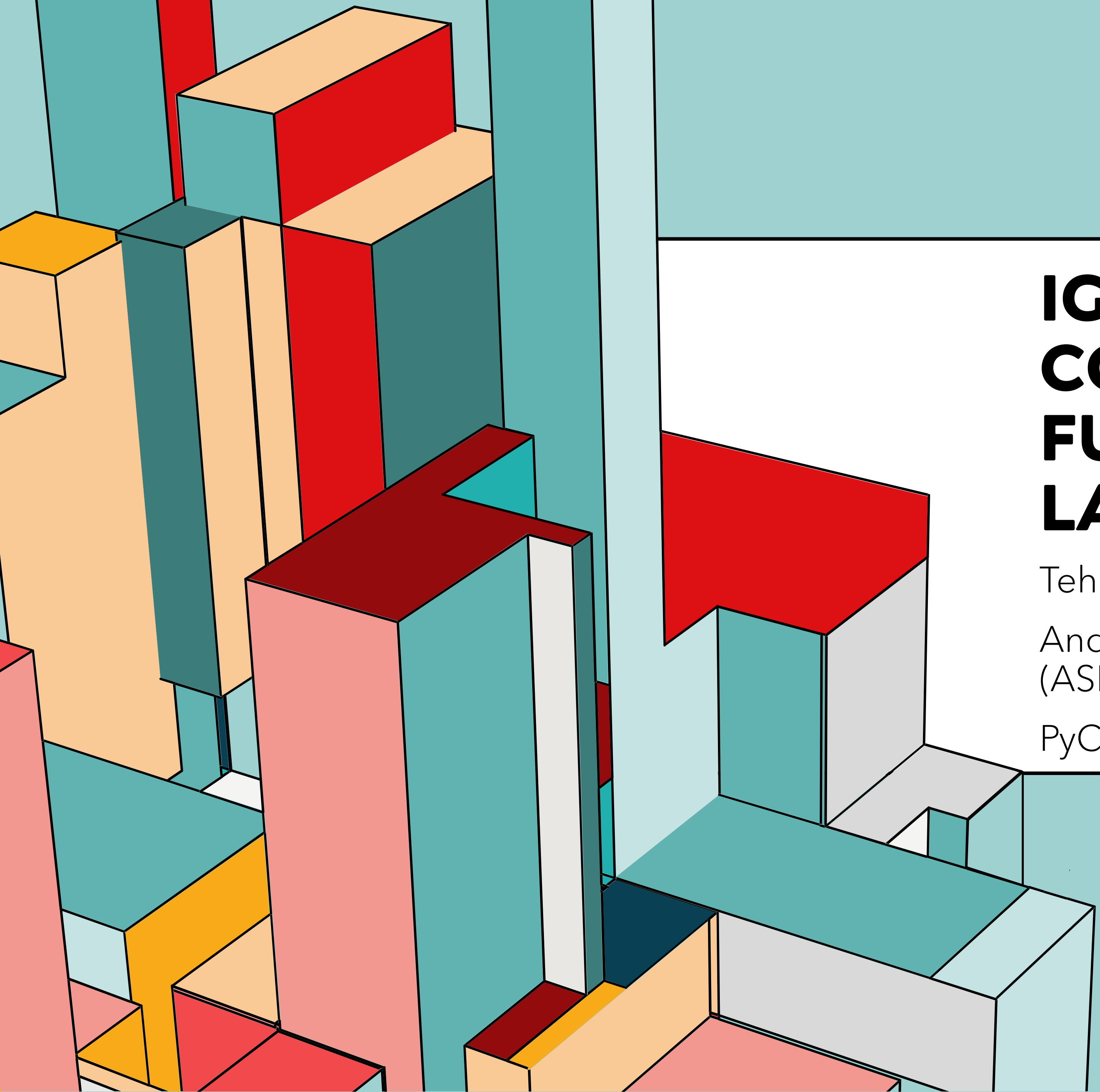


IGNITING CONVERSATIONS, FUSING FILES: LANGCHAIN

Teh Kim Wee

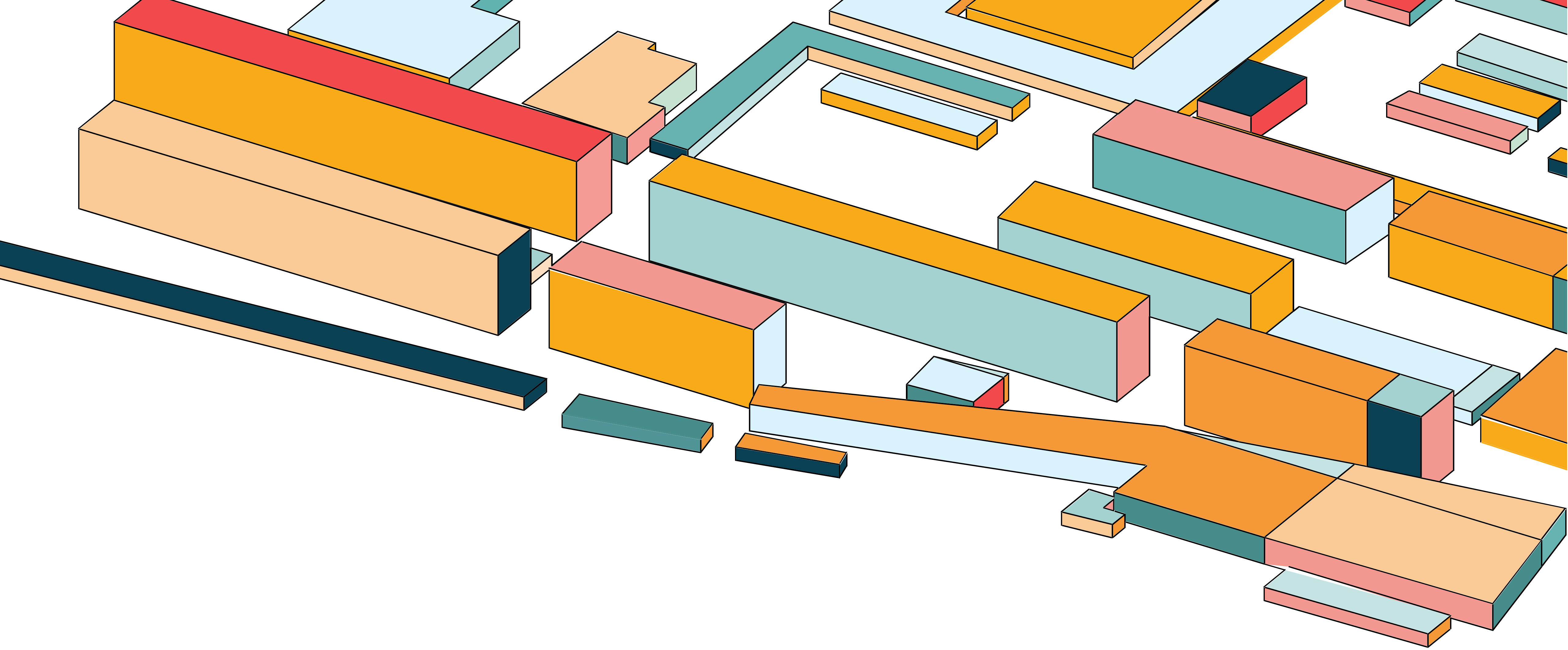
Anderson Serangoon Junior College (ASRJC)

PyCon SG Education Summit 2023

LANGCHAIN

🦜🔗 LangChain

- Open-Source Framework
- Gives CHATGPT the ability to read your files!
- Chain multiple tools to be used with GPT

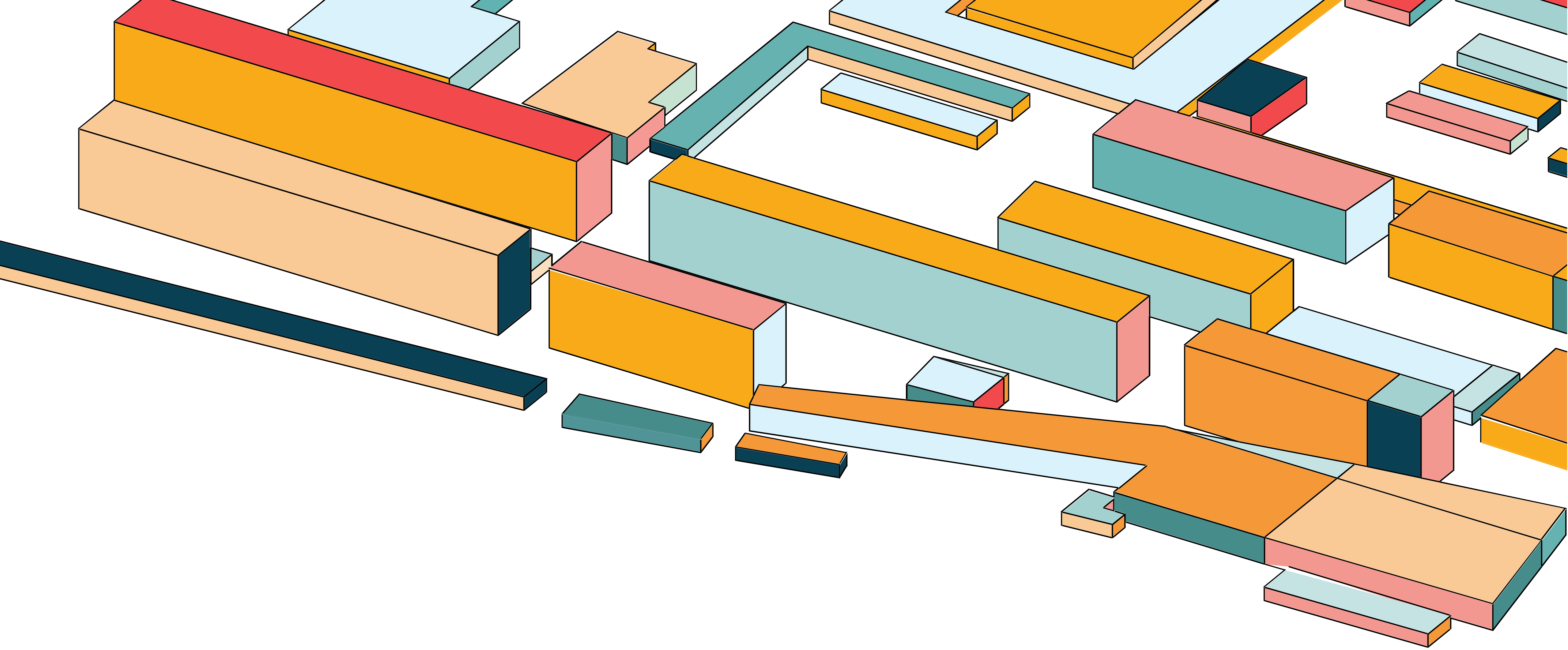


ALL YOU EVER NEED...

GOOGLE COLAB

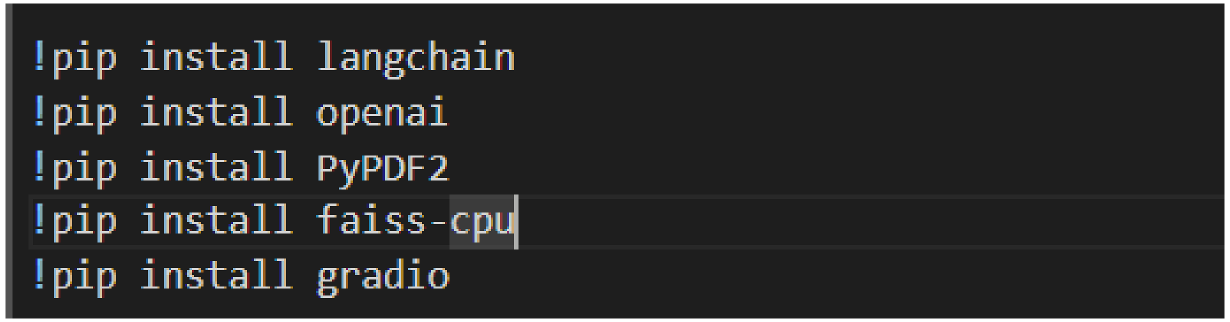
- File → Make a Copy

Google Colaboratory



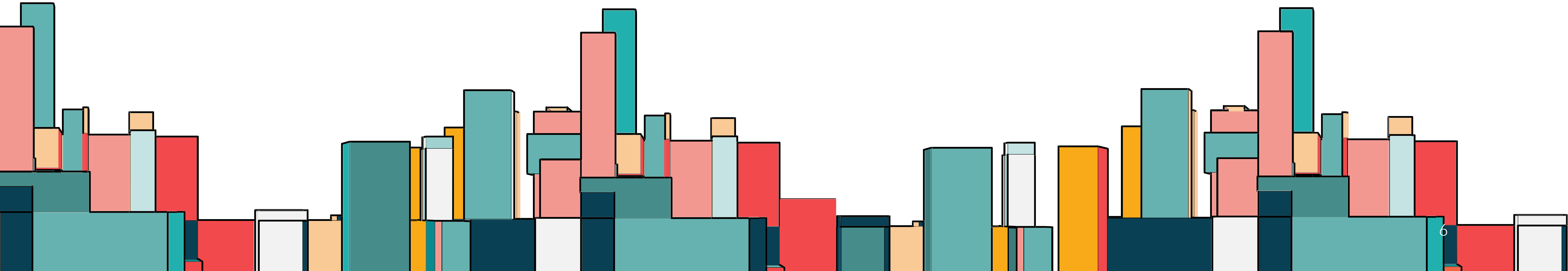
SETTING-UP

DEPENDENCIES

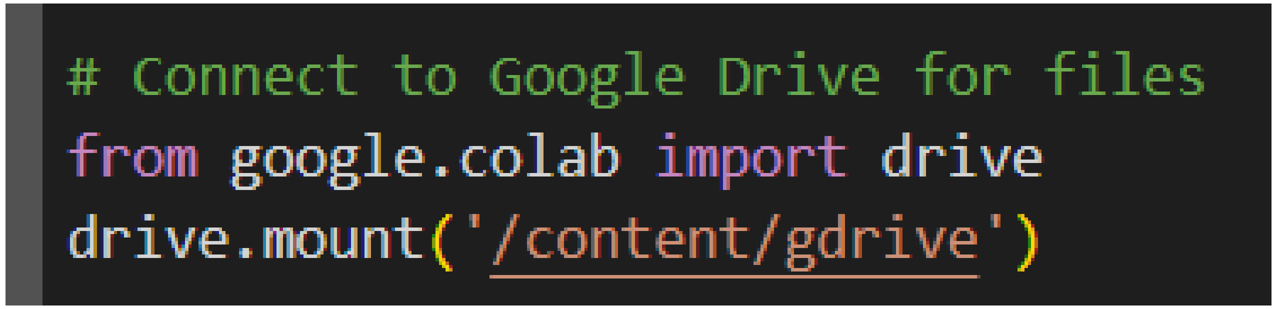


```
!pip install langchain
!pip install openai
!pip install PyPDF2
!pip install faiss-cpu
!pip install gradio
```

