

Introduction to large language models (LLMs)

INTRODUCTION TO LLMS IN PYTHON

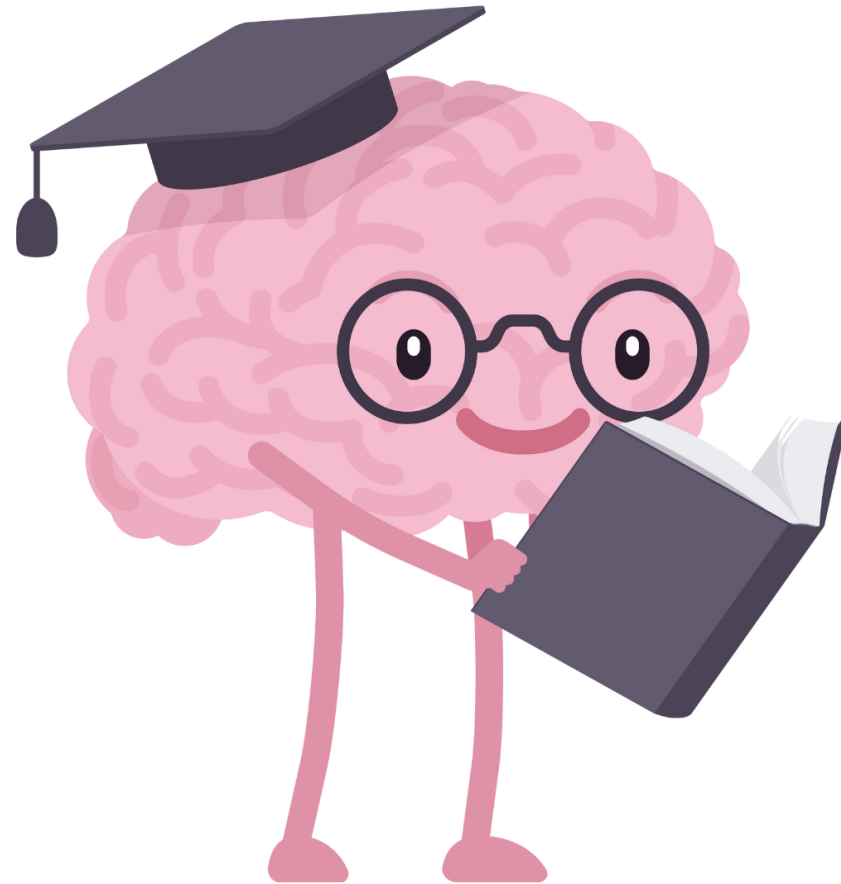


Jasmin Ludolf

Senior Data Science Content Developer,
DataCamp

Previous knowledge

- Navigating the Hugging Face Hub
- Deep learning models



Introduction to LLMs



Understanding

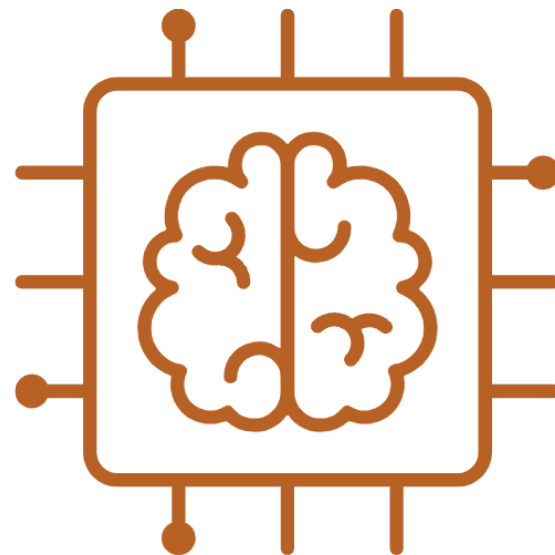
Introduction to LLMs

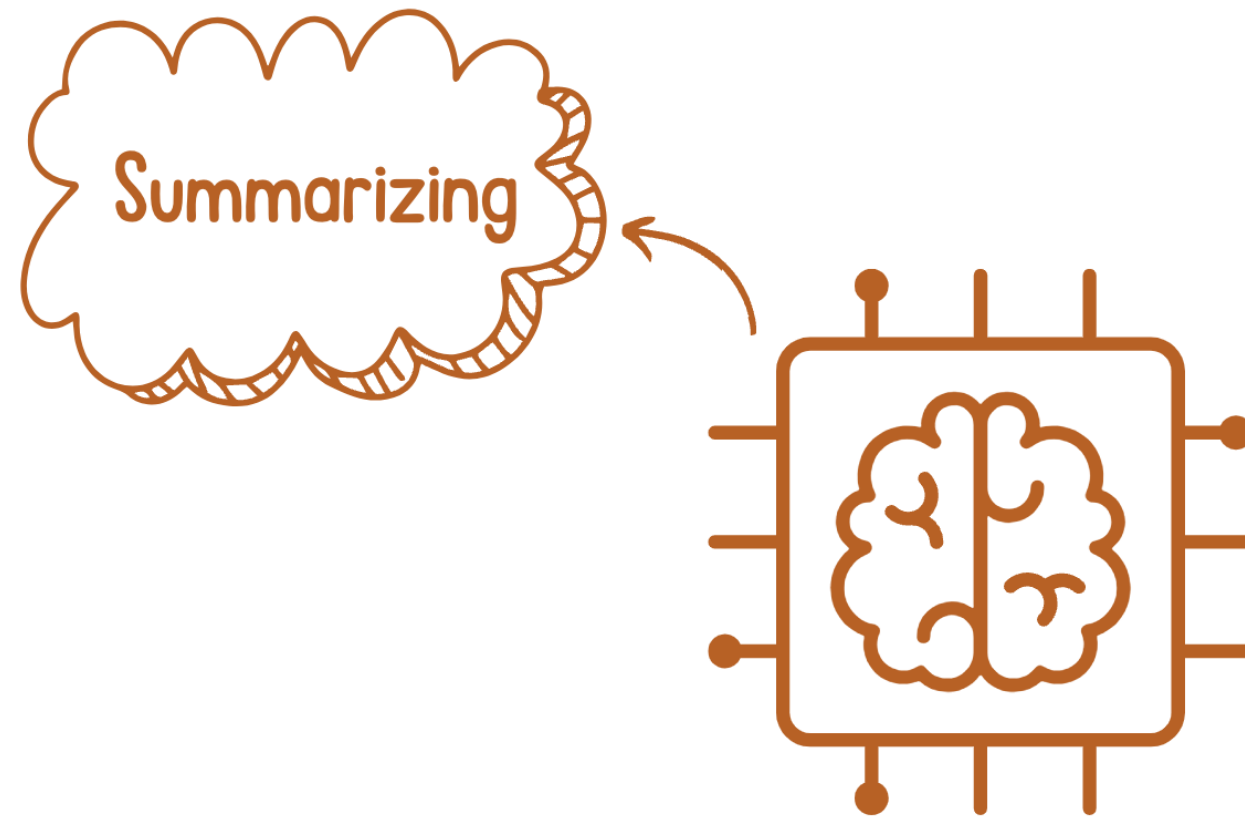


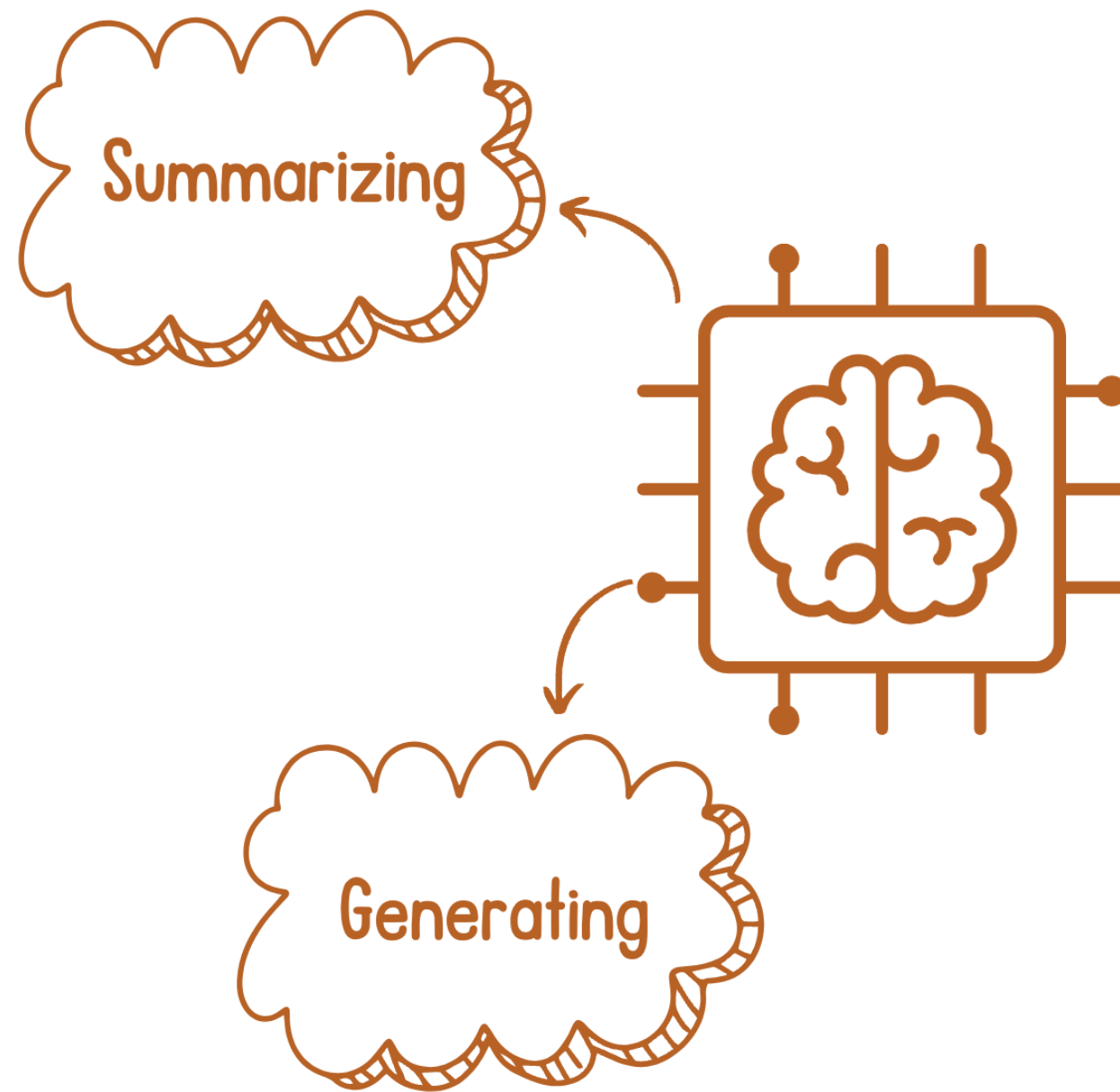
Understanding

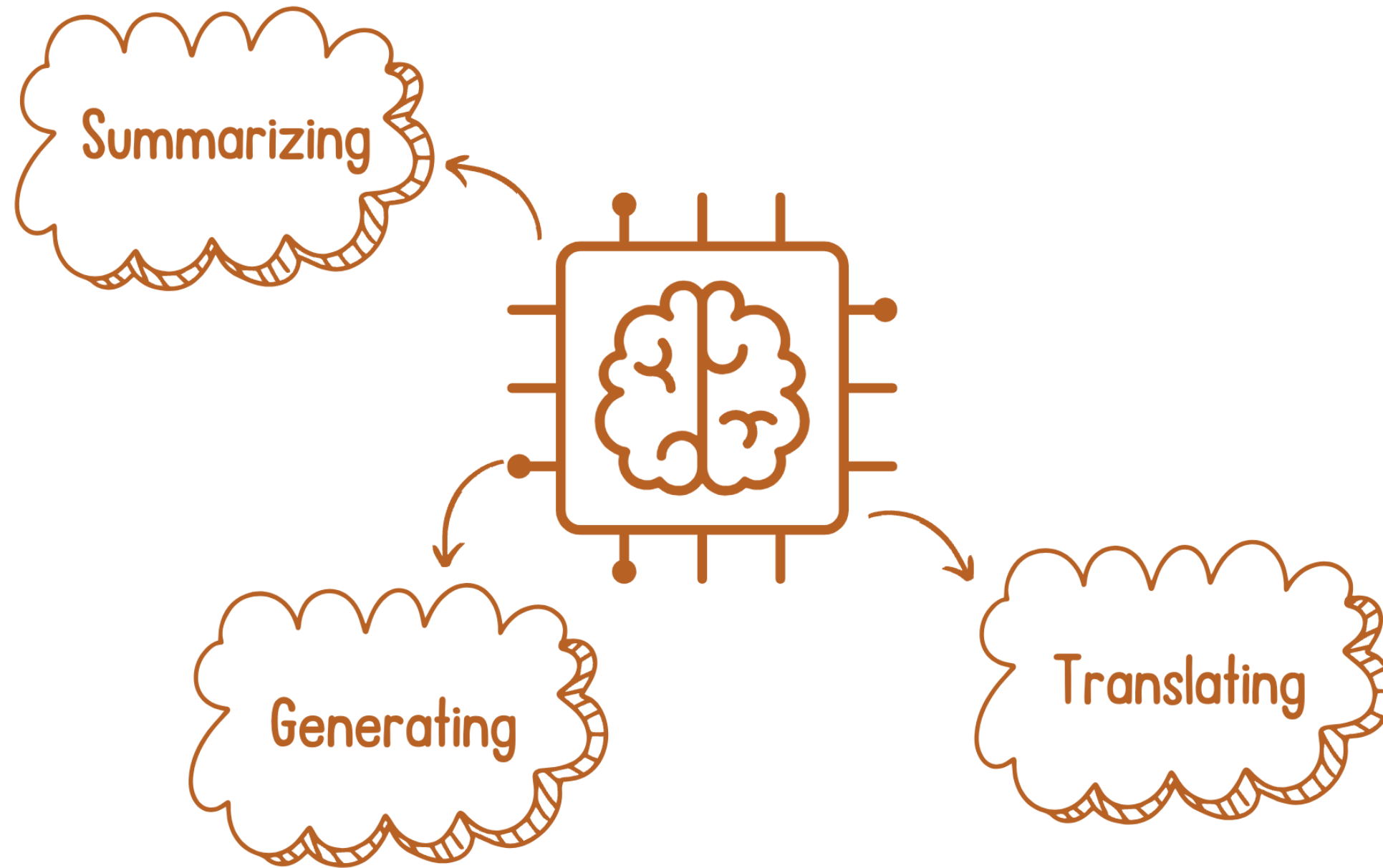


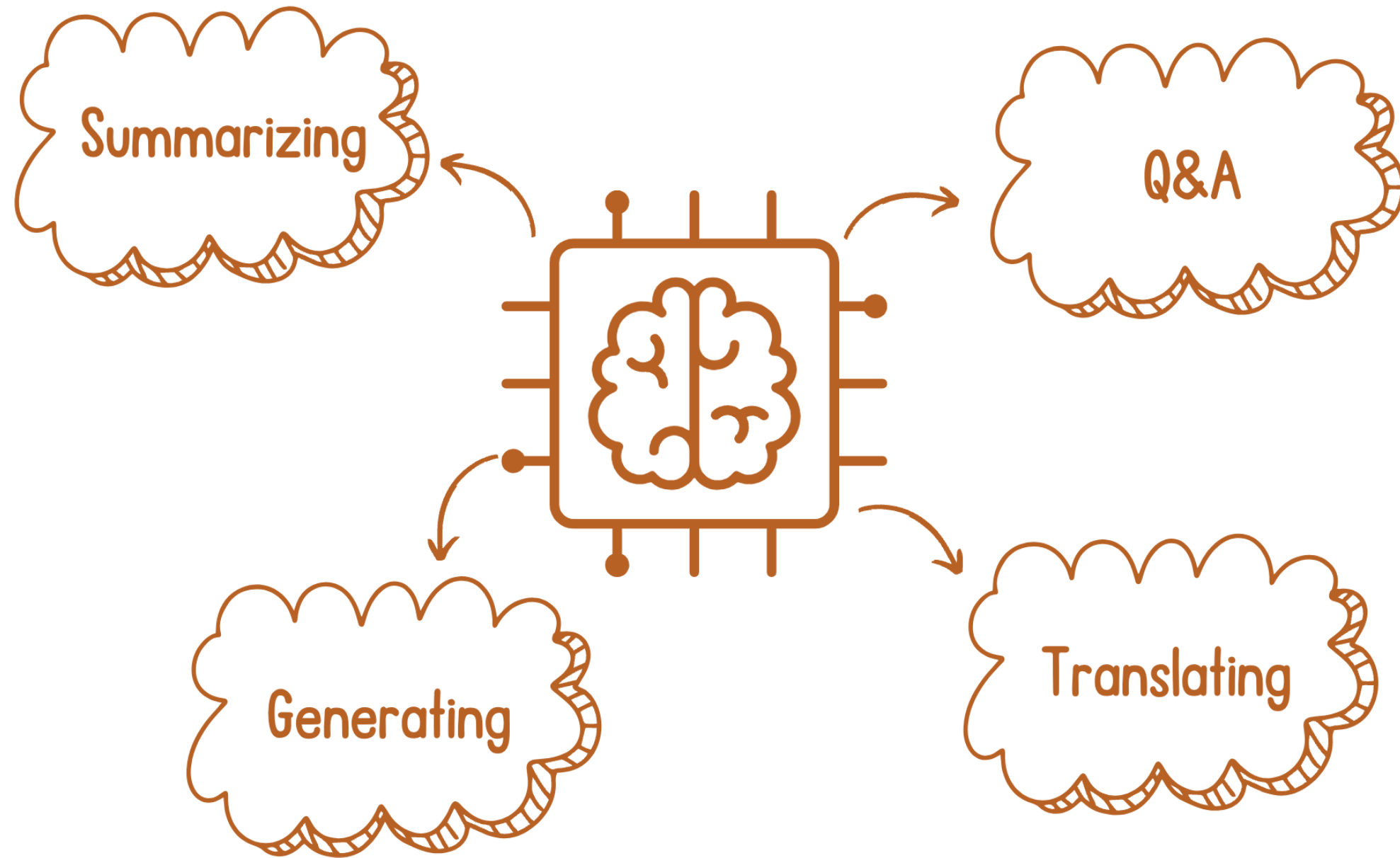
Using











LLaMA
by  **Meta**  **Gemini**

LLMs

- Based on **deep learning** architectures
- Most commonly **transformers**
- Huge **neural networks** with lots of parameters and text data



