main analytics

Beyond Scripting

A Beginner’s Guide to Accelerating Automation with Executables

www.mainanalytics.de

main analytics

…on a calm Friday afternoon

main analytics

...on a calm Friday afternoon

Colleague

Michel

Are you still working?

main analytics

Reference[1]

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

… say no

...on a calm Friday afternoon

Colleague

Michel

Are you still working?

Yes, sure.
How can I help you?

main analytics

Reference[1]

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

Yes, sure.
How can I help you?

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

Yes, sure.
How can I help you?

… say no

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

… and add a ToC?

Yes, sure.
How can I help you?

… say no

main analytics

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

… and add a ToC?

Yes, sure.
How can I help you?

… say **no**

main analytics

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

… and add a ToC?

… and sequential page numbers?

Yes, sure.
How can I help you?

… say **no**

main analytics

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

… and add a ToC?

… and sequential page numbers?

.. also, not many—just about ~1,000

Yes, sure.
How can I help you?

… say **NO**

...on a calm Friday afternoon

Colleague

Michel

Are you still working?

Do you know how to easily merge PDFs?

... and add a ToC?

... and sequential page numbers?

.. also, not many—just about ~1,000

Yes, sure.
How can I help you?

... say **NO**

main analytics

…on a calm Friday afternoon

Colleague

Michel

Are you still working?

Yes, sure.
How can I help you?

Do you know how to easily merge PDFs?

… and add a ToC?

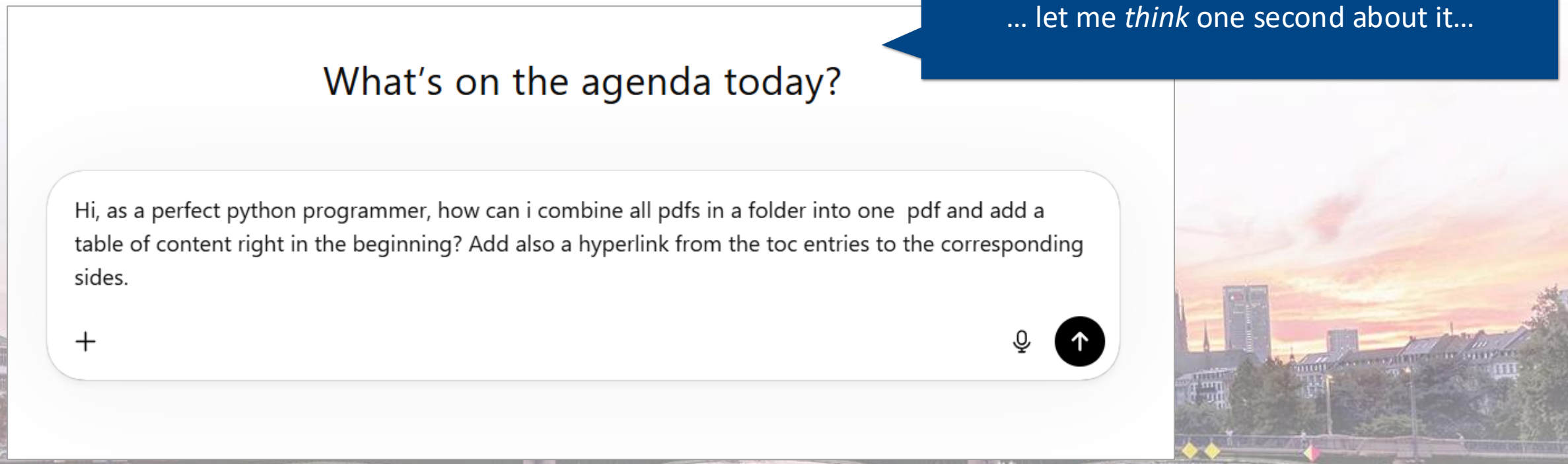
… let me *think* one second about it…

… and sequential page numbers?

.. also, not many—just about ~1,000

main analytics

...so, what to do?



main analytics

…so, what to do?

You can absolutely do this in Python—but it's a bit more involved than just merging PDFs because:

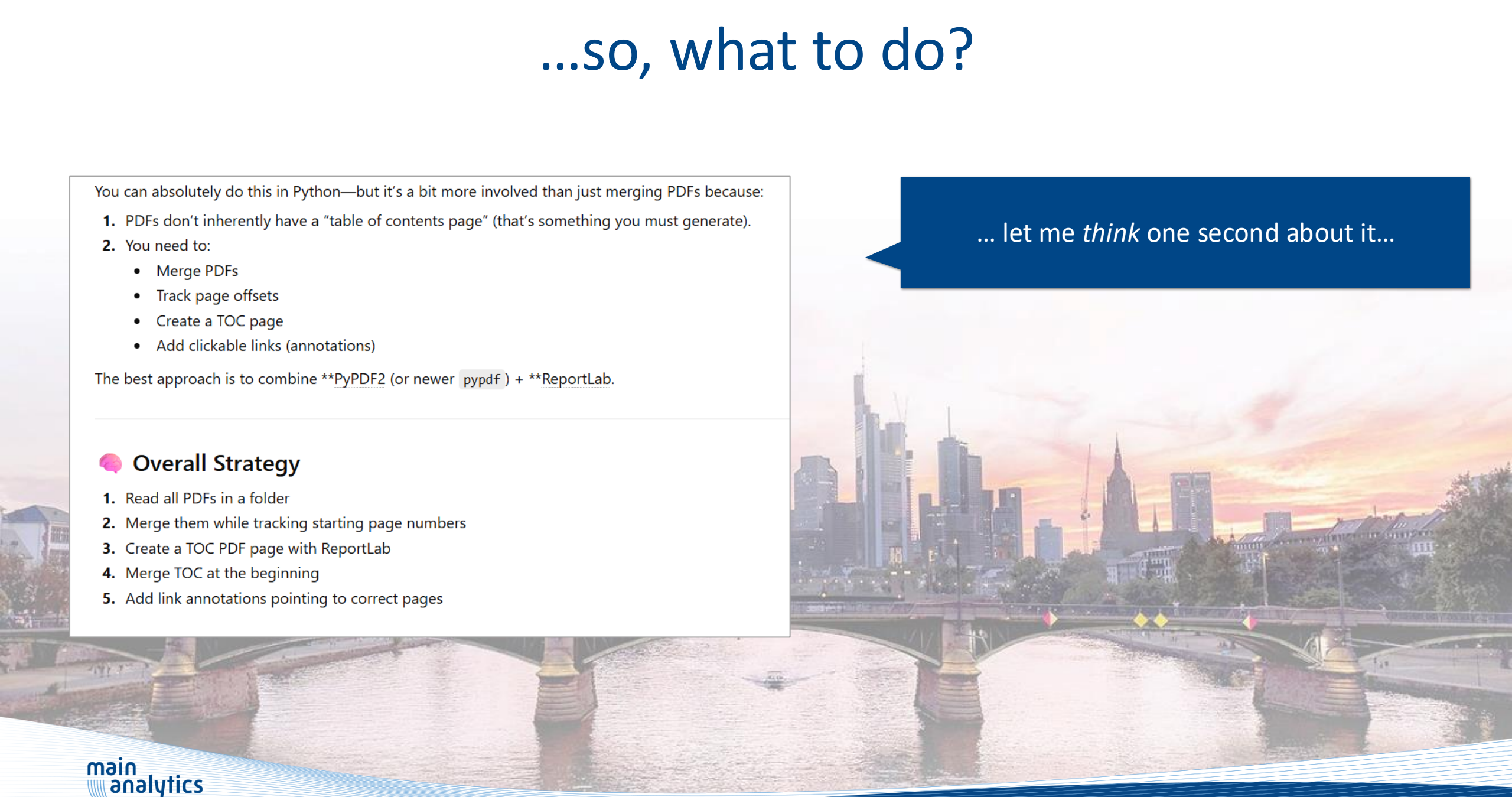
1. PDFs don't inherently have a "table of contents page" (that's something you must generate).
2. You need to:
   - Merge PDFs
   - Track page offsets
   - Create a TOC page
   - Add clickable links (annotations)

The best approach is to combine **PyPDF2 (or newer `pypdf`) + **ReportLab.

🧠 Overall Strategy

1. Read all PDFs in a folder
2. Merge them while tracking starting page numbers
3. Create a TOC PDF page with ReportLab
4. Merge TOC at the beginning
5. Add link annotations pointing to correct pages

… let me *think* one second about it…



main analytics

...so, what to do?

