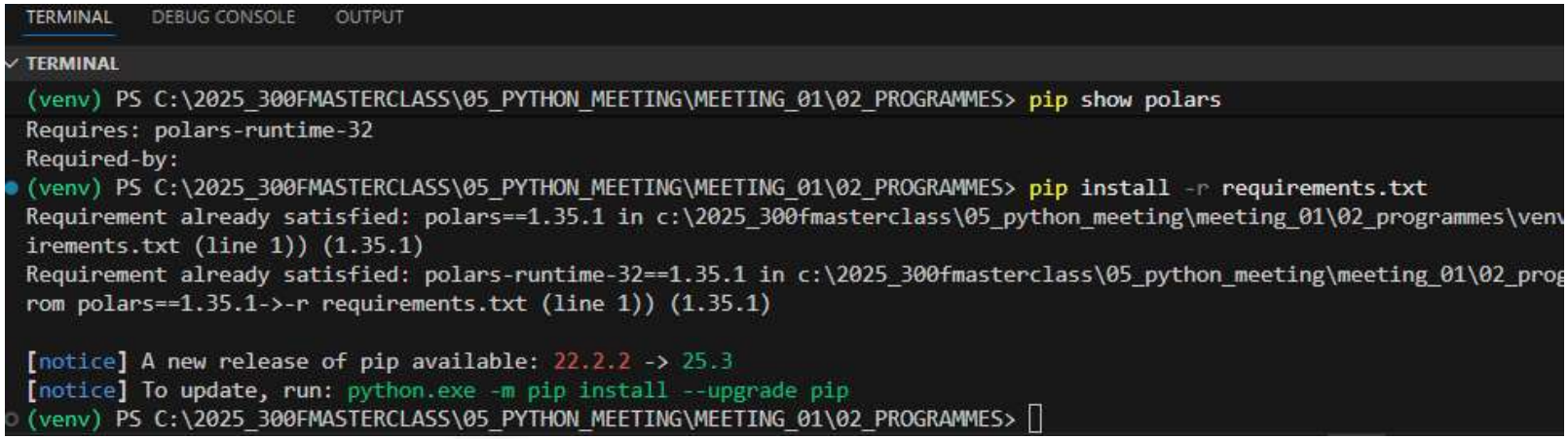


Tip – ensure autosave is on



Basic functions – install a library

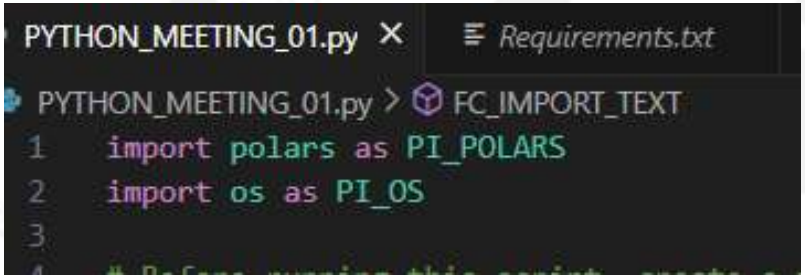
Test run requirements.txt pip install -r requirements.txt



Basic functions – import your library

Put at the top of your script:

import polars as PI_POLARS
import os as PI_OS



Basic functions – create a variable

Create variables

```
ZV_NU_THRESHOLD = 10000
ZV_ST_SOURCES_FOLDER ='C:/2025_300FMASTERCLASS/05_PYTHON_MEETING/MEETING_01/01_SOURCES/'
ZV_ST_SOURCE_FILE_EKKO='20240510_152615_1_EKKO.csv'
ZV_ST_SOURCE_FILE_EKPO='20251025_180621_1_EKPO.csv'
ZV_ST_SOURCE_FILE_APPROVLIM='A_APPROVAL_LIMITS.json'
ZV_ST_RESULTS_FOLDER='C:/2025_300FMASTERCLASS/05_PYTHON_MEETING/MEETING_01/03_RESULTS/'
```

