Python meeting

Python meeting 8

3rd June 2026

The Python Meeting will start soon

COURSE PYTHON SCHEDULE

01: 20250114, 9am: 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: 20250121,10am: Melt, split, for – Case-studies: D01: Trial balance, P01_19: Supplier debtors, O01_19: Customer creditors

- Using melt, split
- For loop

03: 20250128, 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: 20250204, 9am: JSON, expressions, rolling Window – Case-study: P02_19: Split POs

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

05: 20250211, 9AM: zip, dynamic fields – Case-study: P02_03: PO whilst blocked

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

COURSE PYTHON SCHEDULE

06: 20250218, 10am: Cumulative sum, dates – Case-study: P02_18: PO slow rotation

- cum_sum for calculating future or past inventory movements

07: 20250225, 9am: Isolation forest for unusual transactions – Case-study: Unusual bank transfers

- Using isolation forest library to score for unusual transactions

08: 20250304: 9am: SpaCy – Case-study: P01_11/ O01_11: Suppliers/ customers that are people

- Using SpaCy NLP object to categorize names

09: 20250311: 9am: Contract review – Case-study: Scoring of contracts

- Using Gemini model to check for paragraphs of similar meaning

10: 20250318: 10am: Open AI API – Case-study: Generating audit reports

- Using API to OpenAI to generate text for audit reports

11: 20250325: 9am: 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: 20250401: 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

Case-study

→ What is the problem we are trying to solve and which is the best tool to solve it?

02

SpaCy code overview

SpaCy can also be used for other things

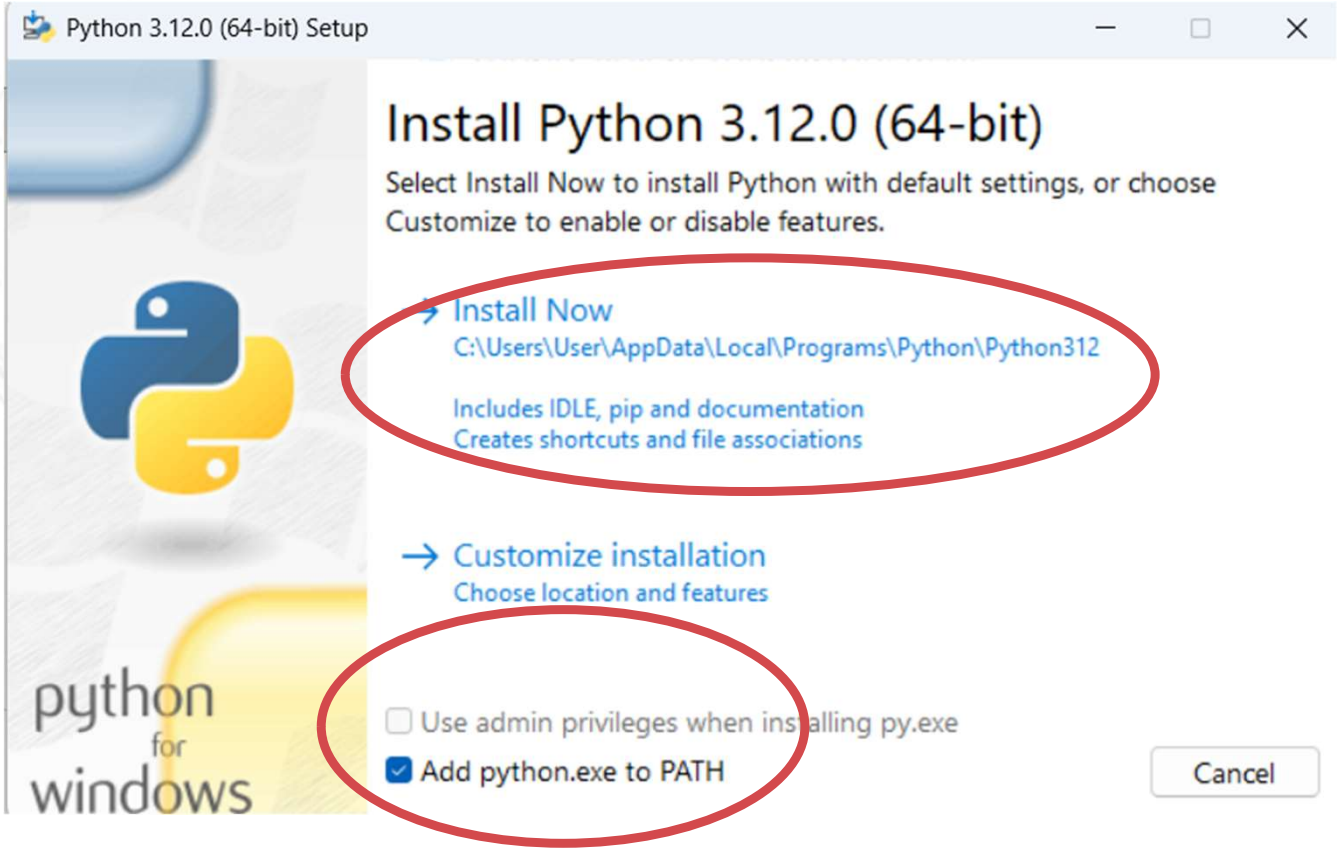
Trouble shooting – SpaCy only supports up to python version 3.12

- The python code for this session uses SpaCy.
- SpaCy only runs on python 3.12.
- Check your version of python by typing python --version in a black CMD box.
- If your version of python is later than 3.12, then you need to also install python 3.12 on your machine (you can keep existing version of python): Different options:
  - Inside the VSC terminal type: winget install Python.Python.3.12
  - Or download from this location:
    - https://www.python.org/ftp/python/3.12.10/python-3.12.10-amd64.exe
  - Or go to:
    - https://www.python.org/downloads/release/python-3120/
    - Scroll-down and click on the Windows 32-bit or 64-bit (depending on your machine) installer:

Version	Operating system	Description	File size	Sigstore	GPG	MD5 checksum
Gzipped source tarball	Source release		25.9 MB	.sigstore	SIG	d6eda3e1399cef5dfde7c4f319b0596c
XZ compressed source tarball	Source release		19.6 MB	.sigstore	SIG	f6f4616584b23254d165f4db90c247d6
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	43.3 MB	.sigstore	SIG	eddf6f35a3cbab94f2f83b2875c5fc27
Windows installer (64-bit)	Windows	Recommended	25.3 MB	.sigstore	SIG	32ab6a1058dfbde76951b7aa7c2335a6
Windows installer (32-bit)	Windows		24.0 MB	.sigstore	SIG	de59862985bf7afa639f2e4f9e2a722c
Windows installer (ARM64)	Windows	Experimental	24.5 MB	.sigstore	SIG	230c703e3b8b3d92765d118afa7b2f78
Windows embeddable package (64-bit)	Windows		10.5 MB	.sigstore	SIG	8e24d2b26a8dbf1da0694b9da1a08b2c
Windows embeddable package (32-bit)	Windows		9.4 MB	.sigstore	SIG	c2047dc270c4936f9c64619bb193b721
Windows embeddable package (ARM64)	Windows		9.8 MB	.sigstore	SIG	3da91ef1a86a8a210a32ea99c709dd93

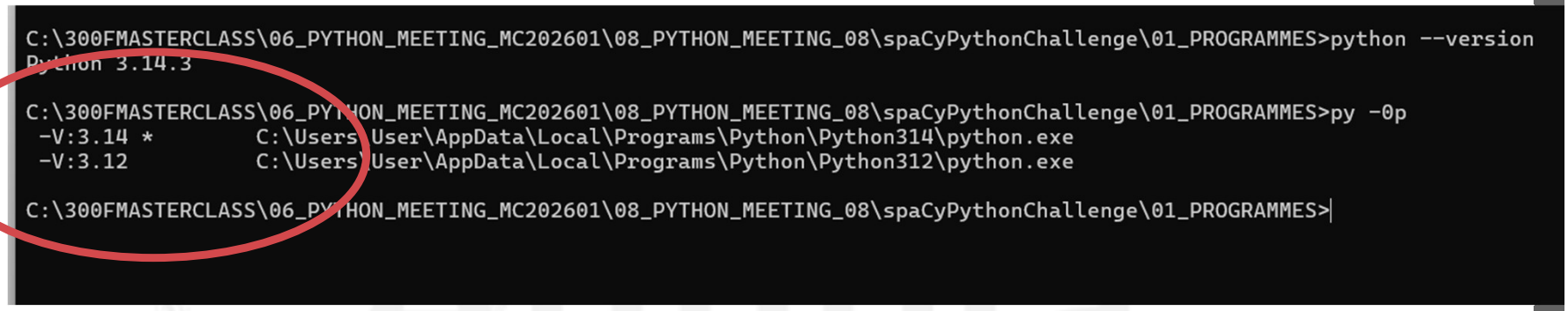
Trouble shooting - SpaCy only supports up to python version 3.12

- Double-click the installer to install it on your machine:
  - When you are installing be careful to tick add python to PATH



Trouble shooting - SpaCy only supports up to python version 3.12

- After installation, run py -0p in a CMD window to check which python versions you have:



- You should now have two python versions on your machine.

Trouble shooting - SpaCy only supports up to python version 3.12

- Now we will recreate our virtual environment, but using the python version 3.12, instead of the python version 3.14.
  - Go to the PROGRAMS folder of the project for this python session
  - If you already have a venv folder in the PROGRAMS folder, then you should delete it
  - In the Windows search bar type CMD
  - In the black box that opens, ensure that it has the path to the PROGRAMS folder and then type code .
  - Then, in Visual Studio Code type:
    - py -3.12 -m venv venv
  - Once the venv is created, you can open it and check the scripts version:

Trouble shooting – requirements.txt – we can update the versions

- Follow the instructions in the README.md for using the Python project:

README.txt

```
1   # 300Framework - P01_11: Suppliers that are people
2   ---
3   ---
4   This is a really exiting version of the spaCy test, bec
    choose to train on incorrect values produced by spaCy i
    continuously improve the result that you get. In this w
    how you can really easily build a light-weight AI proje
    feedback loop from your auditors for continuous improve
    example is really easy to understand and use - and give
    foundations for understanding and using more advanced m
5
6   ## Instructions
7   Please note: spaCy currently only supports Python versi
    therefore we use a Python envrionment 3.12 for this pro
8
9   ### 1/ Create Virtual Environment
10
11  Ensure that you have Python version 3.12 on your machin
    have it, download it from https://www.python.org/downlo
12
13  In Visual Studio Code, open a PowerShell terminal (File
    the location: (you can use cd to navigate to the 02_PRO
14
15  ```text
16  PS C...\02_PROGRAMMES
17  ```
18  Ensure that the address in the terminal matches
    C:\YourProjectRootFolder\02_PROGRAMMES.
19  The rest of the instructions are commands that we will run in the
    Powershell terminal at this location.
```

README

300Framework – P01_11: Suppliers that are people

This is a really exiting version of the spaCy test, because you can choose to train on incorrect values produced by spaCy in order to continuously improve the result that you get. In this way, you can see how you can really easily build a light-weight AI project that supports feedback loop from your auditors for continuous improvement. This spaCy example is really easy to understand and use - and gives us the foundations for understanding and using more advanced models.

Instructions

Please note: spaCy currently only supports Python versions up to 3.13, therefore we use a Python envrionment 3.12 for this project

1/ Create Virtual Environment

Ensure that you have Python version 3.12 on your machine. If you do not have it, download it from https://www.python.org/downloads.