You can absolutely do this in Python—but it's a bit more involved than just merging PDFs because:

1. PDFs don't inherently have a "table of contents page" (that's something you must generate).
2. You need to:
   - Merge PDFs
   - Track page offsets
   - Create a TOC page
   - Add clickable links (annotations)

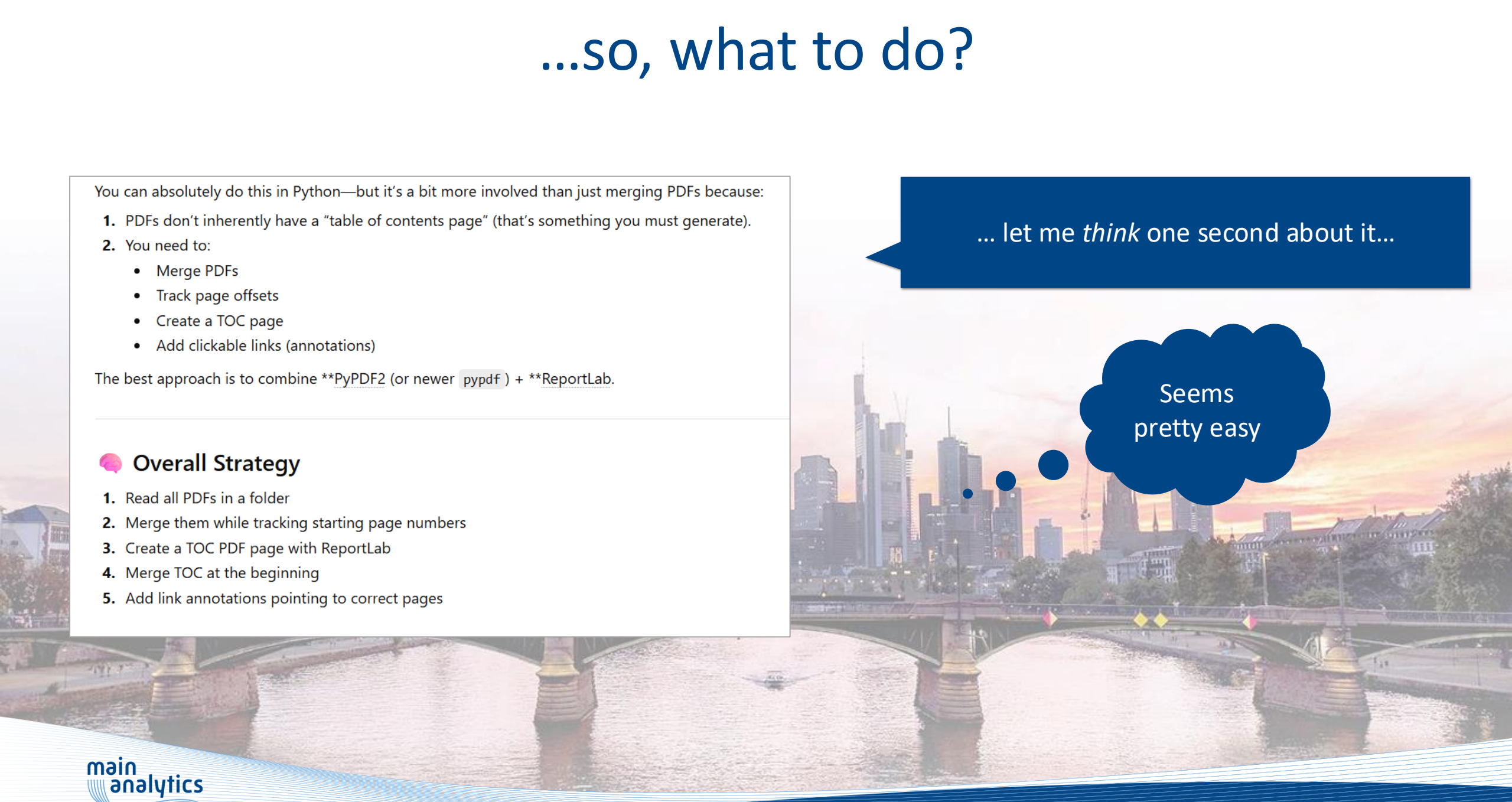
The best approach is to combine **PyPDF2 (or newer `pypdf`) + **ReportLab.

🧠 Overall Strategy

1. Read all PDFs in a folder
2. Merge them while tracking starting page numbers
3. Create a TOC PDF page with ReportLab
4. Merge TOC at the beginning
5. Add link annotations pointing to correct pages

… let me *think* one second about it…

Seems pretty easy



main analytics

The solution (almost)...

✅ Full Working Example

Install dependencies:

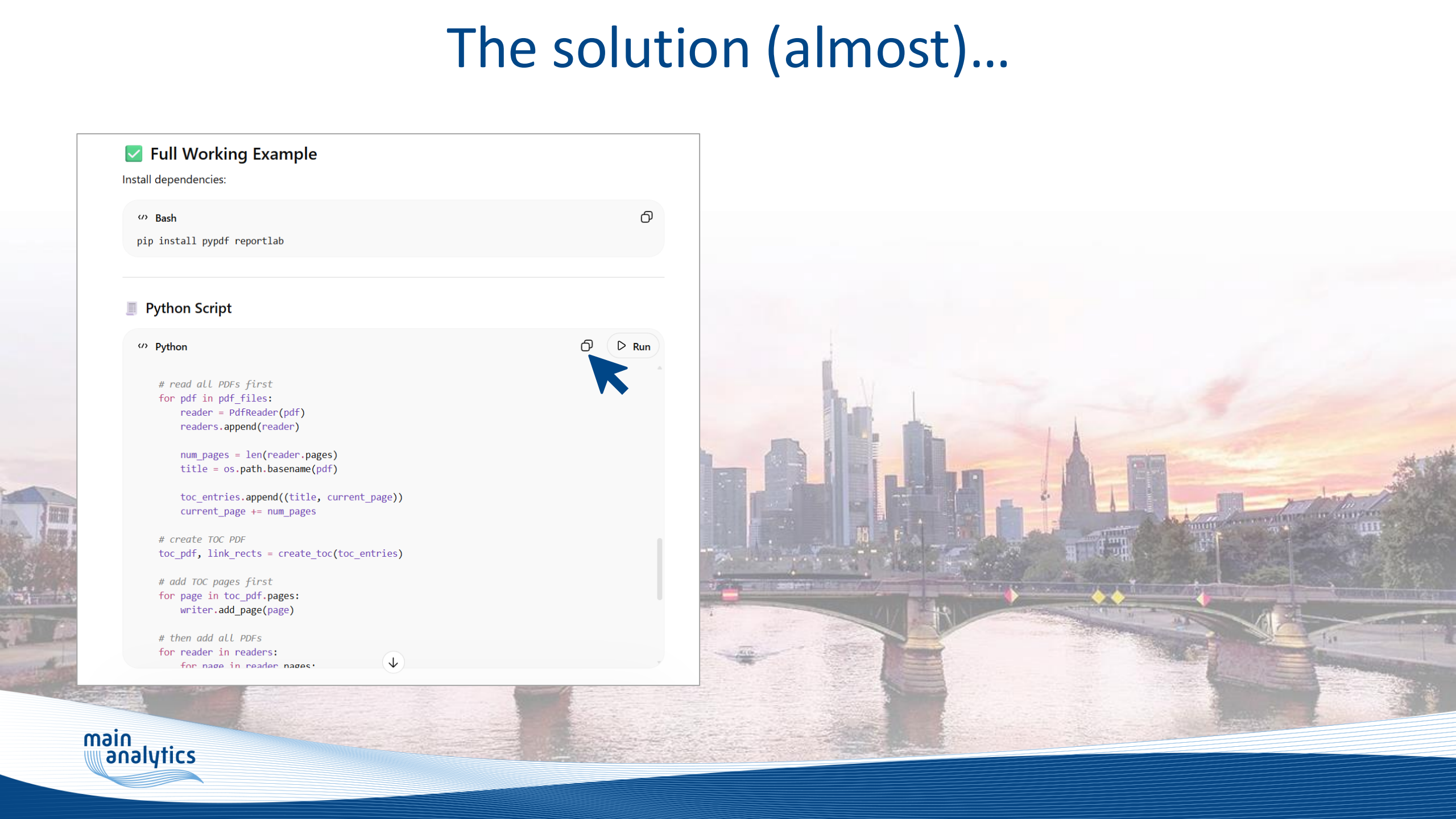
Bash

```
pip install pypdf reportlab
```

Python Script

Python Run

```
    # read all PDFs first
    for pdf in pdf_files:
        reader = PdfReader(pdf)
        readers.append(reader)

        num_pages = len(reader.pages)
        title = os.path.basename(pdf)

        toc_entries.append((title, current_page))
        current_page += num_pages

    # create TOC PDF
    toc_pdf, link_rects = create_toc(toc_entries)

    # add TOC pages first
    for page in toc_pdf.pages:
        writer.add_page(page)

    # then add all PDFs
    for reader in readers:
        for page in reader.pages:
```



main analytics

The solution (almost)...

✅ Full Working Example

Install dependencies:

Bash

```
pip install pypdf reportlab
```

Python Script

Python — Run

```
    # read all PDFs first
    for pdf in pdf_files:
        reader = PdfReader(pdf)
        readers.append(reader)

        num_pages = len(reader.pages)
        title = os.path.basename(pdf)

        toc_entries.append((title, current_page))
        current_page += num_pages

    # create TOC PDF
    toc_pdf, link_rects = create_toc(toc_entries)

    # add TOC pages first
    for page in toc_pdf.pages:
        writer.add_page(page)

    # then add all PDFs
    for reader in readers:
        for page in reader.pages:
```

... the problem

- pdfs are on a sponsor's system
- only SAS is available

main analytics

The solution (for real) …

✅ Full Working Example

Install dependencies:

Bash

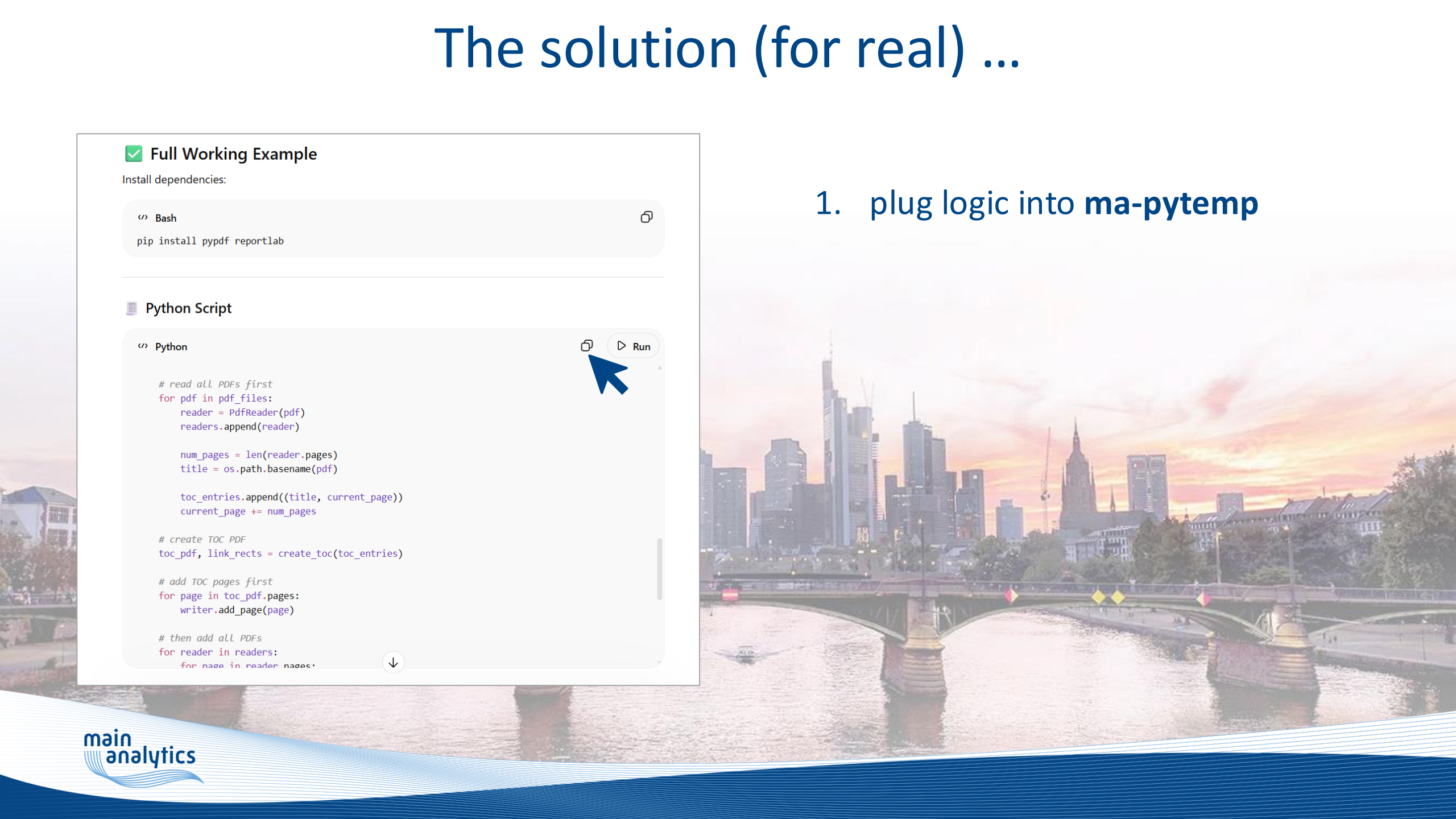
```
pip install pypdf reportlab
```

Python Script

Python — Run

```
    # read all PDFs first
    for pdf in pdf_files:
        reader = PdfReader(pdf)
        readers.append(reader)

        num_pages = len(reader.pages)
        title = os.path.basename(pdf)

        toc_entries.append((title, current_page))
        current_page += num_pages

    # create TOC PDF
    toc_pdf, link_rects = create_toc(toc_entries)

    # add TOC pages first
    for page in toc_pdf.pages:
        writer.add_page(page)

    # then add all PDFs
    for reader in readers:
        for page in reader.pages:
```

1. plug logic into **ma-pytemp**



main analytics

The solution (for real) …

✅ Full Working Example

Install dependencies:

Bash

```
pip install pypdf reportlab
```

Python Script

Python — Run

```
    # read all PDFs first
    for pdf in pdf_files:
        reader = PdfReader(pdf)
        readers.append(reader)

        num_pages = len(reader.pages)
        title = os.path.basename(pdf)

        toc_entries.append((title, current_page))
        current_page += num_pages

    # create TOC PDF
    toc_pdf, link_rects = create_toc(toc_entries)

    # add TOC pages first
    for page in toc_pdf.pages:
        writer.add_page(page)

    # then add all PDFs
    for reader in readers:
        for page in reader.pages:
```

1. plug logic into **ma-pytemp**
2. build **executable** and deploy

ma_pdf-compiler.exe

main analytics

…1 hour later

You’re the best

I created this executable:
path/to/**ma_pdf-compiler.exe**

just click on it

main analytics

Reference[1]

… on a beautiful Friday evening

Executables – What and Why?

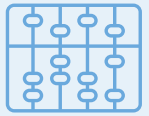
self-sustained programs	no further dependencies no installation
unlocking the power of	high-level languages libraries rest of the world
for all environments	GxP SCE
complexity overhead	develop build deploy
abstract complexity	standardize template prebuild focus on automation

main analytics

Executables – What?

Simple answer

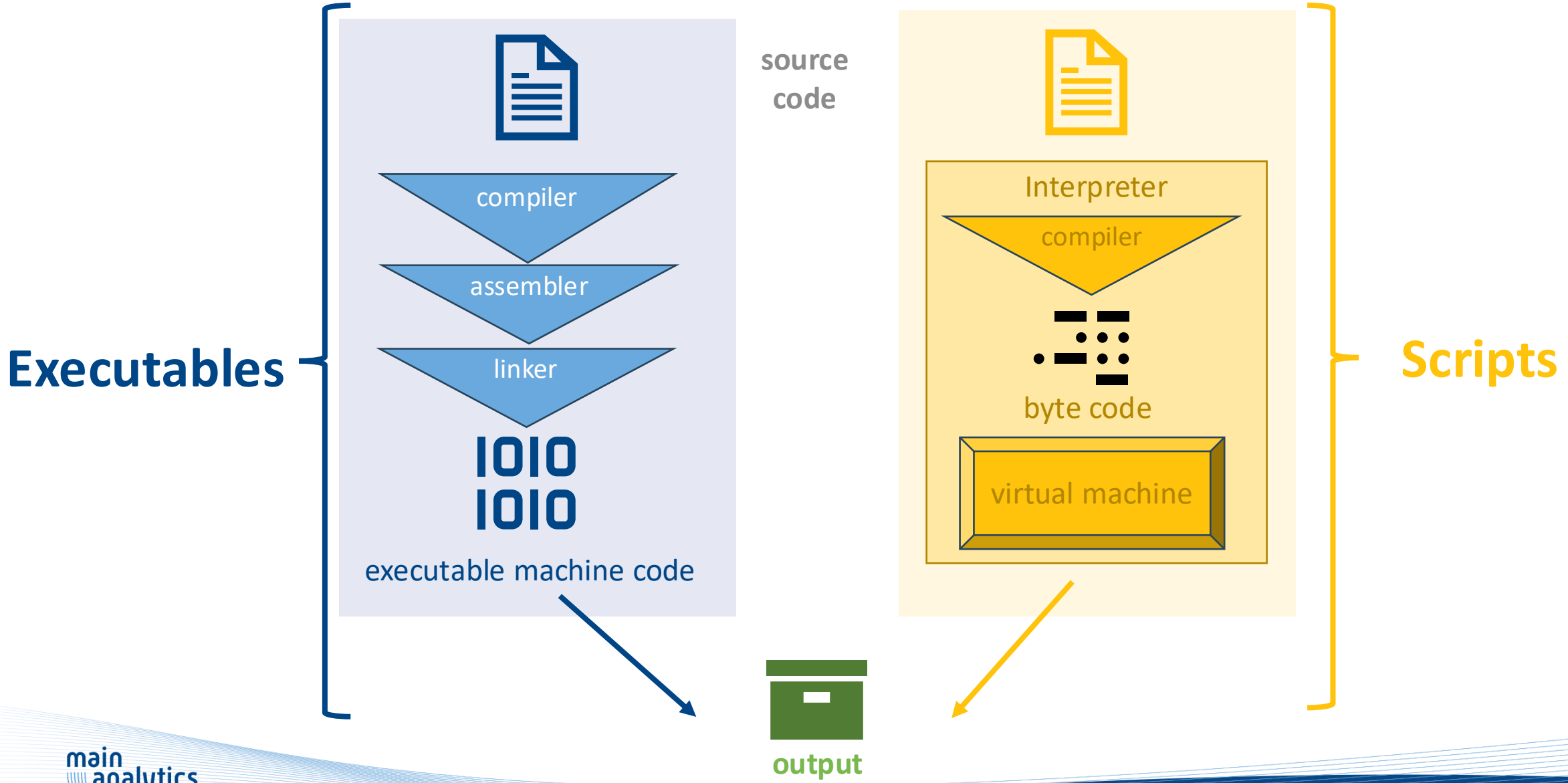
- .exe
- “click and play”



Detailed Answer

- program without needing further compilation
- **machine code** specific to CPU architecture operation system
- **deliverable** of software engineering

Executables vs. Scripts



Executables – Why?

- Many common languages are interpreted
  Java, Python, R, C#,…
- Require “extra” software -> often not available

Executables – Why?

- Many common languages are interpreted
  Java, Python, R, C#,…
- Require “extra” software -> often not available

Why not using what is already available?

Executables – Why?