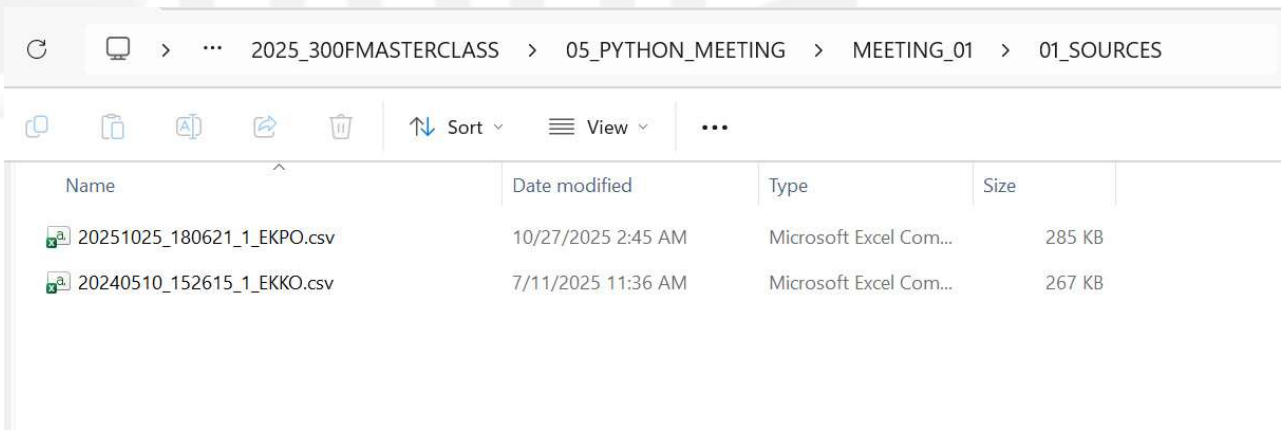
Tip – make sure no spaces in folder names and change \ to / and put / at the end

Basic functions – import and rename

Put these two files in your sources folder:

A_EKPO

A_EKKO



Basic functions – import and rename

Import function – copy to your python program

Note: in python – functions must go above where they are used – so usually we put them at the top or in a separate script

```
# 2.1/ Import
def FC_IMPORT_TEXT(
        ZVFCI_ST_SOURCE_FILE,
        ZVFCI_ST_FOLDER,
        ZVFCI_ST_DELIMITER
    ):
    ZV_ST_FILE_PATH = PI_OS.path.join(ZVFCI_ST_FOLDER, ZVFCI_ST_SOURCE_FILE)


    ZV_DF_HEADER = PI_POLARS.read_csv(
        ZV_ST_FILE_PATH,
        separator=ZVFCI_ST_DELIMITER,
        encoding='utf8',
        has_header=True,
        n_rows=1,
        infer_schema_length=0,
        quote_char=None
    )
    ZV_LI_COLS = ZV_DF_HEADER.columns


    ZV_DI_SCHEMA_OVERRIDES = {ZV_COL: PI_POLARS.Utf8 for ZV_COL in ZV_LI_COLS}


    ZV_DF = PI_POLARS.read_csv(
        ZV_ST_FILE_PATH,
        separator=ZVFCI_ST_DELIMITER,
        encoding='utf8',
        ignore_errors=True,
        has_header=True,
        schema_overrides=ZV_DI_SCHEMA_OVERRIDES,
        infer_schema_length=0,
        null_values=[""],
        quote_char=None


    ).fill_null('')


    return ZV_DF
```

Basic functions – import and rename

Call the import function

```
# Step 3/ Import the files
ZV_DF_EKKO = FC_IMPORT_TEXT(ZV_ST_SOURCE_FILE_EKKO,ZV_ST_SOURCES_FOLDER,'\t')
ZV_DF_EKPO = FC_IMPORT_TEXT(ZV_ST_SOURCE_FILE_EKPO,ZV_ST_SOURCES_FOLDER,'|')


print(ZV_DF_EKKO.head())
print(ZV_DF_EKPO.head())
```

Tip – ensure that things are working as expected use print(<DF_NAME>.head())

