

Evaluation of Prompting Tools Across Diverse AI Platforms: ChatGPT, Claude, Bard, Cohere Command, and Meta

Aim

To compare the performance, user experience, and response quality of different AI platforms (ChatGPT, Claude, Bard, Cohere Command, and Meta) for a specific use case: learning Python programming for beginners. This evaluation will test each platform's ability to explain key programming concepts, suggest resources, and highlight common mistakes.

Procedure:

1. Define the Use Case

The selected task for this evaluation is to generate a beginner's guide to learning Python. This task aims to assess each platform's ability to provide structured, easy-to-understand responses with relevant programming advice.

Example Use Case

Task: Generate a beginner's guide to learning Python, covering key programming concepts, useful resources, and common errors to avoid.

Reason: This task evaluates each platform's ability to provide clear guidance, resources, and insights into common pitfalls in Python programming for beginners.

2. Create a Set of Prompts

A uniform set of prompts will be used to test how each platform performs in the selected use case. These prompts are designed to ensure consistency across platforms.

Prompts

- **Main Prompt:**
"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."
- **Additional Prompts for Versatility:**
 - "Explain Python's control flow with examples."
 - "What are the best online resources for learning Python?"
 - "What are common mistakes beginners make when writing Python code?"

These prompts are intended to evaluate each platform's capacity to deliver comprehensive programming guidance.

3. Run the Experiment on Each AI Platform

The prepared prompts were input into each platform: ChatGPT, Claude, Bard, Cohere Command, and Meta, ensuring all platforms received the same inputs under similar conditions.

Steps

1. Input the primary Python learning guide prompt into each platform.
2. Gather responses and record any differences in:
 - **Response Time:** How quickly does each platform respond?
 - **Ease of Interaction:** How intuitive is the user experience on each platform?
 - **Technical Issues:** Were there any errors or unusual delays during interaction?
3. Repeat the process with the additional prompts to gather more versatile results.
4. Document the responses from each platform for comparison.

4. Evaluate Response Quality

The quality of the responses was assessed using the following criteria:

Evaluation Criteria

- **Accuracy:** Are the programming concepts explained correctly? Are the suggested resources and tips useful and up-to-date?
- **Clarity:** Is the information easy to understand for a beginner? Does the response simplify complex programming ideas?
- **Depth:** Does the response cover the key aspects of Python learning, or does it provide only surface-level information?
- **Relevance:** Does the response stay on topic and focus on important elements of learning Python, such as key concepts, resources, and common mistakes?

5. Compare Performance

Using the collected data, the performance of each platform was compared across the evaluation criteria.

Key Observations

- **Response Quality:** Which platforms provided the most structured and informative Python learning guide?
- **User Experience:** Were some platforms easier to use or faster to respond?
- **Strengths & Weaknesses:** Identify platform-specific advantages. For example, did one platform excel in clarity while another provided more detailed explanations?

Example Response (ChatGPT)

Input:

"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."

ChatGPT Python Learning Guide for Beginners:

1. Introduction to Python

- **Why Learn Python?**

Python is beginner-friendly due to its simple syntax and readability. It's widely used in web development, data analysis, automation, and artificial intelligence.

2. Key Concepts to Learn

- **Variables and Data Types:**

Learn about integers, floats, strings, and booleans.

```
age = 25 # integer
name = "John" # string
is_student = True # boolean
```

- **Control Flow:**

Master if-else statements and loops.

```
if age > 18:
    print("You are an adult.")

for i in range(5):
    print(i)
```

- **Functions:**

Learn how to define and call functions.

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Alice"))
```

- **Lists and Dictionaries:**

Learn to store multiple items in a list or key-value pairs in a dictionary.

```
numbers = [1, 2, 3, 4]
person = {"name": "Alice", "age": 25}
```

3. Best Resources for Learning Python

- Official Python Documentation: Comprehensive guide on python.org.
- Online Courses:
 - Codecademy Python Course: Interactive lessons.
 - Coursera - Python for Everybody: Beginner-friendly course.

- Books:
 - "Python Crash Course" by Eric Matthes.
 - "Automate the Boring Stuff with Python" by Al Sweigart.

4. Common Mistakes to Avoid

- **Mixing Up Data Types:**

```
age = 25

print("You are " + str(age) + " years old.") # Correct
```
- **Indentation Errors:**

```
if age > 18:
    print("Adult") # Incorrect indentation will cause an error.
```

5. Project Ideas for Practice

- Build a Simple Calculator.
- To-Do List Application.
- Weather App using an API.
- Number Guessing Game.

Example Response (Claude)

Input:

"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."

Claude Python Learning Guide for Beginners:

1. Introduction to Python

- **Why Learn Python?**

Python is a versatile language ideal for beginners due to its clear syntax and readability. It's widely utilized in various fields, including web development, data science, and automation.

2. Key Concepts to Learn

- **Variables and Data Types:**

Understand the different types of data you can work with in Python, such as integers, strings, and lists.

```
age = 30 # integer
name = "Alice" # string
colors = ["red", "green", "blue"] # list
```
- **Control Flow:**

Learn to use conditional statements and loops.

```
If age < 18:
    print("Minor")
else:
    print("Adult")

for i in range(5):
    print(i)
```

- **Functions:**

Functions are essential for organizing your code.