In Visual Studio Code, open a PowerShell terminal (File-> Terminal) at the location: (you can use cd to navigate to the 02_PROGRAMMES folder)

```
PS C...\02_PROGRAMMES
```

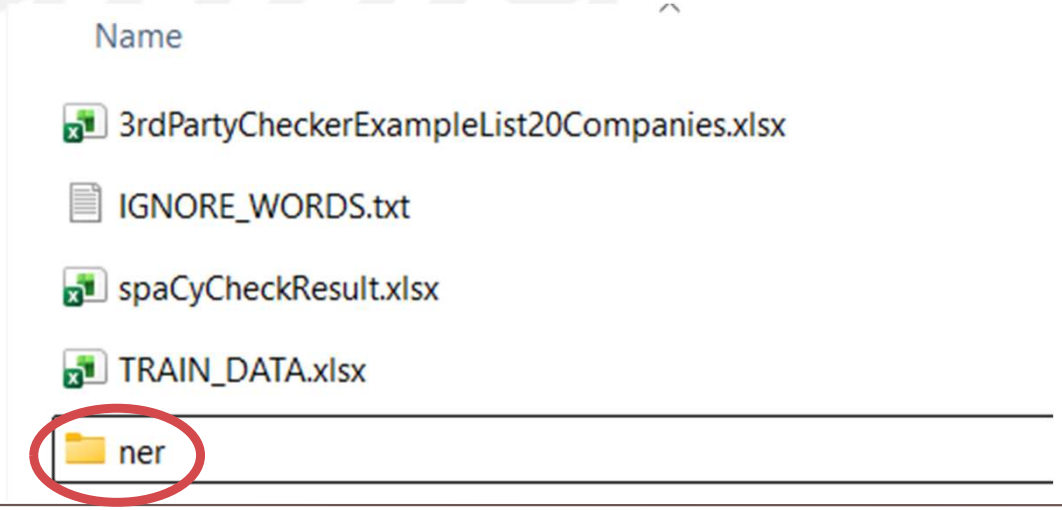
Ensure that the address in the terminal matches C:\YourProjectRootFolder\02_PROGRAMMES. The rest of the instructions are commands that we will run in the Powershell terminal at this location.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(venv) PS C:\300FMASTERCLASS\03_PYTHON_MC202602\08_PYTHON_MEETING_08\P01_11> gi
```

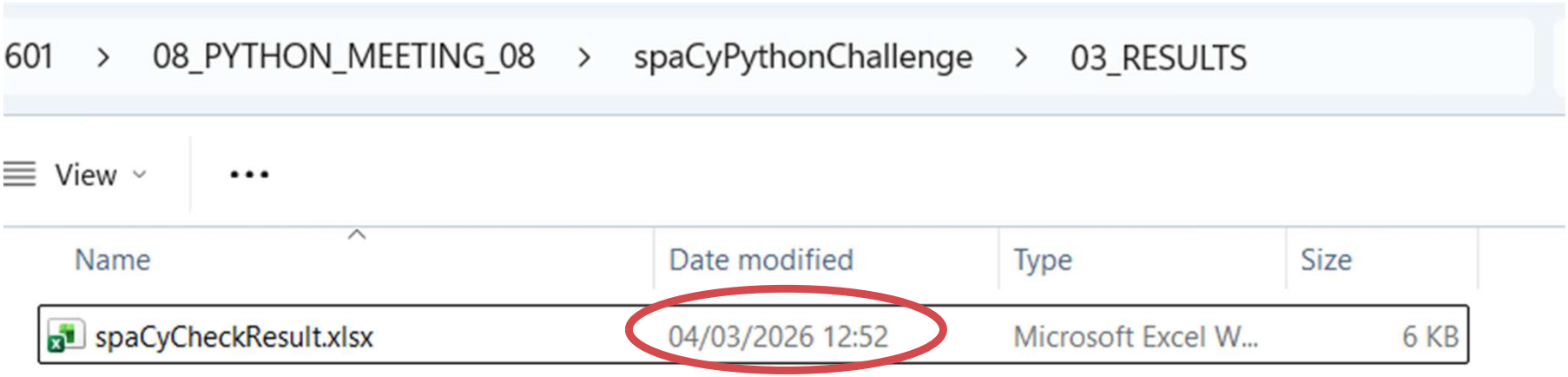
Trouble shooting – missing ner folder

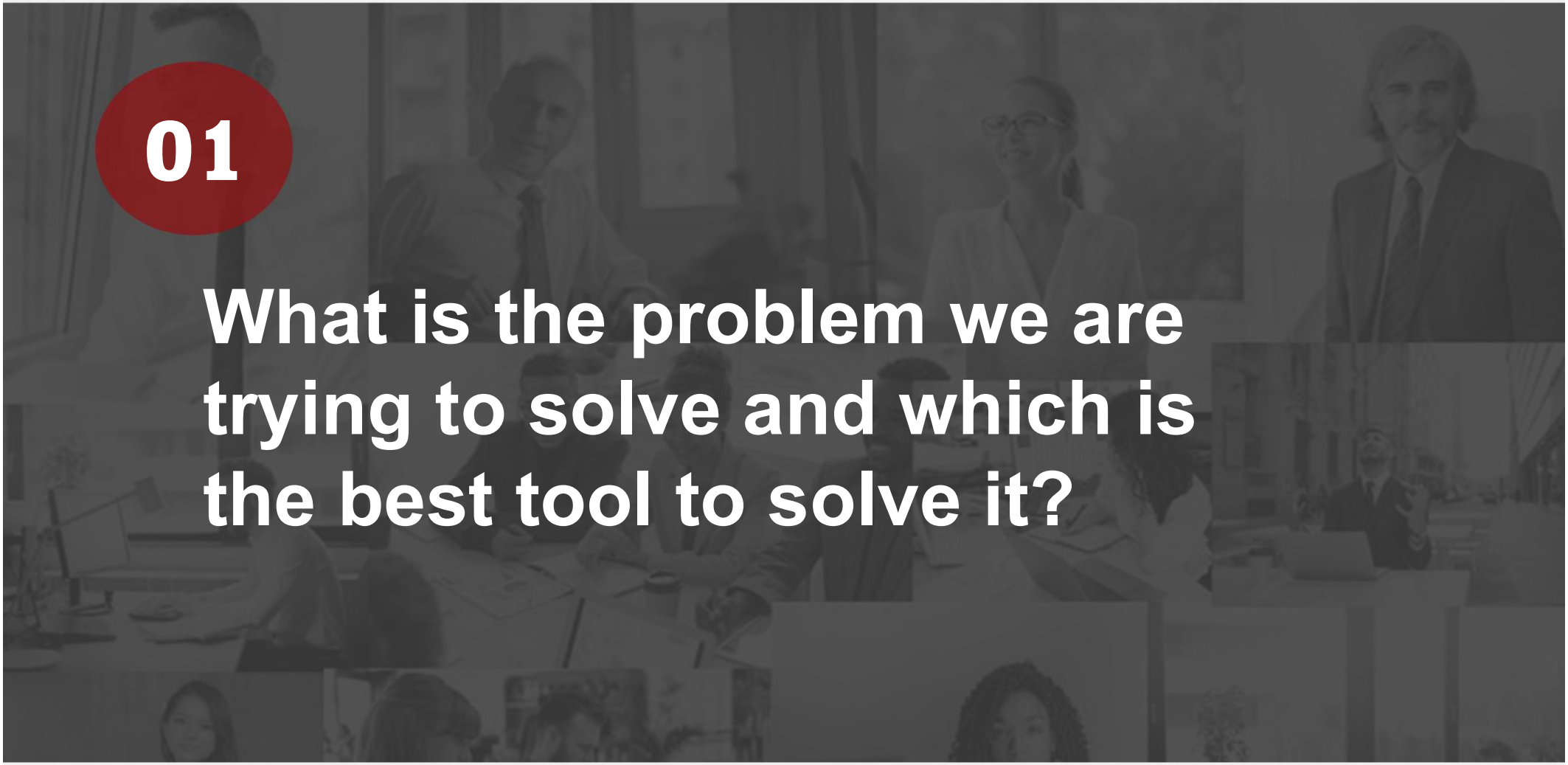
- Before running the program, ensure that the ner folder is in /sources
  - The ner folder was too heavy to put in the GitHub, therefore, you need to get it from here:
    - https://drive.google.com/file/d/1icy08Wut8BZMlulYvZT8yGOQ9CJMC9mA/view?usp=sharing
  - Download the ner.zip
  - Unzip it
  - Put the whole /ner folder in the /02_SOURCES folder
  - (ensure that you don´t change the folder name)



Trouble shooting – check it all runs!

- Now try running the program and you should see that the results file in the 03_RESULTS folder is updated:





01

What is the problem we are trying to solve and which is the best tool to solve it?

The problem we are trying to solve

- We want to quickly isolate all suppliers that sound like humans

	A	B	C	D	E
1	Supplier_number	Supplier_name	Country	Tax_code	Country_code
40786	1200	Finanzamt Frankfurt	Germany	DE657981456	DE
40787	1920	Versicherungen ABC	Germany	DE654871564	DE
40788	#12	Infritil Ltd	New Zealand		NZ
40789	1098	Productos Argentinos Imp. S.A	Argentina	GB963852741	AR
40790	1100	Phunix GmbH	Germany	DE123469789	DE
40791	#15	Z Energy Ltd	New Zealand		NZ
40792	0000068241	FINHARMONY	France	4.39067E+13	FR
40793	0000604797	CHARLOTTEHAVEN A/S	Denmark	DK27348408	DK
40794	0000088561	SEGULA ENGINEERING FRANCE	France		FR
40795	0006012744	SC AVANT BUSINESS & TRAVEL SOLUTION	Romania	27413971	RO
40796	0000603836	CIMI BEAUTY BAGS APS	Denmark	DK27167411	DK
40797	0001003539	XYLEM ANALYTICS UK LIMITED	United Kingdom	GB828197985	GB
40798	0006011932	BAUER & GINDELE SRL	Romania	RO36502385	RO
40799	#14	Pushpay	New Zealand		NZ
40800	0000071864	Capgemini Technology Services	France	479766842	FR
40801	0006030618	EUROLOG NETORK LOGISTICS SERVICES	Romania	RO29227576	RO
40802	1910	Schumacher AG	Germany	DE664565782	DE
40803	0006011824	PETRALUPS PROD SRL	Romania	RO6838694	RO
40804	0001002546	CGG DATA MANAGEMENT (UK) LTD	United Kingdom		GB
40805	#13	Kiwibank	New Zealand		NZ
40806	0000602769	KRÜGER AQUACARE	Denmark	DK57446412	DK