Using Hugging Face models

```
from transformers import pipeline
```

```
summarizer = pipeline(task="summarization", model="facebook/bart-large-cnn")
```

```
text = "Walking amid Gion's Machiya wooden houses is a mesmerizing experience. The beautifully preserved structures exuded an old-world charm that transports visitors back in time, making them feel like they had stepped into a living museum. The glow of lanterns lining the narrow streets add to the enchanting ambiance, making each stroll a memorable journey through Japan's rich cultural history."
```

```
summary = summarizer(text, max_length=50)
```

- `clean_up_tokenization_spaces=True` : remove unnecessary white space

Model outputs

```
print(summary)
```

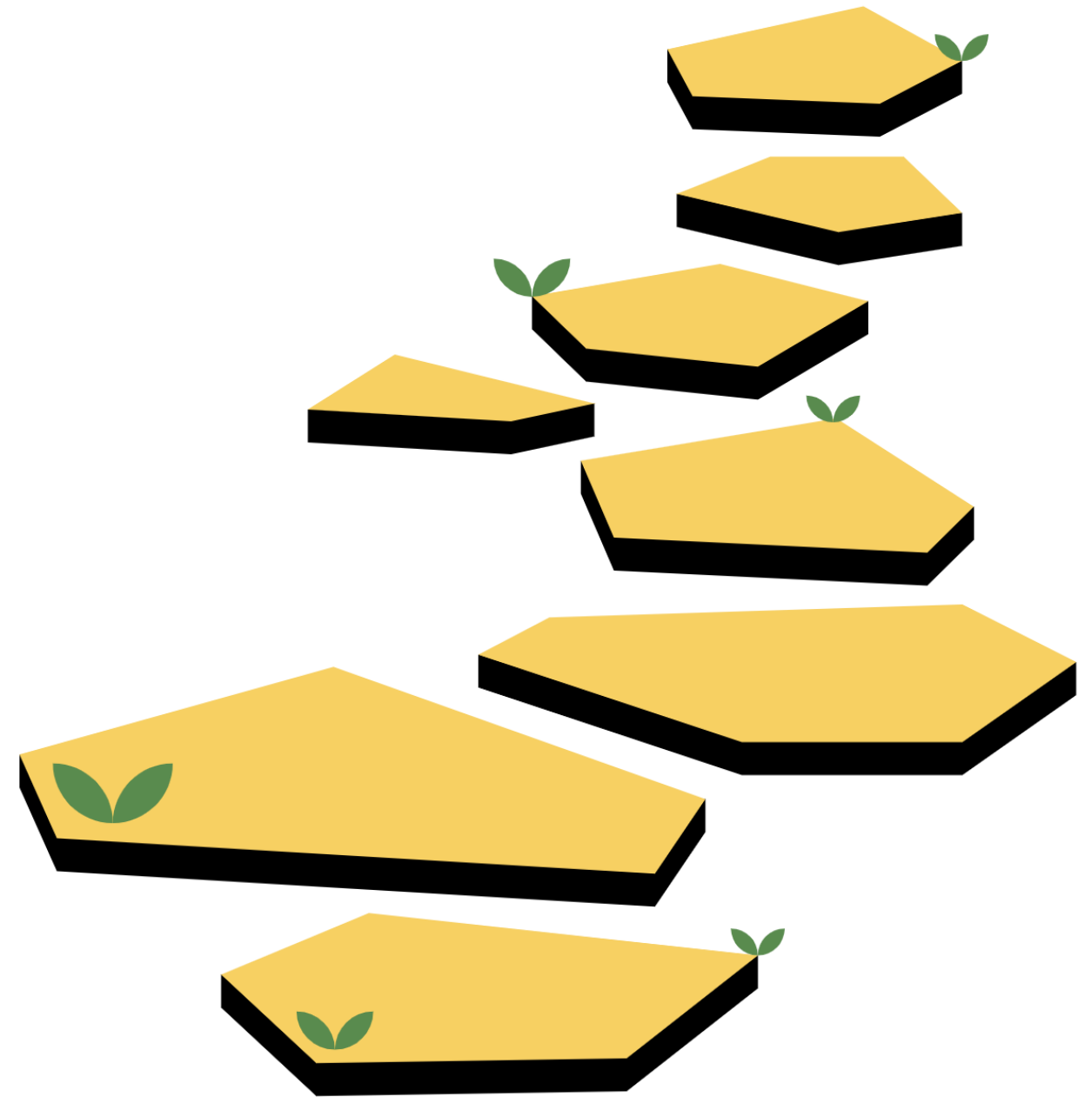
```
[{'summary_text': "Gion's Machiya  
wooden houses exuded an old-world  
charm that transports visitors back  
in time. The glow of lanterns lining  
the narrow streets add to the  
enchanted ambiance, making each  
stroll a memorable journey through  
Japan's"}]
```

```
print(summary[0]["summary_text"])
```

```
Gion's Machiya wooden houses exuded an  
old-world charm that transports  
visitors back in time. The glow of  
lanterns lining the narrow streets add  
to the enchanted ambiance, making each  
stroll a memorable journey through  
Japan's
```

Up next

- Build on existing LLM knowledge
- Perform new tasks
- See how LLMs are built
- Fine-tune LLMs
- Evaluate LLM performance



Let's practice!

