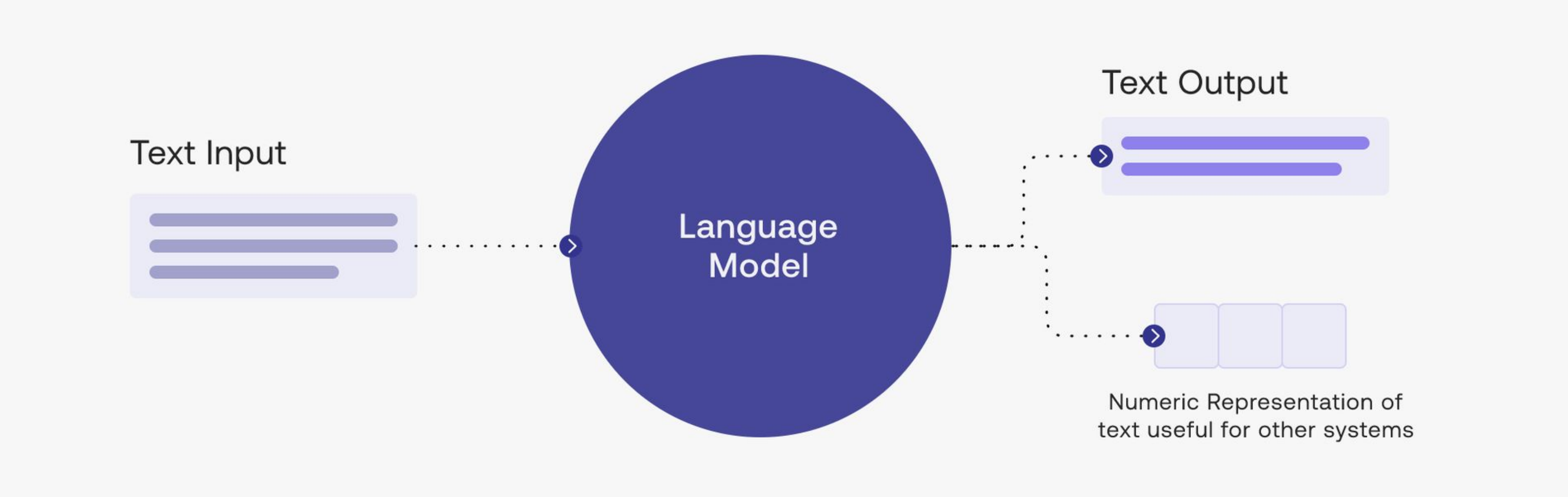
ChatGPT for Programmers

Diana PHOLO STONE

PREDICTIVE
INSIGHTS

Introduction

- LLMs offer unique opportunities to enhance educational experiences through language generation capabilities + vast knowledge base.
- Can simulate conversations creating dynamic and engaging interactions

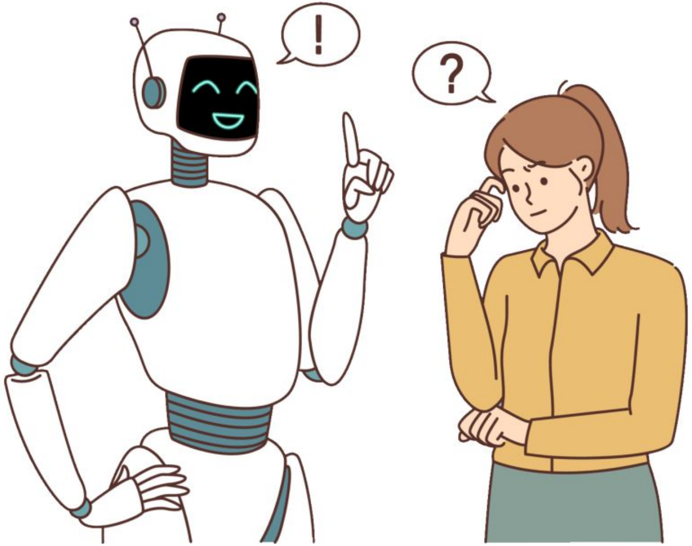


Introduction (cont.)

- LLMs are built on deep neural networks.
- These networks consist of layers of attention mechanisms
  - Attention Mechanism allows model to focus on specific words in sentence.
  - Captures relationships between words and context.
- LLMs are trained on massive text corpora
  - They learn grammar, facts, reasoning abilities, and more.

Effective Prompt Engineering

- Prompts = precise blueprints for output you expect from an AI model.
- Prompt engineering = crafting effective instructions / queries to guide language model's response.
  - Helps shape output by providing context and constraints



Effective Prompt Engineering (cont.)

