Bucksmore Industry Day

Prompt Engineering Challenge

Contents

1. Setup

1.1 Requirements

- Python 3.x
- OpenAI API Key
- Internet Access
- Code Editor (or Jupyter Notebook), e.g.
    - Visual Studio Code
    - Google Colab

1.2 Setup

1.2.1 Install OpenAI Python Library

```
!pip install openai
```

1.2.2 Setup your API Key

```
from openai import OpenAI

client = OpenAI(
    api_key="OPENAI_API_KEY",
)
```

Note on API Key Security: *In real-world applications, you should not embed your API key directly into code as it can be accidentally exposed. Sensitive credentials should be stored in .env files or using environment variables so that they can be loaded securely. However, for the sake of simplicity in this activity, you may insert the API key directly into the code. The keys we use will be deleted following the activity.*

1.2.3 Example Usage

```
response = client.responses.create(
    model="gpt-4o-mini",
    instructions="You are a coding assistant that talks like a pirate.",
    input="How do I check if a Python object is an instance of a class?",
)

print(response.output_text)
```

1.3 Documentation

For detailed documentation on OpenAI API usage, please use:

- Overview - OpenAI API

- GitHub - openai/openai-python: The official Python library for the OpenAI API

For help with Python:

- Python Tutorial - Learn Python Programming Language - GeeksforGeeks
- Python Tutorial
- Welcome to Python.org

Setting up Python:

- How to Install Python on Your System: A Guide – Real Python

2. Challenge 1 – Rewrite for Specificity

Use prompts to generate a text output. The goal is to improve the clarity and specificity of your instructions, starting with a vague prompt and refining it step-by-step. Observe how changes to the prompt affect the quality and relevance of the output.

2.1 Task

- Write a Python function that sends the prompt(s) and displays or saves the output.
- Start with a very simple and very vague prompt such as "Write about AI".
- Rewrite it 5 times to be more specific each time. For example:
    - Write about AI
    - Explain how AI is used in daily life
    - Describe how AI is used in schools and classrooms
    - Write a paragraph about how AI helps teachers personalise learning

2.2 Optional

- Give the AI a role, for example "You are a teacher explaining AI to a classroom of 8-year-old students". This could be done within the instructions, or in the prompt (input) itself – try both and see if there is any difference.

3. Challenge 2 – Creative Writing

Use prompts to generate a 100-word short story. The subject can be anything you like, but some examples are:

- A story set on a space station where something goes wrong
- A robot learns to lie for the first time
- A police report from the perspective of a talking dog

3.1 Task

- Write a Python function that sends the prompt(s) and displays or saves the output.
- Experiment with tone and genre (e.g. horror, comedy).

3.2 Optional

- Add a parameter to control the word count (e.g. generate a 50-word or 150-word story).
- Include a twist ending in the story.
- Generate a story using only dialogue, with no narration.

4. Challenge 3 - Summarisation

Using a long block of text, for example a news report or a Wikipedia article, generate a concise summary of the main points. This should be no more than three sentences long.

For example, you could leverage a sample of the following articles:

- Why don't we trust technology in sport? - BBC Sport
- Large language model - Wikipedia
- We're looking at further online safety rules, says minister - BBC News

4.1 Task

- Write a Python function that sends the prompt(s) and displays or saves the output
- Limit the response to no more than five sentences, or less if possible.

4.2 Optional

- Generate different summaries for different target audiences, for example
    - Kids
    - Subject Experts
- Load a .txt file and summarise the text it contains.
- Use the Wikipedia Python library to load input text.

5. Challenge 4 – Generating Structured Outputs

Extract a structured output from an unstructured (free form) block of text.

5.1 Task 1 – Named Entity Recognition

Using the text provided (ner_sample_text.txt), extract **named entities** and classify them into specific types. A named entity is any real-world object such as a person, company, or place.

- Write a Python function that sends the prompt(s) and generates the output.
- Extract entities and their types from the text. Limit the extracted entities to the following types:
    - Person
    - Organisation
    - Location
- Store the results in a structured Python list or dictionary.

5.1.1 Optional

- Count the number of times each entity is mentioned in the text.
- Use a Wikipedia article as your input text.

5.2 Task 2 – Meeting Actions

Using the text provided (meeting_actions_sample.txt), extract tasks (to do items) and as appropriate. For each task, include the **task description**, and where appropriate, the **assigned person** and **due date**.

- Write a Python function that sends the prompt(s) and generates the output.
- Store the results in a structured Python list or dictionary.

5.2.1 Optional

- Allow the prompt to be **strict** (only extract definite tasks) or **lenient** (include vague suggestions like "someone should" or "maybe").
- Add a **priority ranking** for each task based on urgency or due date (e.g. high, medium, low).
- Generate sample meeting notes using a separate/different prompt. Use these meeting notes as the input text to your Python function.