- LangChain Framework
- OpenAI (to use the API)
- PYPDF2: Read PDFs
- FAISS-CPU: Vector Search
- Gradio: Spin up a simple web interface

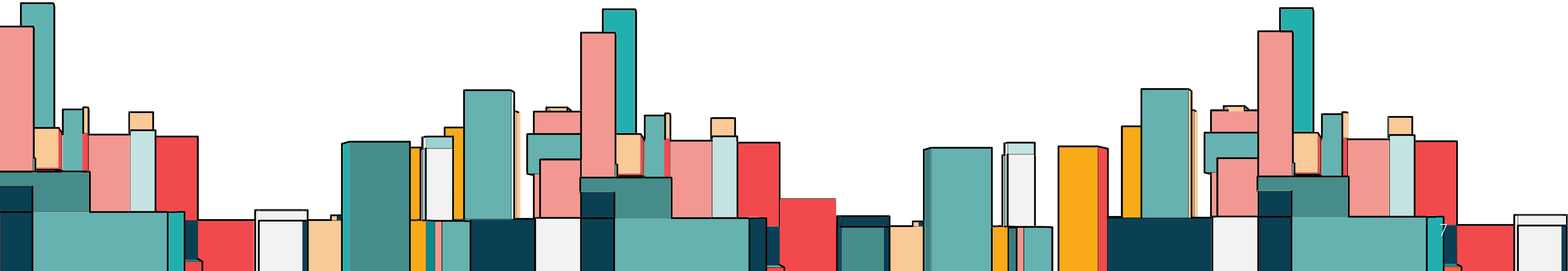


USING GOOGLE DRIVE



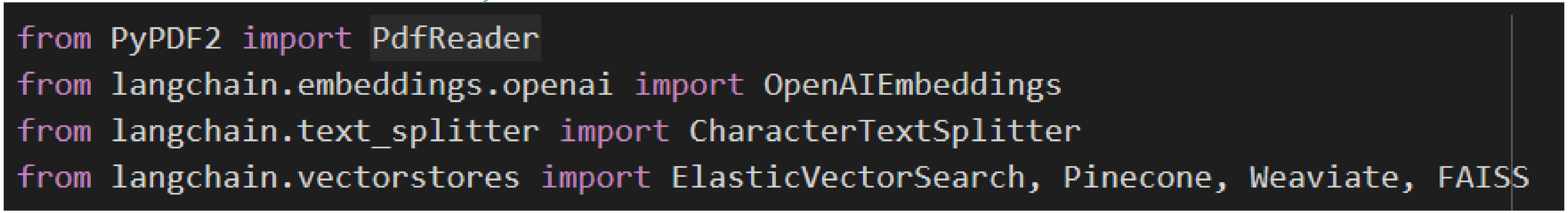
```
# Connect to Google Drive for files
from google.colab import drive
drive.mount('/content/gdrive')
```

MOUNT GOOGLE DRIVE TO ACCESS FILES THERE

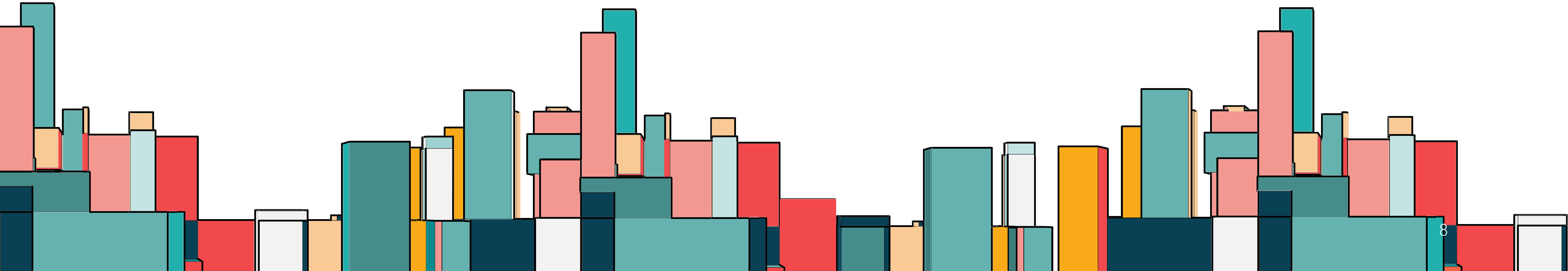


IMPORTING IMPORTANT STUFF

- Read PDFs

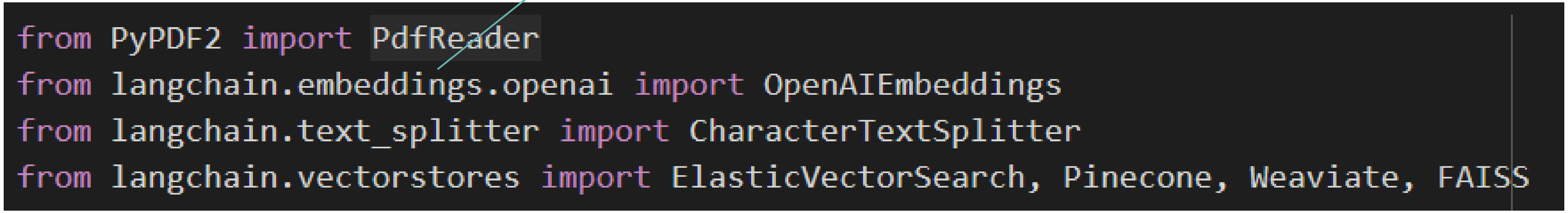


```python
from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import ElasticVectorSearch, Pinecone, Weaviate, FAISS
```

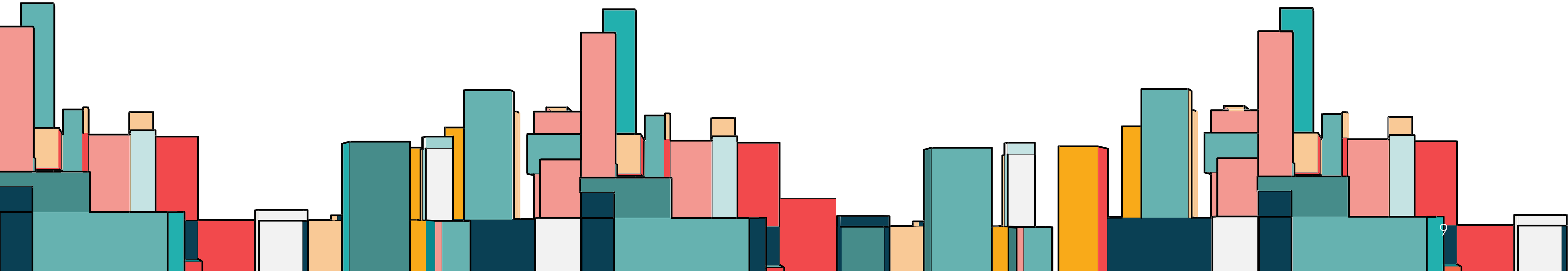


IMPORTING IMPORTANT STUFF

- Get OpenAI Embeddings

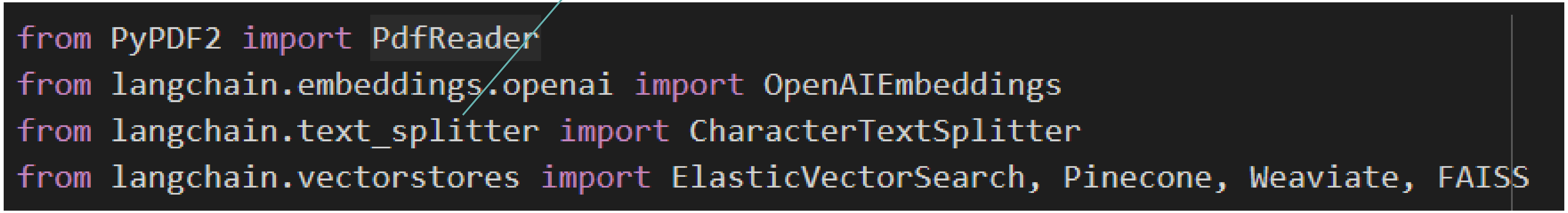


```python
from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import ElasticVectorSearch, Pinecone, Weaviate, FAISS
```

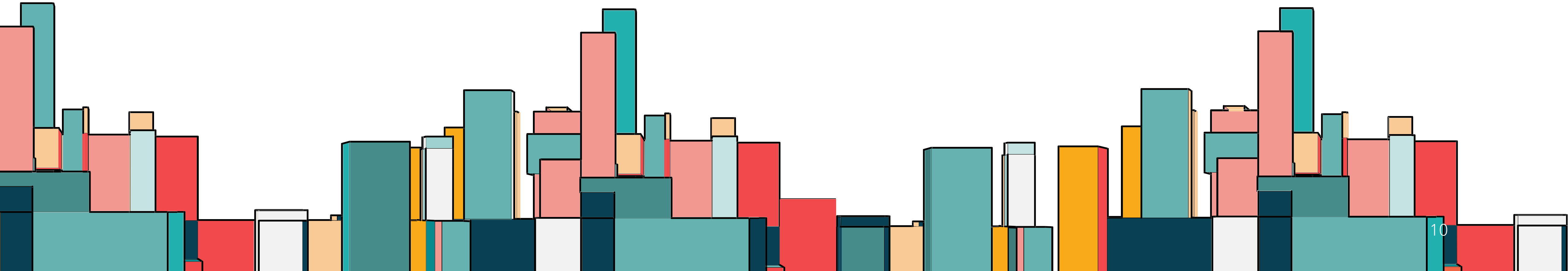


IMPORTING IMPORTANT STUFF

- Split the text to meet Token Requirements

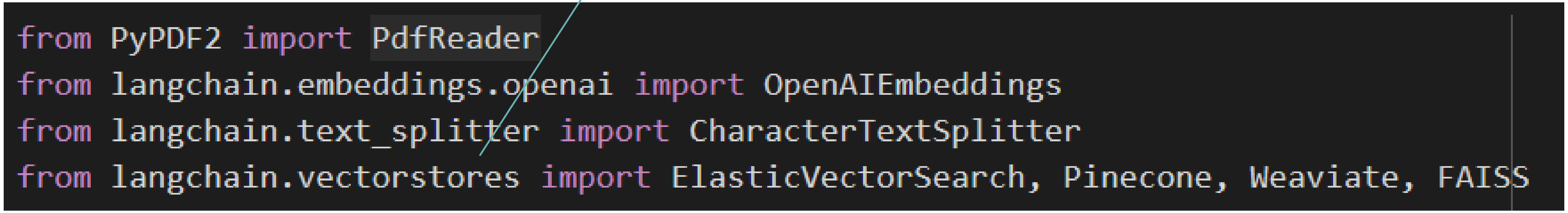


```
from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import ElasticVectorSearch, Pinecone, Weaviate, FAISS
```

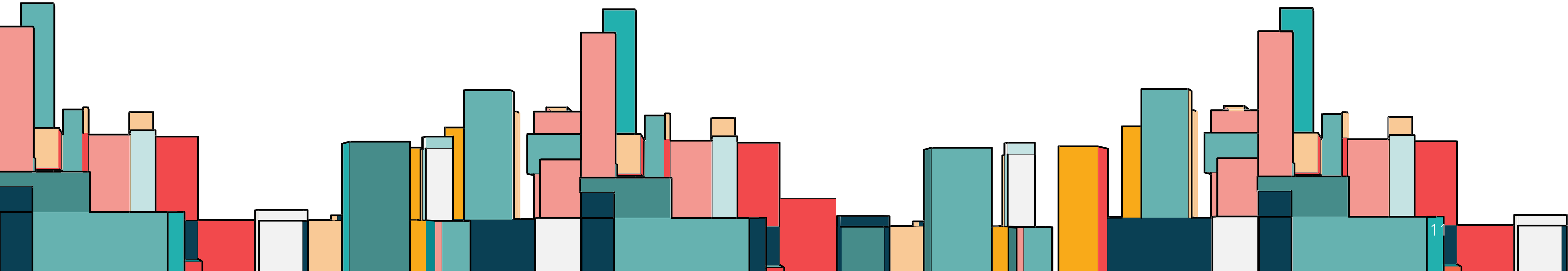


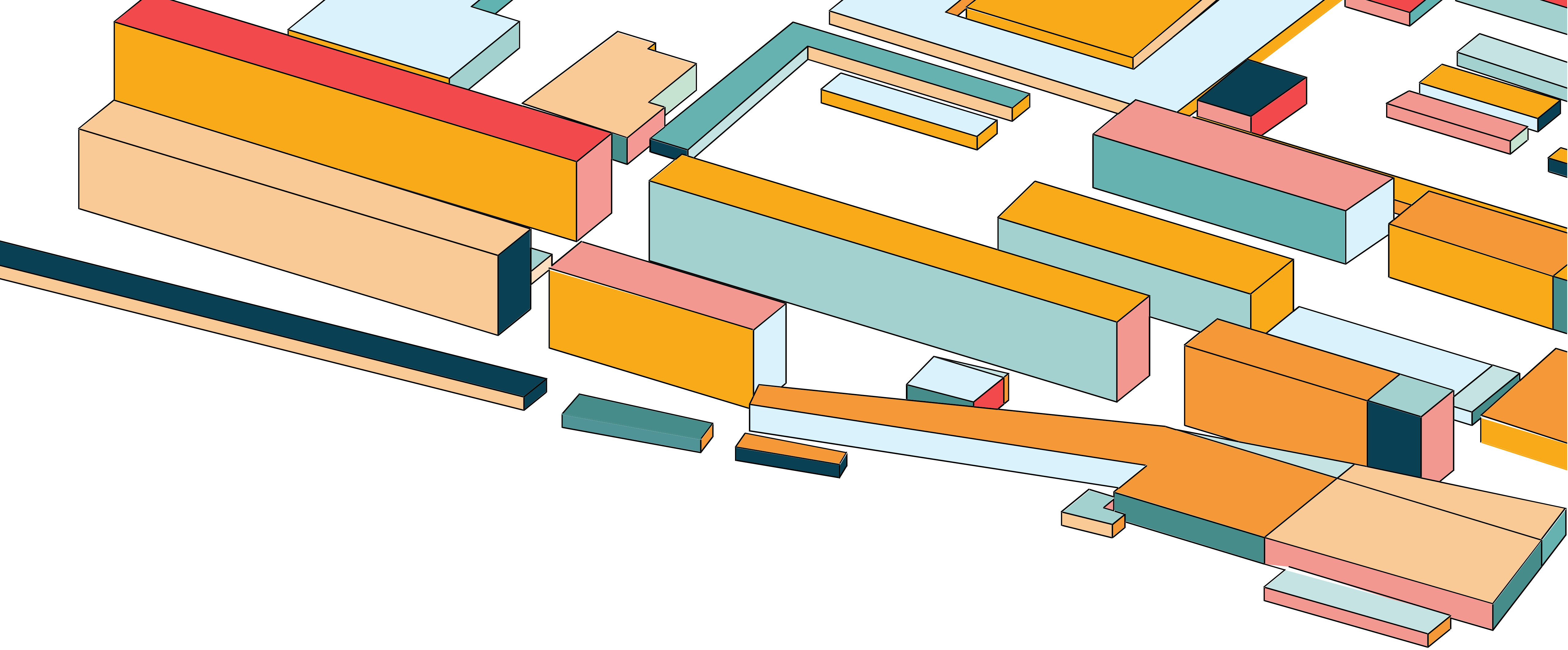
IMPORTING IMPORTANT STUFF

- Database to perform “Search”



```
from PyPDF2 import PdfReader
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import ElasticVectorSearch, Pinecone, Weaviate, FAISS
```



