

# One String, Many Prompts

A Hands-On Workshop in AI Code Generation

Geoffrey Goetz - January 2026



```
{  
  "recorded": 1768237200,  
  "location": "den",  
  "sensor": "bmp280",  
  "measurement": "temperature",  
  "units": "C",  
  "value": 22.3  
}
```

# CODE



# G GEOFFRE



# FEEDBACK



Command your IoT Projects from the Terminal

# Hands-On Pre-Requisites

MINIMAL REQUIREMENTS

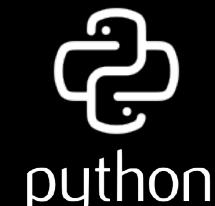
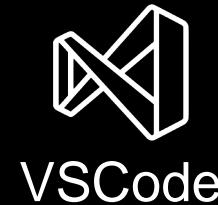
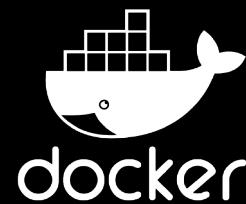
## TOOLING SETUP

Install **Docker**, **VS Code** and **Ollama** (*optional*)

Create GitHub CoPilot Account

*Create Docker* (*optional*)

*Create Postman and/or Ollama Account* (*optional*)



Select primary **Development Language**

Run Docker Image for development language

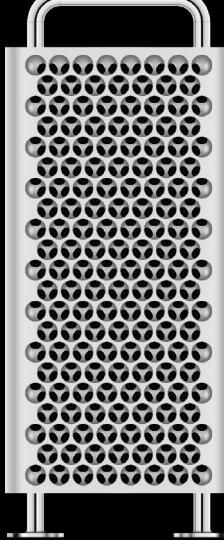
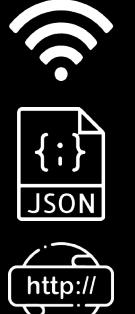
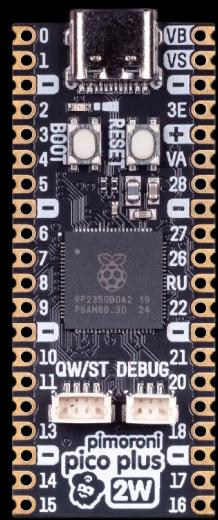
Select primary **Database Server**

Run Docker Image for database server

*Download Ollama LLM* (*optional*)

## ENVIRONMENT SETUP

```
{  
  "recorded" : 1768237200,  
  "location" : "den",  
  "sensor" : "bmp280",  
  "measurement": "temperature",  
  "units" : "C",  
  "value" : 22.3  
}
```



5  
Sensors

5  
Platforms

5  
Microcontrollers

10K  
Lines of Code

5  
AI LLMs

1  
String

$\sim 5^5$   
Combinations

5  
Languages

5  
Frameworks

5  
Databases

1K  
AI Prompts



---

## FUNCTIONAL REQUIREMENT

Gather data from diverse sensors across multiple sites, then save this information in a way that allows it to be easily examined.