- Improved Model Outputs: carefully crafted prompts can yield more accurate and relevant results
- Reduced Bias: Appropriate prompts can help mitigate biases in model outputs
- Personalised Learning: can address individual learning needs
- Critical Thinking Development
- Creative Expression

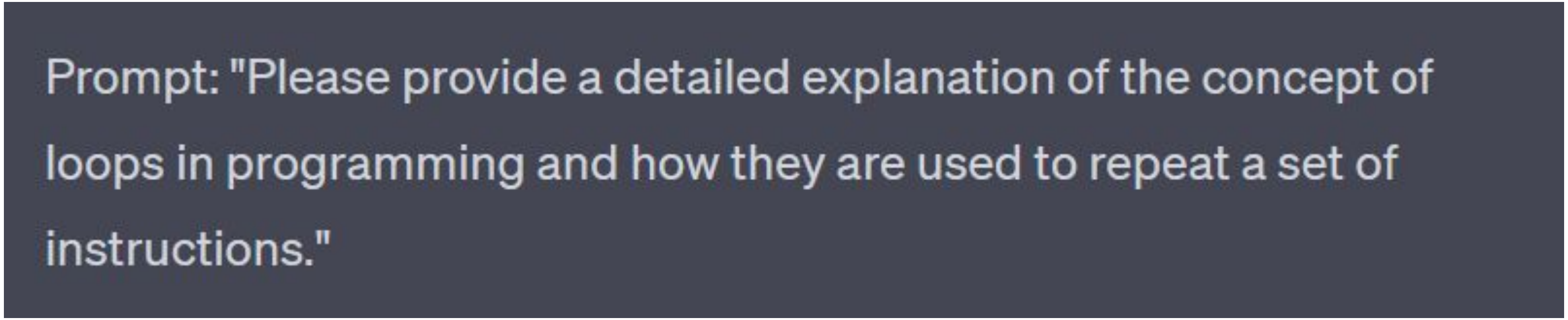
Types of prompts

- There are several types of prompts that can be used with language models like ChatGPT.
- Instruction Prompts
- Contextual Prompts
- Completion Prompts
- Question-Answer Prompts

Effective Prompting

Instruction Prompts : Descriptive Prompt

- Provides specific instructions that prompt ChatGPT to generate detailed explanation
- When to use it?
    - Concept Exploration
    - Elaboration and Contextualization
    - Instructional Support

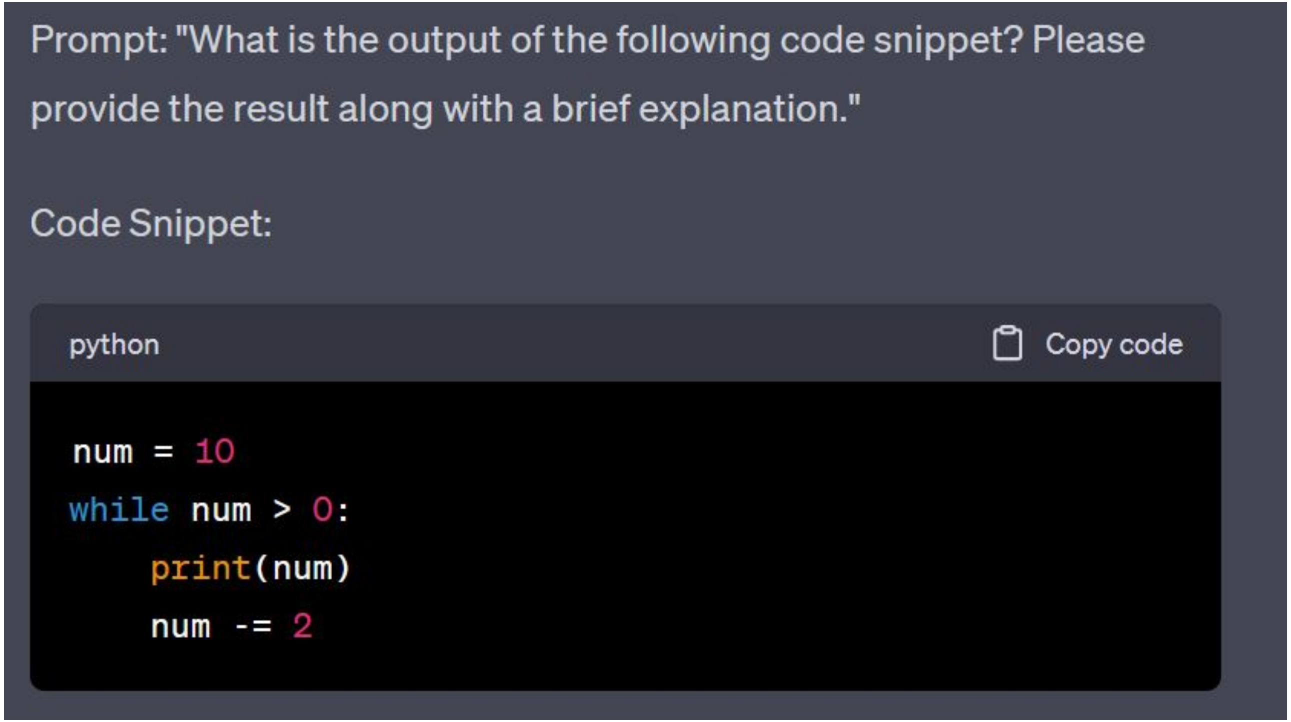


Prompt: "Please provide a detailed explanation of the concept of loops in programming and how they are used to repeat a set of instructions."

Instruction Prompts : Query Prompt

- Specific instruction or question designed to elicit a direct response or provide a specific piece of information
- When to use it?
    - Fact Retrieval
    - Quick Information Retrieval: data points, statistics, or definitions.
    - Comparative Analysis
    - Problem-Solving: guide the language model in solving specific problems or providing step-by-step solutions.