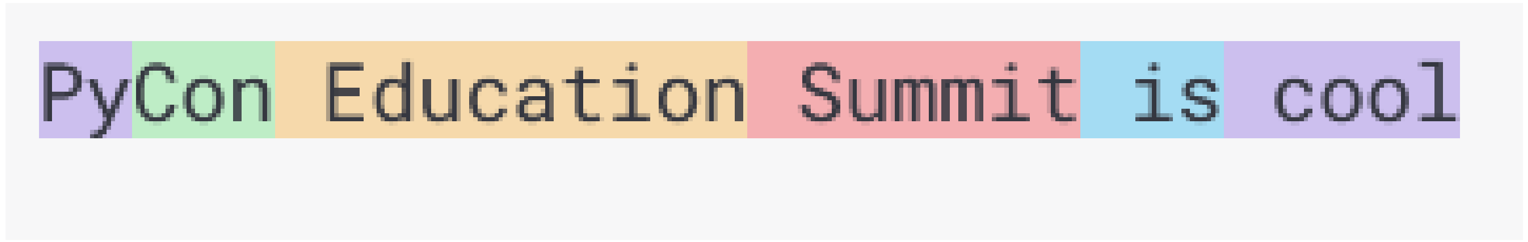


READING THE PDF FOR GPT

TOKENS

- LLM views "words" differently from us

- https://platform.openai.com/tokenizer

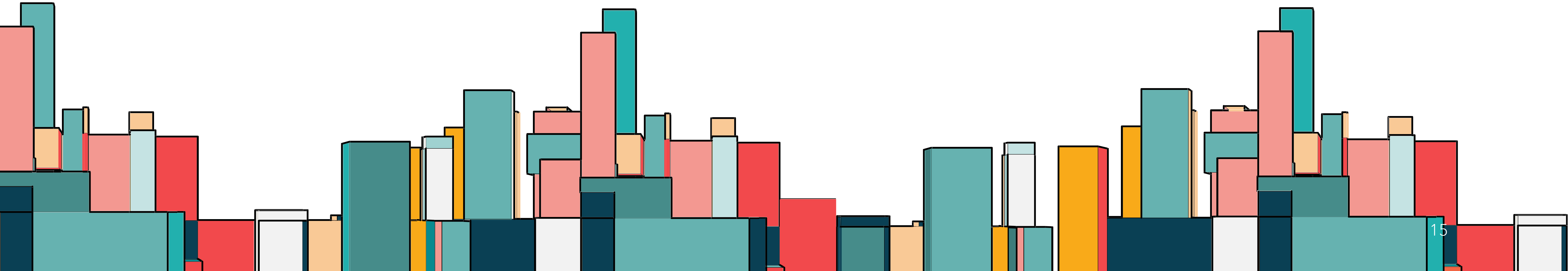

PyCon Education Summit is cool

LIMITATIONS OF GPT

- GPT-3.5-Turbo has limit of 4096 tokens ☹
- LangChain helps to settle this by breaking down the text into smaller chunk
- GPT-3.5-16K now supports 16K Tokens!

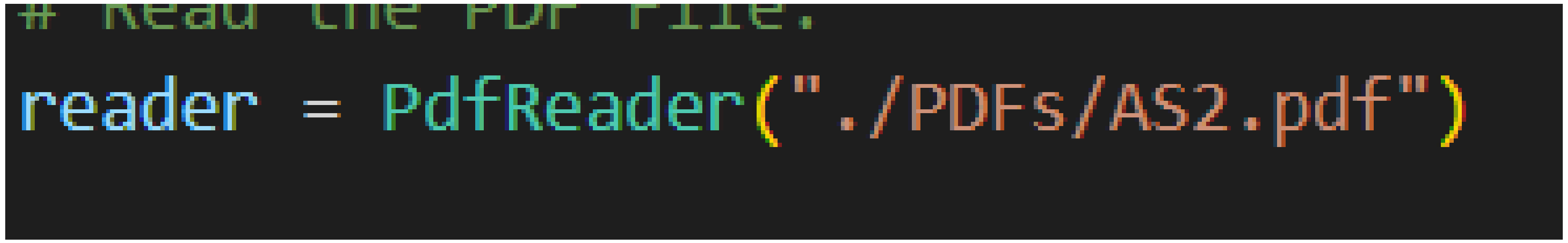
COUNTER IT!

- CharacterTextSplitter:
- Splits the Text into Multiple Chunks before passing it into GPT

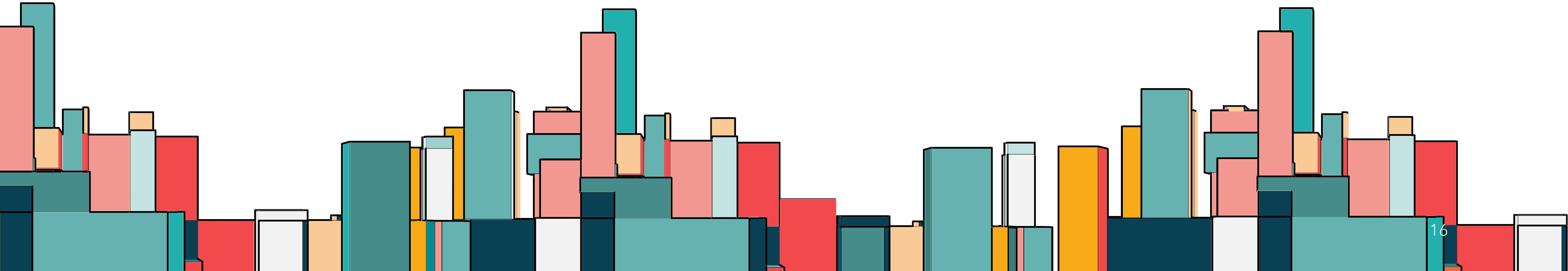


PDF READER

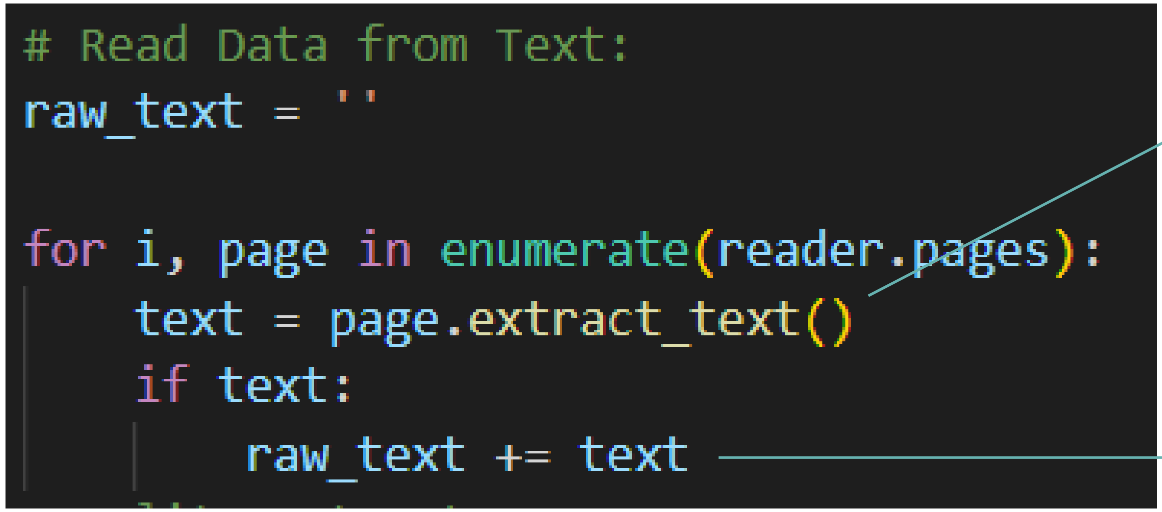
- Read PDF:



```
# Read the PDF File.
reader = PdfReader("./PDFs/AS2.pdf")
```



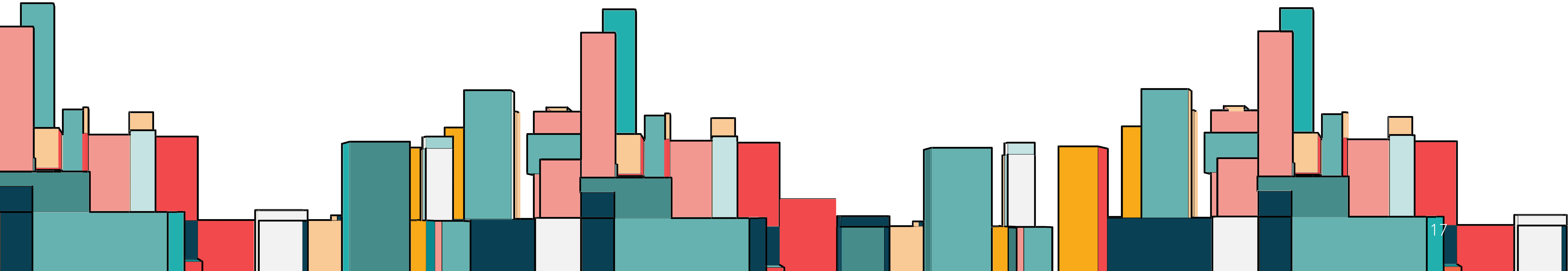
DEALING WITH TEXT



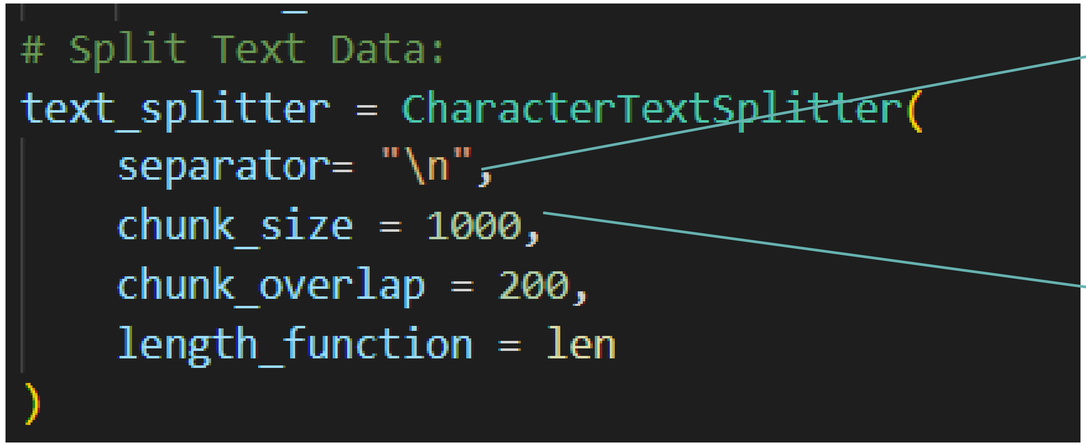
```
# Read Data from Text:
raw_text = ''

for i, page in enumerate(reader.pages):
    text = page.extract_text()
    if text:
        raw_text += text
```

- Extract Text

- Append it to raw_text



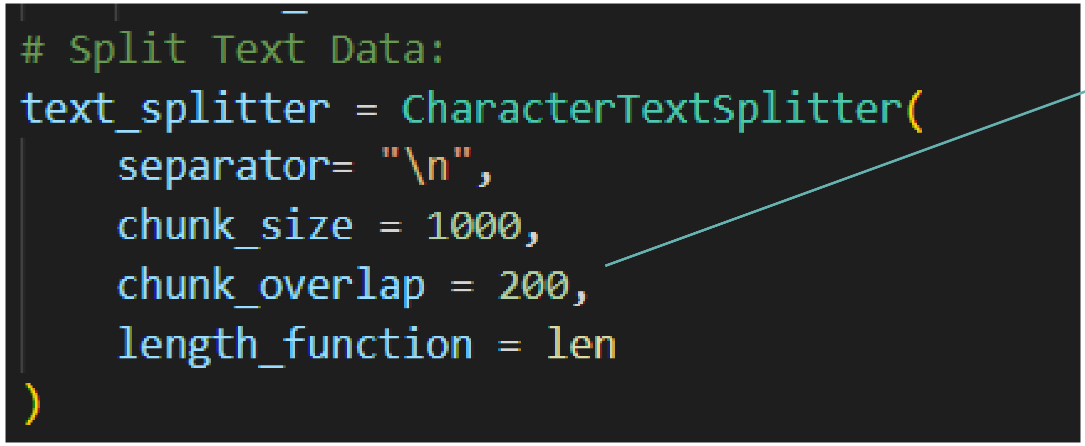
CHARACTERTEXTSPLITTER



```
# Split Text Data:
text_splitter = CharacterTextSplitter(
    separator= "\n",
    chunk_size = 1000,
    chunk_overlap = 200,
    length_function = len
)
```

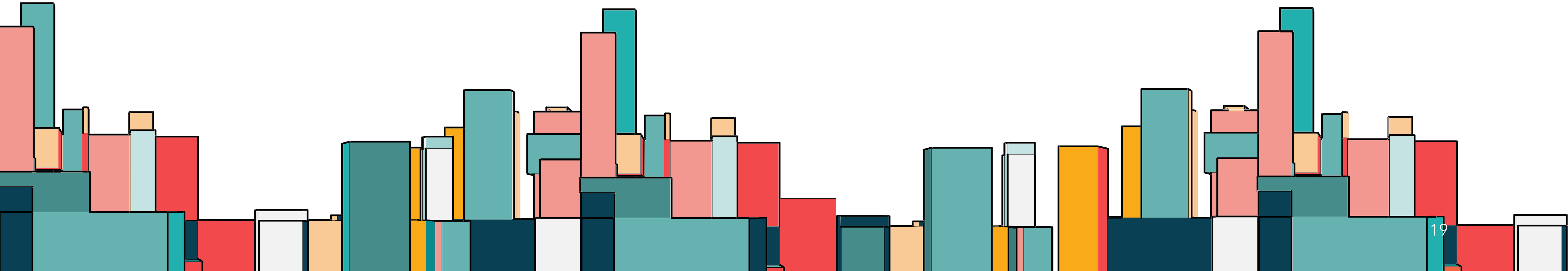
- Sees Text according to new line
- Split into chunks of 1000 tokens

