TUTORIAL 9: ADVANCED USES OF LLMS & PYTHON PACKAGES

Creating Business Value with Generative AI
Fall 2025

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

31. October and 3. November 2025

Magnus Bender
Assistant Professor

TODAY'S TUTORIAL

TODAY'S TUTORIAL

1. Use `pip`
   - Get an overview of installed packages
   - Install more packages
   - Get a list of packages to install the same selection elsewhere

TODAY'S TUTORIAL

1. Use `pip`
   - Get an overview of installed packages
   - Install more packages
   - Get a list of packages to install the same selection elsewhere
2. Implement a RAG-pipeline on your own
   - There is a partly implemented pipeline in a notebook
   - There is a GUI using (importing) your pipeline from the notebook
   - Code examples for text chunking and text extraction from PDFs

TODAY'S TUTORIAL

1. Use `pip`
   - Get an overview of installed packages
   - Install more packages
   - Get a list of packages to install the same selection elsewhere
2. Implement a RAG-pipeline on your own
   - There is a partly implemented pipeline in a notebook
   - There is a GUI using (importing) your pipeline from the notebook
   - Code examples for text chunking and text extraction from PDFs
3. Use RAG for your project (*optional*)

PYTHON PACKAGES

- Why we need them
- How to manage them
- Practice in uCloud

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

31. October and 3. November 2025

Magnus Bender
Assistant Professor

IMPORT CUSTOM MODULES

example_import.py

```python
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

IMPORT CUSTOM MODULES

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

- Create two files: `example_import.py` and `name.py`

IMPORT CUSTOM MODULES

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`

IMPORT CUSTOM MODULES

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

```
<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
                    <function hello at 0x100be6200>
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`
  - `example_import` becomes a module

IMPORT CUSTOM MODULES

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

```
<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
                  <function hello at 0x100be6200>
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`
  - `example_import` becomes a module
  - Variables, classes, and functions in `example_import` are accessible via `example_import.<variable_name>`, `example_import.<class name>`, and `example_import.<function_name>`

IMPORT CUSTOM MODULES

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

```
<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
                  <function hello at 0x100be6200>

False
Bye!
Hello World!
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`
  - `example_import` becomes a module
  - Variables, classes, and functions in `example_import` are accessible via `example_import.<variable_name>`, `example_import.<class name>`, and `example_import.<function_name>`

MORE IMPORT STATEMENTS

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

MORE IMPORT STATEMENTS

Import a name from a module

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

MORE IMPORT STATEMENTS

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

The same function, but two ways to access it.

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

MORE IMPORT STATEMENTS

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

Import two modules with one import statement

MORE IMPORT STATEMENTS

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

Import all variables, classes, and fuctions from a module

MORE IMPORT STATEMENTS

example_import.py

```
def hello():
  print("Hello World!")

def bye():
  print("Bye!")

VARIABLE = False
```

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

Import with a different name, use `ei.<name>`

HOW TO GET PACKAGES & MODULES

HOW TO GET PACKAGES & MODULES

- Modules may be part of Python (standard library), e.g., `json, csv, time`
  - No need to install, are already available

HOW TO GET PACKAGES & MODULES

- Modules may be part of Python (standard library), e.g., `json, csv, time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic, nicegui, openai`

HOW TO GET PACKAGES & MODULES

- Modules may be part of Python (standard library), e.g., `json, csv, time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic, nicegui, openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (https://pypi.org/)

HOW TO GET PACKAGES & MODULES

- Modules may be part of Python (standard library), e.g., `json, csv, time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic, nicegui, openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (https://pypi.org/)
    - Possible to search there for relevant packages
    - Possible to ask ChatGPT
    - We introduced you to some useful packages

HOW TO GET PACKAGES & MODULES

- Modules may be part of Python (standard library), e.g., `json, csv, time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic, nicegui, openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (https://pypi.org/)
    - Possible to search there for relevant packages
    - Possible to ask ChatGPT
    - We introduced you to some useful packages
- Modules may be self-created, custom modules
  - Easy way: Place multiple Python files in same directory and use `import <file name>`

PIP: PYTHON PACKAGE INSTALLER

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:
  - Each of you has your own Python environment

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via `pip`

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via `pip`
    - See the `requirements.txt` →

`requirements.txt`

```
requests
tqdm
pdoc

pydantic
nicegui

nltk
numpy
scipy
scikit-learn

openai
ipykernel

pdfminer.six
```

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via `pip`
    - See the `requirements.txt` →
  - You may install more packages if your need them