INTRODUCTION TO LLMS IN PYTHON

Using pre-trained LLMs

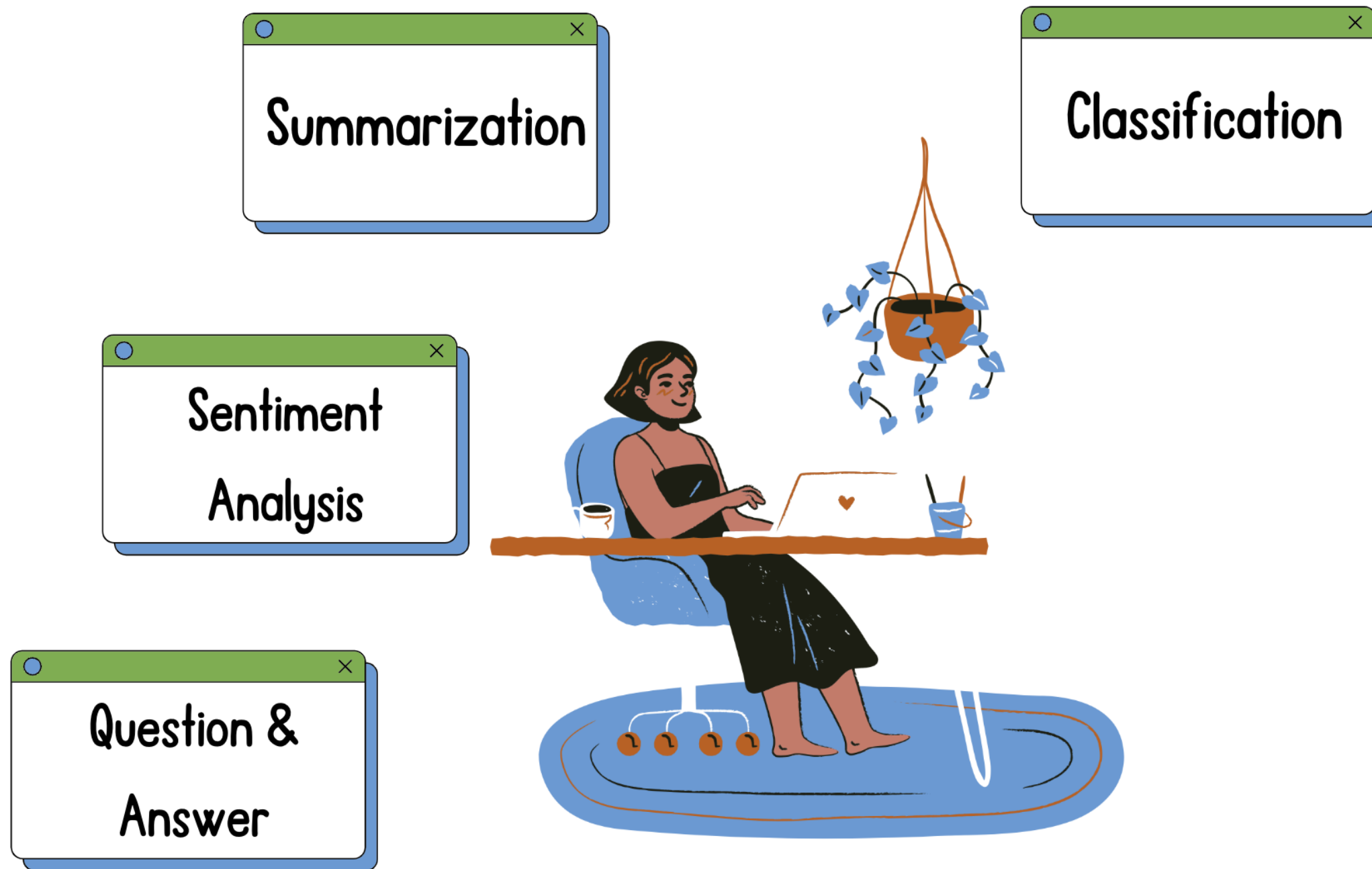
INTRODUCTION TO LLMS IN PYTHON



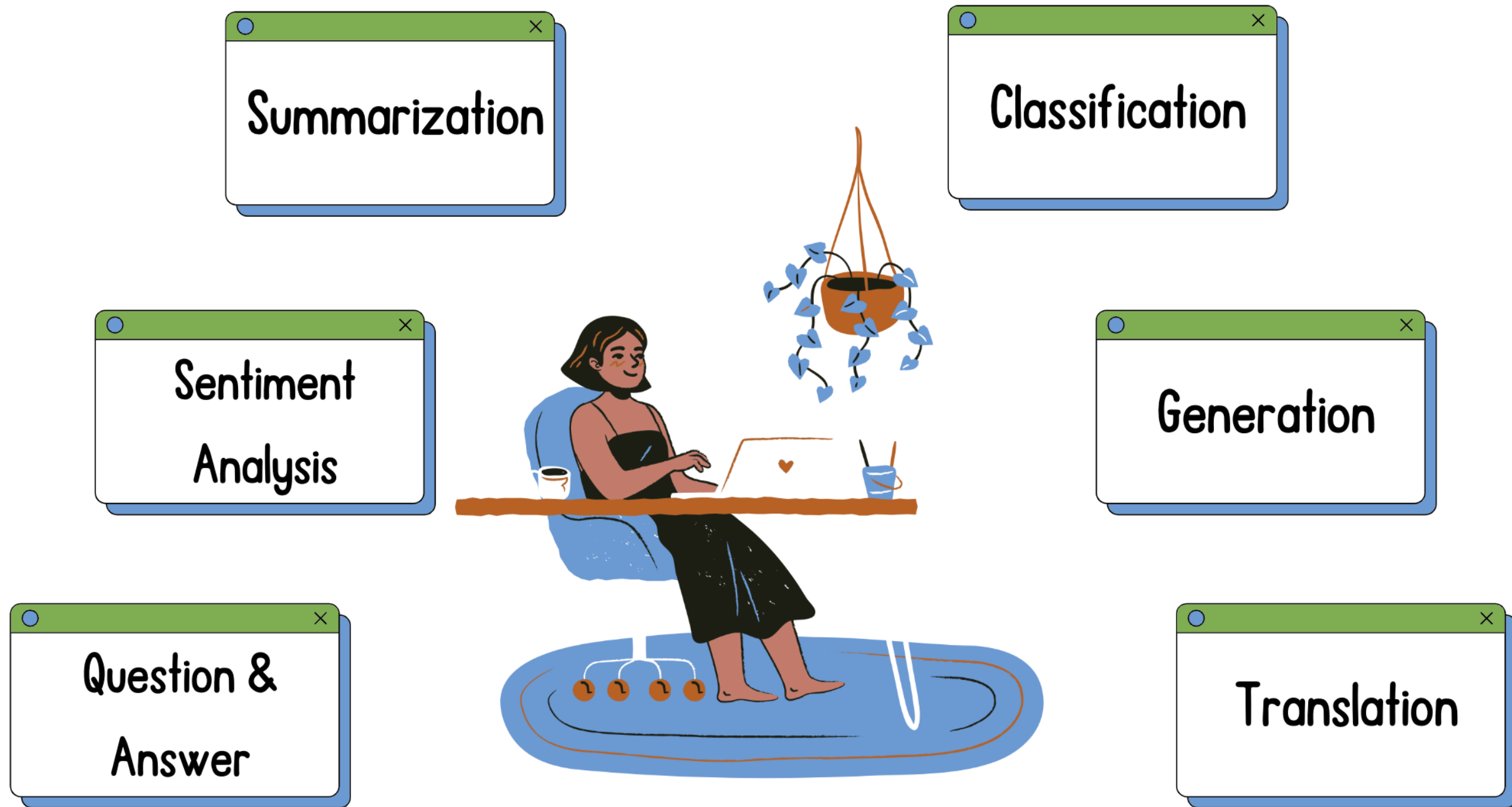
Jasmin Ludolf

Senior Data Science Content Developer,
DataCamp

Language understanding



Language generation



Text generation

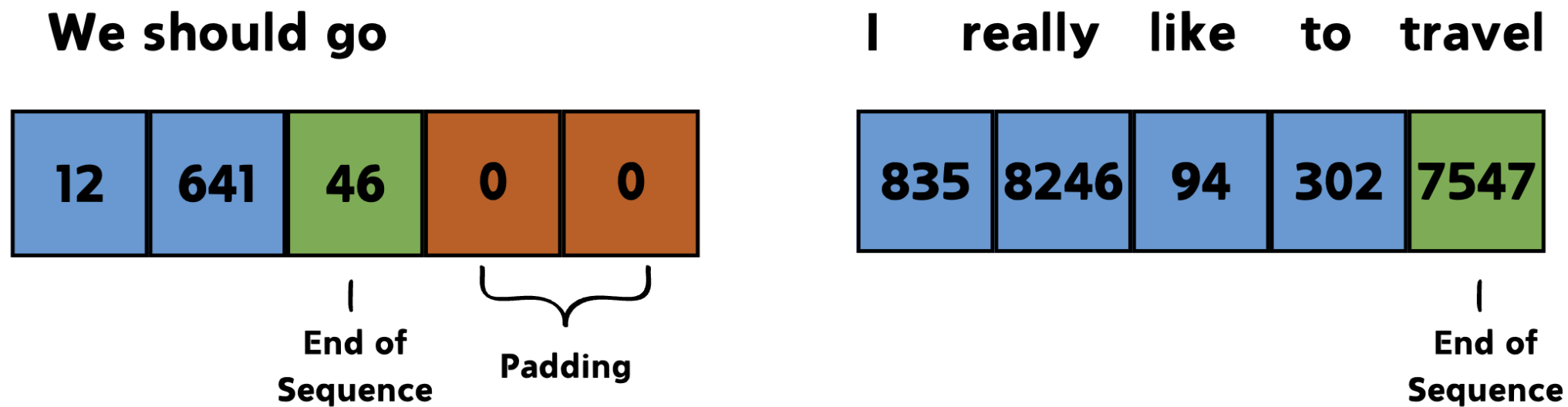
```
generator = pipeline(task="text-generation", model="distilgpt2")

prompt = "The Gion neighborhood in Kyoto is famous for"

output = generator(prompt, max_length=100, pad_token_id=generator.tokenizer.eos_token_id)
```

- Coherent
- Meaningful
- Human-like text
- `eos_token_id` : end-of-sequence token ID

Text generation



- `pad_token_id` : fills in extra space up to `max_length`
- **Padding**: adding tokens
- Setting to `generator.tokenizer.eos_token_id` marks the end of meaningful text, learned through training
- Model generates up to `max_length` or `pad_token_id`
- `truncation = True`