CHARACTERTEXTSPLITTER

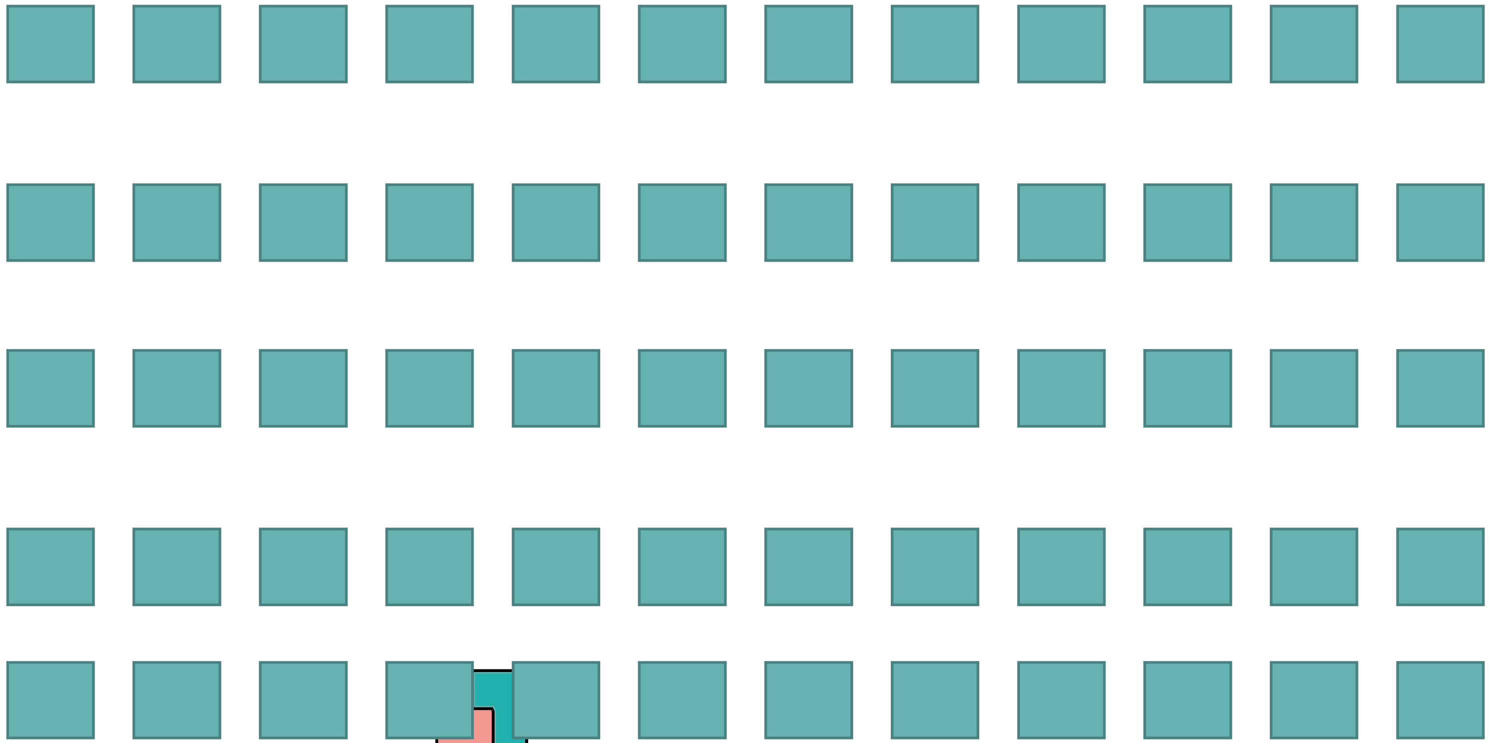


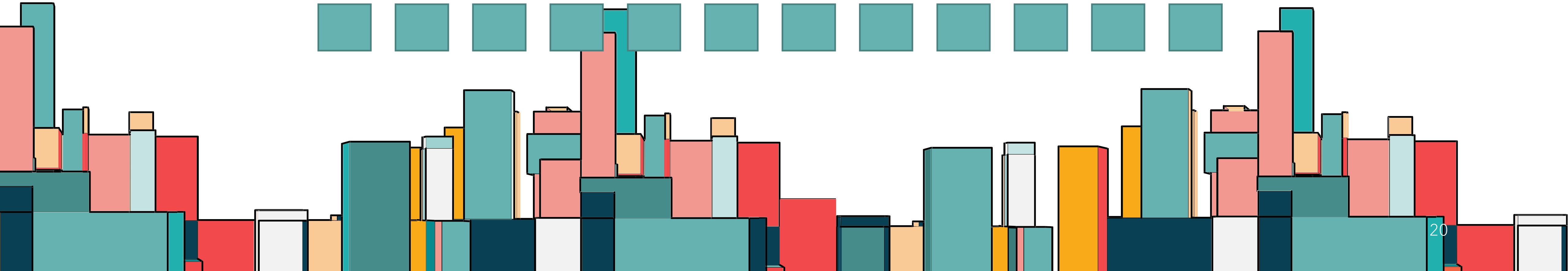
```
# Split Text Data:
text_splitter = CharacterTextSplitter(
    separator= "\n",
    chunk_size = 1000,
    chunk_overlap = 200,
    length_function = len
)
```

- Overlapping Tokens between chunks

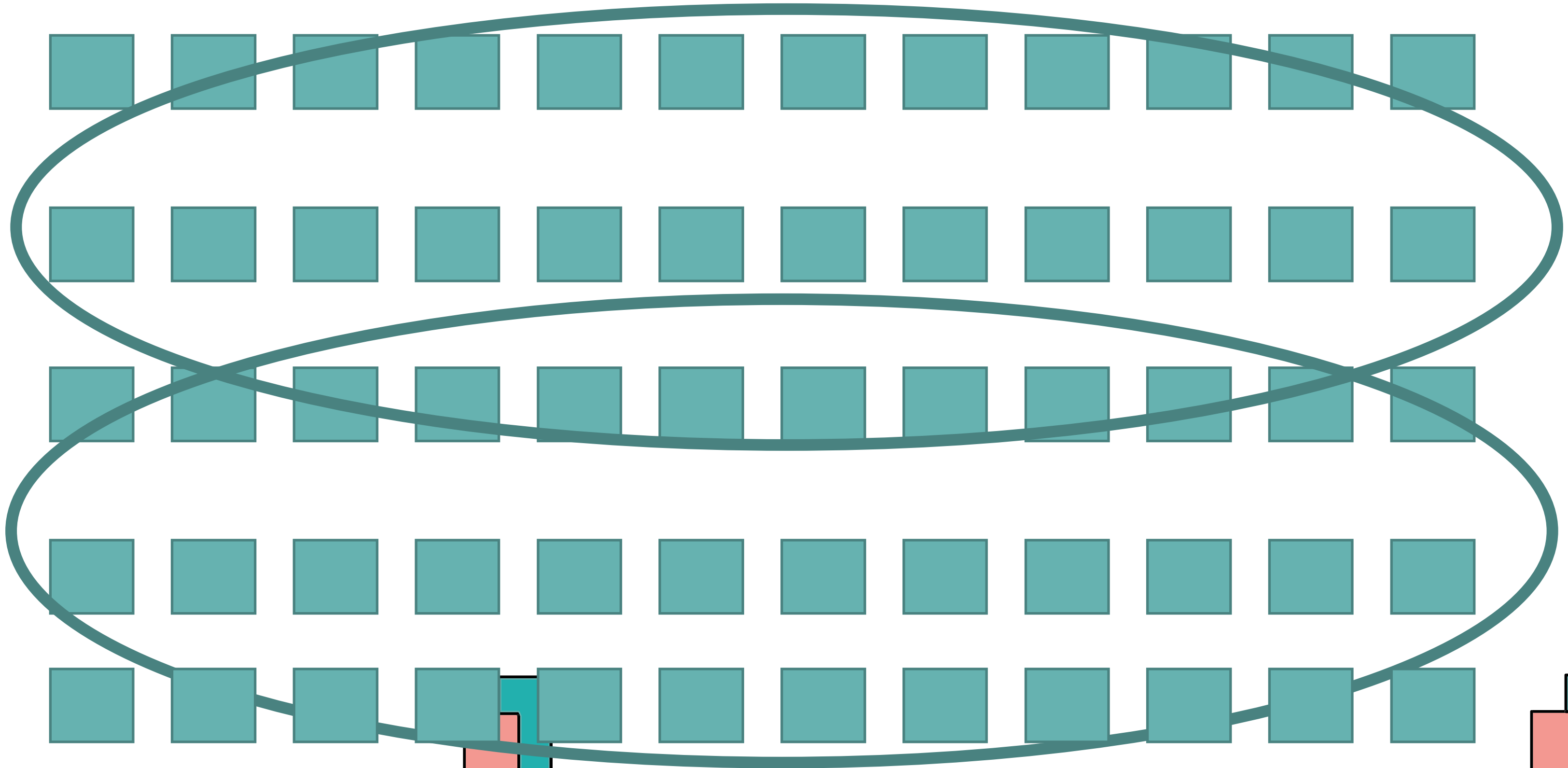


CHARACTERTEXTSPLITTER

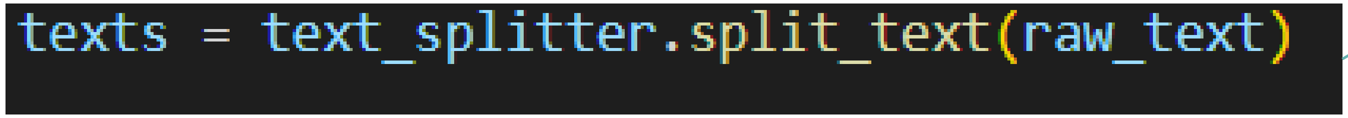




CHARACTERTEXTSPLITTER

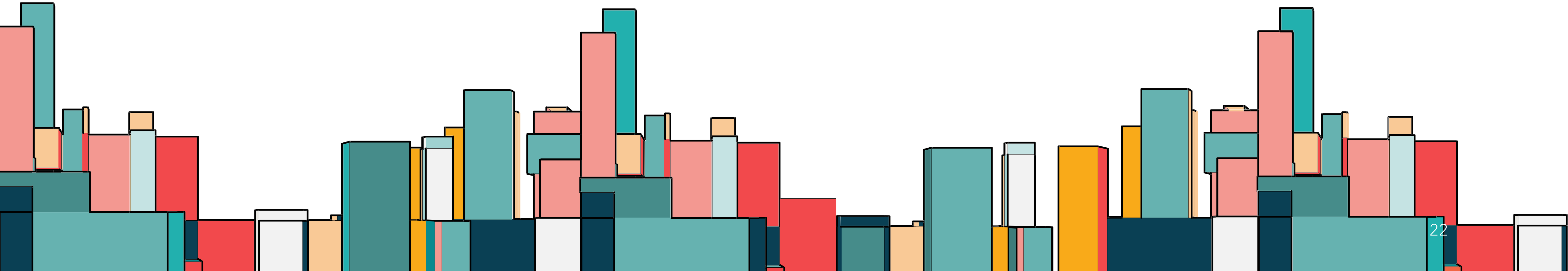


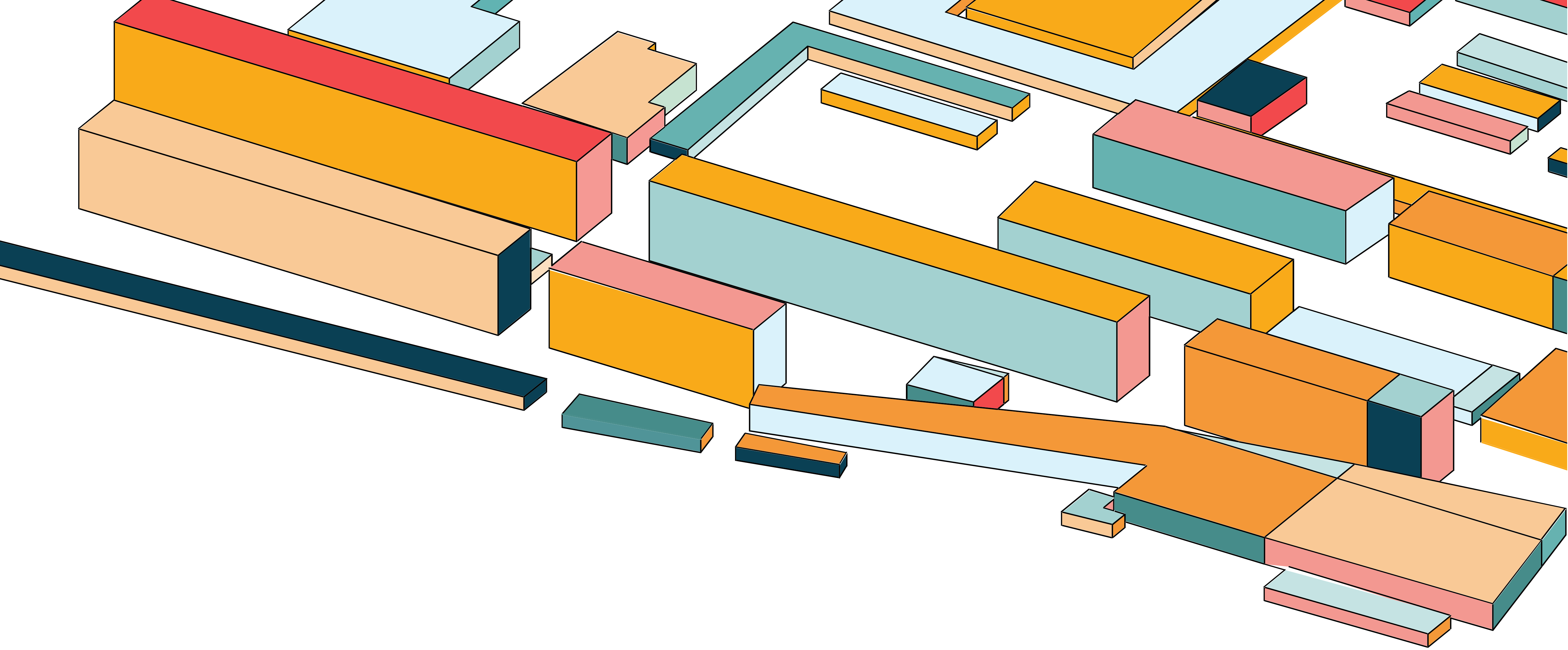
APPLY IT TO OUR EXTRACTED TEXT



```
texts = text_splitter.split_text(raw_text)
```

