```
!pip install PyPDF2 langchain langchain-google-genai google-generativeai
```

```
  Downloading google_generativeai-0.3.0-py3-none-any.whl.metadata (5.8 kB)
  Downloading google_generativeai-0.2.2-py3-none-any.whl.metadata (3.1 kB)
  Downloading google_generativeai-0.2.1-py3-none-any.whl.metadata (3.1 kB)
  Downloading google_generativeai-0.2.0-py3-none-any.whl.metadata (3.1 kB)
  Downloading google_generativeai-0.1.0-py3-none-any.whl.metadata (3.0 kB)
Collecting langchain-google-genai
  Downloading langchain_google_genai-2.1.4-py3-none-any.whl.metadata (5.2 kB)
  Downloading langchain_google_genai-2.1.3-py3-none-any.whl.metadata (4.7 kB)
  Downloading langchain_google_genai-2.1.2-py3-none-any.whl.metadata (4.7 kB)
  Downloading langchain_google_genai-2.1.1-py3-none-any.whl.metadata (4.7 kB)
  Downloading langchain_google_genai-2.1.0-py3-none-any.whl.metadata (3.6 kB)
  Downloading langchain_google_genai-2.0.11-py3-none-any.whl.metadata (3.6 kB)
  Downloading langchain_google_genai-2.0.10-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in /usr/local/lib/python3.11/dist-packages (from google-generativ
Requirement already satisfied: google-api-core in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.25.0)
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.171.
Requirement already satisfied: google-auth>=2.15.0 in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (2.38.0)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (5.29.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (4.67.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from google-generativeai) (4.14.0)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from google-ai-generativelang
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /usr/local/lib/python3.11/dist-packages (from google-api-co
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from google-auth>=2.15.0->google-ge
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth>=2.15.0->google-gen
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packages (from google-auth>=2.15.0->google-generativea
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.11/dist-packages (from langchain-core<1.0.0,
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/dist-packages (from langchain-core<1.0.0,>=0.3.58->l
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.11/dist-packages (from langchain-core<1.0.0,>=0.3.58->la
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from langsmith<0.4,>=0.1.17->langchain) (
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.11/dist-packages (from langsmith<0.4,>=0.1.17->langcha
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from langsmith<0.4,>=0.1.1
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from langsmith<0.4,>=0.1.17->lan
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3.0.0,>=2.7.4->langch
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3.0.0,>=2.7.4->langcha
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3.0.0,>=2.7.4->lang
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2->langchain) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2->langchain) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2->langchain) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2->langchain) (2025.4
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages (from SQLAlchemy<3,>=1.4->langchain) (3.2.2)
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in /usr/local/lib/python3.11/dist-packages (from google-api-python-client->goo
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from google-api-python-
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from google-api-python-client->googl
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in /usr/local/lib/python3.11/dist-packages (from google-api-core[grpc]!=2.0.*,!=
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in /usr/local/lib/python3.11/dist-packages (from google-api-core[grpc]!=2
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.11/dist-packages (from h
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->langsmith<0.4,>=0.1.17->langc
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->langsmith<0.4,>=0.1.1
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->langsmith<
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.11/dist-packages (from jsonpatch<2.0,>=1.33->langchain-core
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules>=0.2.1->google-au
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio->httpx<1,>=0.23.0->langsmith<0.4,>
Downloading langchain_google_genai-2.0.10-py3-none-any.whl (41 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.0/42.0 kB 3.0 MB/s eta 0:00:00
Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Installing collected packages: filetype, langchain-google-genai
Successfully installed filetype-1.2.0 langchain-google-genai-2.0.10
```

```
import os
os.environ["GOOGLE_API_KEY"] = "AIzaSyDyISlBjQ_oYINPse2p-b7mZFSz8o0QhEw"  # Replace with your Gemini API key
```

```
import PyPDF2

def extract_text_from_pdf(pdf_path):
    with open(pdf_path, "rb") as file:
        reader = PyPDF2.PdfReader(file)
        text = ""
        for page in reader.pages:
            text += page.extract_text()
    return text

# Upload PDF
from google.colab import files
uploaded = files.upload()
pdf_path = list(uploaded.keys())[0]
```

```
# Extract content
study_material = extract_text_from_pdf(pdf_path)
print(study_material[:1000])  # Preview
```

```
Choose files Prompt Engineering.pdf
• Prompt Engineering.pdf(application/pdf) - 1549480 bytes, last modified: 13/06/2025 - 100% done
Saving Prompt Engineering.pdf to Prompt Engineering (1).pdf
Prompt Engineering
Slides by Elvis Saravia https://www.promptingguide.ai/  and images
from other sourcesAgenda
•Introduction  to Prompt  Engineering
•Advanced  Techniques  for Prompt  Engineering
•Conclusion  & Future DirectionsRise of In -context Learning
Brown, Tom B. et al. "Language Models are Few -Shot Learners." ArXiv  abs/2005.14165 (2020): n. pag.What  are prompts?