- Many common languages are interpreted
  Java, Python, R, C#,…
- Require “extra” software -> often not available

Why not using what is already available?

Executables – Why?

- Many common languages are interpreted
  Java, Python, R, C#,…
- Require “extra” software -> often not available

Why not using what is already available?

→ unleash the internet

→ unleash AI

Challenges of Executables

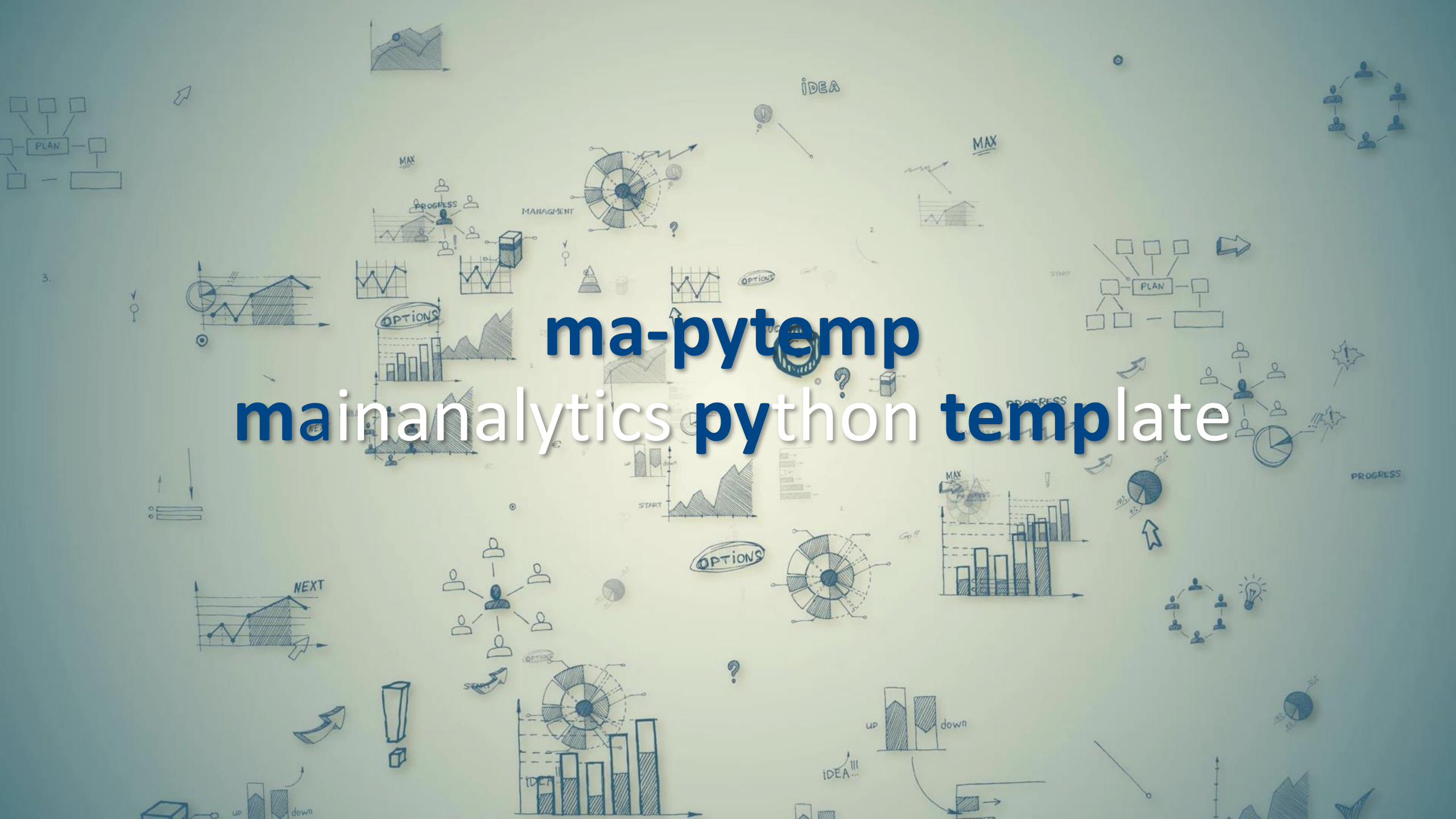
Programming language	Dependency management	User Interface (Framework)	Adaptivity
Hard coded resources	Robustness	Error & Exception handling	IO
User feedback	Logging	Unit Testing	Validation
Version control	Build	Building machine vs. target machine	Deployment

main analytics

Executables as Tools in Statistical Programming

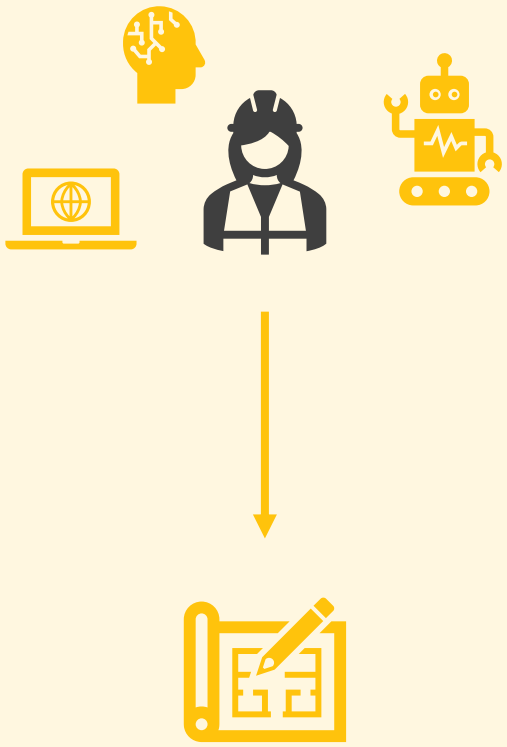
powerful and modern
even for GxP environments

high **development complexity**
for the “box” around although
core logic often **simple**


