

Code Smarter, Not Harder: Leveraging AI for Enhanced Programming

[Library event page](#)

Sep 18, 2023

Theme:

- Could be **using these tools to do something simple.**
- I don't know Python or R, but I need to read a CSV. How do you get the tool to show you how to do this.
- Prompt: Can you show me different ways to do X task and show me the code to accomplish that?
- **Teaching how to use this well/effectively.**
- **How to prompt well for coding.**
- **The process of deepening your prompt to achieve X task. And prompting bigger.** (It's just a word predictor.)
 - How much information to give it before you ask your question.
 - Ask it to give you multiple ways of doing something.
- Activity: Have them try prompting to achieve a task.

Prompting strategies

1. Deeper
 - a. Provide context
 - i. Provide language, packages in use, etc.
 - ii. Include the "role" you want AI to play
 - b. Be clear and specific with what you want in return
 - c. Anticipate ambiguity
 - d. Ask about required packages/libraries, they are sometime forgotten
2. Broader/Bigger
 - a. Initially ask about the problem you are trying to solve
 - b. Take a step back and ask if you are asking the right question (e.g., should I be using JSON or something else for this task?)
 - c.
3. Ask for multiple solutions
 - a. Ask for two or three options
 - b. Ask it to provide the simplest solution first
4. Ask for explanation of the reasoning

- a. Ask for step-by-step instructions
- b. Add that you want comments in the code

Draft outline: (90 minutes)

- [5 min.] Welcome and introductions [Jason]
- [10 min.] What is Generative Artificial Intelligence (GAI) and how it works
 - Visualization of how GAI works.
- [5 min.] Overview of Available GAI tools [Ryan]
 - Current
 - Microsoft Co-pilot [Describe - Ryan]
 - OpenAI ChatGPT, Free [Justine]
 - OpenAI ChatGPT4, Paid [John]
 - Google Bard
 - IDE integration
- [30 min.] Activity
- [10 min.] Tips for prompting [Jason]
- [10 min.] Pros and cons of strategies [Justine]
 - Comparisons
 - Limitations
 - Ways to improve
 - Ethics
 - Using in class vs work vs side-projects
 -
- [15 min.] Q&A [everyone]

Potential data / examples

NASA global temps:

Bar chart:

[World of Change: Global Temperatures](#)

Raw data (txt file):

https://data.giss.nasa.gov/gistemp/graphs/graph_data/Global_Mean_Estimates_based_on_Land_and_Ocean_Data/graph.txt

Original source:

<https://climate.nasa.gov/vital-signs/global-temperature/>

- **Deepening** – use ChatGPT as a partner by asking it for instance to respond to your thoughts and questions with refined questions. You may go a step further by asking it to outline how to explore one of those questions.

- My title:

- **Partner to explore deeper ideas**

- ChatGPT:

This encourages collaboration and brainstorming. In the context of AI, this could mean using the AI as a "thought partner" to explore the boundaries of a particular topic or problem. It might involve asking the AI to **outline various approaches** to a coding problem, **speculate on the potential outcomes** of a data analysis, or list the known and unknown variables in a complex scenario.

Example:

Prompt: "What are some different techniques to handle missing data in a machine learning project?"

- **Broader/Bigger** – give ChatGPT a question to help move towards solving a problem. This is useful for when you don't know the answer, or where you may go. For a more simple approach, assign it a role/identity, succinctly describe the context, unambiguously state the goal, and provide an open-ended question.

- My title:

- **Explore broadly**

- ChatGPT:

Before narrowing down to a specific solution, it can be valuable to get a comprehensive view of the landscape. This strategy can involve asking the AI to **enumerate multiple avenues or solutions** that could be pursued. By exploring broadly, you avoid prematurely committing to a single path.

Example:

Prompt: "List programming languages suitable for data analysis, along with their pros and cons."

- **Solutions** – once you have identified a path to go down, ask ChatGPT to give you advice/solutions to the problem.

- Ex. "What are some open problems in X field?"

- My title:

- **Ask for multiple solutions**

- ChatGPT:

For specific problems, especially in coding or data analysis, there's rarely only one correct answer. By asking the AI for multiple solutions, you can **compare approaches for efficiency, simplicity, or creativity**. This is particularly useful for coding challenges, algorithmic problems, or statistical analyses.

Example:

Prompt: "Provide three different Python functions that can calculate the mean of a list of numbers."

- **Explained reasoning** – ask ChatGPT to explain all or part of its reasoning for the solution it provides.

- Ex. "Can you explain to me why you suggested X?"

- My title:

- **Ask for explanation of the reasoning**

- ChatGPT:

A solution is much more valuable when you understand the reasoning behind it. This is crucial for code or data analysis where understanding the 'why' can often be as important as the 'what.' This can be a follow-up where you ask the AI to **explain why a particular method or solution is recommended**, or what the trade-offs are.