Basic functions – import and rename

Rename fields – copy to your python program

```
# Step 4/ EKKO
# 4.1/ Select / rename
ZV_DF_EKKO = (
    ZV_DF_EKKO
    .select(
        [
            'EBELN',
            'BEDAT',
            'ERNAM',
            'LIFNR'
        ]
    )
    .rename(
        {
            'EBELN':'EKKO_EBELN',
            'BEDAT':'EKKO_BEDAT',
            'ERNAM':'EKKO_ERNAM',
            'LIFNR':'EKKO_LIFNR',

        }
    )
)
```

```
# Step 5/ EKPO
# 5.1/ Select / rename
ZV_DF_EKPO= (
    ZV_DF_EKPO
    .select(
        [
            'EBELN',
            'EBELP',
            'NETPR',
            'NETWR',
            'MATNR',
            'BPUMZ',
            'BPUMN'
        ]
    )
    .with_columns(
        [
            PI_POLARS.col('NETPR')
            .cast(PI_POLARS.Float64)
            .alias('NETPR'),


            PI_POLARS.col('NETWR')
            .cast(PI_POLARS.Float64)
            .alias('NETWR'),


            PI_POLARS.col('BPUMZ')
            .cast(PI_POLARS.Float64)
            .alias('BPUMZ'),


            PI_POLARS.col('BPUMN')
            .cast(PI_POLARS.Float64)
            .alias('BPUMN')
        ]
    )
    .rename(
        {
            'EBELN':'EKPO_EBELN',
            'EBELP':'EKPO_EBELP',
            'NETPR':'EKPO_NETPR',
            'NETWR':'EKPO_NETWR',
            'MATNR':'EKPO_MATNR',
            'BPUMZ':'EKPO_BPUMZ',
            'BPUMN':'EKPO_BPUMN',
```

```
print(ZV_DF_EKKO.head())
print(ZV_DF_EKPO.head())
```

Basic functions – joining

Add PO header to PO detail:

```
# Step 6/ Join
ZV_DF_EKPO_EKKO = (
    ZV_DF_EKPO
    .join(
        ZV_DF_EKKO,
        left_on='EKPO_EBELN',
        right_on='EKKO_EBELN',
        how='inner',
        suffix='_2'
    )
)


print(ZV_DF_EKPO_EKKO.head())
```

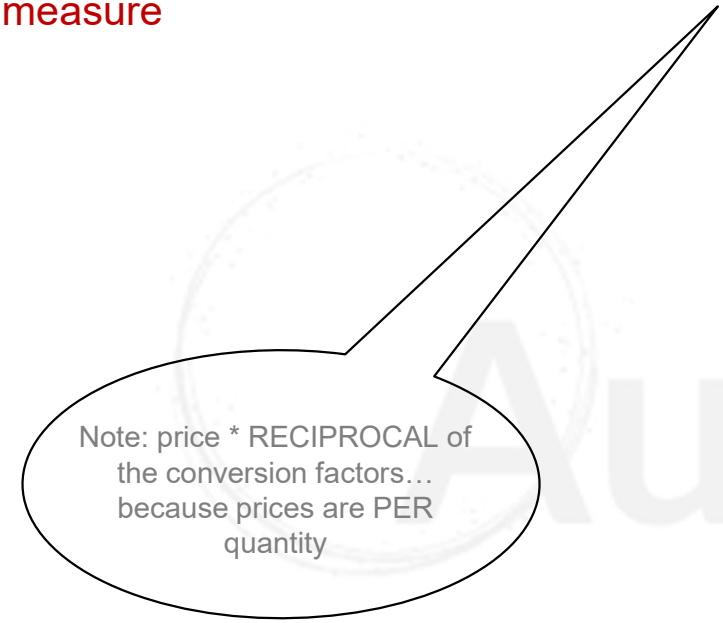
Inner join will eliminate any rows in EKPO that are not in EKKO