`requirements.txt`

```
requests
tqdm
pdoc

pydantic
nicegui

nltk
numpy
scipy
scikit-learn

openai
ipykernel

pdfminer.six
```

PIP: PYTHON PACKAGE INSTALLER

- Each of us already uses `pip` on uCloud
  - But we did not tell you about (only select „initialization `setup.sh`“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via `pip`
    - See the `requirements.txt` →
  - You may install more packages if your need them
    - `pip list`
    - `pip install <package name>`

`requirements.txt`

```
requests
tqdm
pdoc

pydantic
nicegui

nltk
numpy
scipy
scikit-learn

openai
ipykernel

pdfminer.six
```

LET'S USE PIP

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:
  - Actually, `pip` is a program available in the terminal/ shell

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:
  - Actually, `pip` is a program available in the terminal/ shell
  - A line starting with `!` in notebook tells it to run this line as shell command

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:
  - Actually, `pip` is a program available in the terminal/ shell
  - A line starting with `!` in notebook tells it to run this line as shell command
- Generally, about packages:
  - You may (have to) install packages for your project

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:
  - Actually, `pip` is a program available in the terminal/ shell
  - A line starting with `!` in notebook tells it to run this line as shell command
- Generally, about packages:
  - You may (have to) install packages for your project
  - But do not directly install any packages without taking a *deeper* look at it

LET'S USE PIP

- There is a notebook in `Tutorials/Tutorial_9/packages.ipynb`
  - It provides a step-by-step guide through the usage of `pip`
- Some remarks before running the notebook:
  - Actually, `pip` is a program available in the terminal/ shell
  - A line starting with `!` in notebook tells it to run this line as shell command
- Generally, about packages:
  - You may (have to) install packages for your project
  - But do not directly install any packages without taking a *deeper* look at it
    - Anyone can publish packages, they may be buggy, not well documented, ...
    - E.g.: Search the web for the name or ask ChatGPT about it

IMPLEMENTATION OF A RAG PIPELINE

- Step by step in a notebook
- Use the notebook with NiceGUI
- RAG pipelines in practice

AARHUS BSS
DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY

31. October and 3. November 2025

Magnus Bender
Assistant Professor

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories
- Some remarks before running the notebook:

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories
- Some remarks before running the notebook:
  - All the function definitions are already present
  - Some functions contain more code, some less → look for `TODO`

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories
- Some remarks before running the notebook:
  - All the function definitions are already present
  - Some functions contain more code, some less → look for `TODO`
  - Nearly each cell has a validation to test your implementation
    `if __name__ == "__main__": …`

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories
- Some remarks before running the notebook:
  - All the function definitions are already present
  - Some functions contain more code, some less → look for `TODO`
  - Nearly each cell has a validation to test your implementation
    `if __name__ == "__main__": …`
- After you implemented all `TODO`s in the notebook and tested in the notebook, you may run the `gui.py`
  - It contains a simple NiceGUI application running your RAG-pipline from `rag.ipynb`:
    `from rag import category_list, answer_with_rag`

NOTEBOOKS AND GUIS: CREATE A RAG PIPELINE WITH A GUI

- There a notebook in `Tutorials/Tutorial_9/rag.ipynb`
  - It provides a step-by-step guide for creating a small RAG pipeline on news articles
  - The news are provided in the `news{_small}.csv` and consist of text and categories
- Some remarks before running the notebook:
  - All the function definitions are already present
  - Some functions contain more code, some less → look for `TODO`
  - Nearly each cell has a validation to test your implementation
    `if __name__ == "__main__": …`
- After you implemented all `TODO`s in the notebook and tested in the notebook, you may run the `gui.py`
  - It contains a simple NiceGUI application running your RAG-pipline from `rag.ipynb`:
    `from rag import category_list, answer_with_rag`
- I will upload a `rag_solution.ipynb` later

SOME INSPIRATION

- Not in a specific order, but may be helpful ...

SOME INSPIRATION

- Not in a specific order, but may be helpful ...

```
json.loads(row['key'])

messages.append({"role": "","content": ""})

response = client.chat.completions.create(
  model="gpt-5-nano",
  messages=[]
)
response.choices[0].message.content


if categories is None or row['key'] in categories:
```

```
rows.append(row)

categories = set()

with open(file_name, 'r') as f:
  reader = DictReader(f)

client.embeddings.create(
  model="..."
)

if i > top_k:
  break
```

EXTRACT TEXT FROM PDFS

EXTRACT TEXT FROM PDFS

- Extracts the text from a PDF file
  - The text needs to be stored as text in the PDF, so no images, scans, etc.