40807	0001005209	THE KEIL CENTRE	United Kingdom	GB890738975	GB
40808	100	C.E.B. BERLIN	Germany	DE456145834	DE
40809	111	KBB Schwarze Pumpe	Germany	DE159458525	DE
40810	200	SMP	United States	DE121145282	US

- With Python we can use a simple library (SpaCy) to help us categorize names.

Exists in Google?

Tax number is incorrect?

Sounds like a person's name/ generic name/ key word

Machine Learning!

How do you solve this problem today?

20260603_01_HowDoYouCheckYourSuppliers? Copy

1. **How do you currently check your suppliers during an audit?** (Single choice)

- We don't check our suppliers because that is second line of defense
- We check only the new ones that were created/ changed and see if Due Diligence check exists
- We get the list of suppliers and we read them to see if the names look normal... if something looks unusual, then we look it up in Google, etc. For example, if the supplier´s name sounds like a person´s name
- We get the top x suppliers in terms of value and we check those for Due Diligence controls as part of a walkthrough
- We use machine learning to compare Due Diligence documentation to supplier documentation
- We use python to run basic checks such as looking up the supplier tax code on government sites
- We use python to check if the supplier has a website
- We use python to check if the google image for the address looks like a residential address
- We get a list of newly created suppliers from the auditee (for example an Excel spreadsheet) and we compare the list to due diligence documentation
- We run artificial intelligence in order to identify suppliers that do not sound like real companies (for example they are people, shell companies), then we check if the values for those suppliers are high - and if they are high, then we check the reason

Why not just ask ChatGPT or an AI SAAS?

Privacy: Most large organizations don't allow you to upload sensitive data to an AI SAAS – so you might need to use your in house versions of them.

Accuracy: The AÍ SAAS systems don't always give you the same or accurate result

Ease of use: You might have to become expert at MARKDOWN in order to get the openAI to give you the output in the format that you want – for example if you want the result as a table

Traceability: If your auditors use the AI manually – will their results be repeatable in the future – because the AI will evolve and you won't know what the auditor used to communicate with the AI

Cost: If you are sending a lot of data to an AI SAAS – you will have to pay per token and the way that the cost is structured is quite hard to understand… you might end up with a surprising bill. You can also have your AI key stolen and someone can rack up huge expenses with it that you have to then try to reclaim.