Text generation

```
generator = pipeline(task="text-generation", model="distilgpt2")

prompt = "The Gion neighborhood in Kyoto is famous for"

output = generator(prompt, max_length=100, pad_token_id=generator.tokenizer.eos_token_id)

print(output[0]["generated_text"])
```

```
The Gion neighborhood in Kyoto is famous for its many colorful green forests, such as the
Red Hill, the Red River and the Red River. The Gion neighborhood is home to the world's
tallest trees.
```

- Output may be suboptimal if prompt is vague

Guiding the output

```
generator = pipeline(task="text-generation", model="distilgpt2")

review = "This book was great. I enjoyed the plot twist in Chapter 10."

response = "Dear reader, thank you for your review."

prompt = f"Book review:\n{review}\n\nBook shop response to the review:\n{response}"

output = generator(prompt, max_length=100, pad_token_id=generator.tokenizer.eos_token_id)

print(output[0]["generated_text"])
```

```
Dear reader, thank you for your review. We'd like to thank you for your reading!
```

Language translation

- Hugging Face has a complete list of translation tasks and models

```
translator = pipeline(task="translation_en_to_es", model="Helsinki-NLP/opus-mt-en-es")

text = "Walking amid Gion's Machiya wooden houses was a mesmerizing experience."

output = translator(text, clean_up_tokenization_spaces=True)

print(output[0]["translation_text"])
```

Caminar entre las casas de madera Machiya de Gion fue una experiencia fascinante.

Let's practice!

INTRODUCTION TO LLMS IN PYTHON

Understanding the transformer

INTRODUCTION TO LLMS IN PYTHON

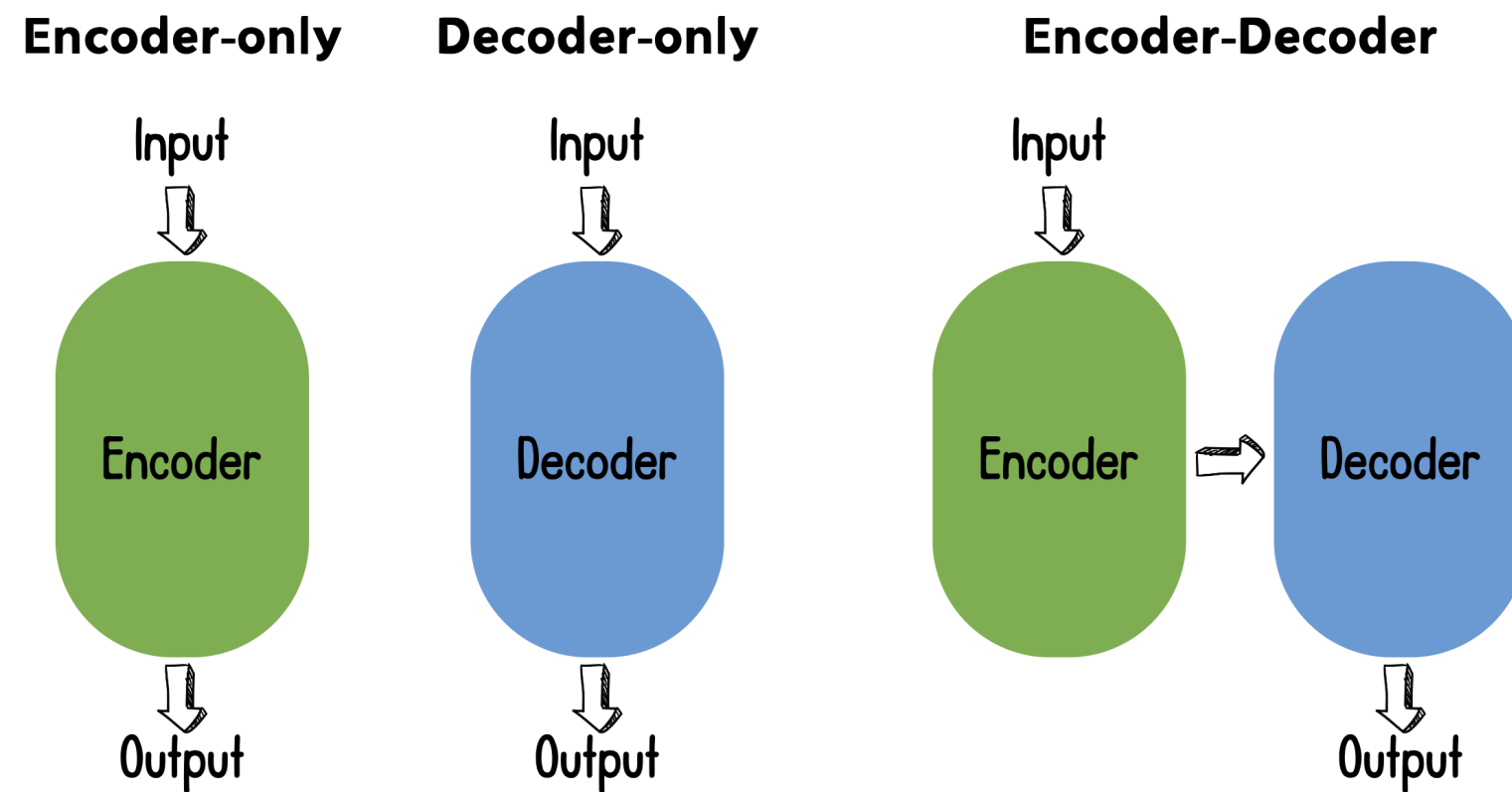


Jasmin Ludolf

Senior Data Science Content Developer,
DataCamp

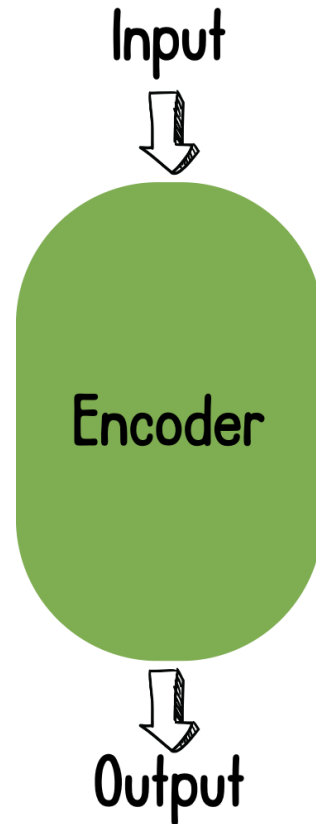
What is a transformer?

- Deep learning architectures
- Processing, understanding, and generating text
- Used in most LLMs
- Handle long text sequences in parallel