Example:

Prompt: "Explain why you recommended using a Random Forest algorithm for this classification problem."

Deeper – A partner to explore deeper ideas

Source: [Prompt Engineering for ChatGPT](#), Vanderbilt University on Coursera

Prompt types:

1. Question refinement
2. Outline expansion

Question refinement

Prompt-1:

I need some help digging deeper into a topic. Whenever I ask you a question, I need you to suggest to me a better version of the question to think about instead.

Prompt-2:

What are the benefits of using ChatGPT to assist me with learning to be a more proficient programmer?

ChatGPT:

A more nuanced question to consider could be: "How can ChatGPT specifically enhance various aspects of my programming skills, from understanding algorithms to debugging, and what are its limitations in the context of programming education?"

Prompt-3:

Please do the same with this question. What are some different techniques to handle missing data in a machine learning project?

ChatGPT:

A more refined question to explore might be: "What are the pros and cons of various techniques for handling missing data in machine learning projects, and how do these methods impact model performance and interpretability?"

Outline expansion

Prompt-1:

I'd like you to act as an **outline expander**. Please generate a bullet point outline based on the input that I give you and then ask me which bullet point you should expand on. Each bullet point can have at most 3 sub bullets. For the bullet point I select, create a new outline. After each outline, ask me which bullet you should expand next.

Prompt-2:

Writing effective ChatGPT prompts for improved programming.

ChatGPT:

Here's the outline based on your input:

1. Introduction

- Importance of effective prompts
- Objectives of the guide
- Who should read this guide

2. Understanding ChatGPT

- How it works
- Limitations
- Use-cases in programming

3. Components of an Effective Prompt

- Clarity
- Specificity
- Contextual information

4. Techniques for Writing Prompts

- The "Ask-Tell" method
- Iterative refinement
- Leveraging token count

5. Examples and Case Studies

- Good vs Bad prompts
- Real-world applications
- Pitfalls to avoid

6. Testing and Validation

- Methods for testing
- Common pitfalls
- Feedback loops

7. Conclusion

- Summary of key points
- Next steps for implementation
- Additional resources

Which bullet point would you like to see expanded next?

...

[SEE GREEN HIGHLIGHT ABOVE]

Prompt: Using this last question you gave me, please go back and be an outline expander. Same rules as above. Please give me an outline I can use to explore this question.

ChatGPT:

Certainly! Here's an outline based on the refined question:

1. Introduction

- Importance of handling missing data
- Scope of the discussion
- Intended audience

2. Common Reasons for Missing Data

- Data collection errors
- Incomplete sources
- Data sparsity

3. Impact of Missing Data

- Model performance
- Bias and fairness
- Interpretability

4. Techniques for Handling Missing Data

- Imputation methods
- Data augmentation
- Utilizing specialized algorithms

5. Pros and Cons of Various Techniques

- Effectiveness in different scenarios
- Computational cost
- Risk of introducing bias

6. Case Studies

- Real-world examples where techniques succeeded
- Real-world examples where techniques failed
- Lessons learned

7. Evaluation Metrics

- How to measure success
- Tools for assessment
- Caveats in evaluation

8. Conclusion

- Summary of key findings
- Recommendations for practice
- Further areas for research

Which bullet point would you like to see expanded next?

Broader/Bigger – Explore broadly

Prompt types:

1. Flipped interaction
2. Persona pattern

Flipped interaction

Prompt-1:

I would like you to ask me questions to help me create a visualization of historical land and ocean temperatures with data from NASA. Ask me questions until you have enough information to identify the easiest approach to creating my visualization. Please ask me one question at a time.

ChatGPT:

Absolutely, let's get started. What is your primary goal for creating this visualization of historical land and ocean temperatures? Are you looking to understand trends, compare regions, or something else?

Prompt-2:

I want to communicate to my undergraduate students the general trend in the data.

ChatGPT:

Great, that helps clarify the purpose. What type of data are you working with? Is it time-series data, geospatial data, or something else?

Prompt-3:

It is time-series data. It starts at the year 1880 and ends on 2022.

ChatGPT:

Excellent, time-series data will make it easier to show trends over time. What software or programming languages are you comfortable using for data visualization? Do you have a preference for tools like Excel, Tableau, or coding languages like Python or R?

Prompt-4:

I prefer to use Excel because I have worked with it for many years. But I am open to learning Python. Generally, I'm looking for what will give me the most flexible output.

ChatGPT:

Understood, both Excel and Python have their merits. Excel is simpler and more immediate, while Python, particularly with libraries like Matplotlib or Seaborn, offers more customization and flexibility. Do you already have the NASA data, and in what format is it available? Is it a CSV file, Excel spreadsheet, or something else?