Preparation: There may be some data preparation steps you want to do before you send your data to the AI SAAS – which is why it can be good to go with an API to a in-house AI platform. So that you can first prepare the data in python and then query the AI platform.

Why not just ask ChatGPT or an AI SAAS?

20260603_02_AnyOtherReasonsWhyNotUseChatGPTSAAS?

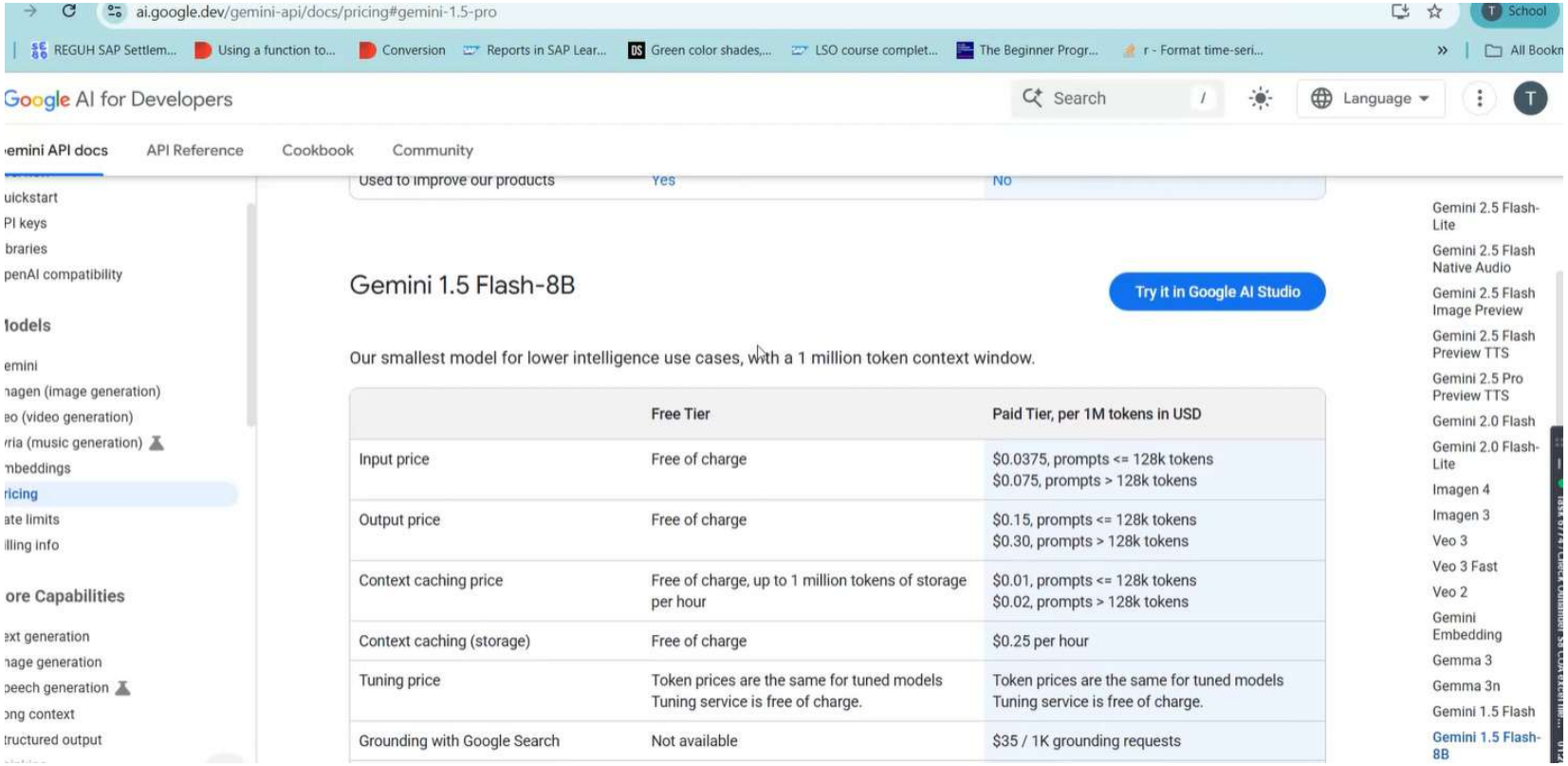
1. Any other reason why you would not use ChatGPT or SAAS for this kind of test? (Short answer)

Short answer (200 characters)

Why not use Gemini for everything?

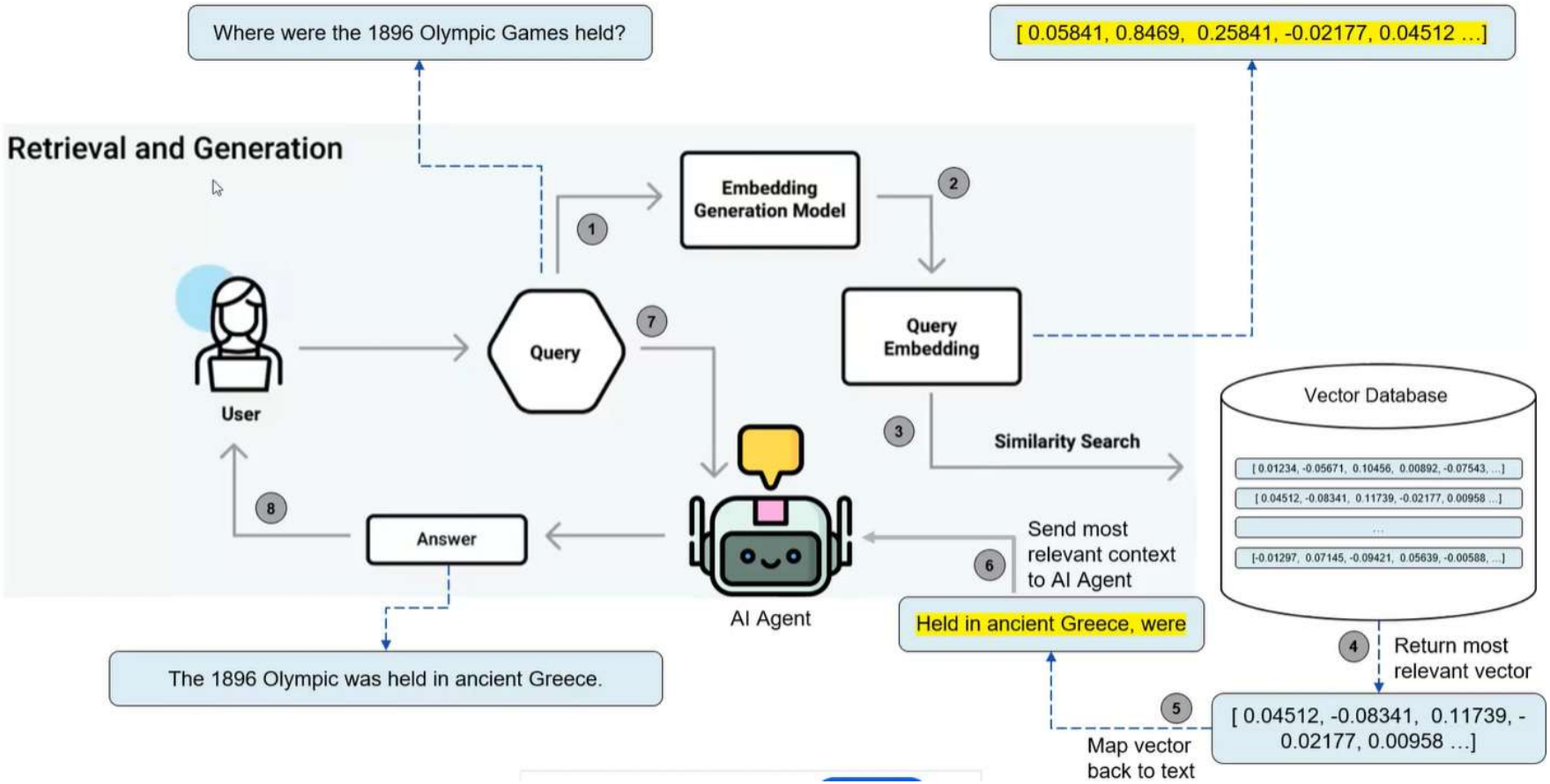
Also – some models require a huge amount of RAM to run.

So that might not be practical on a PC or laptop.



If data is too sensitive or voluminous, we can use RAG

We can use a RAG model in order to vectorize our data source first and then query based on vectors.



The tool that we use can depend on the situation, but it helps if python is pivotal

Structured data

Unstructured

Web

Python program: receives instructions and data sources, runs data analytics, Artificial Intelligence and generates results

Front-end

Upload files

Run risk analysis

View/ download report

KPI dashboard

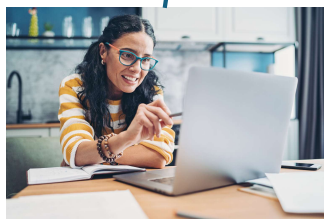
AWS Bedrock

openAI

GoogleNotebook

Python libraries scikitLearn/ SpaCy

Local GEMMA model



Auditor

You might want to think about using python as a central pivot – so that you can choose the AI model (local or SAAS) that you want, pre-prepare data, have traceability concerning instructions to AI sent through an API

Enables use of all python libraries: https://www.python.org/

Why did we choose python for 300Framework?

For 300Framework, we used to use (and we still do for some customers):

- ACL
- Qlik Sense
- PBI
- SQL
- Tableau

However, we decided to switch to python + JavaScript visualization interface for the following reasons:

Python is free

Python has endless libraries – with which you can do practically anything – all data analytics and all Artificial Intelligence

You can even use python streamlit library to make a web interface

Python is fast and the libraries are improving every day

Most Artificial Intelligence platforms are written in python and have python APIs

1. Core AI libraries are Python-first

Most of the major AI frameworks are written with Python as the main interface:

- TensorFlow
- PyTorch
- scikit-learn
- spaCy
- Hugging Face Transformers

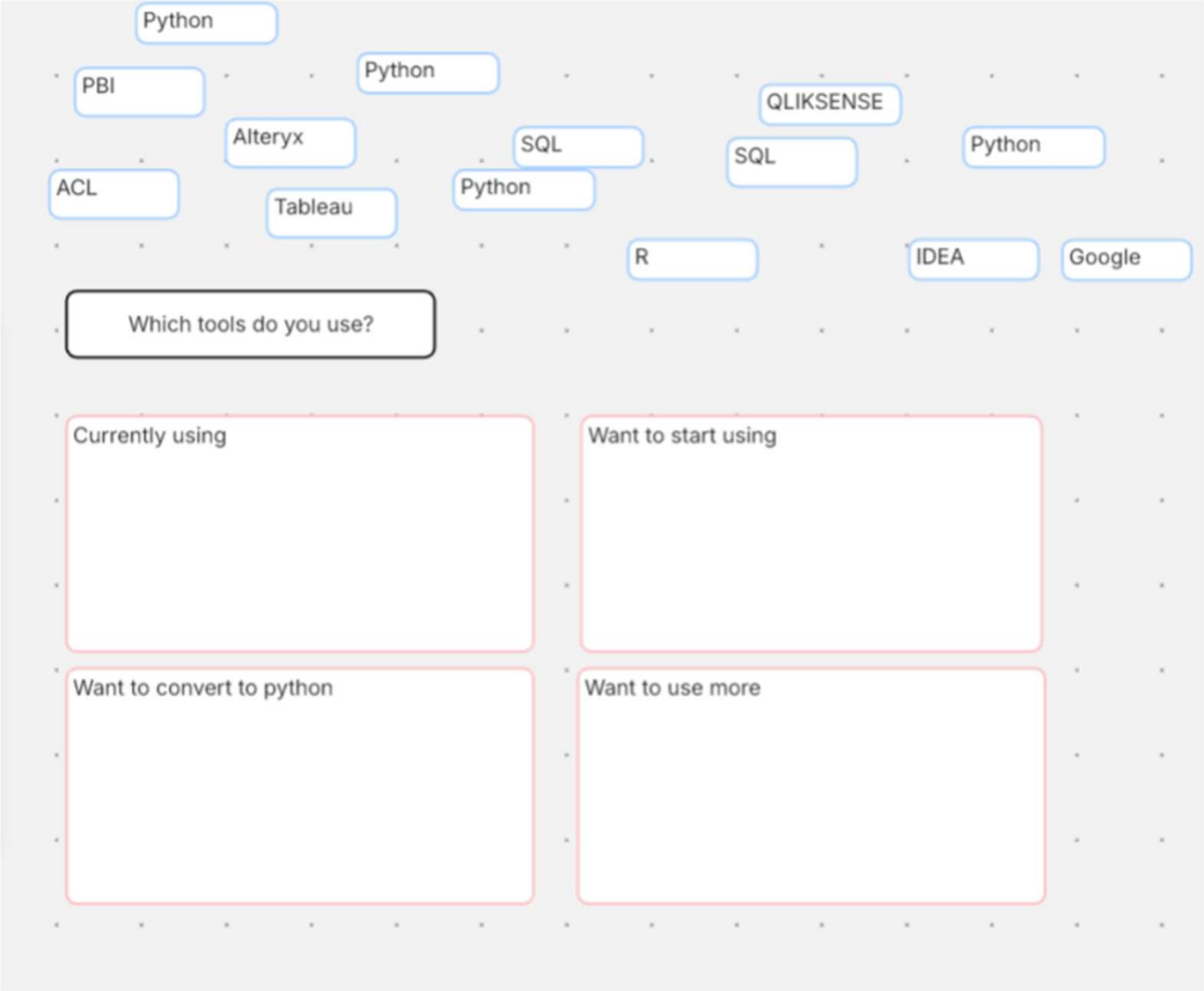
Even when the **core engine is written in C++**, Python is the **primary user interface.**