---

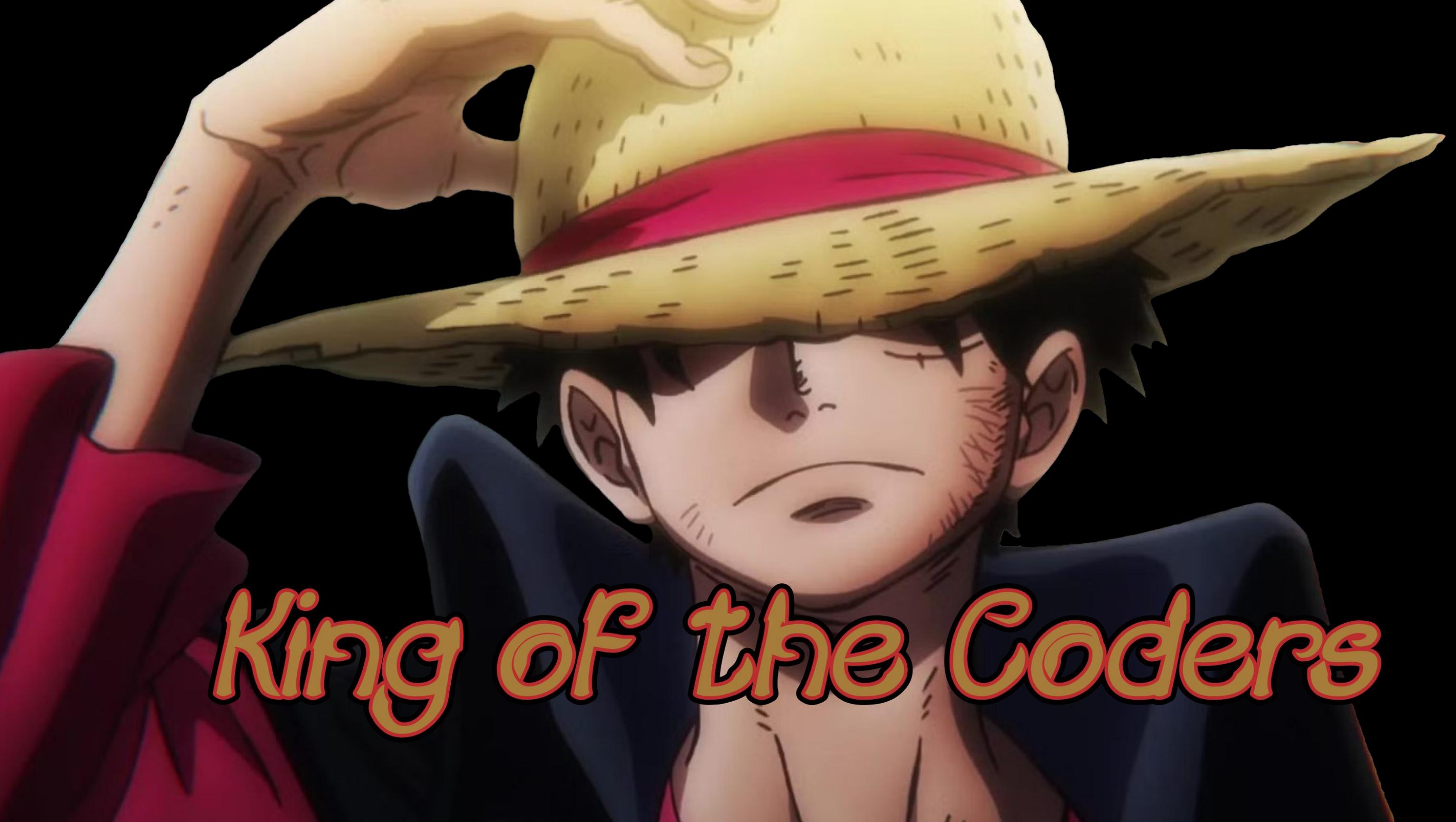
## NON-FUNCTIONAL CONSTRAINTS

The system requires a single, standardized JSON data format universally compatible across five different types of sensors, microcontroller platforms, HTTP frameworks, programming languages, and database systems.

---

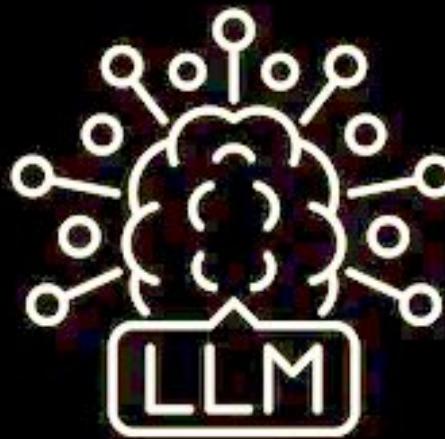
## OVERALL GOAL

The goal is to repeatedly use AI to solve the same core problem across a wide variety of technical stacks, including different languages, platforms, chipsets, and databases. This approach aims to cultivate proficiency in prompting AI for functional code generation, while also developing a deeper understanding of the unique interoperability challenges and nuances that arise when mixing these diverse technologies.