EXTRACT TEXT FROM PDFS

- Extracts the text from a PDF file
  - The text needs to be stored as text in the PDF, so no images, scans, etc.

```
from pdfminer.high_level import extract_text

plain_text = extract_text("my-file.pdf")
```

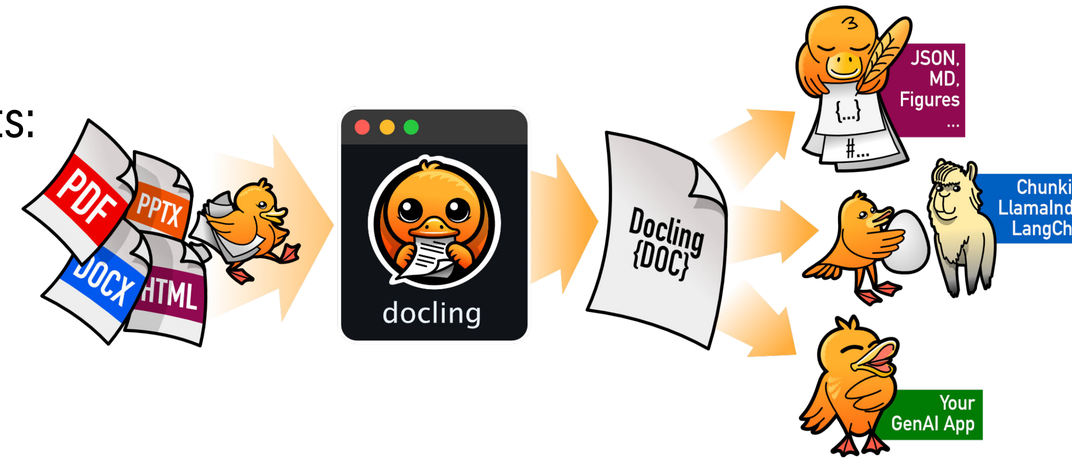
EXTRACT TEXT FROM PDFS

- Extracts the text from a PDF file
  - The text needs to be stored as text in the PDF, so no images, scans, etc.

```
from pdfminer.high_level import extract_text

plain_text = extract_text("my-file.pdf")
```

- In case of more complex text extractions or other file formats:
  - Take a look at `docling`



https://github.com/docling-project/docling

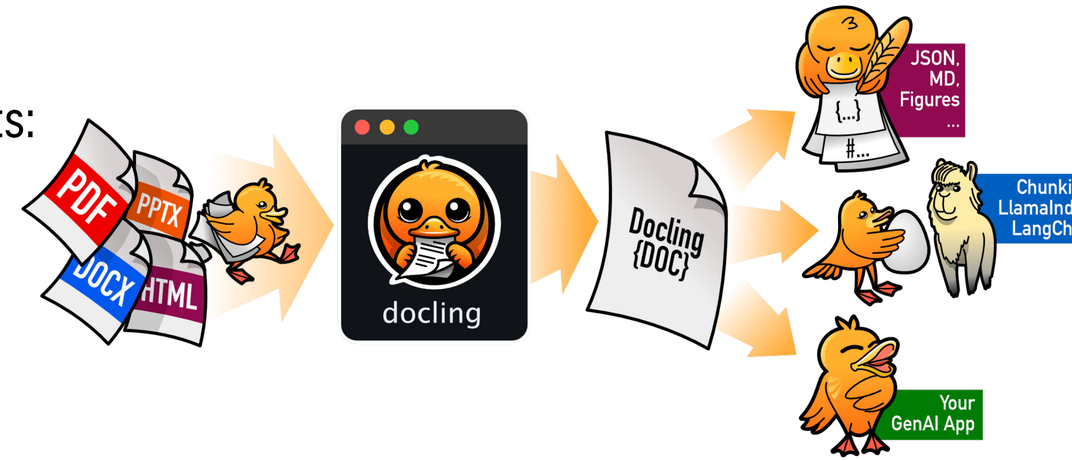
EXTRACT TEXT FROM PDFS

- Extracts the text from a PDF file
  - The text needs to be stored as text in the PDF, so no images, scans, etc.

```
from pdfminer.high_level import extract_text

plain_text = extract_text("my-file.pdf")
```

- In case of more complex text extractions or other file formats:
  - Take a look at `docling`
  - It is slightly more complex, but much more powerful
  - It may require more resources



https://github.com/docling-project/docling

CHUNK TEXTS

CHUNK TEXTS

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string
  - Set a chunk size

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string
  - Set a chunk size
  - Create a list for the chunks

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string
  - Set a chunk size
  - Create a list for the chunks
  - Iterate from 0 to length of the full text with the step size of chunk size

