In [1]:
```
import os
import openai

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
openai.api_key = os.environ['OPENAI_API_KEY']
```

In [29]:
```
#llm we will be using
llm_model= 'gpt-3.5-turbo-0301'
```

LangChain Output Parsers

We get a string in response of the input we give to the LLM, In case if you want a structured output like output in the form of Json or any other structured format you can use output parsers.

Formating Output without Parsers:

Lets say we have a Customer Review, and we want to get information in form of Json.

In [30]:
```
customer_review = """\
This leaf blower is pretty amazing.  It has four settings:\
candle blower, gentle breeze, windy city, and tornado. \
It arrived in two days, just in time for my wife's \
anniversary present. \
I think my wife liked it so much she was speechless. \
So far I've been the only one using it, and I've been \
using it every other morning to clear the leaves on our lawn. \
It's slightly more expensive than the other leaf blowers \
out there, but I think it's worth it for the extra features.
"""
```

Our output should look like.

In [31]:
```
{
  "gift": False,
  "delivery_days": 5,
  "price_value": "pretty affordable!"
}
```

```
{'gift': False, 'delivery_days': 5, 'price_value': 'pretty affordable!'}
```

Lets define Prompt String Template that we will use to make a Prompt.

In [32]:
```
review_template = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? \
Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the product \
to arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,\
and output them as a comma separated Python list.

Format the output as JSON with the following keys:
gift
delivery_days
price_value

text: {text}
"""
```

Making a Prompt template:

In [33]:
```
from langchain.prompts import ChatPromptTemplate

prompt_template = ChatPromptTemplate.from_template(review_template)
print(prompt_template)
```

```
input_variables=['text'] output_parser=None partial_variables={} messages=[Hu
manMessagePromptTemplate(prompt=PromptTemplate(input_variables=['text'], outp
ut_parser=None, partial_variables={}, template='For the following text, extra
ct the following information:\n\ngift: Was the item purchased as a gift for s
omeone else? Answer True if yes, False if not or unknown.\n\ndelivery_days: H
ow many days did it take for the product to arrive? If this information is no
t found, output -1.\n\nprice_value: Extract any sentences about the value or
price,and output them as a comma separated Python list.\n\nFormat the output
as JSON with the following keys:\ngift\ndelivery_days\nprice_value\n\ntext:
{text}\n', template_format='f-string', validate_template=True), additional_kw
args={})]
```

Making the message by combining review and prompt template.

In [34]:
```
message= prompt_template.format_messages(text= customer_review)
```

In [35]:
```
from langchain.chat_models import ChatOpenAI
chat= ChatOpenAI(temperature= 0.0, model= llm_model)
```

In [36]:
```
#passing message to chat model of Langchain that is our llm in back of the s
response= chat(message)
print(response.content)
```

```
{
    "gift": true,
    "delivery_days": 2,
    "price_value": ["It's slightly more expensive than the other leaf blowers
out there, but I think it's worth it for the extra features."]
}
```

Now if you check the data type of response:

In [39]:
```
print(type(response.content))
```

```
<class 'str'>
```

As you can see it string and if you apply json/python dictionary function, it will throw you an error.

In [40]:
```
response.content.get('gift')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[40], line 1
----> 1 response.content.get('gift')

AttributeError: 'str' object has no attribute 'get'
```

Now here is when we use parsers.

Formating Output with Parser:

In [41]:
```
from langchain.output_parsers import ResponseSchema
from langchain.output_parsers import StructuredOutputParser
```

So first step is to make Response Schema. This will show how you want your output to be and what will be included in it.