•Prompts  involve  instructions  and context  passed  to a
language  model to achieve a desired task
•Prompt  engineering  is the practice  of developing  and
optimizing  prompts  to efficiently  use language  models
(LMs)  for a variety  of applications
•Prompt  engineering is  a useful  skill for  AI engineers  and
researchers  to improve  and efficiently  use language  models
What is prompt engineering?
Prompt engineering  is a process of creating  a set of prompts,
or questions,  that are used to guide the user toward a desired
outcome.  It is an effective  tool for designers  to create user
```

```
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash", temperature=0.3)
```

```
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain

summary_prompt = PromptTemplate(
    input_variables=["text"],
    template="Summarize the following study material into concise bullet points:\n\n{text}"
)

summary_chain = LLMChain(llm=llm, prompt=summary_prompt)
summary = summary_chain.run(study_material)
print("Summary:\n", summary)
```

```
<ipython-input-17-1483531208>:9: LangChainDeprecationWarning: The class `LLMChain` was deprecated in LangChain 0.1.17 and will be remove
  summary_chain = LLMChain(llm=llm, prompt=summary_prompt)
<ipython-input-17-1483531208>:10: LangChainDeprecationWarning: The method `Chain.run` was deprecated in langchain 0.1.0 and will be remo
  summary = summary_chain.run(study_material)
Summary:
 Here's a concise bullet point summary of the provided prompt engineering study material:

**I. Introduction to Prompt Engineering**

*   **Prompts:** Instructions and context given to a language model (LM) to achieve a task.
*   **Prompt Engineering:** Developing and optimizing prompts for efficient LM use.
*   **Importance:** Crucial for research, testing LM limitations, and enabling innovative applications.
*   **Decoding Parameters:**
    *   **Temperature:** Controls randomness (0-1). Lower = Sharper, more repetitive. Higher = More diverse.
    *   **Top P:** Selects tokens with cumulative probability exceeding p (0-1). Lower = More repetitive.
*   **Prompt Elements:** Instructions, context, input data, output indicator.
*   **Settings:** Temperature and Top_p affect determinism. Keep low for exact answers, high for diverse responses.

**II. Designing Prompts for Different Tasks**

*   **Common Tasks:** Text summarization, question answering, text classification, role playing, code generation, reasoning.
*   **Examples:** The slides provide examples of prompts for each task, demonstrating how to structure the prompt with context and instr

**III. Advanced Prompt Engineering Techniques**

*   **Few-Shot Prompts:** Provide examples in the prompt to guide the model.
*   **Chain-of-Thought (CoT) Prompting:** Instruct the model to reason step-by-step. Can be combined with few-shot or used in a zero-sho
*   **Self-Consistency:** Sample multiple reasoning paths using CoT and select the most consistent answer.
*   **Knowledge Generation Prompting:** Generate additional knowledge as part of the context to improve results.
*   **Program-Aided Language Model (PAL):** Uses an LLM to generate programs as intermediate reasoning steps, offloading the solution tc
*   **ReAct:** Interleaves reasoning traces and task-specific actions, allowing interaction with external tools.
*   **Directional Stimulus Prompting:** Uses a tuneable policy LM to generate hints that guide a black-box frozen LLM.
```

```
**IV. Risks**

*   **Prompt Injection:** Hijacking LM output by injecting untrusted commands.
*   **Prompt Leaking:** Forcing the model to reveal information about its own prompt.
*   **Jailbreaking:** Bypassing safety and moderation features.
```

```
question_prompt = PromptTemplate(
    input_variables=["summary"],
    template="""
From the following summary, create 3 multiple-choice quiz questions.
Each question must include 4 options (a-d) and clearly state the correct answer.

Summary:
{summary}
"""
)

question_chain = LLMChain(llm=llm, prompt=question_prompt)
questions = question_chain.run(summary)
print("Quiz Questions:\n", questions)
```

```
Quiz Questions:
 Here are three multiple-choice quiz questions based on the provided summary:

**Question 1:**

Which of the following best describes the purpose of "Temperature" as a decoding parameter in prompt engineering?

a) It defines the length of the generated text.
b) It controls the complexity of the language model.
c) It controls the randomness of the generated text.
d) It determines the number of examples used in few-shot prompting.

**Correct Answer: c) It controls the randomness of the generated text.**

**Question 2:**

Which advanced prompt engineering technique involves providing examples within the prompt to guide the language model?

a) Chain-of-Thought (CoT) Prompting
b) Self-Consistency
c) Few-Shot Prompts
d) Knowledge Generation Prompting

**Correct Answer: c) Few-Shot Prompts**

**Question 3:**

Which of the following is a risk associated with prompt engineering, where an attacker can inject commands to manipulate the language mc

a) Prompt Optimization
b) Prompt Injection
c) Knowledge Generation
d) Decoding Parameter Tuning

**Correct Answer: b) Prompt Injection**
```