ma-pytemp
mainanalytics python template

ma-pytemp: Idea

ma-pytemp: Idea



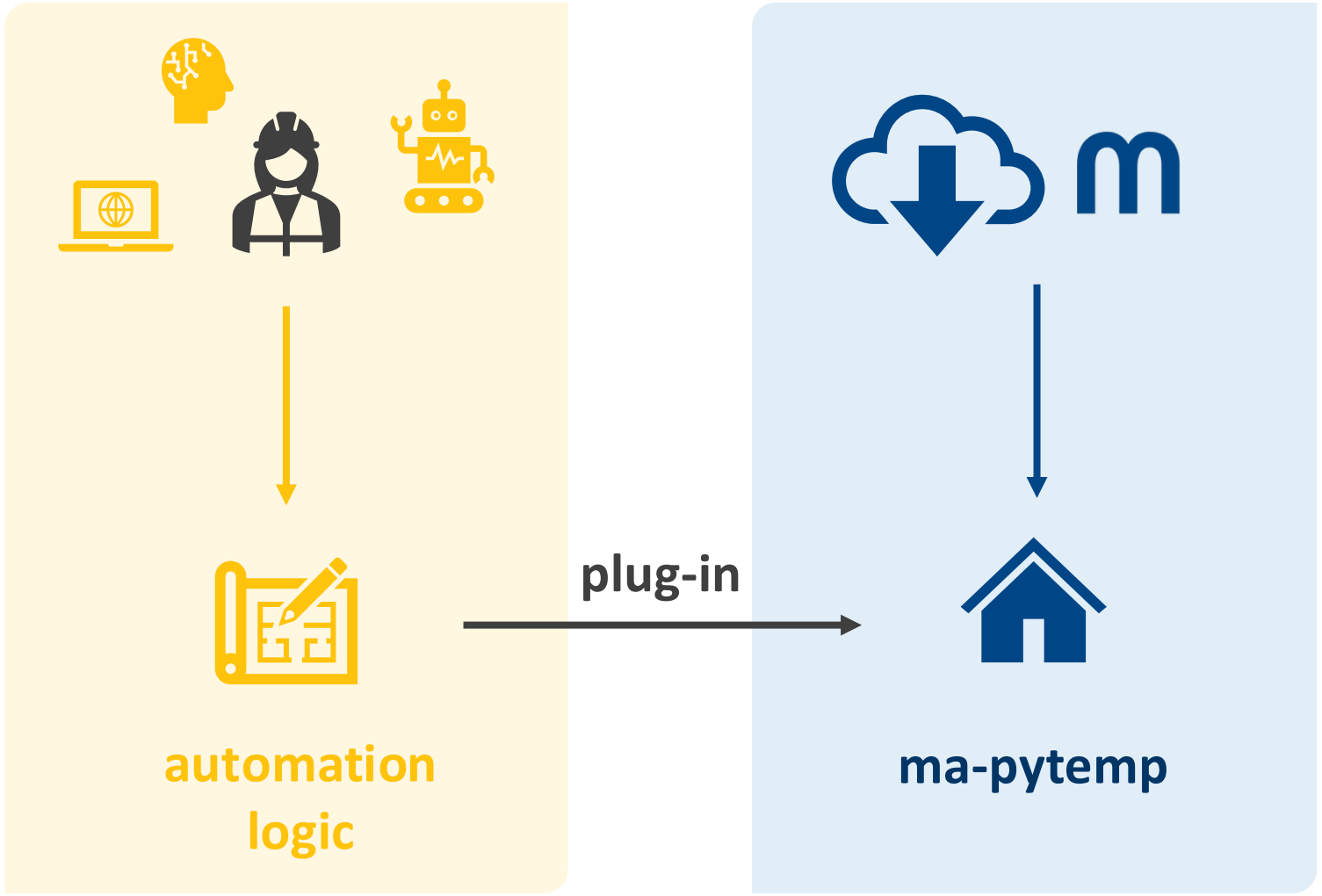
→ **search for solutions**

- Libraries
- Repos
- Internet

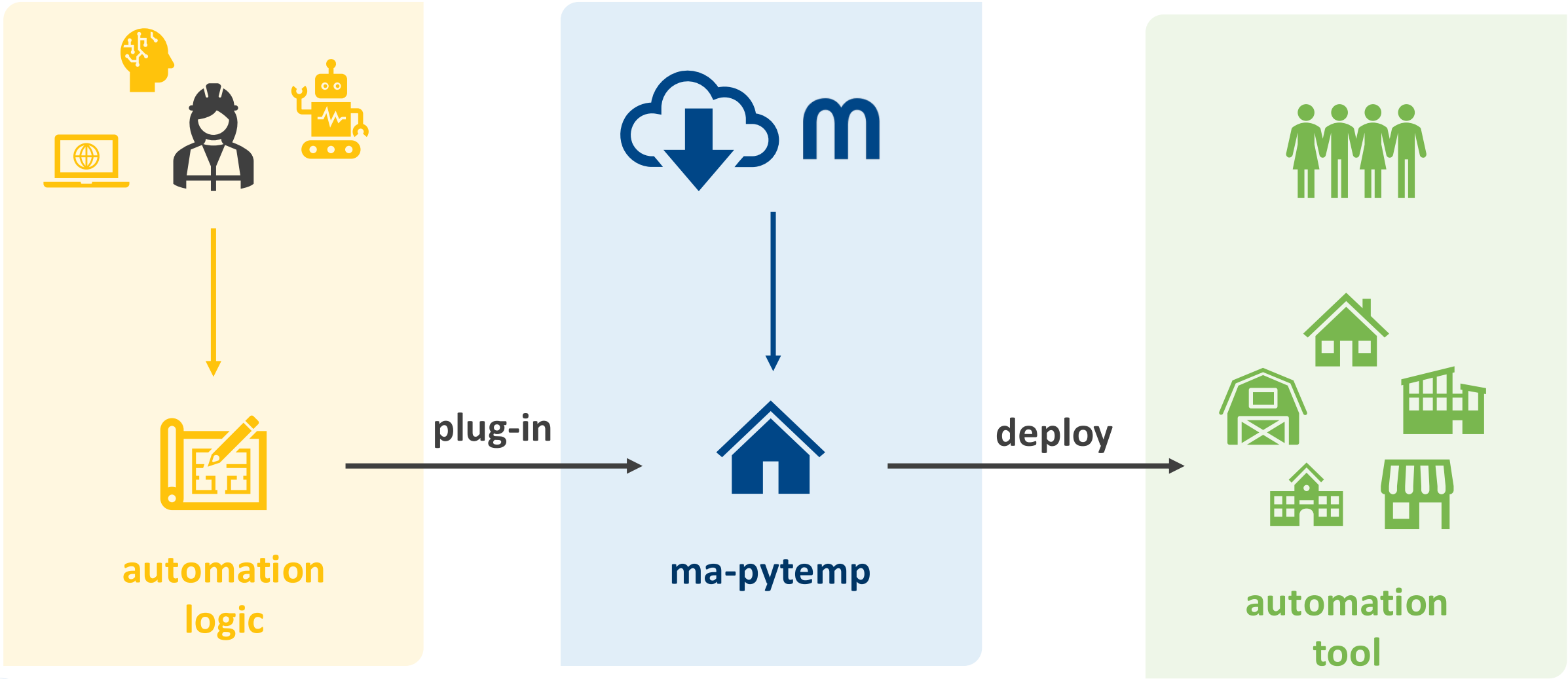
→ **Vibe-coding**

→ **Use your own skills**

ma-pytemp: Idea



ma-pytemp: Idea



ma-pytemp: Features

Key features	Python
	Dependency & Project Management
	GUI Framework
	Structured architecture
	Logging
	Adaptability
	Validation
	CICD

ma-pytemp: Features

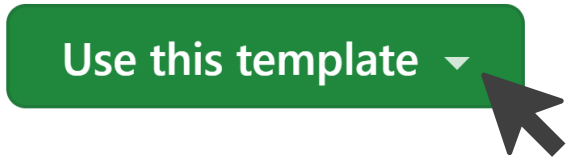
Key features	Python
	Dependency & Project Management
	GUI Framework
	Structured architecture
	Logging
	Adaptability
	Validation
	CICD

More features

- Template git repository
- Version control
- Documentation
- Basic code quality
- Customization
- .ico creation
- multi-threading
- …

how to use ma-pytemp?

1



https://github.com/mainanalytics

how to use ma-pytemp?

1

Use this template

https://github.com/mainanalytics

2

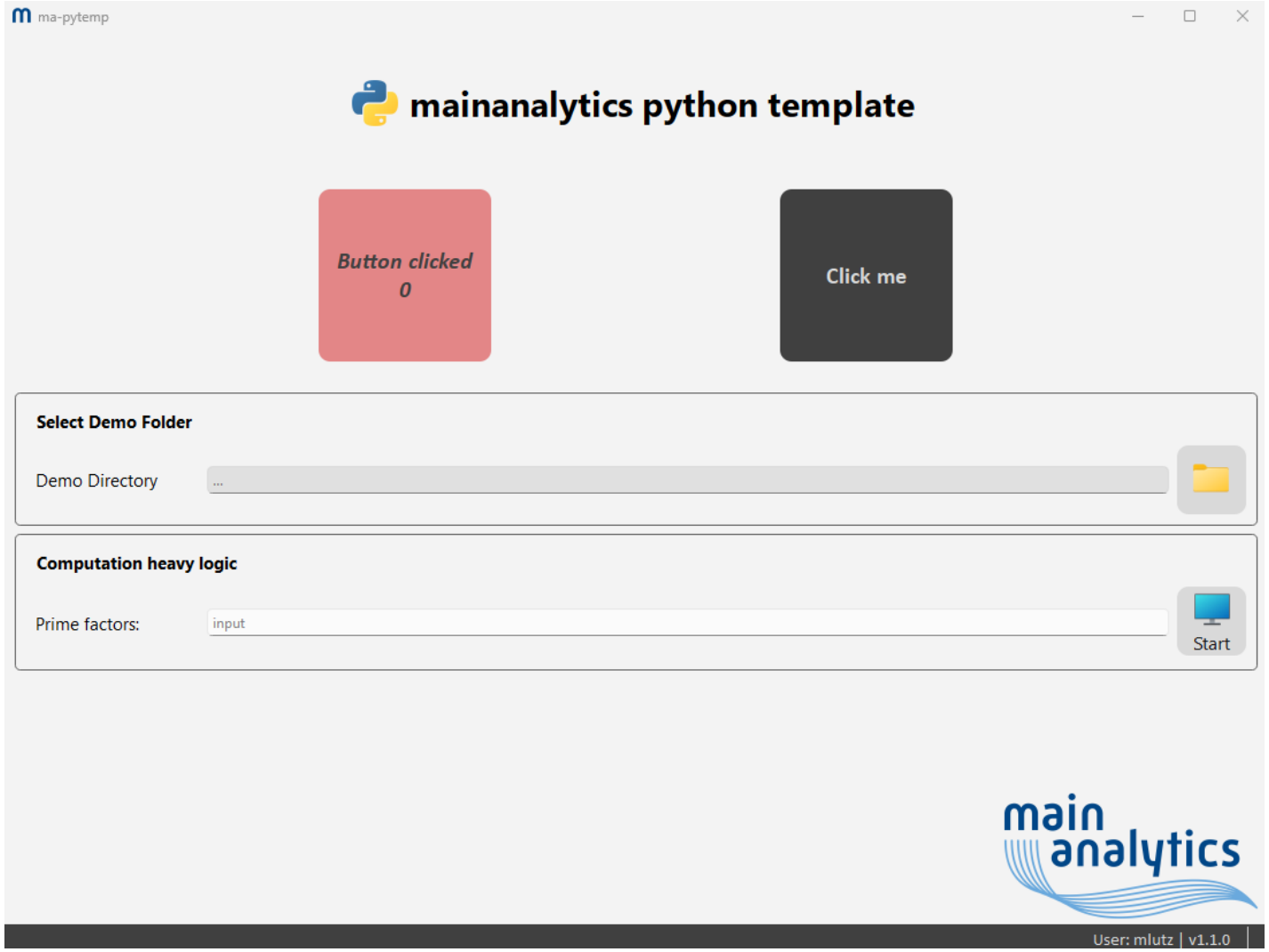
Poetry

initialize project
create VENV
install dependencies
...