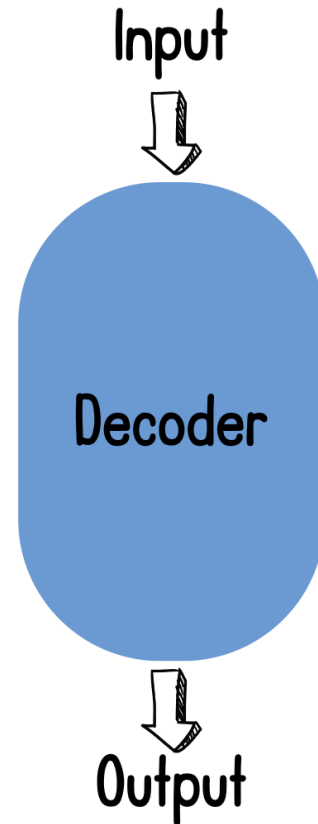


Transformer architectures

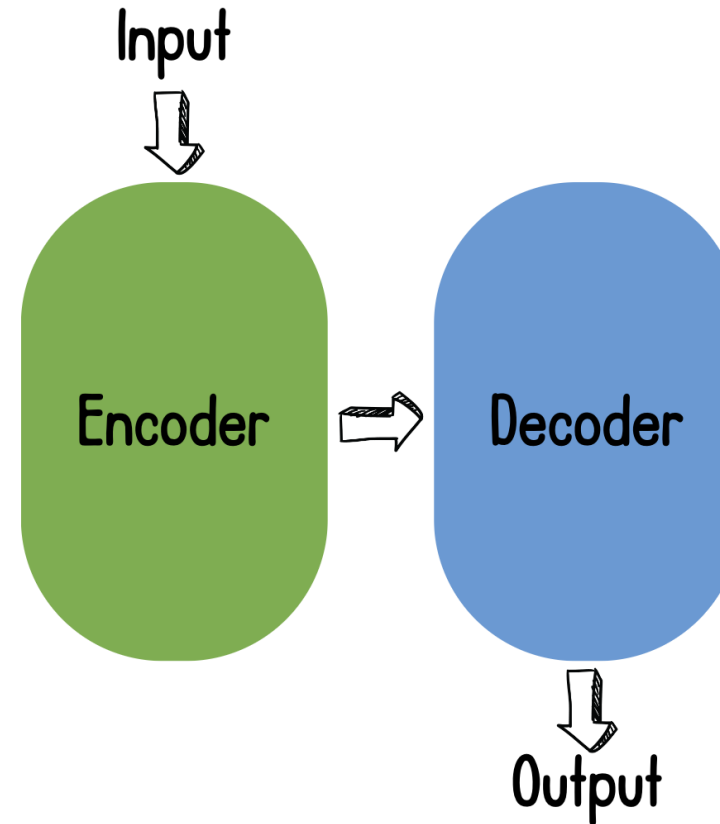
Encoder-only



Decoder-only

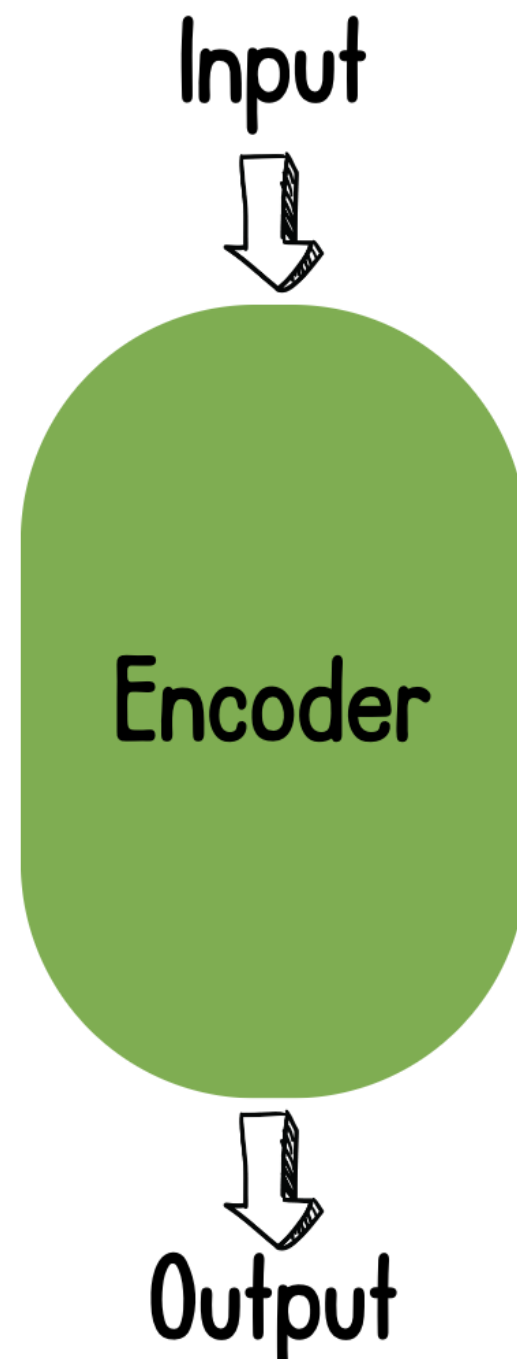


Encoder-Decoder



- Find the architecture details in the Hugging Face model card

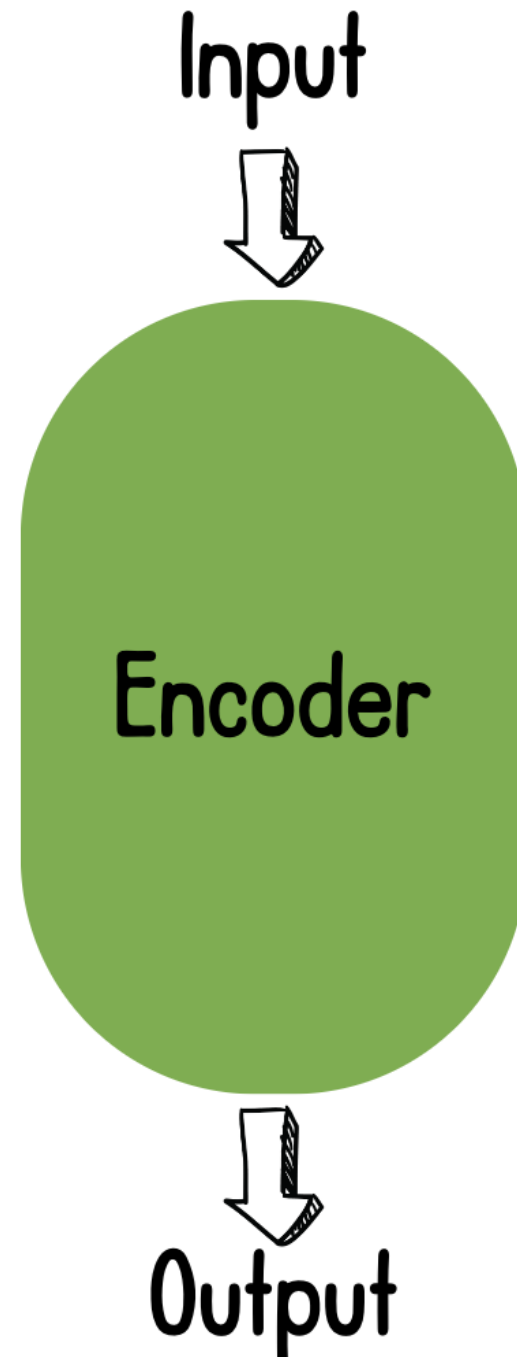
Encoder-only



- Understanding the input text
- No sequential output
- Common tasks:
 - Text classification
 - Sentiment analysis
 - Extractive question-answering (extract or label)
- BERT models
- Example:

```
"distilbert-base-uncased-distilled-squad"
```

Encoder-only



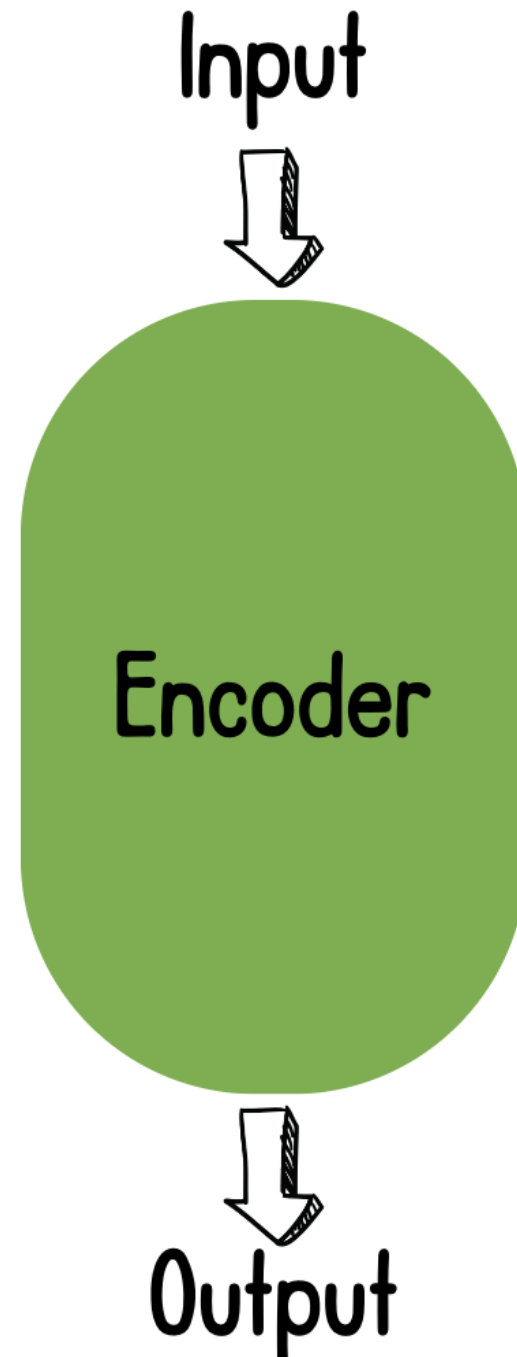
```
llm = pipeline(model="bert-base-uncased")  
print(llm.model)
```

```
BertForMaskedLM(  
  (bert): ...  
)  
  (encoder): BertEncoder(  
    ...
```

```
print(llm.model.config)
```

```
BertConfig {  
  ...  
  "architectures": [  
    "BertForMaskedLM"  
  ]  
  ...
```