Example: Query Prompt - Fact Retrieval

- 

Prompt: "What is the output of the following code snippet? Please provide the result along with a brief explanation."

Code Snippet:

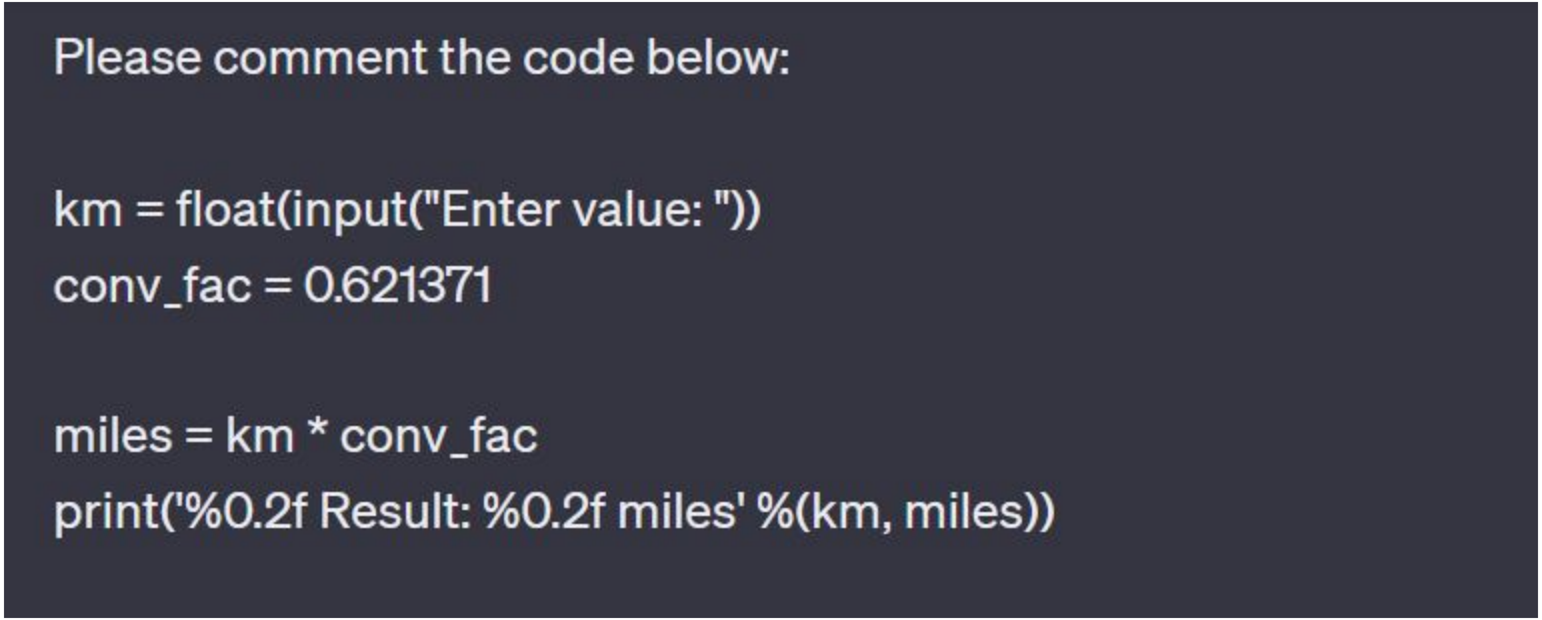
```python
num = 10
while num > 0:
    print(num)
    num -= 2
```

Instruction Prompts : Query Prompt (cont.)

Applications

- Debugging Assistance
- Language Syntax and Usage
- Code Generation
- Optimisation and Performance