Prompt-5:

Yes, I have the data in a CSV file. Let's go with Python.

ChatGPT:

Great, Python will offer you a lot of flexibility for creating detailed visualizations. One last question: Do you have any specific design elements you want to include, like annotations to point out significant years or events, or is a straightforward plot sufficient for your needs?

Prompt-6:

No, I just need to be sure that red, green, or yellow colors are not used for accessibility purposes.

ChatGPT:

Thank you for sharing that important detail about accessibility. Given all the information you've provided, the easiest approach for you would be to use Python with the Matplotlib library to plot your time-series data. You can customize the color palette to avoid red, green, and yellow. Your data is already in a CSV file, which makes it easy to import into Python for manipulation and visualization.

Would you like to proceed with some code examples or further guidance on how to go about this?

Prompt-7:

Yes, please give me some code examples I can start with.

ChatGPT:

Certainly! Below is a basic Python code snippet that utilizes the Pandas library for data handling and Matplotlib for plotting. This code assumes that your NASA data in the CSV file has two columns: one for the year ('Year') and one for the temperature ('Temperature'). Feel free to adjust the column names as they appear in your dataset.

First, you'll need to install the required Python packages if you haven't done so already:

```
pip install pandas matplotlib
```

Here's a sample code snippet:

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data from the CSV file
data = pd.read_csv('path/to/your/file.csv')

# Extract the Year and Temperature data
years = data['Year']
```

```
temperatures = data['Temperature']

# Create the plot
plt.figure(figsize=(12, 6))

# Plot the data
plt.plot(years, temperatures, color='blue', linewidth=2)

# Customize the plot
plt.title('Historical Land and Ocean Temperatures')
plt.xlabel('Year')
plt.ylabel('Temperature (°C)')
plt.grid(True)

# To ensure the colors are accessible, we use 'blue' for the line
plt.gca().set_facecolor('white')

# Save or show the plot
plt.savefig('Historical_Temperatures.png') # Save the plot as a PNG file
# plt.show() # Uncomment this line if you want to display the plot
```

Make sure to replace 'path/to/your/file.csv' with the actual path to your CSV file.

This code will produce a simple line plot with the x-axis representing the years and the y-axis representing the temperatures. The line is colored blue for accessibility purposes.

Feel free to adjust the code to suit your specific needs, and let me know if you have any further questions!

Solutions – Ask for multiple solutions

Prompt types:

1. Alternative approaches (to problem solving – solutions and feedback)

Note: The example explored below was given by ChatGPT when asking it to interpret the 4 strategies.

Alternative approaches

Prompt-1:

I want to explore something with multiple solutions and problems. For example, I know there can be different functions in Python to calculate the mean of a list of numbers. First please generate a list of 20 random numbers. Then using those numbers, please show me 3 different Python functions for calculating the mean of that number list. I'd like to see the functions ranked from least to most efficient.

ChatGPT:

Certainly! To begin, let's generate a list of 20 random numbers. We'll use Python's `random` library for this.

```
import random
```

```
# Generate 20 random numbers between 1 and 100
random_numbers = [random.randint(1, 100) for _ in range(20)]
```

```
print("List of 20 random numbers:", random_numbers)
```

Once you've run this code snippet, you should see a list of 20 random numbers. We'll use this list in the following examples.

Next, let's move on to the three Python functions to calculate the mean of this list of numbers, ranked from least to most efficient.

Least Efficient: Manual Iteration

Here, we manually loop through the list to find the sum and then divide by the number of elements.

```
def manual_mean(numbers):
    total = 0
    for number in numbers:
        total += number
    return total / len(numbers)
```

```
print("Manual mean:", manual_mean(random_numbers))
```

Moderately Efficient: Using 'sum()' and 'len()'

This approach uses Python's built-in 'sum()' and 'len()' functions to achieve the same result but in a more Pythonic way.

```
def builtin_mean(numbers):  
    return sum(numbers) / len(numbers)  
  
print("Built-in mean:", builtin_mean(random_numbers))
```

Most Efficient: Using NumPy

Using NumPy's `mean()` function is the most efficient way when dealing with large datasets, as NumPy is optimized for numerical calculations.

First, install NumPy if you haven't already:

```
pip install numpy
```

Then, run this code:

```
import numpy as np  
  
def numpy_mean(numbers):  
    return np.mean(numbers)  
  
print("NumPy mean:", numpy_mean(random_numbers))
```

Each method will give you the same result, but their efficiency varies. The NumPy approach is generally the fastest, especially for large datasets, followed by the built-in `sum()` and `len()` methods. The manual loop is typically the least efficient.