- C:\Users\mlutz> cd ma-pytemp
- C:\Users\mlutz> poetry install

User Interface

Graphical User Interface

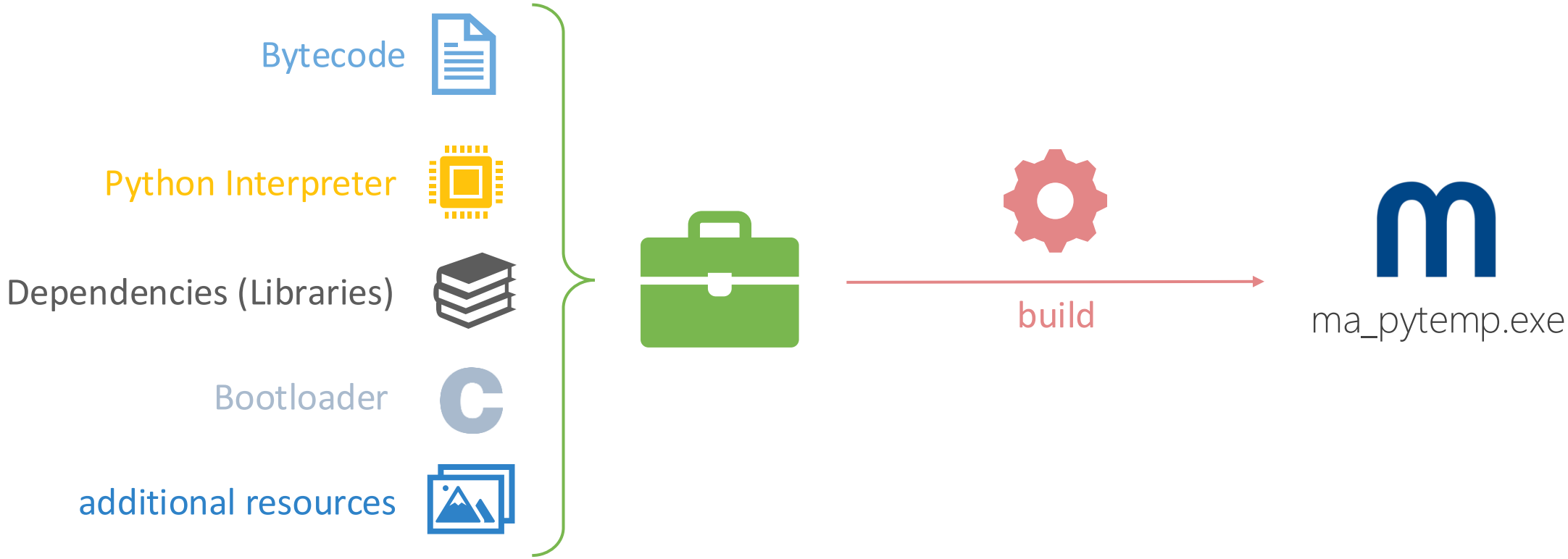


Build

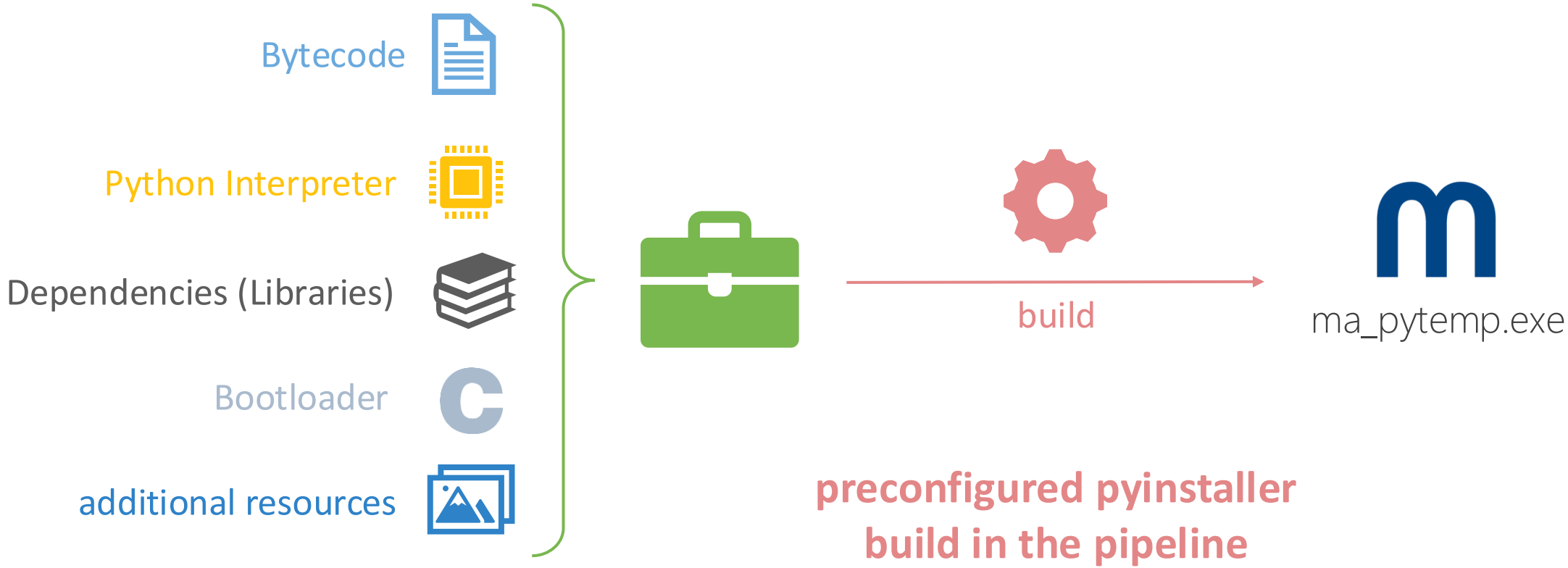
build

creating the executable

I'm packing my suitcase



I'm packing my suitcase



preconfigured pyinstaller build in the pipeline

Deployment

ma-pytemp: Deployment

predefined CICD workflow

pytest

Ruff

Pylint

Star your Python code!

ma-pytemp: Workflow

Workflow file for this run

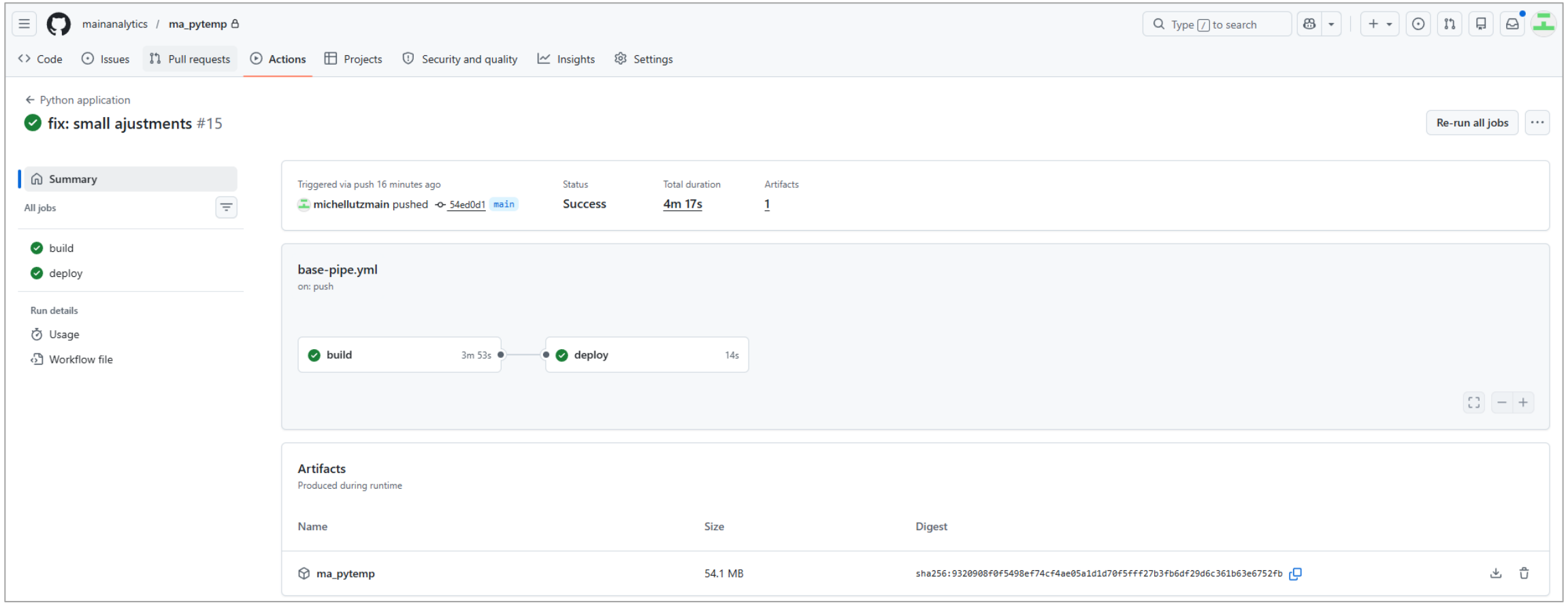
.github/workflows/base-pipe.yml at 84d487c

```yaml
# This workflow will install Python dependencies, run tests and lint with a single version of Python
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python

name: Python application

on:
  push:
    branches:
      - main
      - dev
  pull_request:
    branches:
      - main
      - dev

permissions:
  contents: read

jobs:
  build:
    strategy:
      fail-fast: false

    runs-on: windows-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v5

    - name: Set up Python 3.13.12
      uses: actions/setup-python@v6
      with:
        python-version: "3.13.12"


    - name: Install poetry
      run: |
        python -m ensurepip
        python -m pip install --upgrade pip
        python -m pip install poetry
```

[Workflow runs · mainanalytics/ma_pytemp]