Example: Commenting

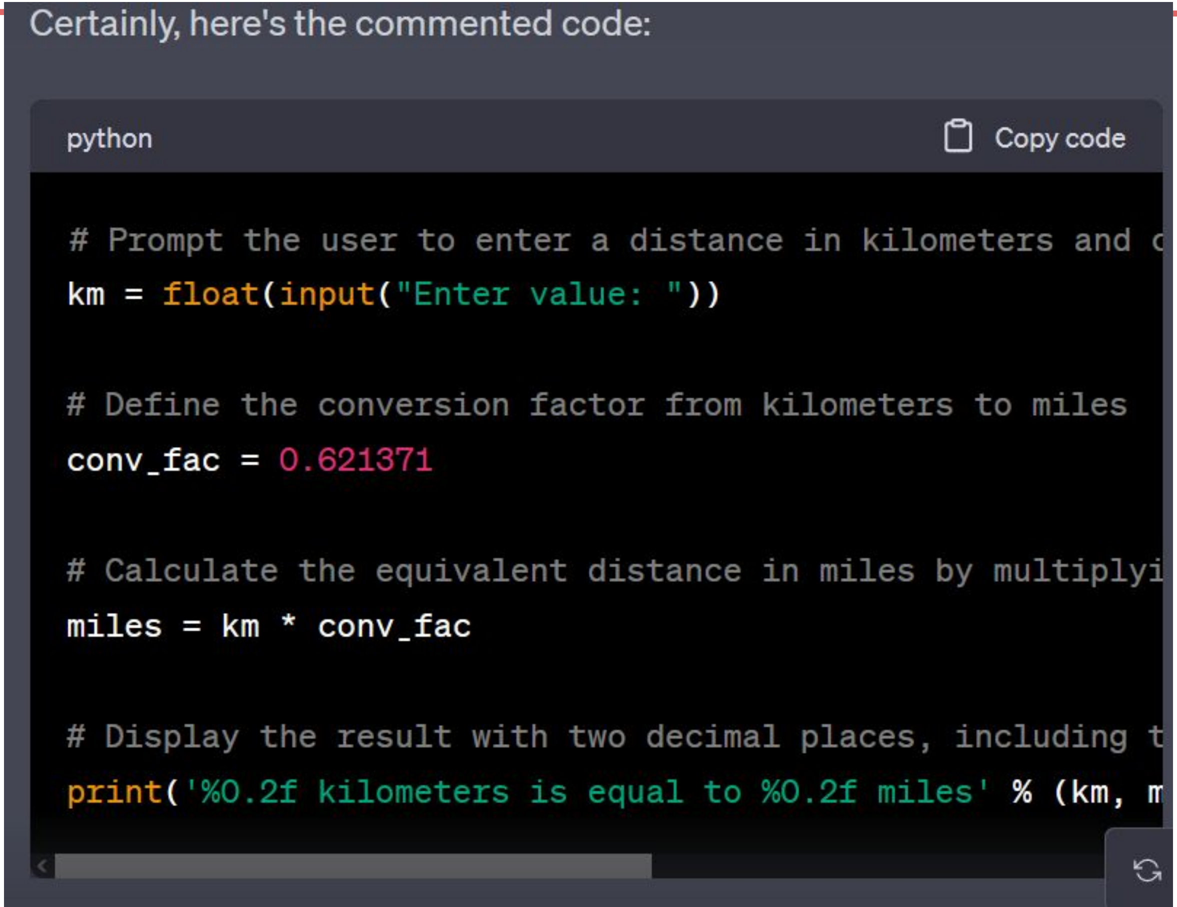


Please comment the code below:

```
km = float(input("Enter value: "))
conv_fac = 0.621371

miles = km * conv_fac
print('%0.2f Result: %0.2f miles' %(km, miles))
```

Example: Commenting (cont.)

-



Certainly, here's the commented code:

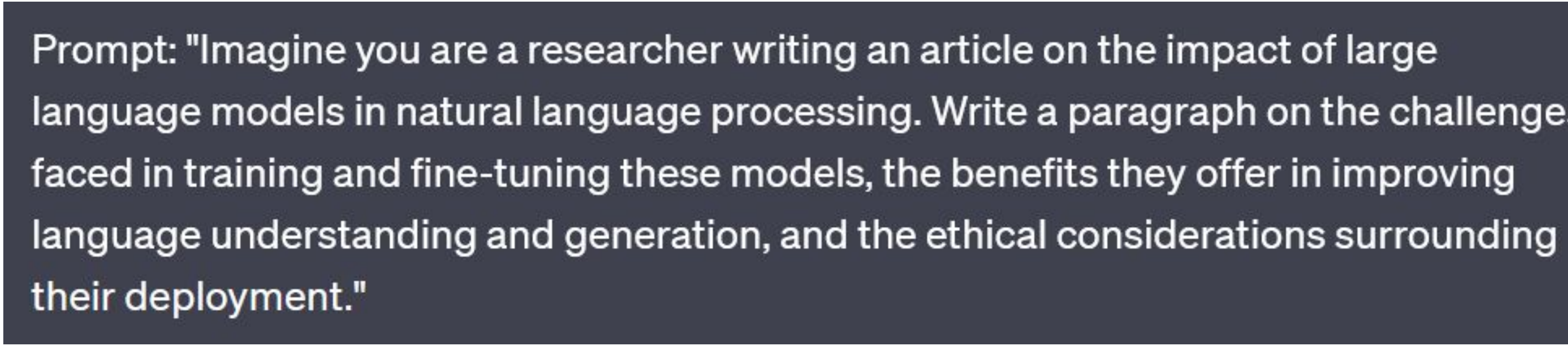
```python
# Prompt the user to enter a distance in kilometers and c
km = float(input("Enter value: "))

# Define the conversion factor from kilometers to miles
conv_fac = 0.621371

# Calculate the equivalent distance in miles by multiplyi
miles = km * conv_fac

# Display the result with two decimal places, including t
print('%0.2f kilometers is equal to %0.2f miles' % (km, m
```

Contextual prompts

- Query that sets a specific context or scenario for generating response
- Directs language model to generate text that aligns with given context
    - more focused and relevant outputs
- Can be used to stimulate creativity, generate imaginative narratives, or simulate conversations within setting

Contextual prompts (cont.)