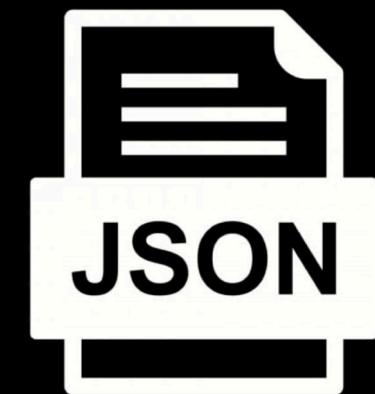
King of the Coders

# Reducing variance in a probabilistic generative system by constraining its conditional distribution.



**Model  
Creation**

*Representation*



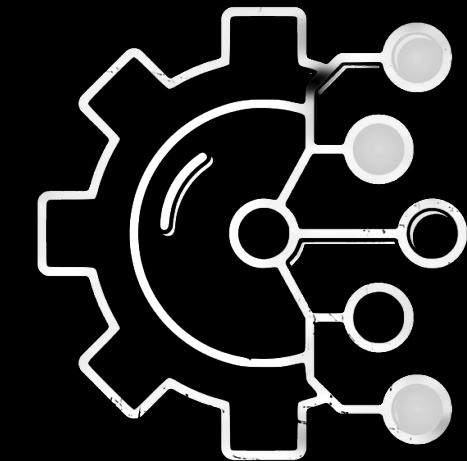
**Fine  
Tuning**

*Preference Shaping*



**Retrieval  
Augmented Generation**

*Epistemic Anchoring*



**Prompt  
Engineering**

*Semantic Narrowing*

# The Definitive and Final AI/LLM Ranking

What are the top five AI LLMs that developers use on a regular basis based on quality of output?



GPT-4/GPT-4o (OpenAI)  
Claude (Anthropic)  
Llama (Meta)  
Gemini (Google)  
Mistral\*



OpenAI's GPT-5  
Anthropic's Claude  
Google DeepMind's Gemini  
Meta's Llama  
DeepSeek-V3\*



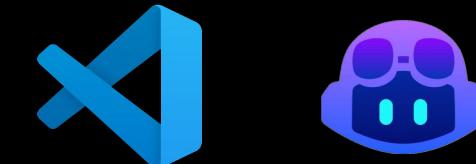
Claude Sonnet 4.5  
GPT-5  
Claude 4.1 Opus  
Grok 4\*  
Gemini 2.5 Pro



GPT-4o / GPT-4  
Claude 3.5 Sonnet  
Gemini 2.5 Pro  
LLaMA 3  
Mistral Large\*



CURSOR



VS Code / CoPilot



Zed / Claude / Olama



NeoVim / Claude



JetBrains / Mellum / CoPilot



Windsurf

The relationships between IDE/editor and “which LLM works best” are more fluid and overlapping than “one-IDE: one-LLM.”