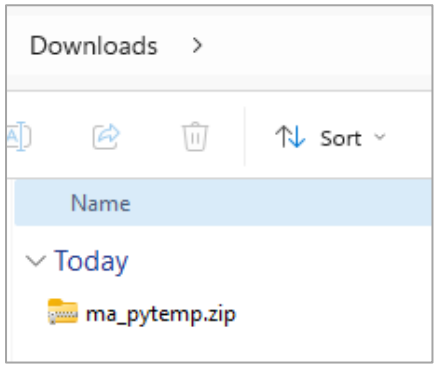
```yaml
    - name: Install dependencies with Poetry
      run:  poetry install --with dev

    - name: Set PYTHONPATH to include the source directory
      run: echo "PYTHONPATH=$PWD/src/ma_pytemp" >> $GITHUB_ENV

    - name: Formatter
      run: poetry run ruff format

    - name: Lint with pylint (fail on warnings/errors)
      run: poetry run pylint src/

    - name: Unit Test and report creation
      run: poetry run pytest

    - name: Build executable
      run: poetry run pyinstaller -y main_standard.spec

    # Upload the compiled executable
    - name: Upload artifacts
      uses: actions/upload-artifact@v7
      with:
        name: ma_pytemp
        path: |
          dist/
          config/
          validation_report/
        retention-days: 10

    # Show artifact id and store in enviroment variable
    - name: Output artifact ID
      run:  |
        echo 'artefact_id=${{ steps.artifact-upload-step.outputs.artifact-id }}' >> $GITHUB_ENV
        echo 'Artifact ID is ${{ steps.artifact-upload-step.outputs.artifact-id }}'

  deploy:
    runs-on: windows-latest
    needs: build

    steps:
    - name: Download artifact
      uses: actions/download-artifact@v7
      with:
        name: ma_pytemp
        path: /opt

    - name: List Artifacts
      run: ls /opt
```

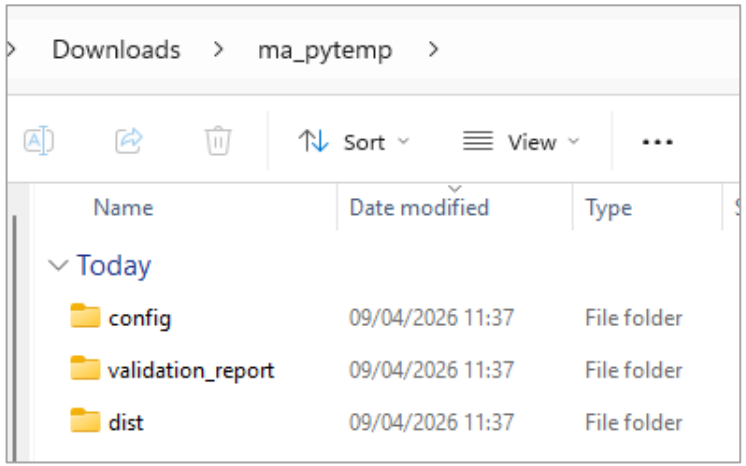
ma-pytemp: Deployment

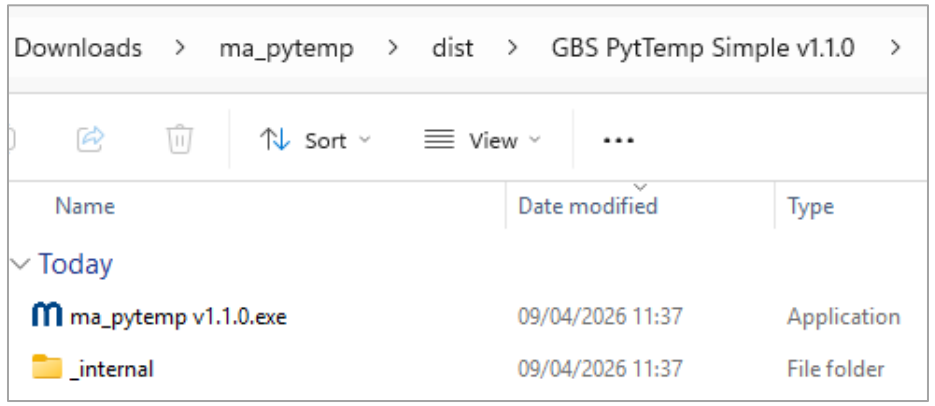


ma-pytemp: Artifact

artifact is a .zip folder

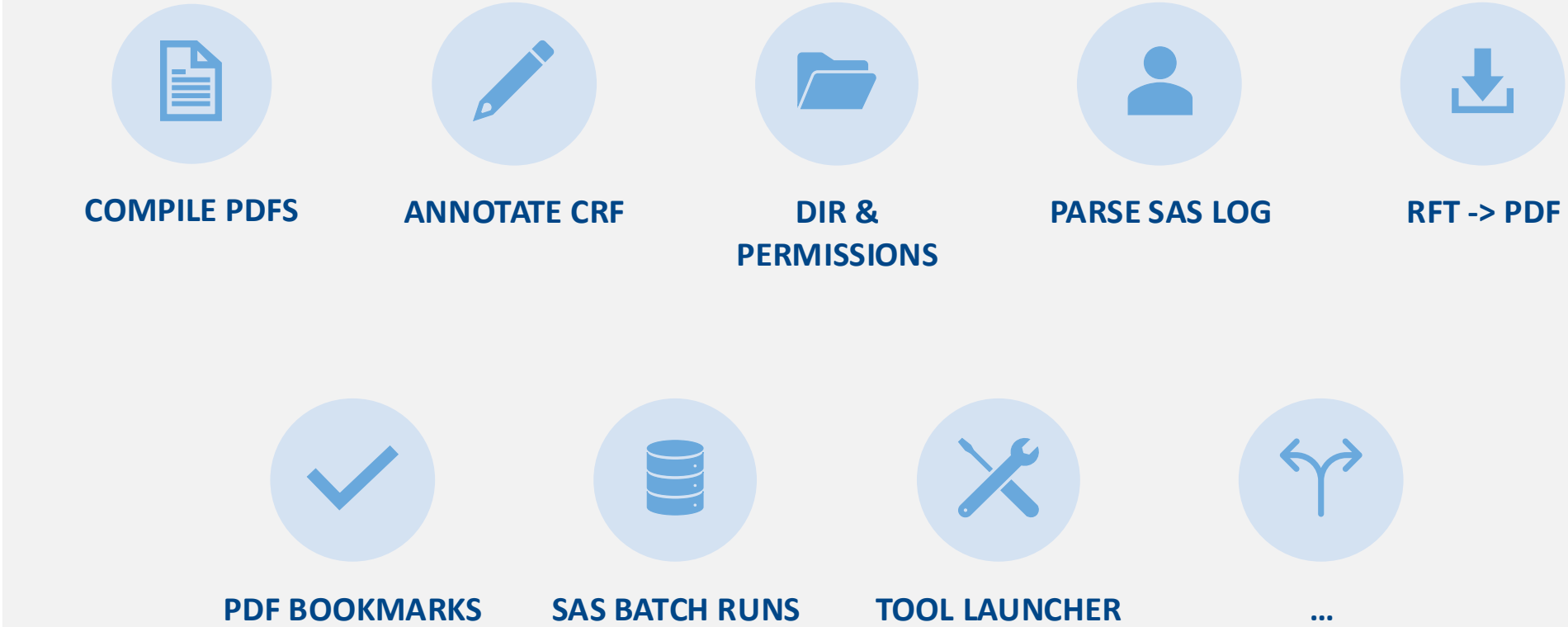







into production

into production



CRF Bookmarks

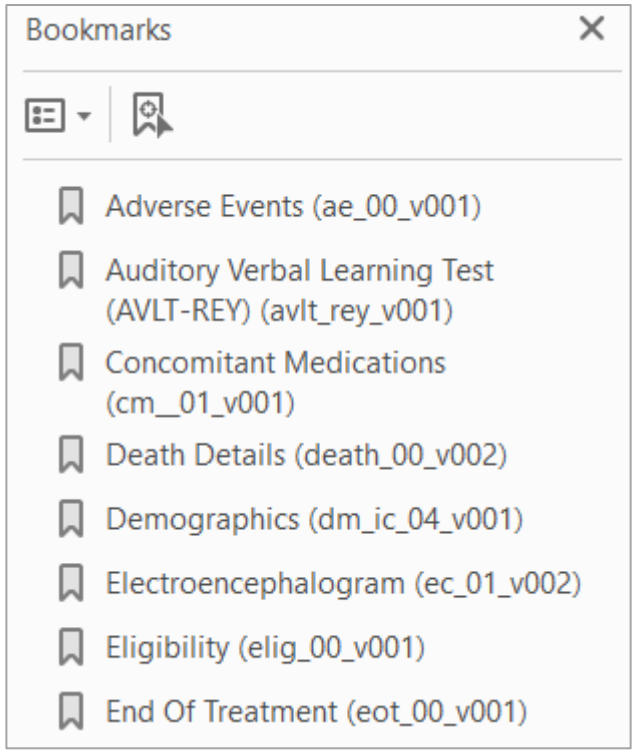
Situation

- a software provider cannot extract CRFs with bookmarks by form and visit
- >> 20 studies
- unclear when or even if this will be fixed