Example

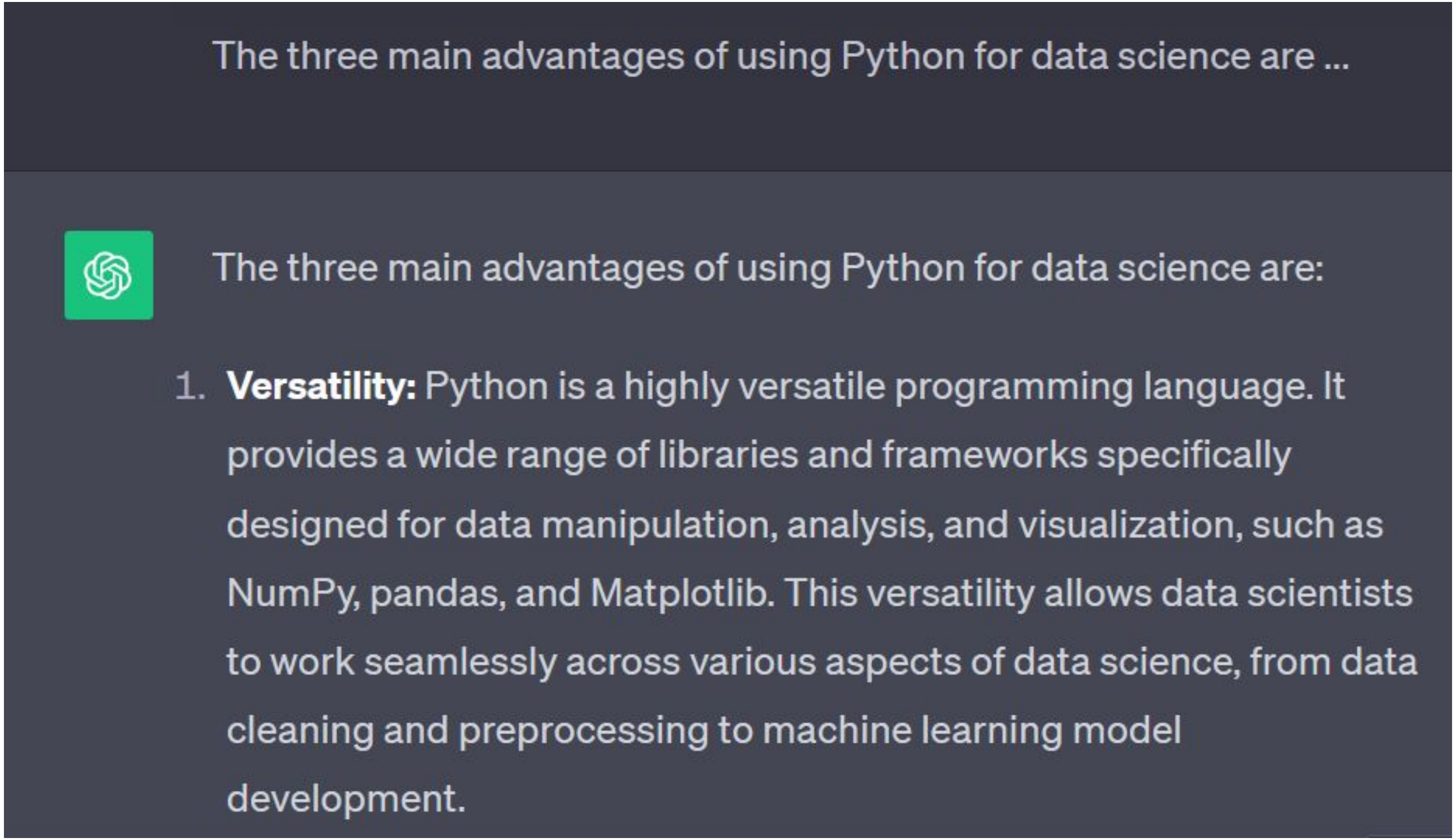


Prompt: "Imagine you are a researcher writing an article on the impact of large language models in natural language processing. Write a paragraph on the challenge faced in training and fine-tuning these models, the benefits they offer in improving language understanding and generation, and the ethical considerations surrounding their deployment."

Completion Prompts

- Involve providing the model with a partial sentence and asking it to complete it.
- Useful when you want the model to generate a continuation of a text or to provide missing information.
- Example:

Completion Prompts (cont.)

- 

The three main advantages of using Python for data science are ...

The three main advantages of using Python for data science are:

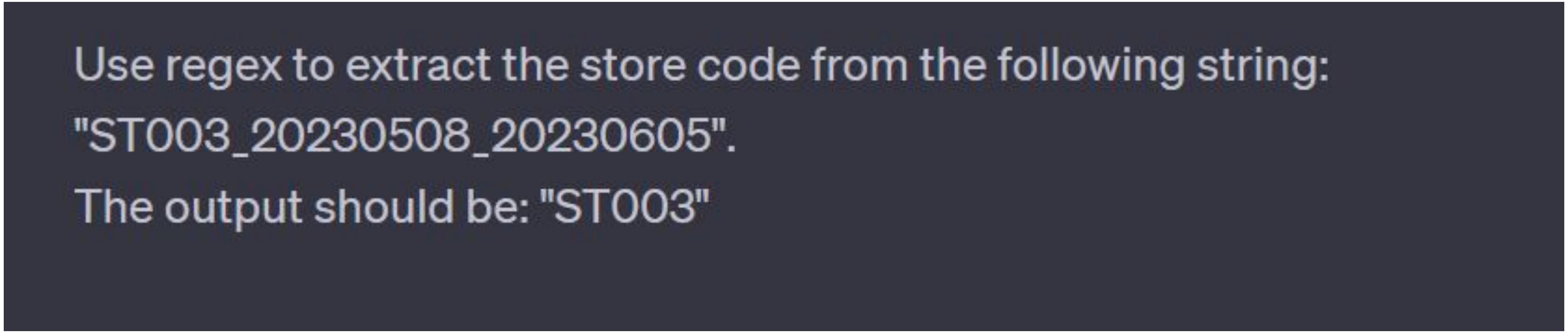
1. **Versatility:** Python is a highly versatile programming language. It provides a wide range of libraries and frameworks specifically designed for data manipulation, analysis, and visualization, such as NumPy, pandas, and Matplotlib. This versatility allows data scientists to work seamlessly across various aspects of data science, from data cleaning and preprocessing to machine learning model development.