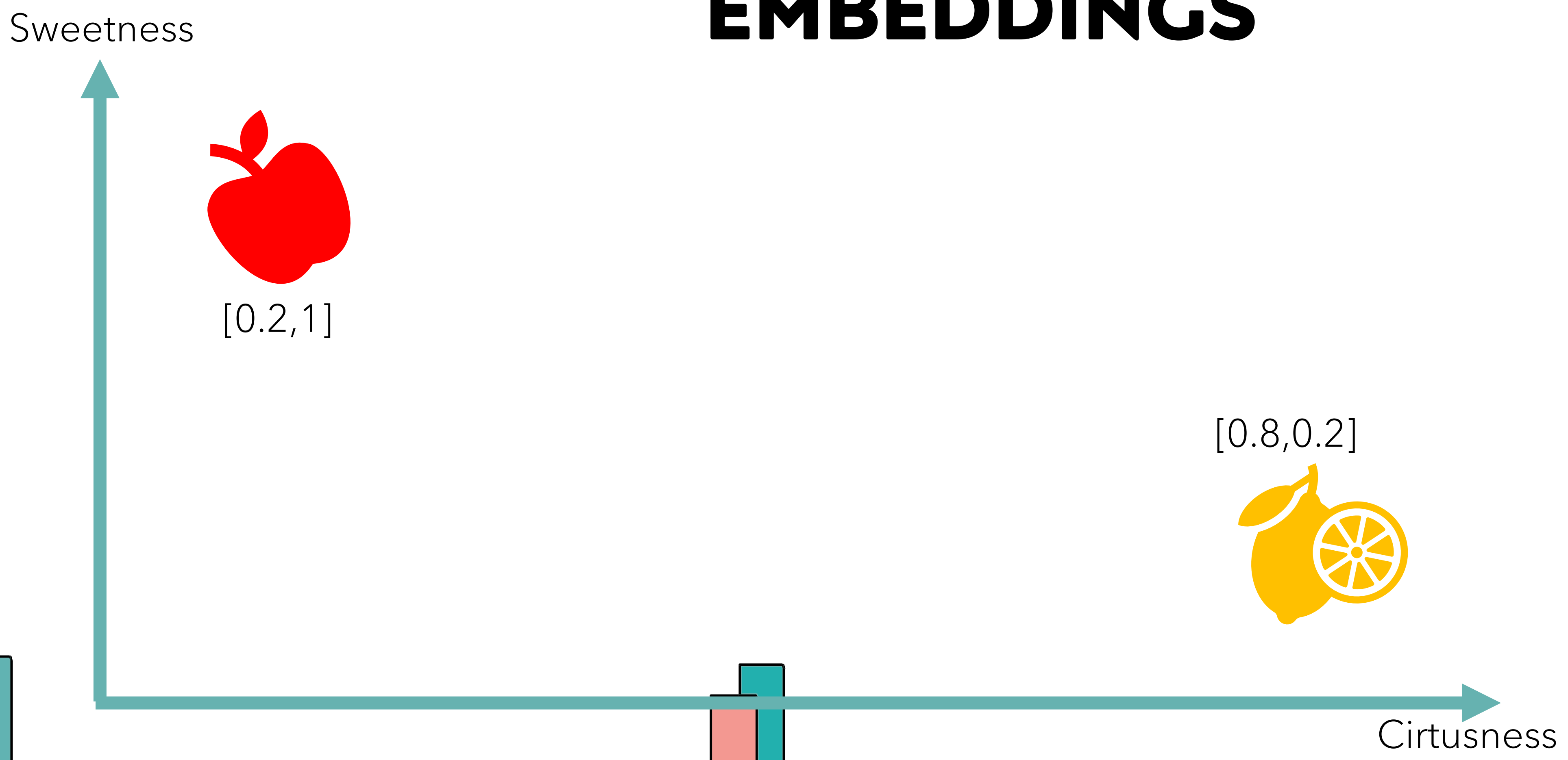
- Apply it to our text



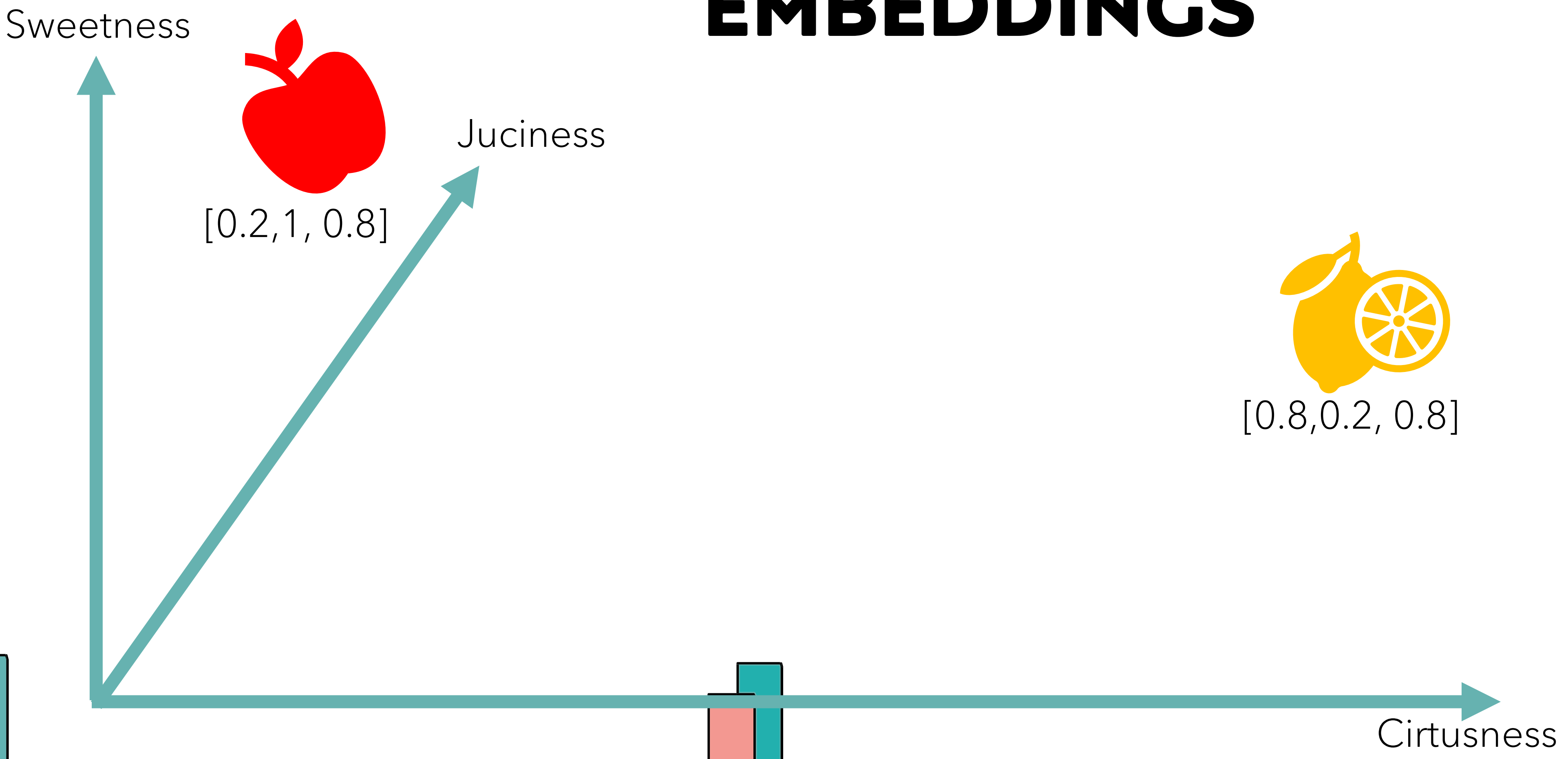


EMBEDDINGS

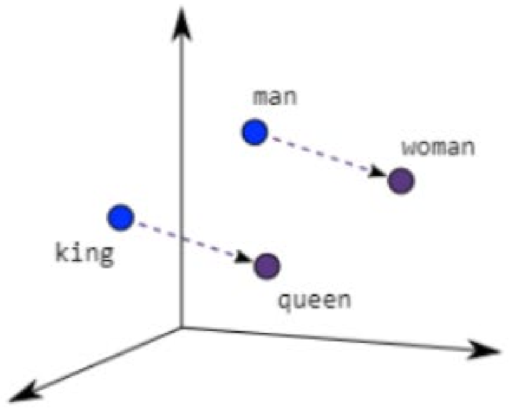
EMBEDDINGS



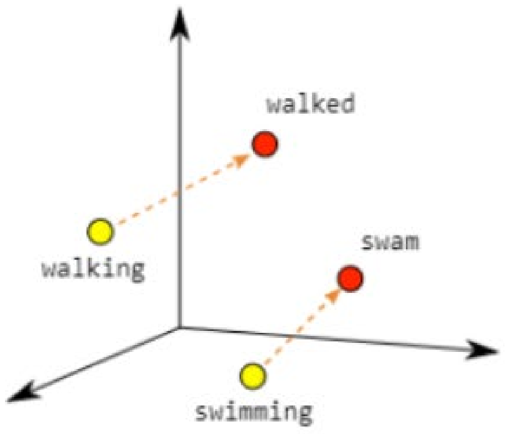
EMBEDDINGS



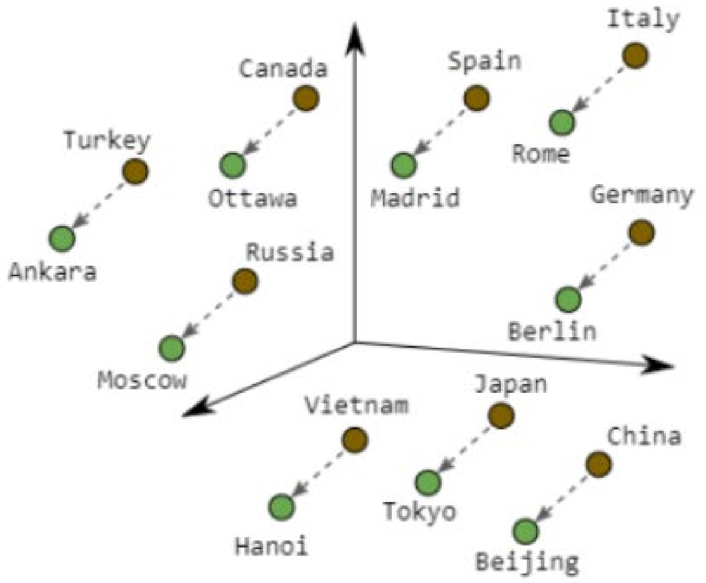
EMBEDDINGS



Male-Female

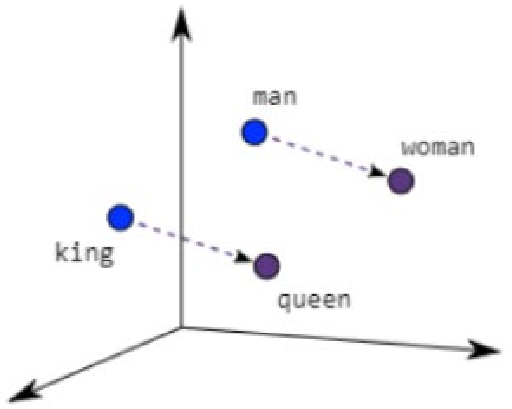


Verb Tense



Country-Capital

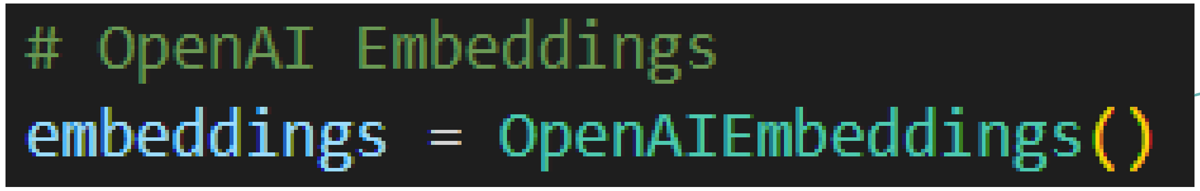
EMBEDDINGS



Male-Female

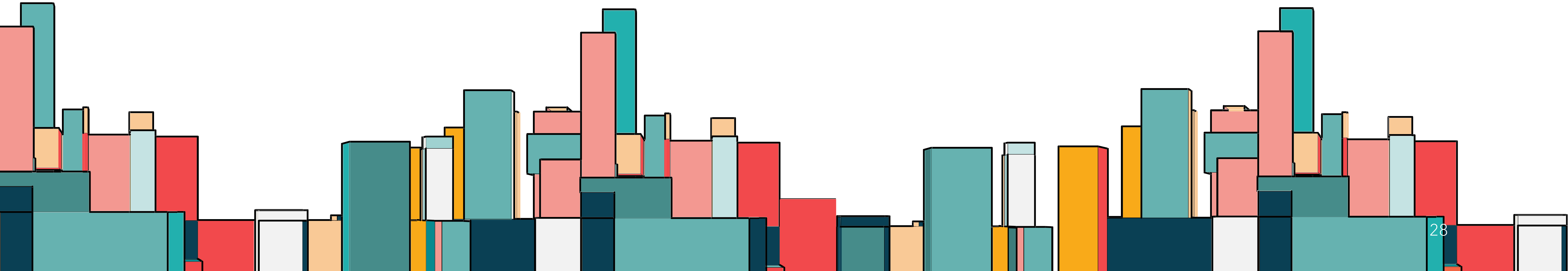
- Words with similar meanings are closer together!

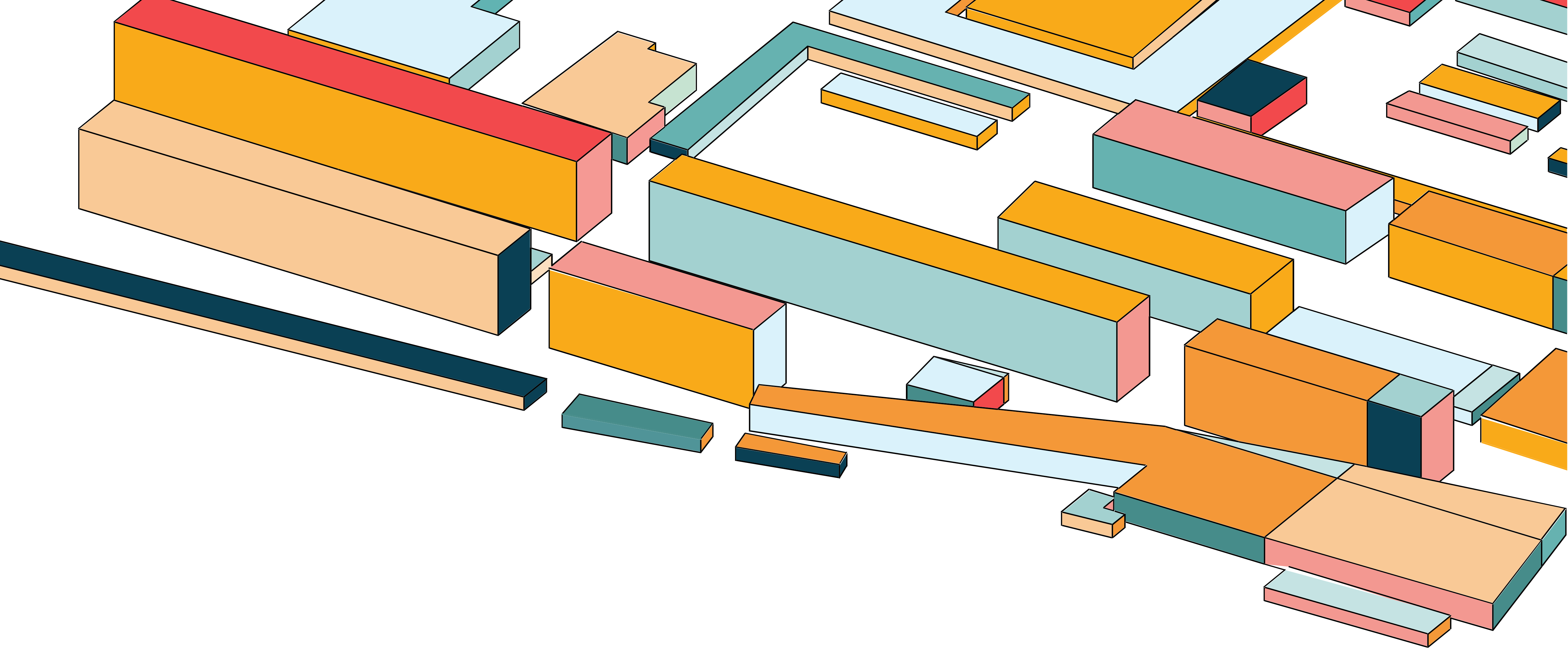
EMBEDDINGS



```
# OpenAI Embeddings
embeddings = OpenAIEmbeddings()
```