```
def add(x, y):
    return x + y

print(add(5, 3))
```

3. Best Resources for Learning Python

- **Online Platforms:**

- **Codecademy:** Offers interactive Python courses.
- **edX - Introduction to Computer Science and Programming Using Python:** A great starting point.

- **Books:**

- "Learning Python" by Mark Lutz.
- "Head First Python" by Paul Barry.

4. Common Mistakes to Avoid

- **Syntax Errors:** Always check your syntax; forgetting a colon or using the wrong indentation can lead to errors.
- **Incorrect Data Types:** Be mindful of using the correct types when performing operations.

```
print("Age: " + age) # Error: age is an integer
```

5. Project Ideas for Practice

- Create a Basic To-Do Application.
- Develop a Basic Chatbot.
- Build a Weather Forecast Application using an API.

Example Response (Bard)

Input:

"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."

Bard Python Learning Guide for Beginners:

1. Introduction to Python

- **Why Learn Python?**

Python is favored for its simplicity and readability, making it an excellent choice for beginners and professionals alike.

2. Key Concepts to Learn

- **Variables:**

Variables store data. Here's how to define them.

```
score = 100 # integer
```

```
player_name = "Bob" # string
```

- **Control Flow:**

Use control flow statements to make decisions in your code.

```
if score >= 50:
```

```
    print("You passed!")
```

- **Functions:**

Functions encapsulate code for reuse.

```
def multiply(a, b):
```

```
    return a * b
```

```
print(multiply(2, 3))
```

3. Best Resources for Learning Python

- **Websites:**

- W3Schools Python Tutorial: Great for quick references.
- Real Python: In-depth tutorials and resources.

- **Books:**

- "Automate the Boring Stuff with Python" by Al Sweigart.

4. Common Mistakes to Avoid

- **Indentation Errors:** Python uses indentation to define blocks of code.
- **Mismatched Parentheses:** Ensure all parentheses are properly closed.

5. Project Ideas for Practice

- Design a Basic Game (e.g., Rock, Paper, Scissors).
- Implement a Basic Expense Tracker.

Example Response (Cohere Command)

Input:

"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."

Cohere Command Python Learning Guide for Beginners:

1. Introduction to Python

- **Why Learn Python?**

Python is a powerful language that's easy to learn. It's widely used for various applications like web development, data analysis, and more.

2. Key Concepts to Learn

- **Data Types:**

Understand the basics of data types in Python.

```
pi = 3.14 # float
```

```
is_active = True # boolean
```

- **Control Flow:**

Mastering control flow is essential for programming logic.

```
if is_active:
```

```
    print("User is active")
```

- **Functions:**

Learn to create and use functions.

```
def divide(x, y):
```

```
    return x / y
```

```
print(divide(10, 2))
```

3. Best Resources for Learning Python

- **Online Courses:**

- Coursera - Python for Everybody.
- Udacity - Introduction to Python Programming.

- **Books:**

- "Python Crash Course" by Eric Matthes.

4. Common Mistakes to Avoid

- **Confusing Lists and Tuples:** Lists are mutable, while tuples are immutable.
- **Forgetting to Import Modules:** Always ensure necessary libraries are imported before use.

5. Project Ideas for Practice

- Create a Basic Blog Application.
- Develop a Simple Quiz Application.

Example Response (Meta)

Input:

"I need a guide to learning Python, covering key concepts, best resources, and common mistakes to avoid."

Meta Python Learning Guide for Beginners:

1. Introduction to Python

- **Why Learn Python?**

Python's syntax is designed to be intuitive, making it a great first language for new programmers.

2. Key Concepts to Learn

- **Variables:**

The backbone of your data manipulation.

```
temperature = 25 # integer
```

```
weather = "Sunny" # string
```

- **Control Flow:**

Conditional logic is fundamental to programming.

```
if temperature > 20:
```

```
    print("It's warm outside!")
```

- **Functions:**

Functions allow you to encapsulate logic.

```
def square(num):
```

```
    return num ** 2
```

```
print(square(4))
```

3. Best Resources for Learning Python

- **Free Resources:**

- Python.org's official tutorial.
- YouTube channels like Corey Schafer for Python tutorials.

- **Books:**

- "Think Python" by Allen B. Downey.

4. Common Mistakes to Avoid

- **Variable Scope Issues:** Understand local vs. global variables.
- **Ignoring Error Messages:** Learn to read and understand error messages.

5. Project Ideas for Practice

- Build a Basic Portfolio Website.
- Develop a Simple Chat Application.

Deliverables

Comparison Table

| AI Platform | Accuracy | Clarity | Depth | Relevance | Response Time |
|----------------|-----------|---------|----------|-------------------|---------------|
| ChatGPT | Excellent | High | In-depth | Highly relevant | Fast |
| Claude | Good | Medium | Moderate | Relevant | Moderate |
| Bard | Good | High | Shallow | Relevant | Fast |
| Cohere Command | Moderate | Low | Minimal | Somewhat relevant | Slow |
| Meta | Moderate | Medium | Moderate | Relevant | Fast |

Summary of Findings:

- ChatGPT provided the most comprehensive and structured responses, making it highly suitable for beginners learning Python.
- Claude offered clear explanations but lacked depth.
- Bard was effective for quick, surface-level guidance but didn’t provide as much detail.
- Cohere Command struggled with clarity and depth, while Meta offered moderate but consistent responses.