Model Tuning

- Fine-Tuning = process of further training pre-trained LLM on specific tasks or specific datasets.
- Purpose: tailors LLM's general language capabilities to perform well on particular tasks.
- Importance: Makes LLMs adaptable and context-aware,
    - improving performance on specialized tasks.
- Example: If you want to use LLM for medical chatbot, fine-tuning it on medical dataset can help it understand medical jargon and context.

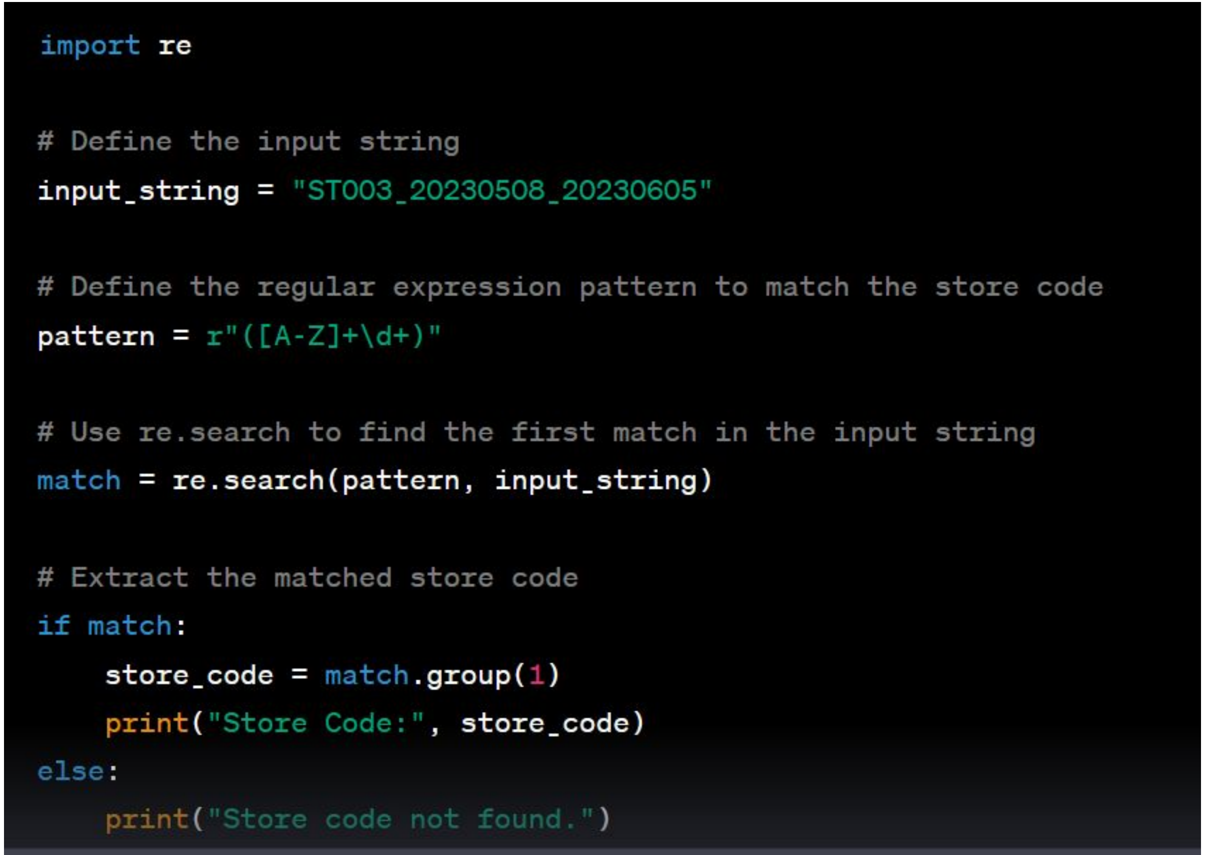
Model Tuning (cont.)

- Zero-shot prompt = simplest type of prompt.
  - Only provides description of task, or some text for GPT-3 to get started with
- One-shot prompt: provides one example that GPT-3 can use to learn how to best complete a task.