2. Python dominates the AI ecosystem

Typical AI stack:

Layer	Typical language
Model training	Python
Data preparation	Python
Experimentation	Python
Notebook analysis	Python
Production inference	Python / C++ / Java

Which analytics tools do you use?




02
SpaCy code overview

Overview of the program

1. Import suppliers, company words and train data

2. Clean the data

3. Create an NLP model

4. Train the NLP model if the option is set to True

Loop (for each name):

5. For each company name:

Here is the real Spacy AI

Apply the NLP model to it (create “doc” object)

A doc spacy object contains all the information about the Word

Check if it has ORG label, if not get first label

Add a column with the label

Cheating here so we can default to ORG if keyword found

Overwrite to ORG for company words

We could also add..If it didn’t get ORG but it was in the train_data set as ORG, then make it ORG

Export result

Before we start…

- We might want to do some other automated processing too.
- For example:
  - Ignore suppliers that are employees: based on the supplier account group category (LFA1_KTOKK→ T077Y_TEXT30)
  - Ignore suppliers that are intercompany
  - Ignore suppliers that are joint ventures
  - Filter the list of suppliers on only those that are in the scope of the audit: for example, only those that are found in LFB1 table, where LFB1 is filtered on the company code (LFB1_BUKRS = T001_BUTXT, where T001_BKTXT is the company you are auditing)
  - Ignore suppliers that are one-time vendors? (LFA1_XCPDK)
  - Ignore suppliers that are deleted? (LOEVM)
  - Or created a long time ago (LFA1_ERDAT)
  - … there might be other criteria that you choose to make the analysis quicker and have less false positives in the results

Is AI going to make data analysts more busy or less busy?

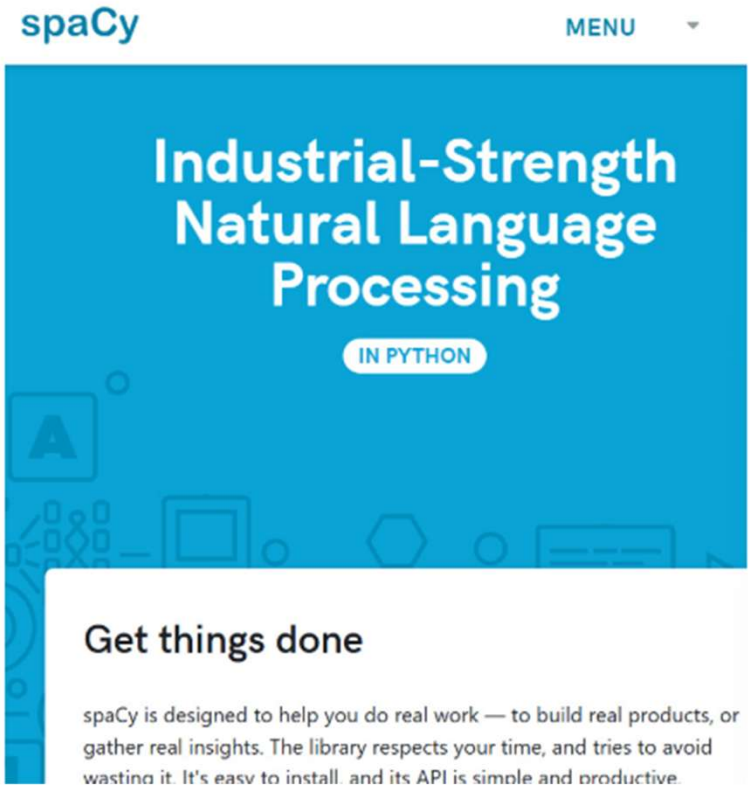
20260603_03_IsAI going to make us more/less busy

1. **Is AI going to make data analysts more or less busy?** (Single choice)

- ○ More busy because the data needs to be AI ready
- ○ Less busy because AI will do everything on its own

SpaCy for supplier categorization

- SpaCy is a python library.
- Uses Natural Language Processing



- Documentation about spaCy can be found here:
- https://spacy.io/
- It is a free python library that helps us to check lots of things using NLP (Natural Language Processing)

Use a pre-trained NER

- NER: Named Entity Recognition
- When can train the NER to make it more intelligent

- The NER is a basic _sm model that we have already pre-trained no 10K lines of data.

Create a spaCy NLP model

- We create an instance of the model

Code example: Create a spaCy nlp model