AI Agent



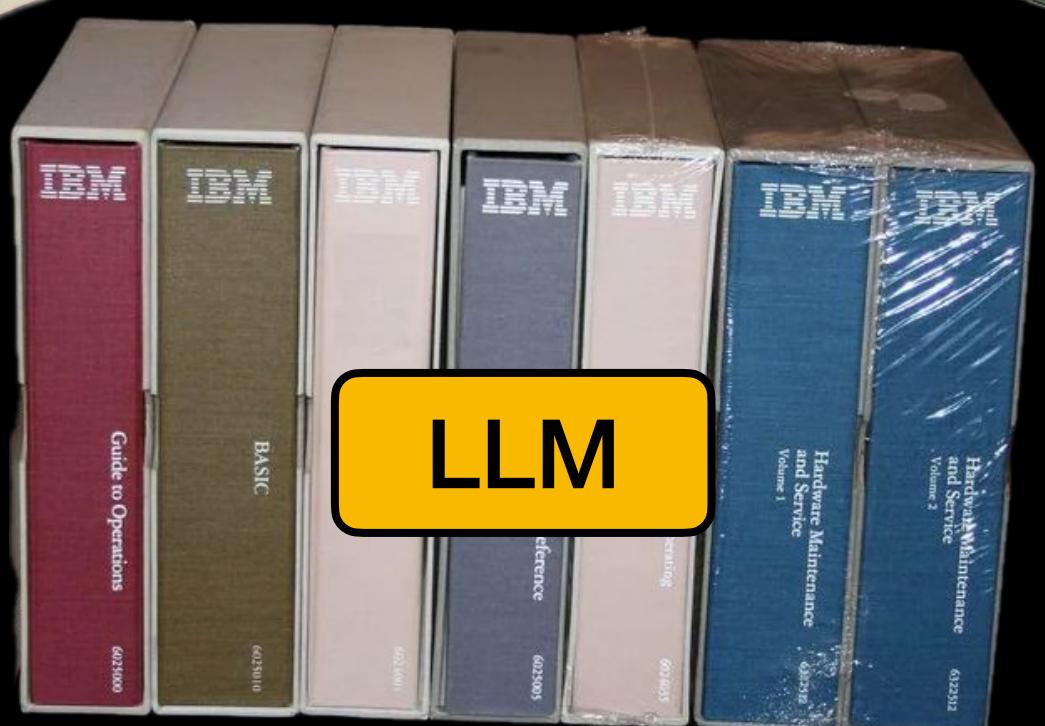
Tokens



IDE



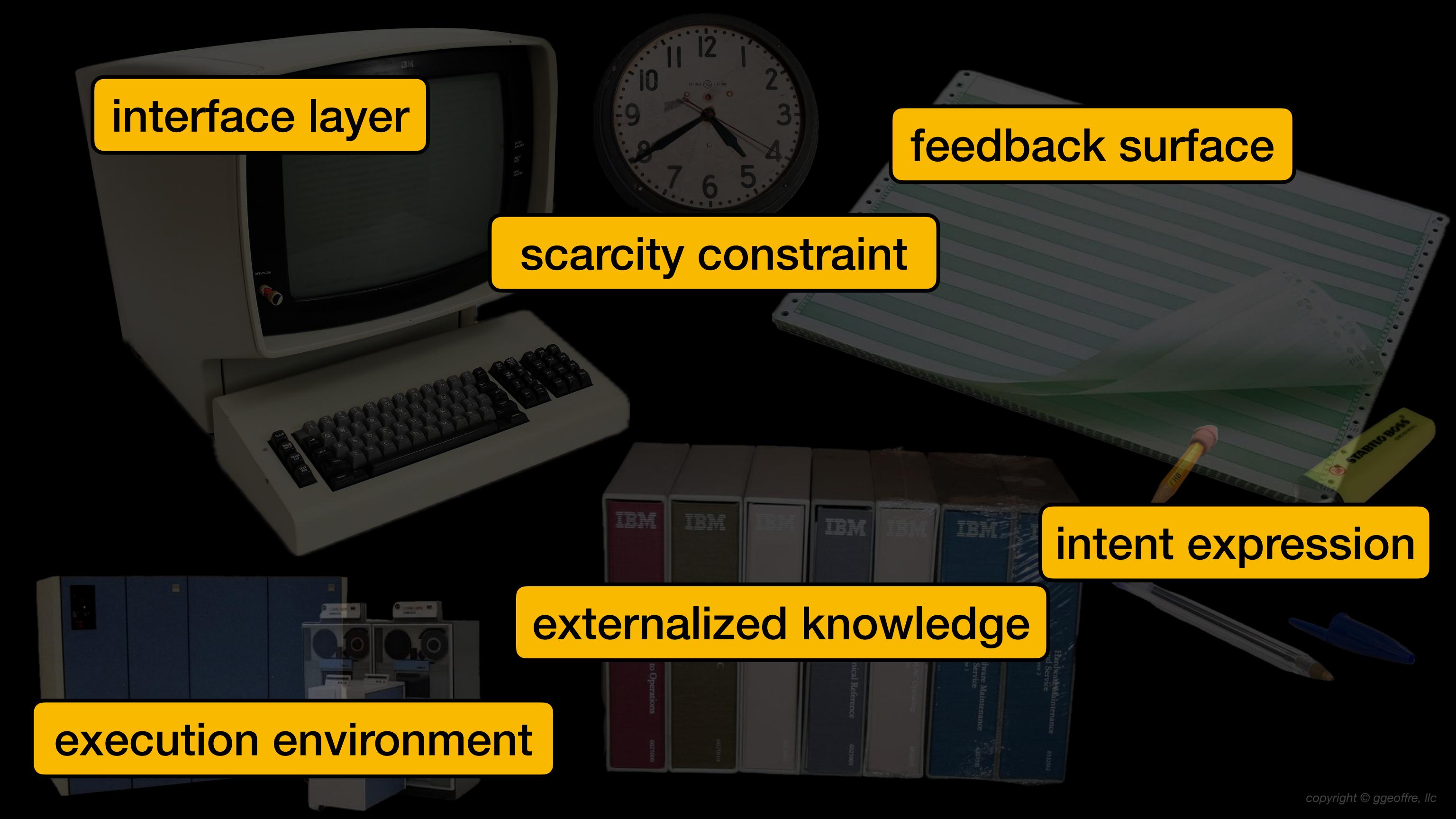