Encoder-only



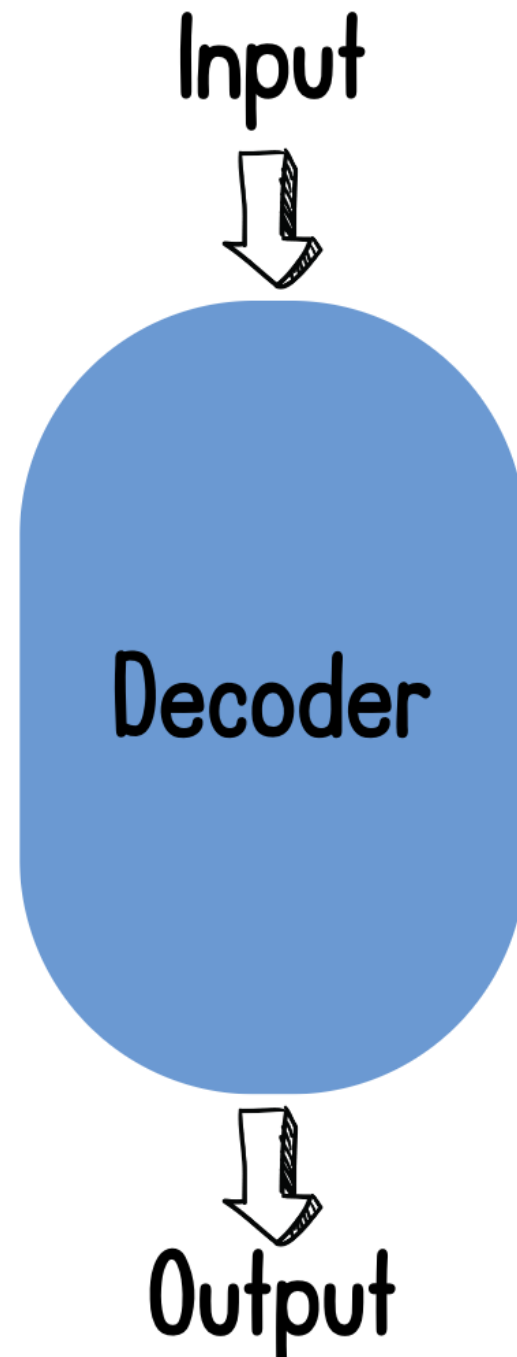
```
print(llm.model.config.is_decoder)
```

False

- Alternatively:

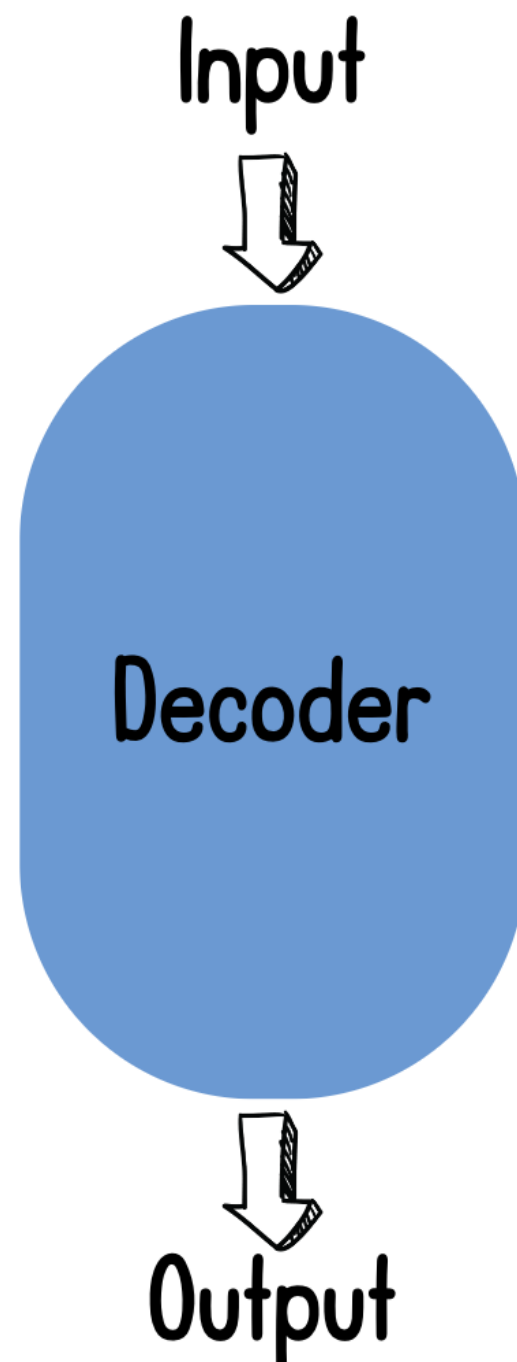
```
llm.model.config.is_encoder_decoder
```

Decoder-only



- Focus shifts to output
- Common tasks:
 - Text generation
 - Generative question-answering (sentence(s) or paragraph(s))
- GPT models
- Example: `"gpt-3.5-turbo"`

Decoder-only



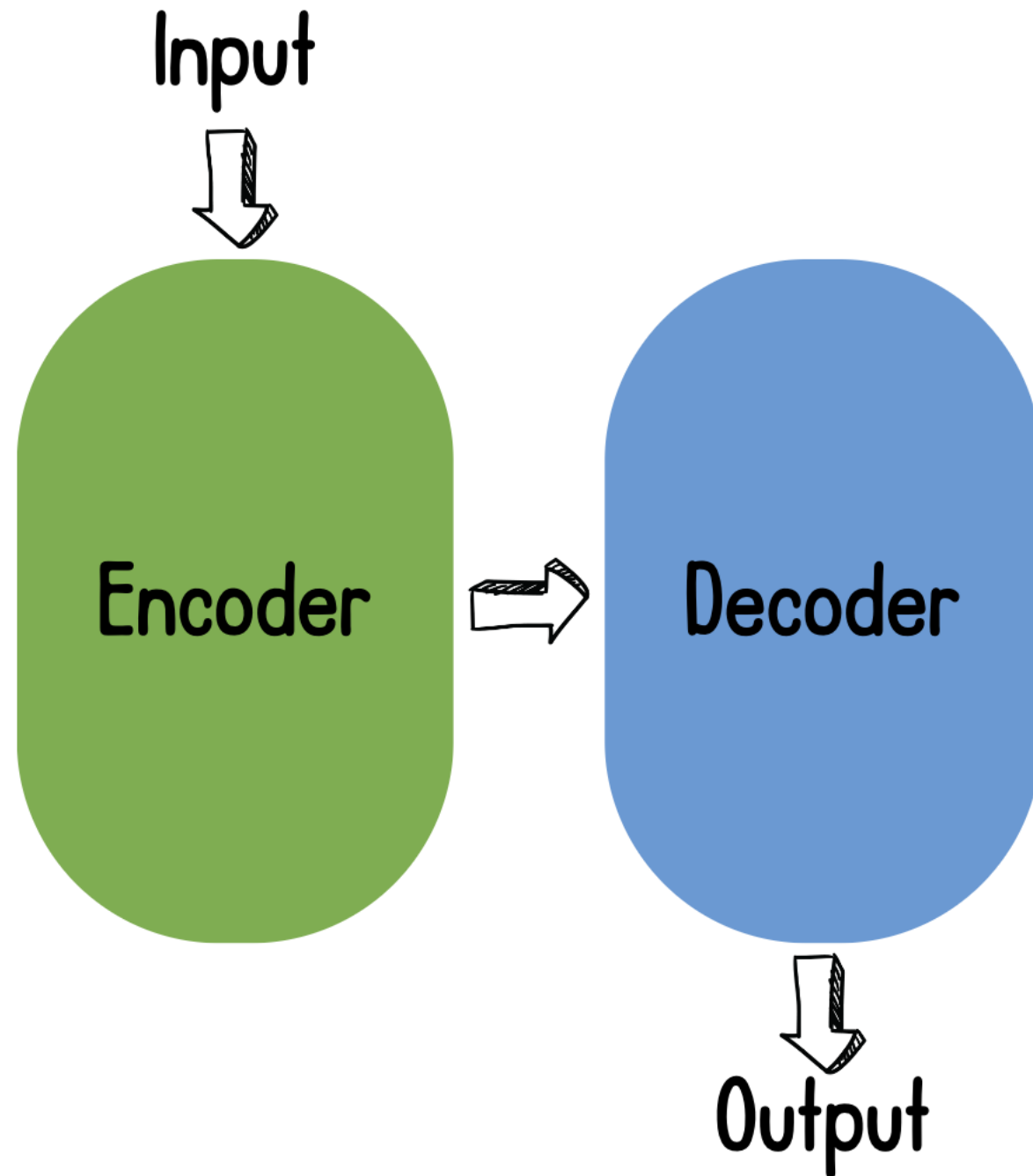
```
llm = pipeline(model="gpt2")  
print(llm.model.config)
```

```
GPT2Config {  
  ...  
  "architectures": [  
    "GPT2LMHeadModel"  
  ],  
  ...  
  "task_specific_params": {  
    "text-generation": {  
      ...  
    }  
  }  
}
```

```
print(llm.model.config.is_decoder)
```