- Import to use OpenAI Embeddings

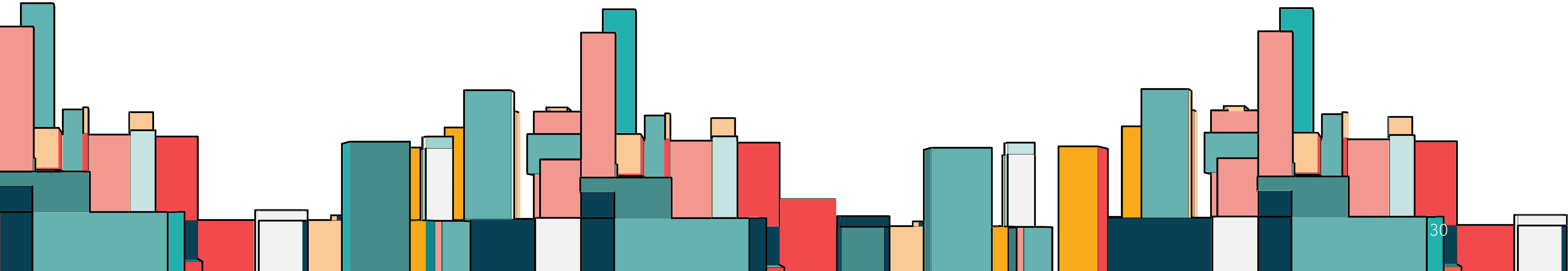




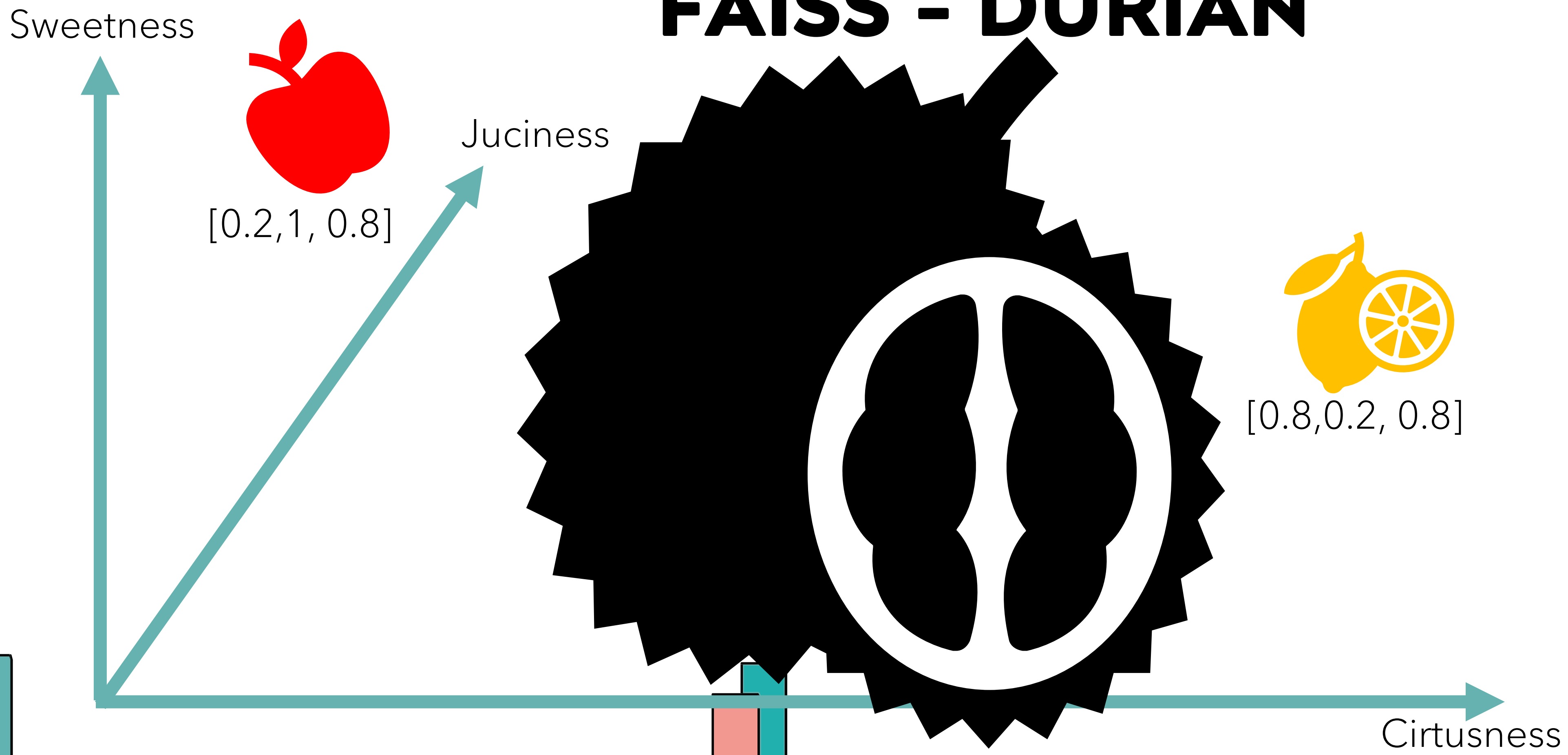
FAISS – FACEBOOK AI SIMILARITY SEARCH

FAISS

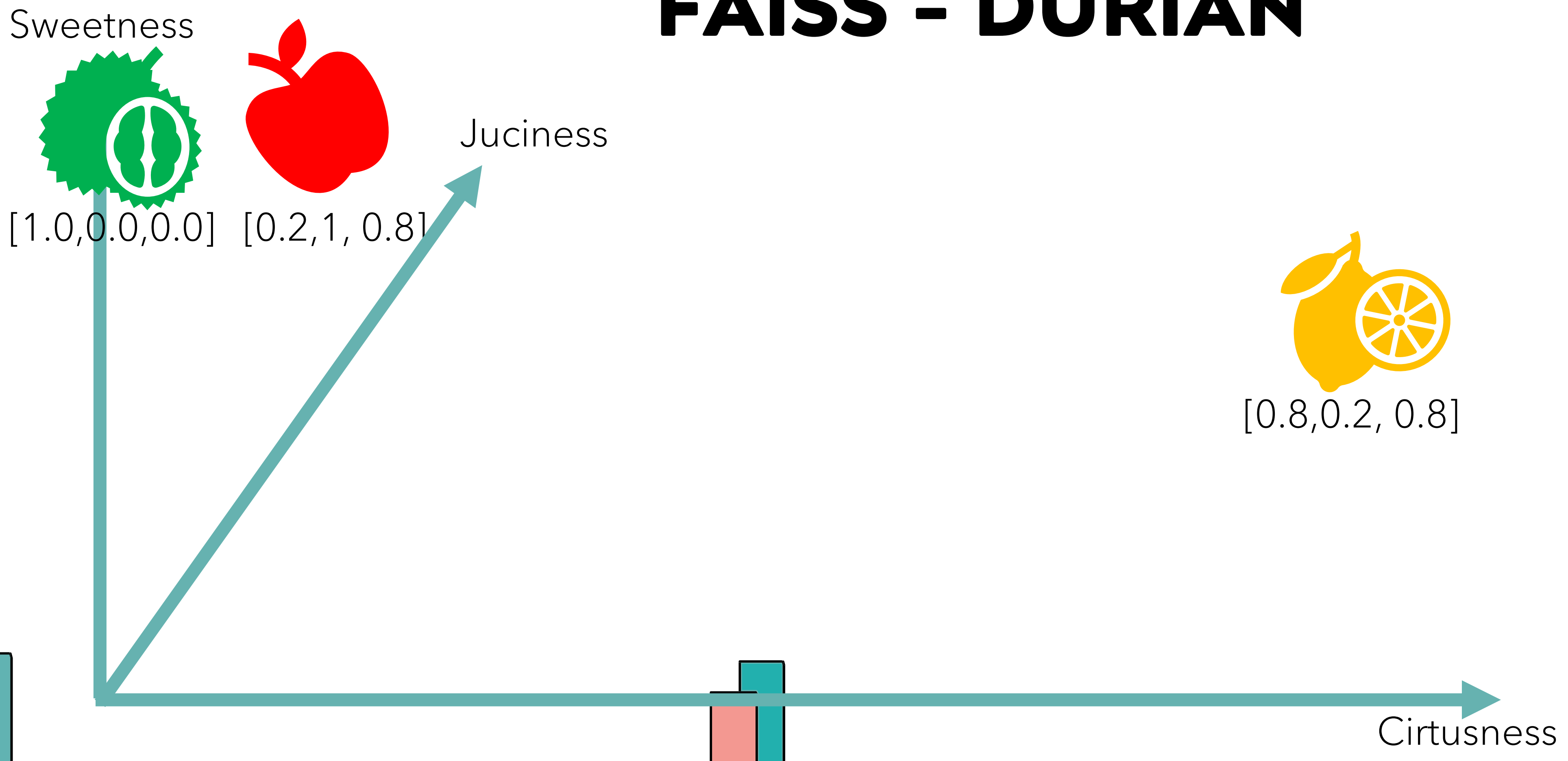
- Similarity Search:
- Find the nearest ‘neighbours’ of a particular word with a set of embeddings