```
nlp = self.create_model()
```

- Create a spaCy nlp object by calling our function create_model(), that we mentioned above

Create a doc object using the model

- Here we create a doc object

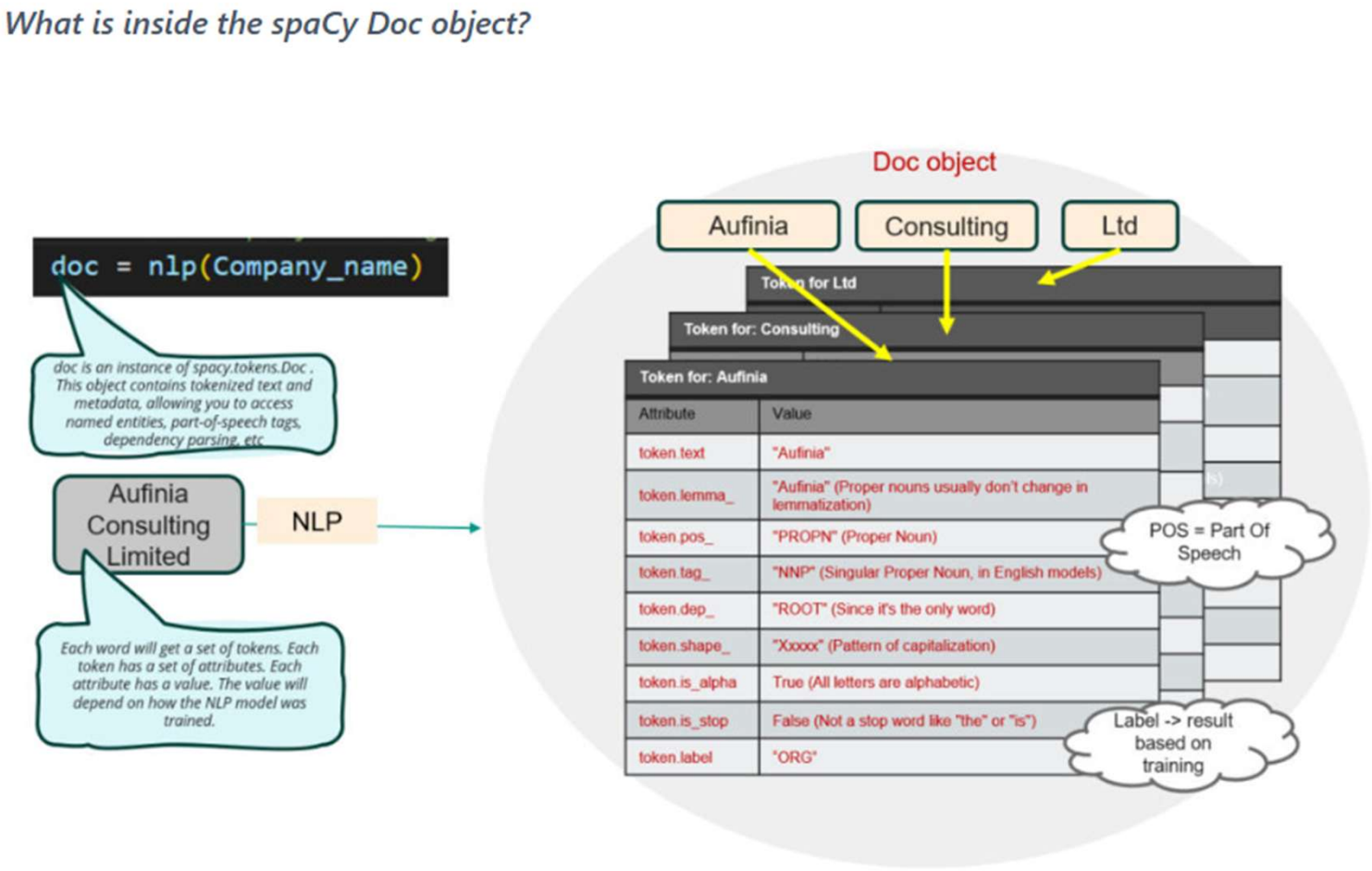
Code example: Create a spaCy Doc object