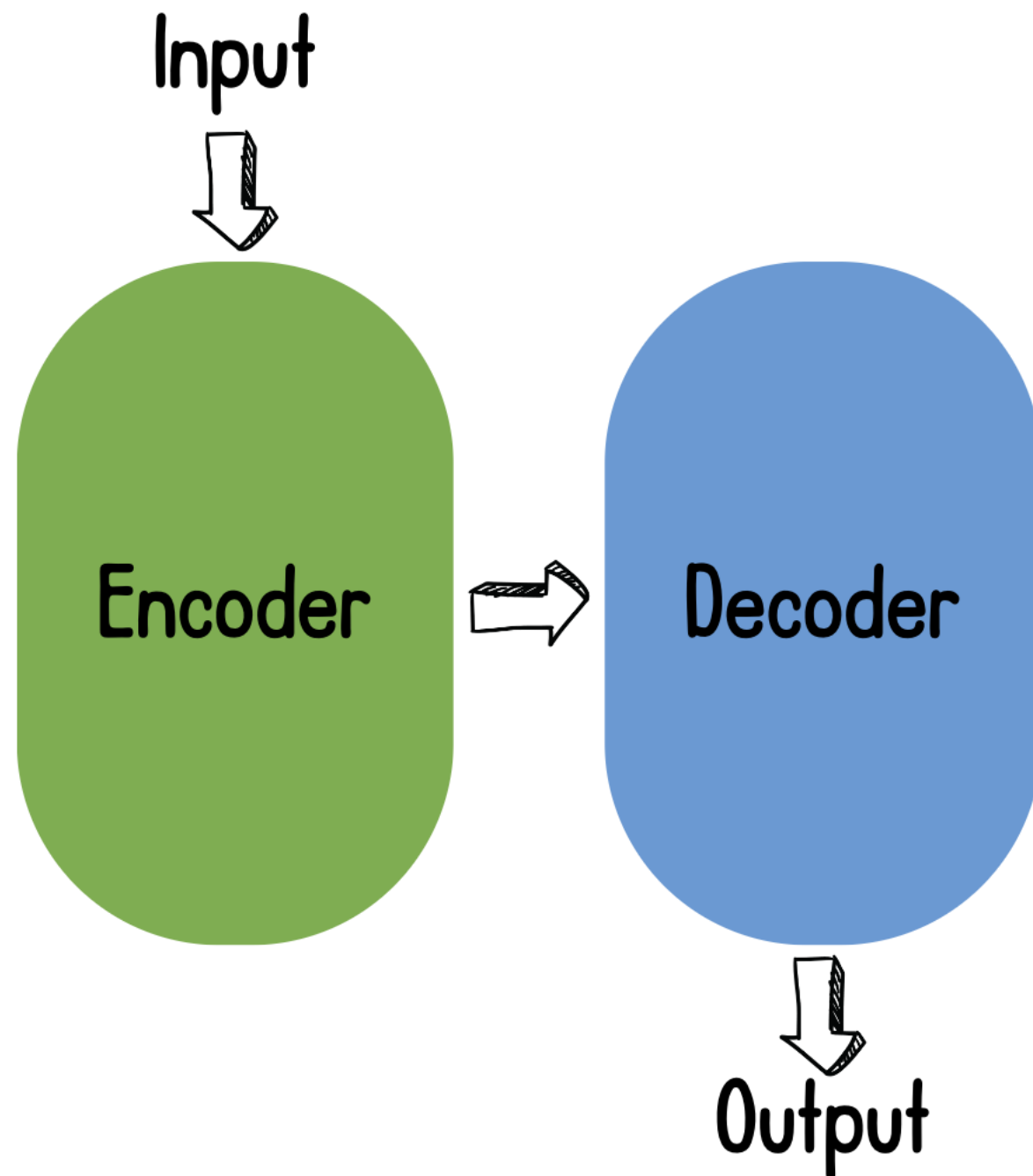
```
False
```

Encoder-decoder



- Understand and process the input and output
- Common tasks:
 - Translation
 - Summarization
- T5, BART models

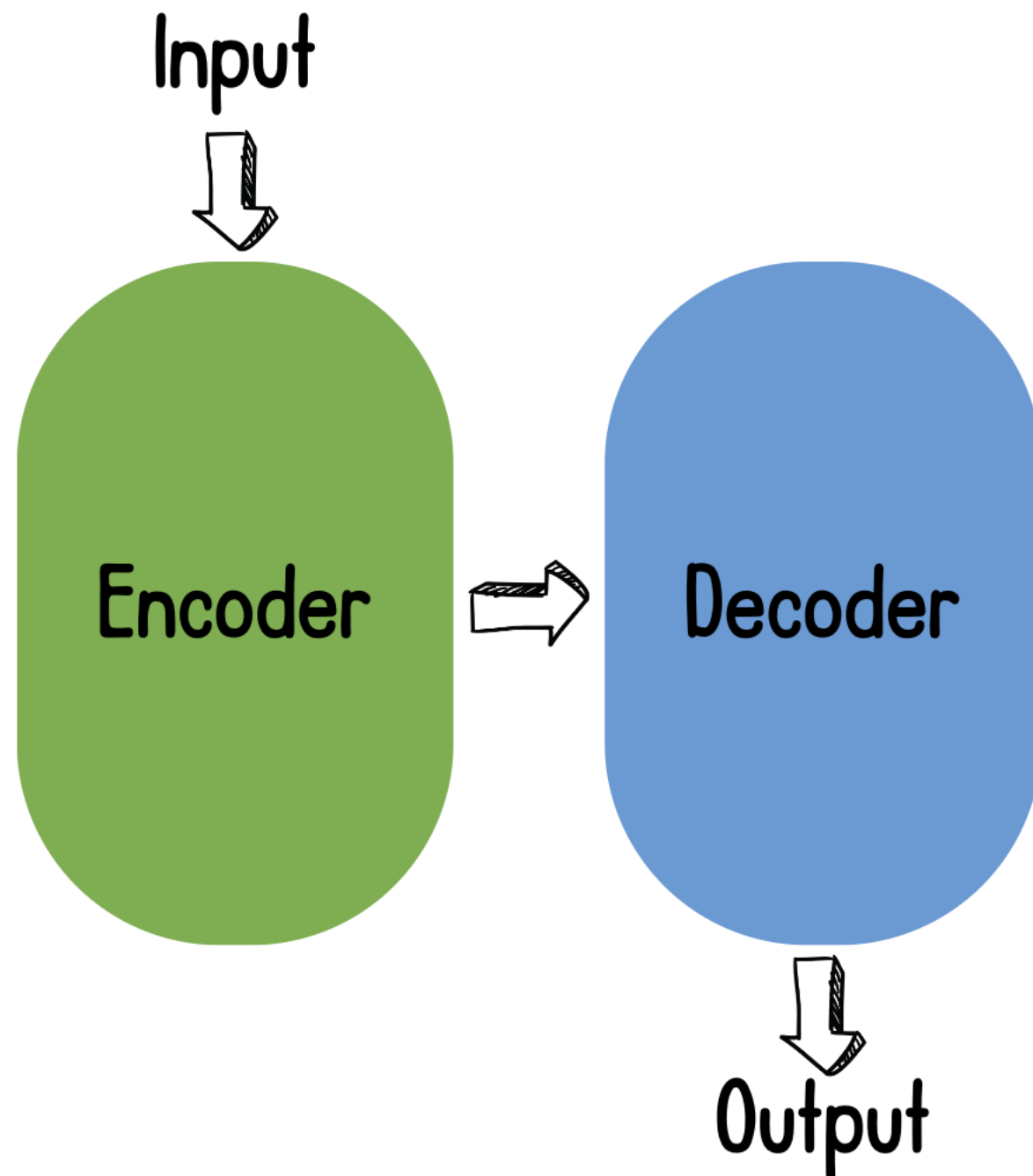
Encoder-decoder



```
llm = pipeline(model="Helsinki-NLP/opus-mt-es-en")  
print(llm.model)
```

```
MarianMTModel(  
...  
    (encoder): MarianEncoder(  
...  
    (decoder): MarianDecoder(  
...  
)
```

Encoder-decoder



```
print(llm.model.config)
```

```
MarianConfig {  
...  
  "decoder_attention_heads": 8,  
...  
  "encoder_attention_heads": 8,  
...  
  "is_encoder_decoder": true,  
...  
}
```

```
print(llm.model.config.is_encoder_decoder)
```

```
True
```

Let's practice!

INTRODUCTION TO LLMS IN PYTHON