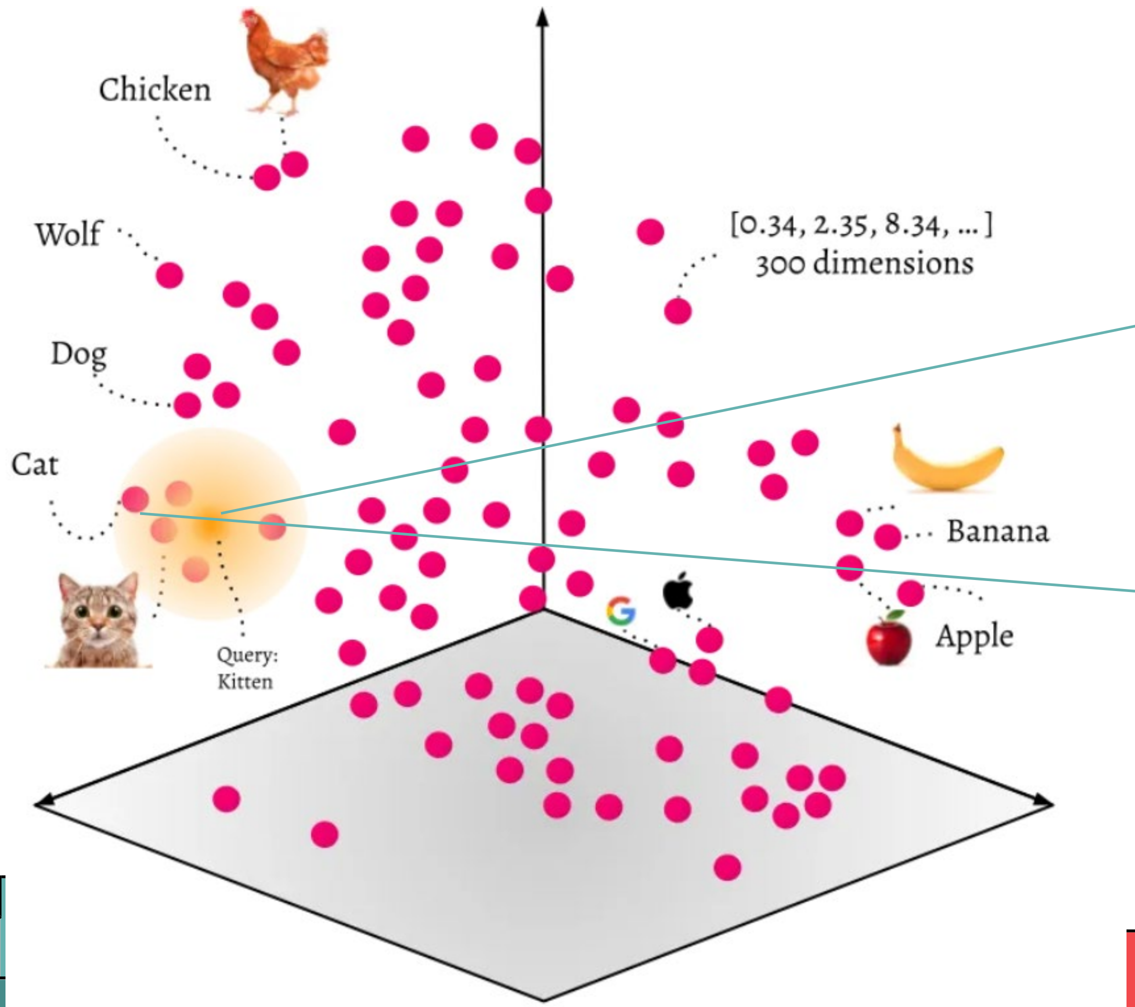
FAISS - DURIAN



FAISS - DURIAN

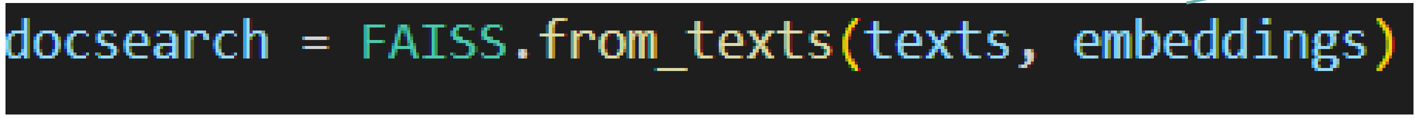


FAISS - KITTEN

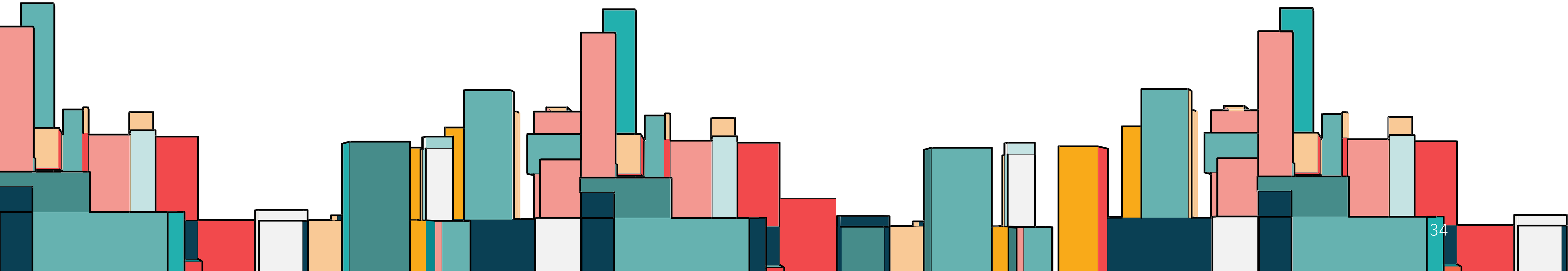


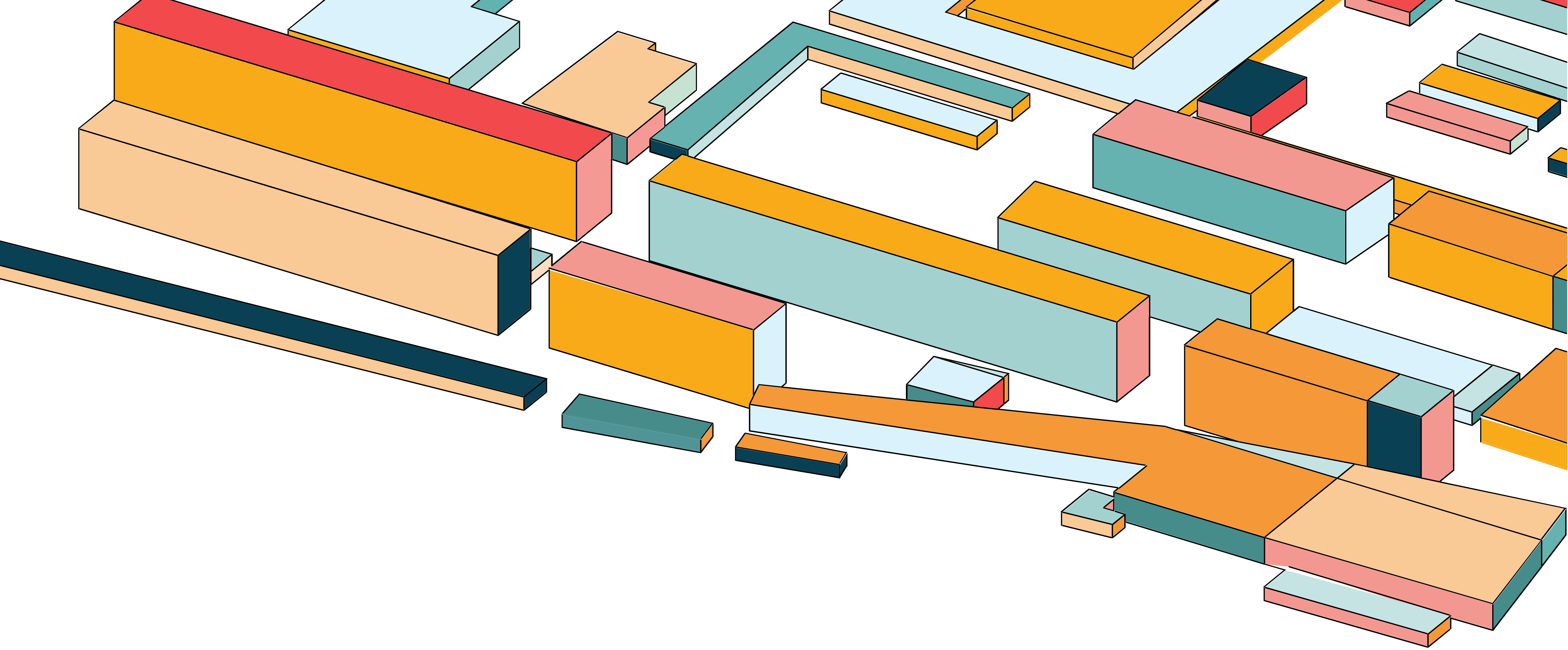
- Document Word
- OpenAI Embeddings

FAISS


```
docsearch = FAISS.from_texts(texts, embeddings)
```

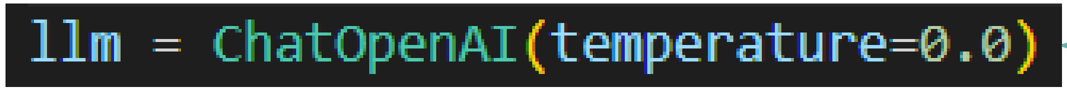
- Do Similarity Search with OpenAI Embeddings





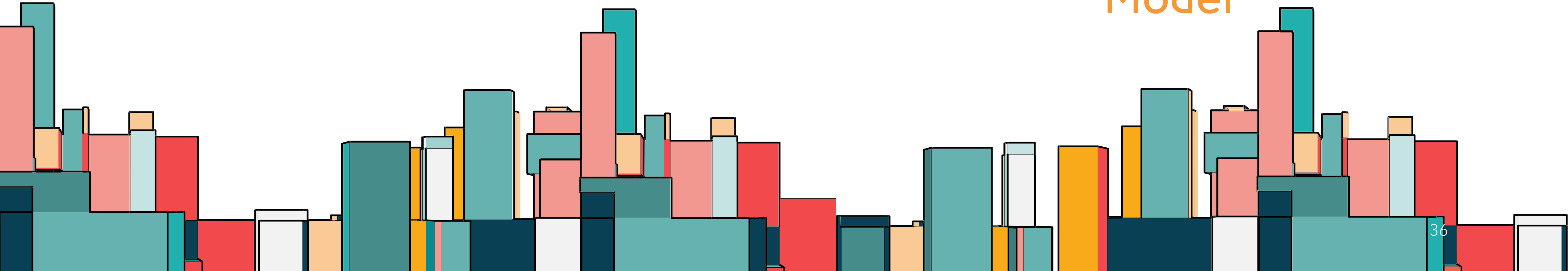
USING LANGCHAIN (FINALLY!)

LARGE LANGUAGE MODEL (LLM)



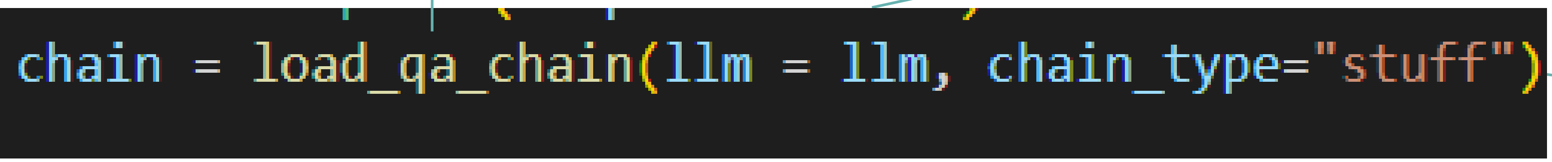
```python
llm = ChatOpenAI(temperature=0.0)
```

- Use GPT-3.5-Turbo
- Restricts Creativity of Model



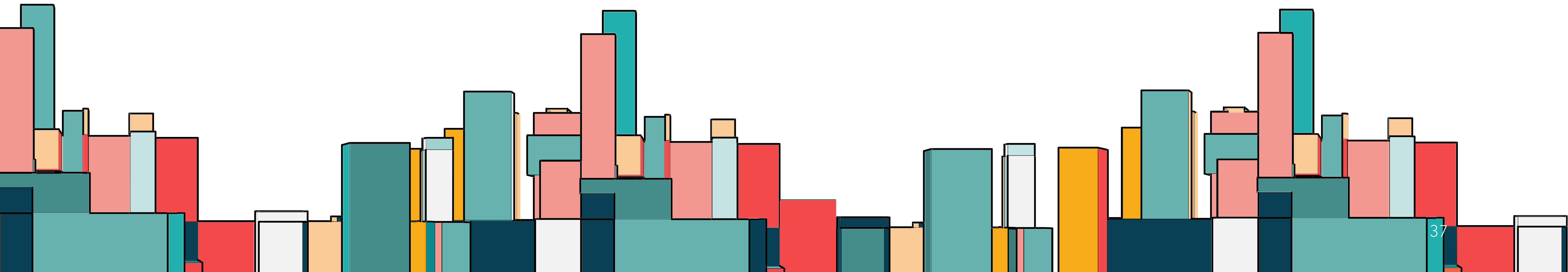
CHAINING

- Question & Answer



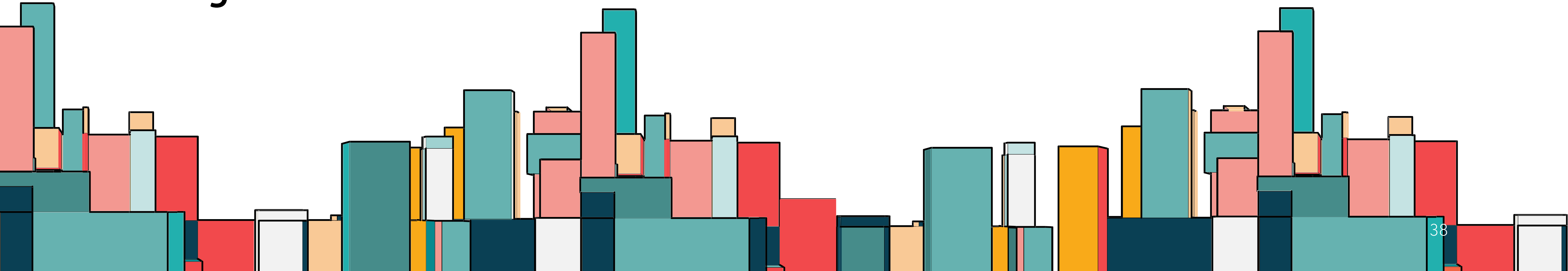
```
chain = load_qa_chain(llm = llm, chain_type="stuff")
```

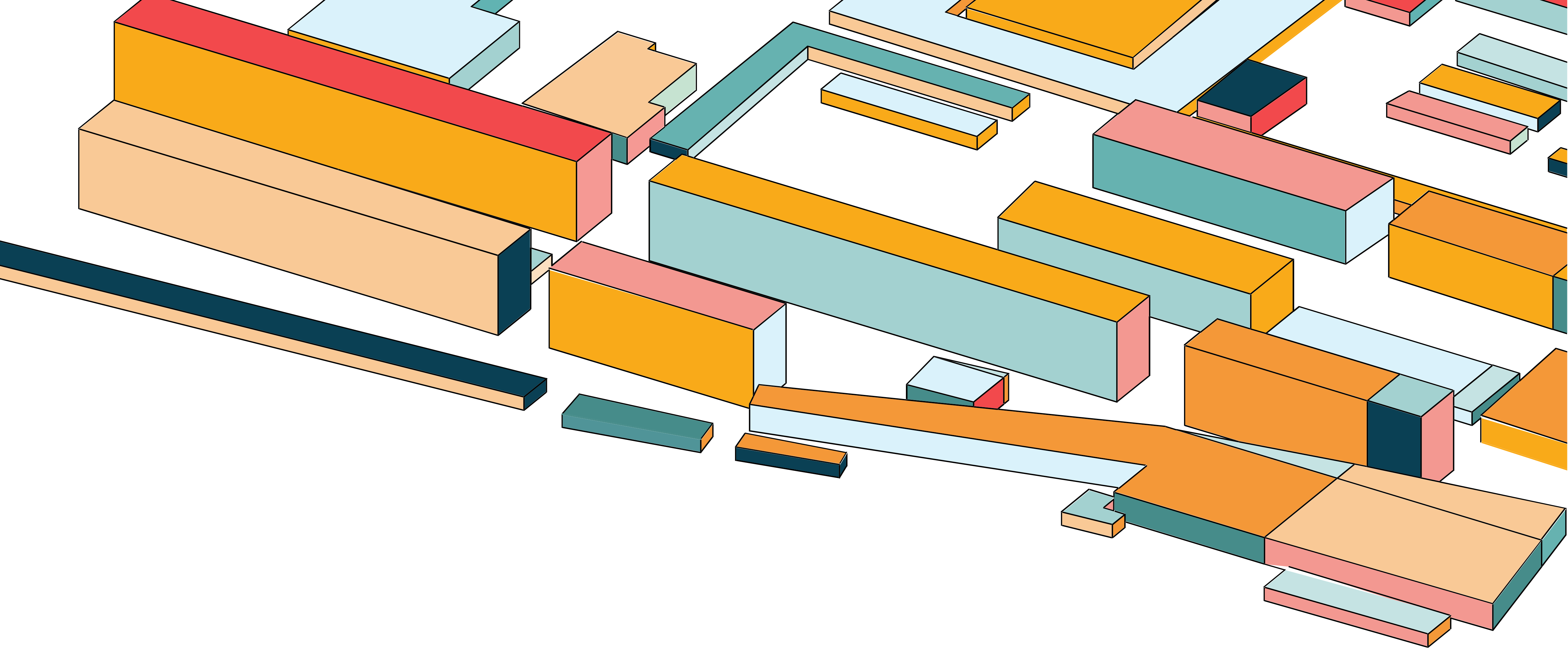
- Use GPT-3.5-Turbo
- Use all Text in Document in prompt



ADDITIONAL INFO: CHAIN TYPE

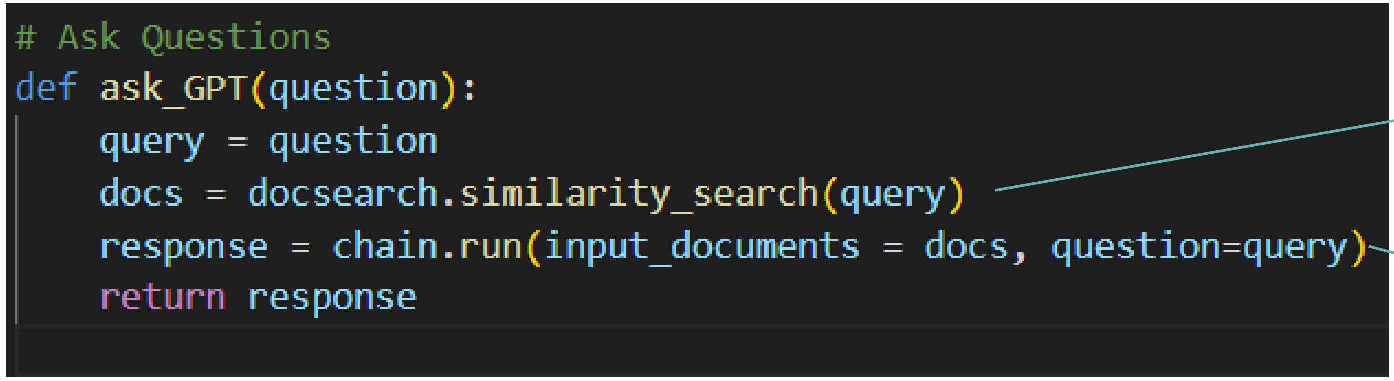
- map_reduce: Separate text into batches before feeding into LLM
- Refine: Separate text into batches -> feed 1st, 2nd etc. batch -> refine answer
- map-rerank: Separate text into batches -> feed 1st 2nd etc. batch -> gives score on answer -> Come up with answer with highest score





