Receives a 'task' and a 'language' as CLI arguments

Uses ChatGPT to generate a code snippet then prints it

Yes, we are doing the PDF project, but later.

Terminal

```
> python main.py \
    --task 'print numbers 1 to 10' \
    --language python
```

Things We'll Need

Python 3 install

OpenAI + LangChain packages installed
pip install langchain openai

OpenAI API key so we can access ChatGPT programatically

Directions to get this key will be in a text lecture right after this one

Code editor + terminal.
I'm using VSCode, you can use anything

LangChain

AlephAlpha	OpenAI
Beam	GPT4All
MosaicML	OctoAIEndpoint
PipelineAI	TextGen

LangChain has a *ton* of classes that wrap up popular text generation models

Almost all interchangeable

Almost all used the exact same way

AlephAlpha

```python
from langchain.llms import AlephAlpha

llm = AlephAlpha(
    model="luminous-extended",
    maximum_tokens=20,
    stop_sequences=["Q:"],
    aleph_alpha_api_key=ALEPH_ALPHA_API_KEY,
)

result = llm('What day comes after Tuesday?')

print(result) # 'Wednesday'
```

TextGen

```python
from langchain.llms import TextGen

llm = TextGen(
    model_url=model_url
)

result = llm('What day comes after Tuesday?')

print(result) # 'Wednesday'
```

GPT4All

```python
from langchain.llms import GPT4All

llm = GPT4All(
    model='./models/model.bin',
    backend="gptj",
    callbacks=[StreamingStdOutCallbackHandler()],
    verbose=True
)

result = llm('What day comes after Tuesday?')

print(result) # 'Wednesday'
```

OpenAI

```python
from langchain.llms import OpenAI

llm = OpenAI(
    openai_api_key='....'
)

result = llm('What day comes after Tuesday?')

print(result) # 'Wednesday'
```

Lots of Loaded Terminology

Large Language Model (LLM)	GPT-3
Completion Model	GPT-4
Chat Model	ChatGPT

Even if you know the difference,
LangChain introduces a little confusion...

Large Language Model (LLM)

An algorithm for generating text based upon a prompt

Predicts the next likely bit of text by looking at the preceding text

Large Language Model

The early bird catches the

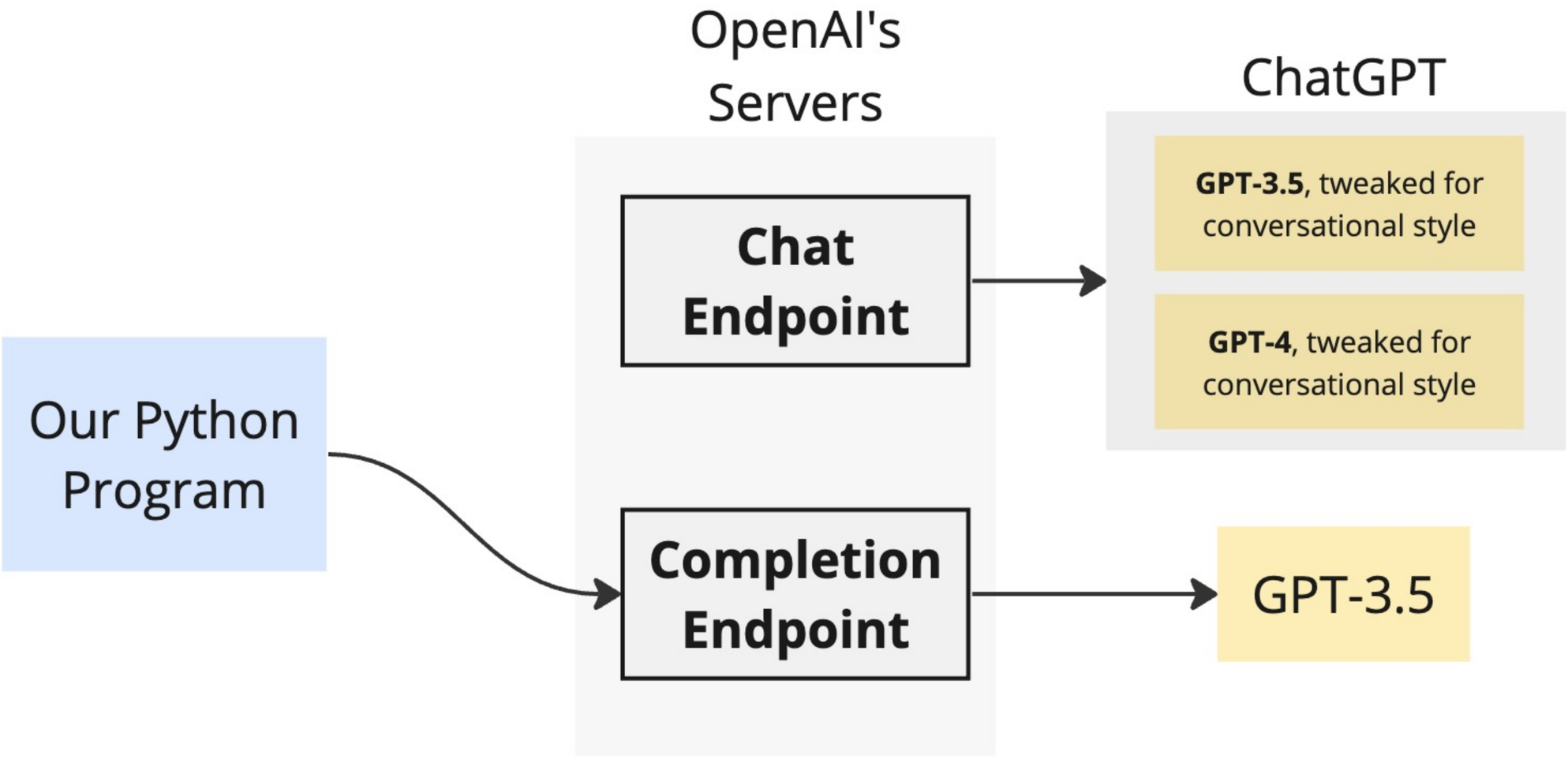
Completion Model

A large language model

Chat Model

A LLM that is customized to generate text in a conversational format

Still, fundamentally, a LLM.


OpenAI's Servers
ChatGPT
Chat Endpoint
GPT-3.5, tweaked for conversational style
GPT-4, tweaked for conversational style
Our Python Program
Completion Endpoint
GPT-3.5

LLM's + LangChain

LangChain refers to **completion**-based models as '**LLM's**

LangChain refers to **chat**-based models as **'Chat Models'**

LLM's + LangChain

```
1   from langchain.llms import OpenAI
2
3   llm = OpenAI()
```

*Gives us an object that will use OpenAI's **completion** endpoint*

```
1   from langchain.chat_models import ChatOpenAI
2
3   chat = ChatOpenAI()
```

*Gives us an object that will use OpenAI's **chat** endpoint*

LLM's + LangChain

Many parts of LangChain assume you are using a Completion-Based LLM

At present, OpenAI's ChatGPT is extremely good + low cost. Using it with LangChain means we have to do a little extra work

Custom LLM's are more likely to be completion based. LangChain might be focused on a future where we use our own specialized models, rather than ChatGPT