Tip: use ‘left’ when you don’t want to eliminate any rows

Tip: use ‘anti’ when you want those that don’t match

Basic functions – create columns

Compute the price into base units of measure




```
# Step 7/ Add columns
ZV_DF_EKPO_EKKO = (
    ZV_DF_EKPO_EKKO
    .with_columns(
        [
            PI_POLARS.when(
                PI_POLARS.col('EKPO_BPUMZ') == 0
            )
            .then(
                PI_POLARS.lit(1)
            )
            .otherwise(
                PI_POLARS.col('EKPO_BPUMZ')
            )
            .alias('ZF_EKPO_BPUMZ'),


            PI_POLARS.when(
                PI_POLARS.col('EKPO_BPUMN') == 0
            )
            .then(
                PI_POLARS.lit(1)
            )
            .otherwise(
                PI_POLARS.col('EKPO_BPUMN')
            )
            .alias('ZF_EKPO_BPUMN'),
        ]


    )
    .with_columns(
        (
            PI_POLARS.col('EKPO_NETPR') *
            (
                PI_POLARS.col('ZF_EKPO_BPUMN')/
                PI_POLARS.col('ZF_EKPO_BPUMZ')
            )
        )
        .alias('ZF_EKPO_NETPR_BPUMNBPUMZ')
    )
)
```

_EKPO_EKKO.head())

Basic functions – group

Group with aggregate – get the average price per material, filter

```
# Step 8/ Group_by
ZV_DF_TT_EKPO_EKKO_GROUP_BY = (
    ZV_DF_EKPO_EKKO
    .filter(
        (
            PI_POLARS.col('EKPO_MATNR') !=''
        ) &
        (
            PI_POLARS.col('ZF_EKPO_NETPR_BPUMNBPUMZ') !=0
        )
    )
    .group_by(
        'EKPO_MATNR'
    )
    .agg(
        [
            PI_POLARS.col('ZF_EKPO_NETPR_BPUMNBPUMZ')
            .mean()
            .alias('ZF_EKPO_NETPR_BPUMNBPUMZ_MEAN')
        ]
    )
)
```

Note: naming convention – keeps our code easy to read!

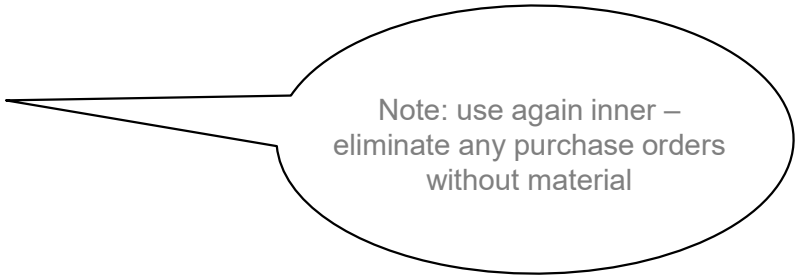
Tip: ignore blank materials and blank suppliers (usually stock transports) to reduce false positives/ noise

Basic functions – join

Add back the average

```
# Step 9/ Join
ZV_DF_EKPO_EKKO = (
    ZV_DF_EKPO_EKKO
    .join(
        ZV_DF_TT_EKPO_EKKO_GROUP_BY,
        on='EKPO_MATNR',
        how='inner',
        suffix='_2'
    )
)


print(ZV_DF_EKPO_EKKO.head())
```



Basic functions – add columns

Calculate the % difference, filter

```
# Step 10/ Add fields/ filter
ZV_DF_EKPO_EKKO_HIGHNETPR = (
    ZV_DF_EKPO_EKKO
    .with_columns(
        (
            (
                (
                    PI_POLARS.col('EKPO_NETPR') -
                    PI_POLARS.col('ZF_EKPO_NETPR_BPUMNBPUM
Z_MEAN')
                ) /
                PI_POLARS.col('ZF_EKPO_NETPR_BPUMNBPUMZ_
MEAN')
            ) * 100
        )
        .alias('ZF_EKPO_NETPR_DIFF_PER')
    )
    .filter(
        (
            PI_POLARS.col('ZF_EKPO_NETPR_DIFF_PER') > 50
        ) &
        (
            PI_POLARS.col('EKPO_NETWR')>ZV_NU_THRESHOLD
        )
    )
)


print(ZV_DF_EKPO_EKKO.head())
```

Tip: use a lot of brackets and put operators always on preceding line: readability and less bugs!