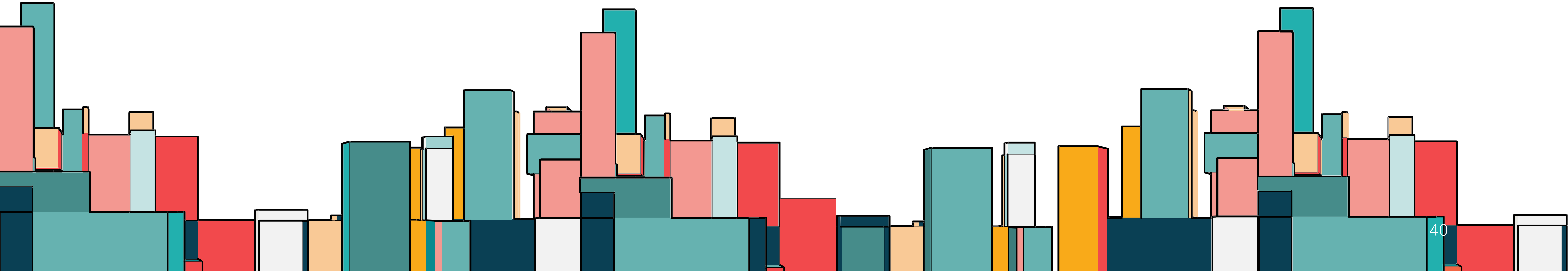
LET'S GO!!! (USING OUR BOT)

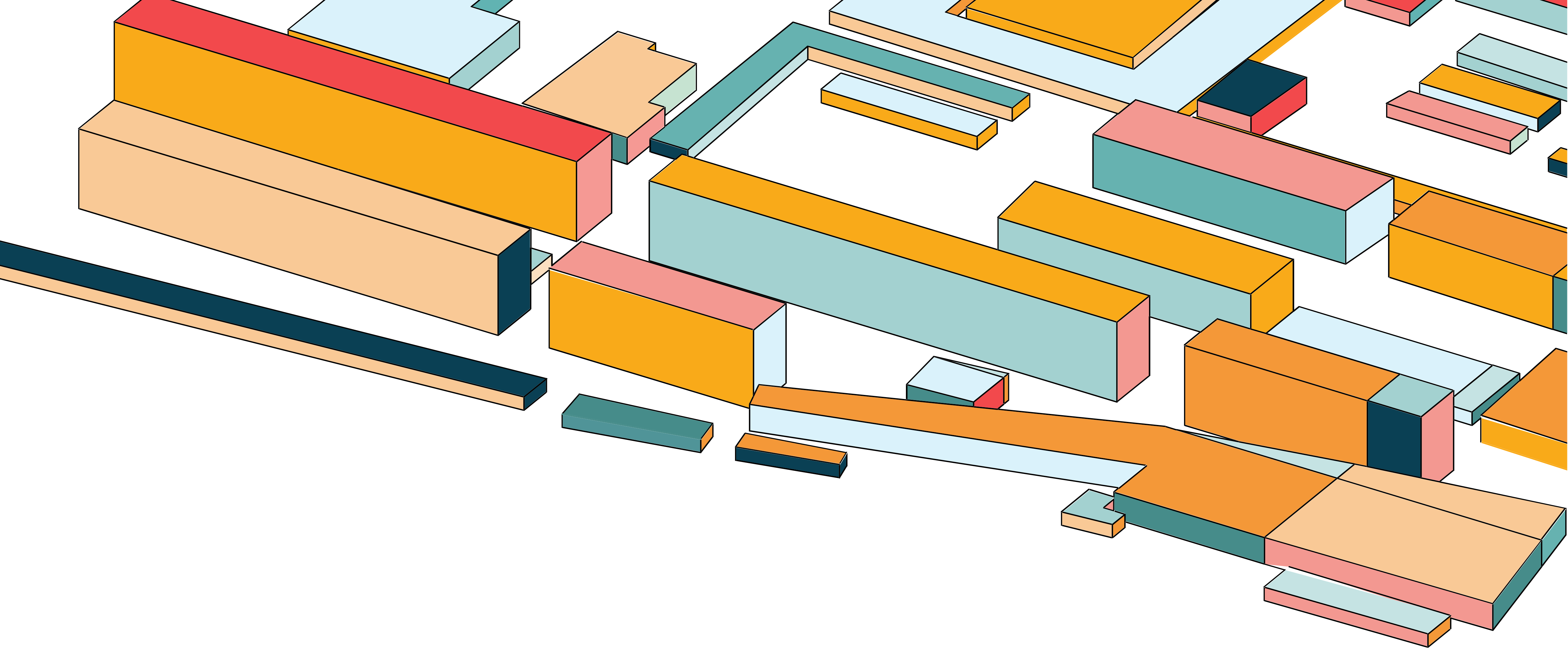
WRITING A FUNCTION



```
# Ask Questions
def ask_GPT(question):
    query = question
    docs = docsearch.similarity_search(query)
    response = chain.run(input_documents = docs, question=query)
    return response
```

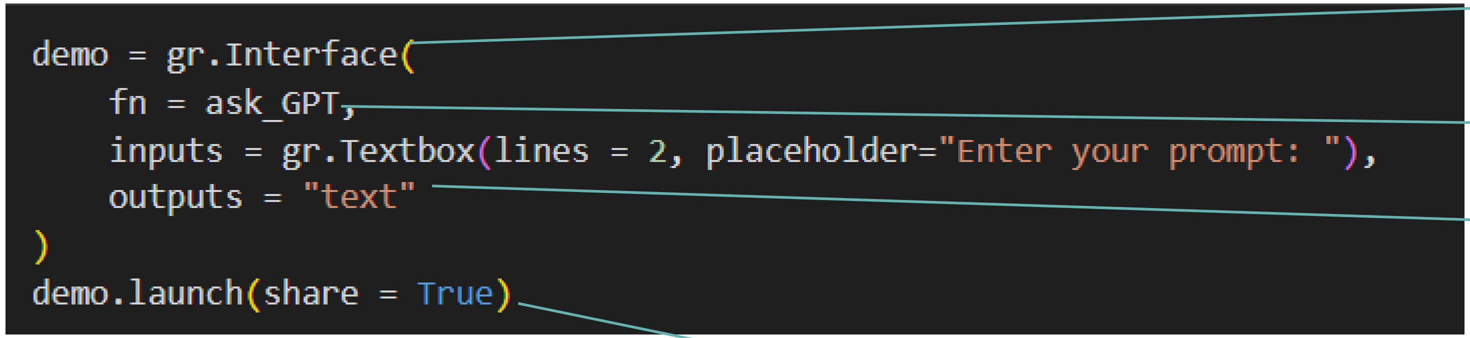
- Similarity Search of our Query
  - Run the Question & Answer chain to generate answer





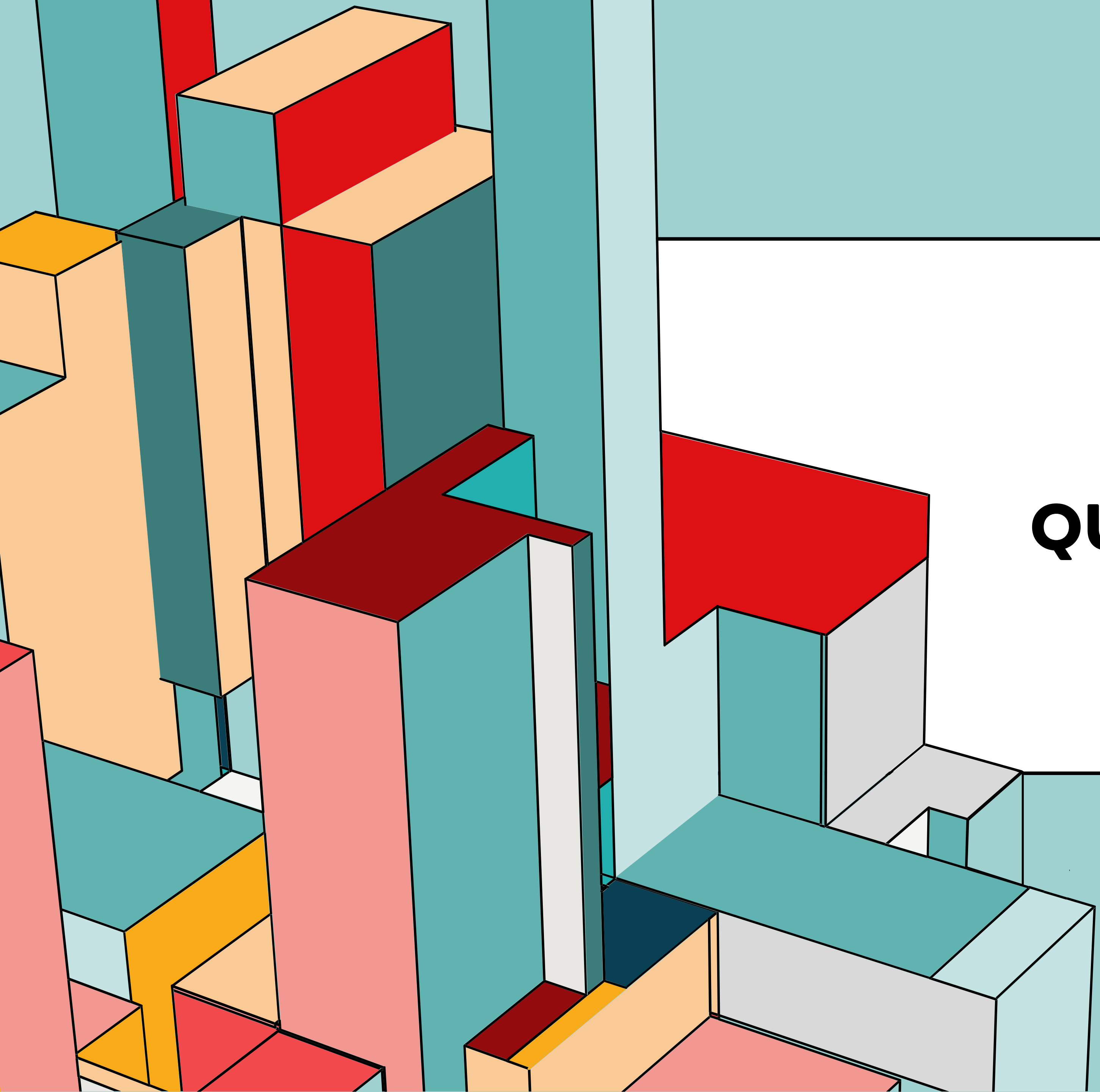
INTO CLASS – QUICK WEB DEPLOYMENT

GRADIO INTERFACE

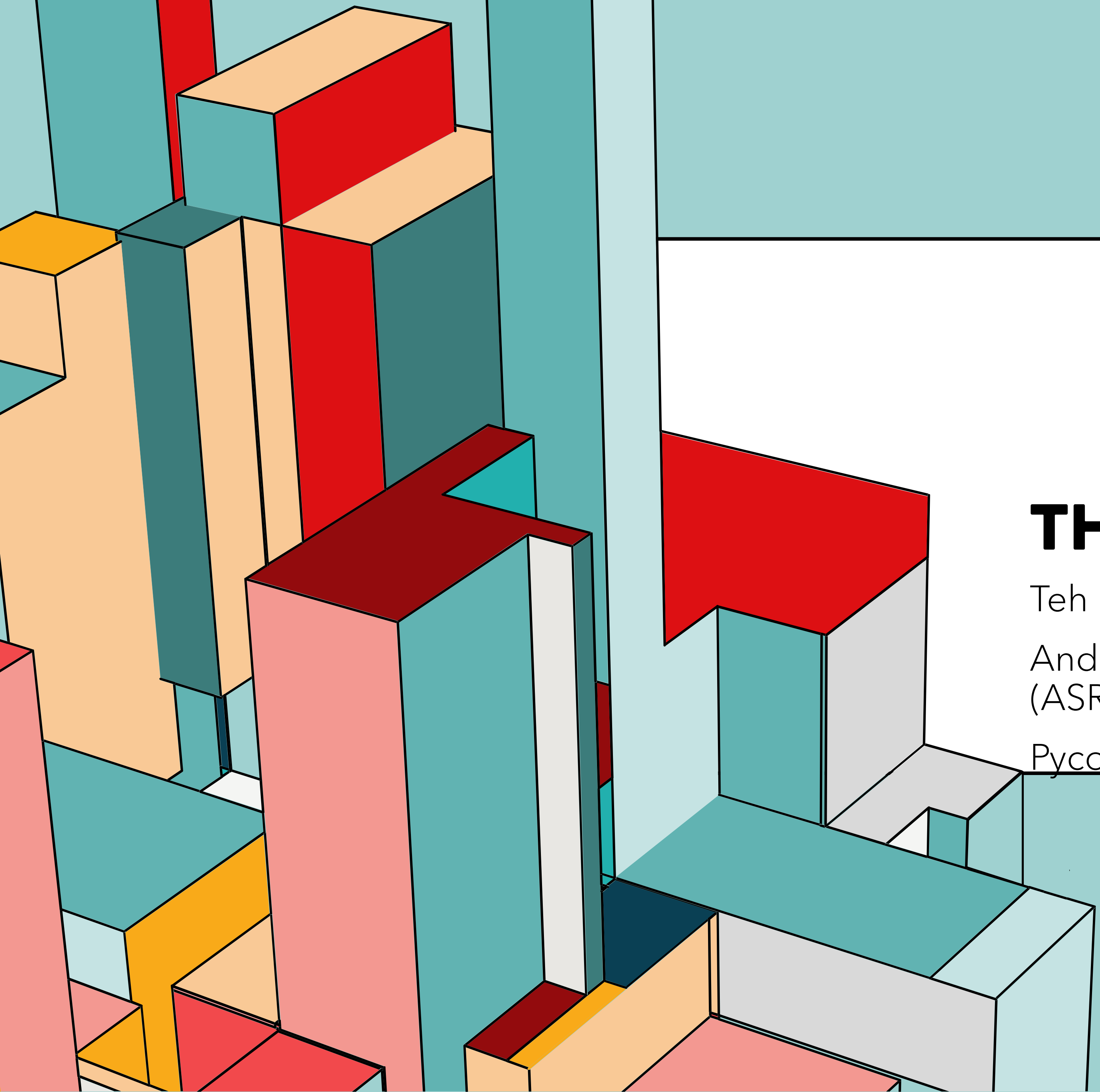


```python
demo = gr.Interface(
    fn = ask_GPT,
    inputs = gr.Textbox(lines = 2, placeholder="Enter your prompt: "),
    outputs = "text"
)
demo.launch(share = True)
```

- Gradio Interface
- Use the Function
- Output response
- *Spawn a live link!*



QUESTIONS?



THANKS

Teh Kim Wee

Anderson Serangoon Junior College (ASRJC)

Pycon Education Summit 2023