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

```
0 150
150 300
300 450
450 600
600 750
750 900
900 1050
1050 1200
1200 1350
1350 1500
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string
  - Set a chunk size
  - Create a list for the chunks
  - Iterate from 0 to length of the full text with the step size of chunk size
  - Use string slicing (same as with lists) to get each chunk

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

```
0 150
150 300
300 450
450 600
600 750
750 900
900 1050
1050 1200
1200 1350
1350 1500
```

CHUNK TEXTS

- Simple Python code to create chunks of a long string
  - Set a chunk size
  - Create a list for the chunks
  - Iterate from 0 to length of the full text with the step size of chunk size
  - Use string slicing (same as with lists) to get each chunk
    - First chunk is substring from character 0 to 149
    - Next 150 to 299
    - ...

```
full_text = "...."
chunk_size = 150

chunks = []
for i in range(0, len(full_text), chunk_size):
  print(i, i+chunk_size)
  chunks.append(full_text[i:i+chunk_size])
```

```
0 150
150 300
300 450
450 600
600 750
750 900
900 1050
1050 1200
1200 1350
1350 1500
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
  api_key="OPEN_AI_KEY ...",
  model_name="text-embedding-3-small"
)
collection = chroma_client.get_or_create_collection("my-data", embedding_function=openai_ef)
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```python
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
  api_key="OPEN_AI_KEY ...",
  model_name="text-embedding-3-small"
)
collection = chroma_client.get_or_create_collection("my-data", embedding_function=openai_ef)

if collection.count() == 0:
  collection.add(ids=["id1", "id2"], documents=["document 1", "document 2"])
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```python
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
  api_key="OPEN_AI_KEY ...",
  model_name="text-embedding-3-small"
)
collection = chroma_client.get_or_create_collection("my-data", embedding_function=openai_ef)

if collection.count() == 0:
  collection.add(ids=["id1", "id2"], documents=["document 1", "document 2"])

print(collection.query(
  query_texts=["This is a query document about hawaii"],
  n_results=1
))
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

```python
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
  api_key="OPEN_AI_KEY ...",
  model_name="text-embedding-3-small"
)
collection = chroma_client.get_or_create_collection("my-data", embedding_function=openai_ef)

if collection.count() == 0:
  collection.add(ids=["id1", "id2"], documents=["document 1", "document 2"])

print(collection.query(
  query_texts=["This is a query document about hawaii"],
  n_results=1
))
```

```
{'ids': [['id1']], 'embeddings': None, 'documents': [['This is a
document about pineapple']], 'uris': None, 'included': ['metadatas',
'documents', 'distances'], 'data': None, 'metadatas': [[None]],
'distances': [[0.9091699719429016]]}
```

EMBEDDINGS AND RETRIEVAL IN PRACTICE: CHROMA DB

- Will take care of storing the documents and embeddings on disk
- Will call the OpenAI API and retrieve the most similar documents based in embeddings
- Source code available on uCloud!

```python
from pathlib import Path
import chromadb # first: pip install chromadb
import chromadb.utils.embedding_functions as embedding_functions

chroma_client = chromadb.PersistentClient(path=Path(__file__).parent/"chroma")
openai_ef = embedding_functions.OpenAIEmbeddingFunction(
  api_key="OPEN_AI_KEY ...",
  model_name="text-embedding-3-small"
)
collection = chroma_client.get_or_create_collection("my-data", embedding_function=openai_ef)

if collection.count() == 0:
  collection.add(ids=["id1", "id2"], documents=["document 1", "document 2"])

print(collection.query(
  query_texts=["This is a query document about hawaii"],
  n_results=1
))
```

```
{'ids': [['id1']], 'embeddings': None, 'documents': [['This is a
document about pineapple']], 'uris': None, 'included': ['metadatas',
'documents', 'distances'], 'data': None, 'metadatas': [[None]],
'distances': [[0.9091699719429016]]}
```

SUMMARY FOR TODAY

SUMMARY FOR TODAY

- RAG and Python packages
- You now have all (or mostly all) the technical prerequisites for your projects

SUMMARY FOR TODAY

- RAG and Python packages
- You now have all (or mostly all) the technical prerequisites for your projects

- If still time:
  - Work on your projects
  - Try to apply RAG, if suitable

SUMMARY FOR TODAY

- RAG and Python packages
- You now have all (or mostly all) the technical prerequisites for your projects

- If still time:
  - Work on your projects
  - Try to apply RAG, if suitable

- Feedback on proposals

DEPARTMENT OF MANAGEMENT
AARHUS UNIVERSITY