```
doc = nlp(Company_name)
```

- We pass each supplier name to our spaCy NLP model, to create a spaCy Doc object for that supplier name
- The Doc object has one ***entity*** per word
- Each ***entity*** has a set of ***tokens***. For example "POS" (Parts Of Speech), or "label"
- Each ***token*** has values, for example: "PROPN" (proper noun) or "ORG" (organization)

Deeper look into a doc object

- What do we find in the doc object?



Obtain the label from the doc object

Code example: Obtain label from Doc object

- We loop around the entities in the Doc object
- If we find an entity that has a ***label*** token with value "ORG", then we classify our supplier as an organization.
- If we find an entity that has a ***label*** token with value "PERSON", then we classify our supplier as a person.

```
for ent in doc.ents:
    if ent.label_ == 'ORG':
        label_EN = ent.label_

    else:
         for word in list_keyword:
             if word in supplier_name_lemma_list:
                 label_EN = 'ORG'
                 break

             else:
                label_EN = ent.label_
```

Tokens and lemmatization

- An object is created using the nlp
- The company name is “lemmatized” using this object

```
85          # tokenizes the text to produce a Doc object
86          doc = nlp(Company_name)
87
88          # lemmatize string => split string into an array => remove empty element from array
89          supplier_name_lemma = ' '.join([token.lemma_ for token in doc])
90          supplier_name_lemma_list = supplier_name_lemma.split()
91          supplier_name_lemma_list = [x for x in supplier_name_lemma_list if x]
```

A doc spacy object contains all the information about the Word

Lemmatization is the process of reducing a word to its base or root form, typically by removing inflections or variations. It aims to group together different forms of a word to analyze them as a single entity.

Lemmatization is crucial in NLP for tasks such as text analysis, sentiment analysis and information retrieval. It helps in standardizing words, reducing dimensionality and improving the accuracy of language processing models.


Questions?