In [43]:
```
#about gift---name of key(in thi case) and what will be included (value in t
gift_schema = ResponseSchema(name="gift",
                             description="Was the item purchased\
                             as a gift for someone else? \
                             Answer True if yes,\
                             False if not or unknown.")

#about delivery days
delivery_days_schema = ResponseSchema(name="delivery_days",
                                      description="How many days\
                                      did it take for the product\
                                      to arrive? If this \
                                      information is not found,\
                                      output -1.")

#about price value
price_value_schema = ResponseSchema(name="price_value",
                                    description="Extract any\
                                    sentences about the value or \
                                    price, and output them as a \
                                    comma separated Python list.")

#combining all in a list to make a main response schema
response_schemas = [gift_schema,
                    delivery_days_schema,
                    price_value_schema]
```

Making output Parser with out schema defined above.

In [44]:
```
output_parser = StructuredOutputParser.from_response_schemas(response_schema
```

format_instruction() will return a string of the Instruction from the Schema.

In [45]:
```
format_instructions = output_parser.get_format_instructions()
```

In [46]:
```
print(format_instructions)
```

```
The output should be a markdown code snippet formatted in the following schem
a, including the leading and trailing "\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item purchased
as a gift for someone else?                                 Answer True if yes,
False if not or unknown.
        "delivery_days": string  // How many days
did it take for the product                                          to arrive? I
f this                                        information is not found,
output -1.
        "price_value": string  // Extract any
sentences about the value or                                        price, and o
utput them as a                                          comma separated Python li
st.
}
```
```

Now in our new Prompt Template we will have the Text of Review and this string returned by format_instructions()

In [47]:
```
review_template_2 = """\
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? \
Answer True if yes, False if not or unknown.

delivery_days: How many days did it take for the product\
to arrive? If this information is not found, output -1.

price_value: Extract any sentences about the value or price,\
and output them as a comma separated Python list.

text: {text}

{format_instructions}
"""

#making prompt template
prompt = ChatPromptTemplate.from_template(template=review_template_2)

#making message
messages = prompt.format_messages(text=customer_review,
                                  format_instructions=format_instructions)
```

In [48]:
```
print(messages[0].content)
```

```
For the following text, extract the following information:

gift: Was the item purchased as a gift for someone else? Answer True if yes,
False if not or unknown.

delivery_days: How many days did it take for the productto arrive? If this in
formation is not found, output -1.

price_value: Extract any sentences about the value or price,and output them a
s a comma separated Python list.

text: This leaf blower is pretty amazing.  It has four settings:candle blowe
r, gentle breeze, windy city, and tornado. It arrived in two days, just in ti
me for my wife's anniversary present. I think my wife liked it so much she wa
s speechless. So far I've been the only one using it, and I've been using it
every other morning to clear the leaves on our lawn. It's slightly more expen
sive than the other leaf blowers out there, but I think it's worth it for the
extra features.


The output should be a markdown code snippet formatted in the following schem
a, including the leading and trailing "\`\`\`json" and "\`\`\`":

```json
{
        "gift": string  // Was the item purchased
as a gift for someone else?                                 Answer True if yes,
False if not or unknown.
        "delivery_days": string  // How many days
did it take for the product                                          to arrive? I
f this                                          information is not found,
output -1.
        "price_value": string  // Extract any
sentences about the value or                                         price, and o
utput them as a                                       comma separated Python li
st.
}
```
```

In [49]:
```
response = chat(messages)
```

In [50]:
```
print(response.content)
```

```
```json
{
        "gift": true,
        "delivery_days": "2",
        "price_value": ["It's slightly more expensive than the other leaf blo
wers out there, but I think it's worth it for the extra features."]
}
```
```

In [51]:
```
print(type(response.content))
```

```
<class 'str'>
```

Using Output Parser:

In [52]:
```
output_dict = output_parser.parse(response.content)
```

In [53]:
```
print(output_dict)
```

```
{'gift': True, 'delivery_days': '2', 'price_value': ["It's slightly more expe
nsive than the other leaf blowers out there, but I think it's worth it for th
e extra features."]}
```

In [54]:
```
type(output_dict)
```

```
dict
```

In [55]:
```
output_dict.get('delivery_days')
```

```
'2'
```

Now you can see we have got out output in the form of python dict.