Use regex to extract the store code from the following string:
"ST003_20230508_20230605".
The output should be: "ST003"

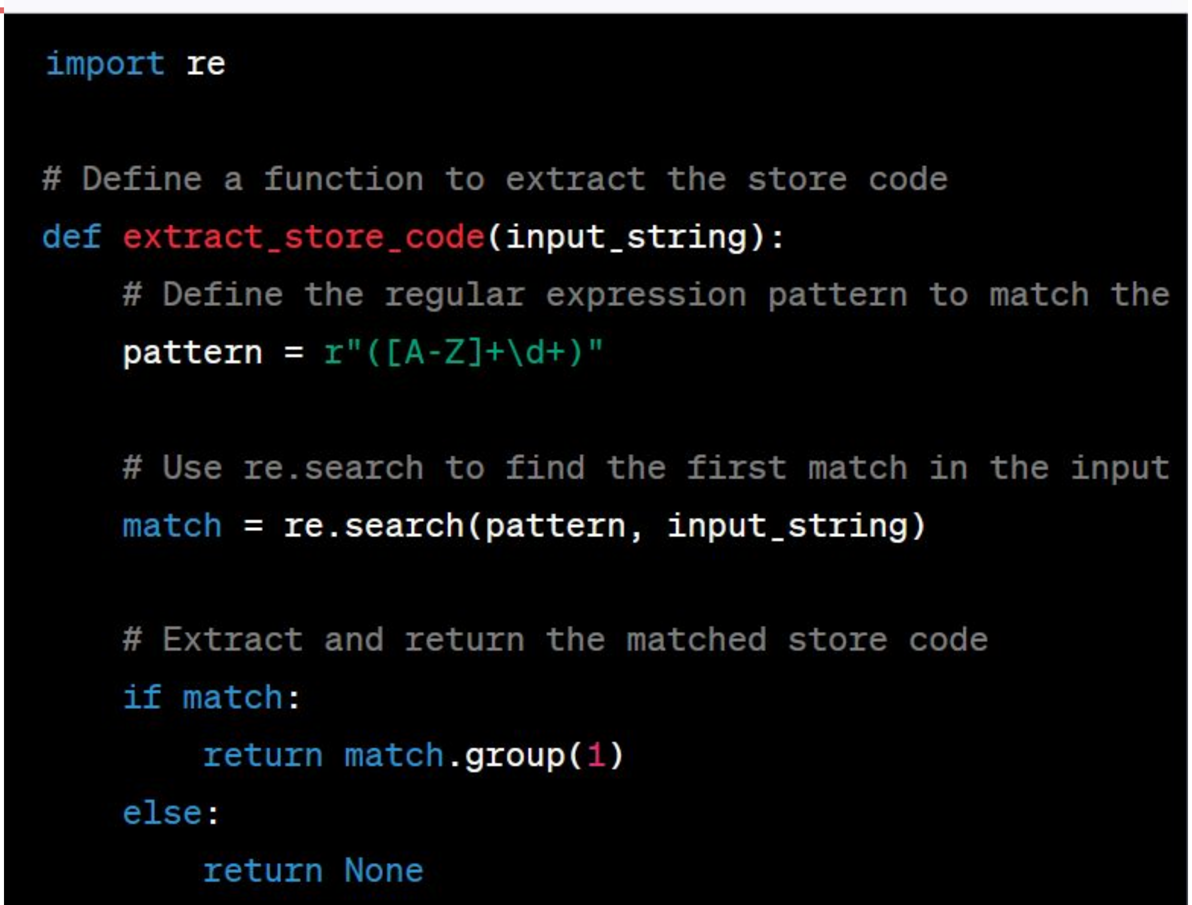
Model Tuning (cont.)



```
import re

# Define the input string
input_string = "ST003_20230508_20230605"

# Define the regular expression pattern to match the store code
pattern = r"([A-Z]+\d+)"

# Use re.search to find the first match in the input string
match = re.search(pattern, input_string)

# Extract the matched store code
if match:
    store_code = match.group(1)
    print("Store Code:", store_code)
else:
    print("Store code not found.")
```

Model Tuning (cont.)