Current solution

- TDMs create bookmarks by hand
- painful and error prone

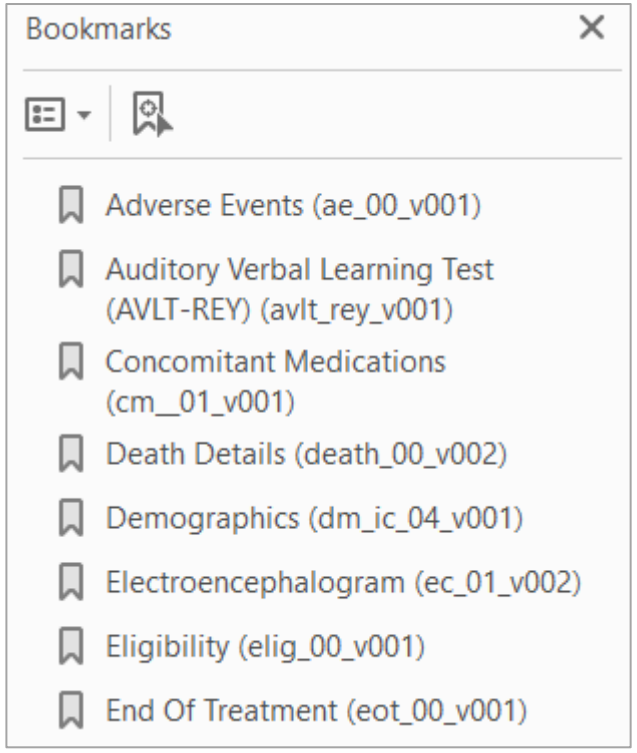
CRF Bookmarks



- first level bookmarks <form_label> (<form_name>)
- no visit information
- Study Definition Specs (SDS) contains visit information

CRF Bookmarks

crf_raw.pdf



sds.xlsx

Form Label	Form Name	Running Records	Screening		Baseline
		Running Records	Screening 1	Screening 2	Baseline
Adverse Events	ae_00_v001	X			
Concomitant Medications	cm__01_v001	X			
Death Details	death_00_v002	X			
Electroencephalogram	ec_01_v002	X			
Exposure	exp_000	X			
Injection Site Reactions	inj_site_00_v001	X			
Demographics	dm_ic_04_v001		X		
Eligibility	elig_00_v001		X		
Informed Consent	ic_01_v001		X		
Medical History	mh_00_v001		X		
Vital Signs (Weight, Height And Temperature)	vs_wgh_hgt_temp_000		X		
Ophthalmic Examination	opth_exm_v001		X	X	X
Auditory Verbal Learning Test (AVLT-REY)	avlt_rey_v001			X	
Vital Signs (Temperature)	vs_temp_000			X	
Patient Health Questionnaire-9 (PHQ-9)	phq_9_v001				X
Satisfaction With Life Survey (SWLS)	swls_000				X
Hamilton Depression Rating Scale - 17 (HAMD-17)	hamd_17_00_v001				X
Vital Signs (Weight And Temperature)	vs_wgh_temp_000				X
Patient Summary	pat_sum_000				
End Of Treatment	eot_00_v001				

CRF Bookmarks

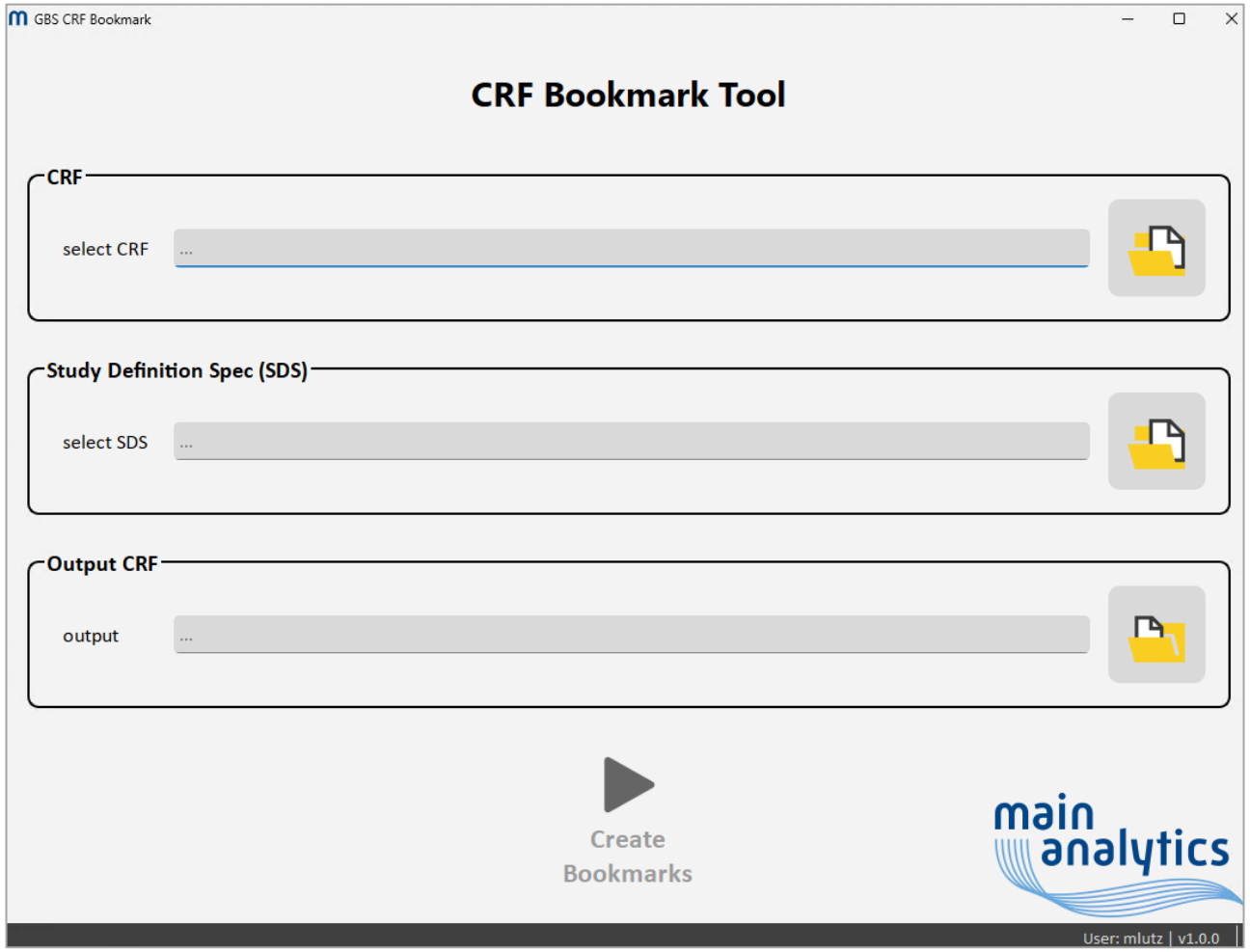
Solution

1. Read in crf_raw.pdf with pymupdf
2. Extract input bookmarks
3. Read in sds.xlsx with pandas
4. Match Form Label and Form Names from original bookmarks with sds
5. Create new bookmarks
   - by Form
   - by Visit
6. Save new file

Deployment

1. Plug logic into **ma_pytemp**
2. Add 3 IO Buttons
3. Add Start button

Example: CRF Bookmarks



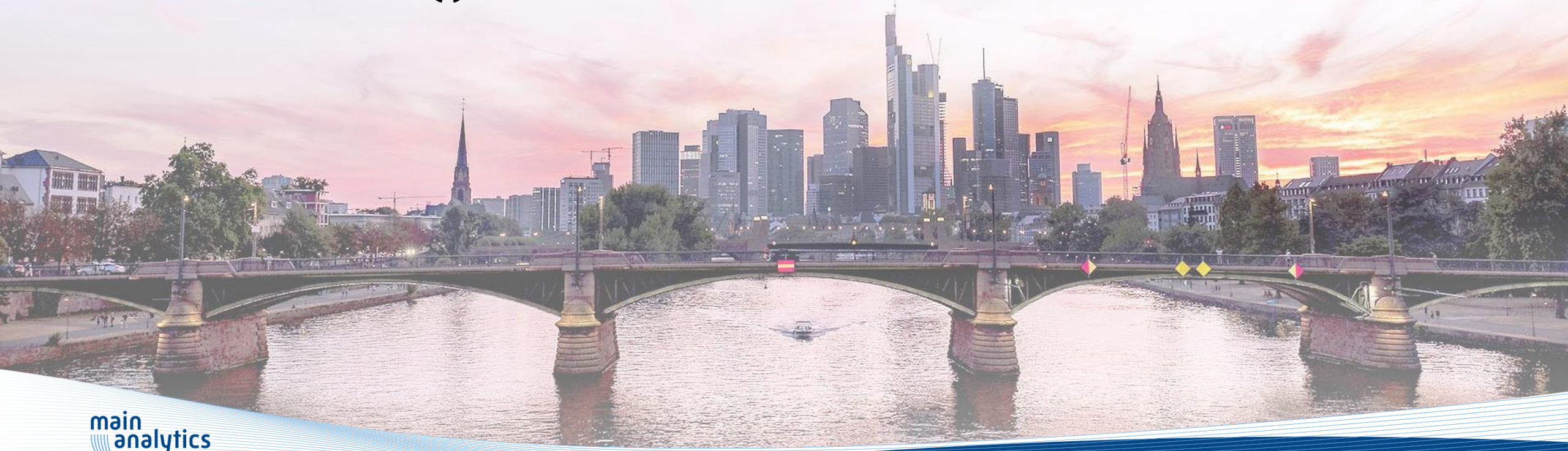
Executables – What and Why?

self-sustained programs	no further dependencies no installation
unlocking the power of	high-level languages libraries rest of the world
for all environments	GxP SCE
complexity overhead	develop build deploy
abstract complexity	standardize template prebuild focus on automation

main analytics

give it a try

https://github.com/mainanalytics/ma_pytemp



main analytics

References

[1] – Image taken from: File:Bud Spencer 1974.png - Wikimedia Commons **22-Apr-2026: i*t is in the public domain***
[2] – Image taken taken from: Michel Lutz – “An enquiry of southeastern Swabia traditions”. All rights beerserved ©