Basic functions – export the result

Export the result – first do:

pip install pandas
pip install openpyxl
pip install pyarrow

Don’t forget to add to your requirements

pip show ….

Then copy-paste version to your requirements.txt

Basic functions – export the result

```
# 11/ Export the result


ZV_ST_RESULTS_FILE =
'ZV_DF_EKPO_EKKO_HIGHNETPR.xlsx'
FC_EXPORT_EXCEL(ZV_DF_EKPO_EKKO_HIGHNETPR,
ZV_ST_RESULTS_FOLDER, ZV_ST_RESULTS_FILE)
```

```
# 2.2/ Export
def FC_EXPORT_EXCEL(ZVFCI_DF_INPUT, ZVFCI_ST_RESULTS_FOLDER,
ZVFCI_ST_RESULTS_FILE):


    PI_OS.makedirs(ZVFCI_ST_RESULTS_FOLDER, exist_ok=True)


    ZV_ST_FILE_PATH = PI_OS.path.join(ZVFCI_ST_RESULTS_FOLDER,
ZVFCI_ST_RESULTS_FILE)


    ZV_DF_PANDAS = ZVFCI_DF_INPUT.to_pandas()

    ZV_DF_PANDAS.to_excel(
        ZV_ST_FILE_PATH,
        index=False,
        engine='openpyxl'
    )


    print(f"✅ Excel file successfully exported to:\n{ZV_ST_FILE_PATH}")
```


03
Python structures

Python structures

A list:

[1, ‘2’,[‘a’,’b’,’c’], 1]

A list is a list of objects: can be different types

A set:

{1, ‘2’,[‘a’,’b’,’c’]}

A set is a list – but no duplicates allowed

An array:

[1, 2,3,4,5]

An array is like a list, but all objects of the same type

A tuple

(1, ‘2’,[‘a’,’b’,’c’], 1)

A tuple is like a list but it cannot be changed or indexed

A dictionary:

{
‘Suppliers’: (‘a’,’b’,’c’)
‘Number’: (1,2,3)
‘Value’:(100,150,300)
}

A dictionary has names that reference objects.

A dataframe:

A dataframe (~table) is like a dictionary, where the names are column names and the objects are arrays (have same type within them).

A series:

If we take a column out it is called a series

09:45 – 09:50

Python structures

(GBP, EUR)
(GBP, USD)
(GBP, ZAF)

Group keys

group_by object

TCURR_GDATU	TCURR_UKURS
20250101	0.61
...	...

TCURR_GDATU	TCURR_UKURS
20250101	0.93
...	...

TCURR_GDATU	TCURR_UKURS
20250101	20.01
20250102	20.03
20250103	20.02
...	...

Dataframes

Python structures

```
group_by([
        Company,
        Material,
        Month
])
```

3 Sub dataframes

Material	Month	Posting date	Input date	Input time	Document	Item	Quantity	Value
Bolts	Jan 2025	1 Jan 2025	1 Jan 2025	00:01:30	00001234	0001	10	200
Bolts	Jan 2025	2 Jan 2025	2 Jan 2025	00:05:20	00001235	0001	3	60
Bolts	Jan 2025	3 Jan 2025	3 Jan 2025	00:16:10	00001236	0001	2	400
Bolts	Jan 2025	25 Jan 2025	25 Jan 2025	00:01:30	00001237	0001	14	280
Bolts	Feb 2025	1 Feb 2025	1 Jan 2025	00:01:30	00001238	0001	5	1000
Bolts	Feb 2025	2 Feb 2025	1 Jan 2025	00:05:20	00001241	0001	6	1200
Bolts	Feb 2025	3 Feb 2025	1 Jan 2025	00:16:10	00001242	0001	17	3400
Bolts	Feb 2025	25 Feb 2025	1 Jan 2025	00:01:30	00001256	0001	2	400
Bolts	Mar 2025	1 Mar 2025	1 Mar 2025	00:01:30	00001281	0001	21	4200
Bolts	Mar 2025	28 Mar 2025	28 Mar 2025	00:05:20	00001290	0001	32	6400

09:50 – 09:55


Questions?