- Few-shot prompt: Provides multiple examples



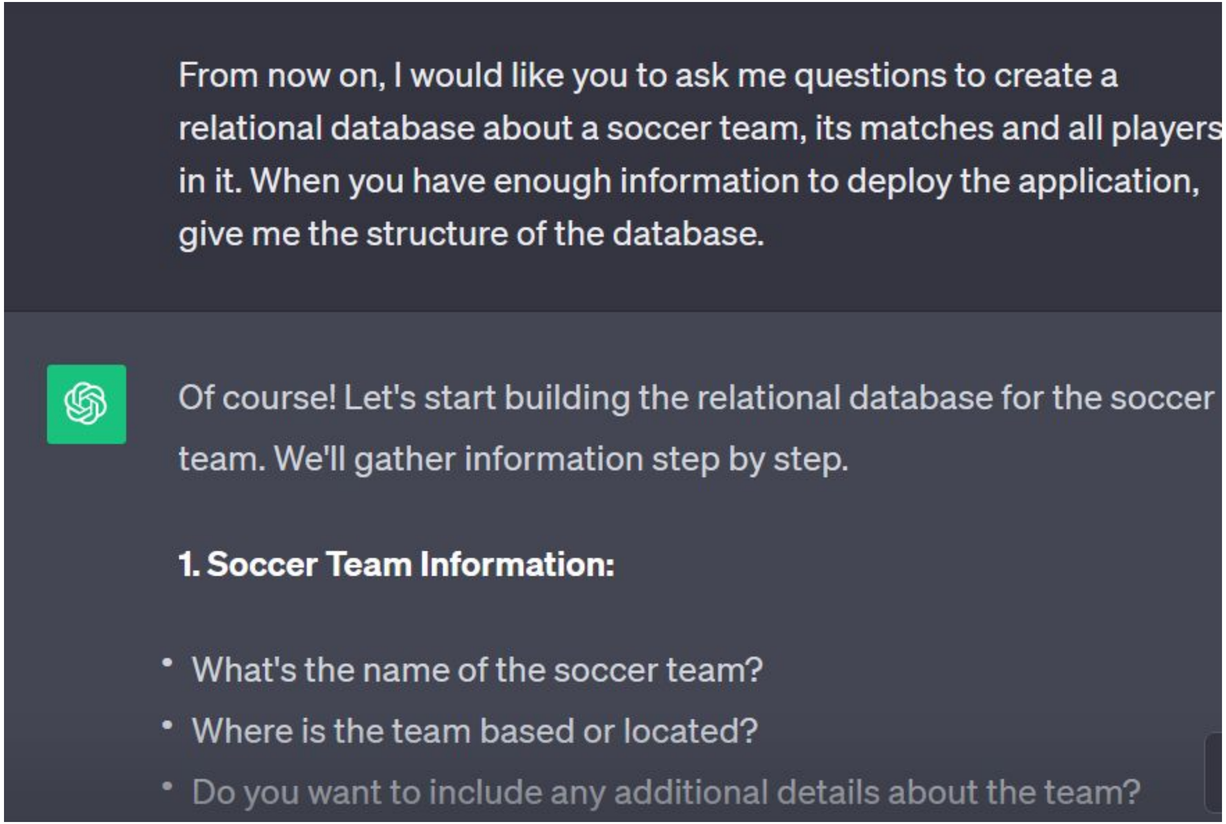
```
import re

# Define a function to extract the store code
def extract_store_code(input_string):
    # Define the regular expression pattern to match the
    pattern = r"([A-Z]+\d+)"

    # Use re.search to find the first match in the input
    match = re.search(pattern, input_string)

    # Extract and return the matched store code
    if match:
        return match.group(1)
    else:
        return None
```

Start Right!

- First prompt should have following structure:
- **Introduction**: Set up context
  - E.g. “Act as a Python software engineer.”
- **Task**
  - E.g. “You are to develop a CRUD application to manage a library.”
- **Contextual Information** about task
  - E.g. “It needs to be a web based application written in Python.”
- **Instructions**
  - E.g. “I want you to generate the code for this application, starting with the ‘create’ part.”
- **Input data**

Start Right! (Cont.)

- Example:



From now on, I would like you to ask me questions to create a relational database about a soccer team, its matches and all players in it. When you have enough information to deploy the application, give me the structure of the database.

Of course! Let's start building the relational database for the soccer team. We'll gather information step by step.

1. Soccer Team Information:

- What's the name of the soccer team?
- Where is the team based or located?
- Do you want to include any additional details about the team?

Challenges of using ChatGPT

Lack of Control

- Lack of full control over model's responses.
- ChatGPT can have unpredictable outputs,
    - challenging to ensure accuracy and reliability of information provided.
- Difficult to enforce strict constraints or prevent undesirable outputs, potentially
    - leading to unintended consequences in certain contexts
    - Can be a problem if used in live applications, for example

Lack of Control (cont.)

- Use Human-in-the-Loop Approach: Incorporate human review or validation of model outputs before delivering responses to end-users.
- Implement Conditional Generation: Use conditioning techniques to guide model's responses, such as specifying keywords or context in the prompt to influence output.
- Leverage Response Ranking: Rank and select model outputs based on relevance, accuracy, and appropriateness, reducing chances of displaying undesirable responses.

Ethical Considerations

- Model's training data, which includes internet text, may contain biased or harmful content, leading to biased or offensive responses.
- Programmers must be vigilant to minimize biases
    - actively monitor the model's outputs to ensure fair and respectful language generation.

Ethical Considerations (Cont.)

- Safeguard user data
    - Adhere to privacy regulations and policies to protect sensitive information.
- Regularly evaluate and mitigate any biases or unfairness in the generated responses
- Use ChatGPT responsibly, taking into account limitations
- Obtain informed consent from users before engaging them with ChatGPT.
- Be open: help users by helping them understand capabilities and limitations of AI systems

Conclusion

-

Nice resources

- A Brief Overview of Attention Mechanism
    - https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129
- A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT
    - https://www.researchgate.net/publication/368716245_A_Prompt_Pattern_Catalog_to_Enhance_Prompt_Engineering_with_ChatGPT
- A step-by-step guide to building a chatbot based on your own documents with GPT
    - https://bootcamp.uxdesign.cc/a-step-by-step-guide-to-building-a-chatbot-based-on-your-own-documents-with-gpt-2d550534eea5

Thank you!

info@predictiveinsights.net

PREDICTIVE
INSIGHTS