Prompt



LLM



Container



interface layer

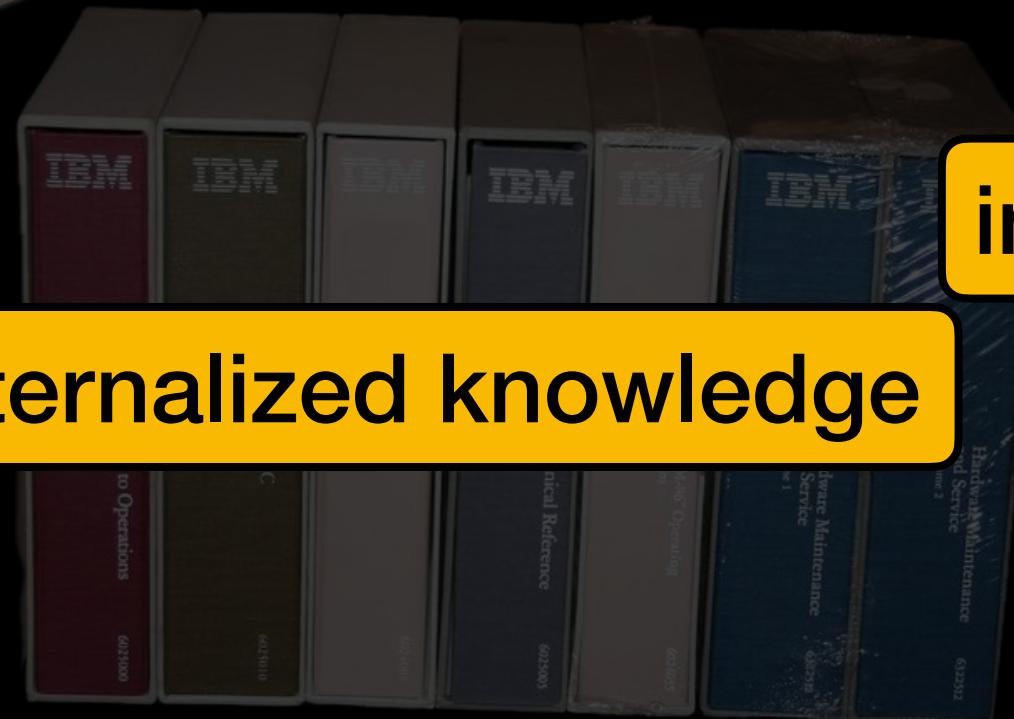
feedback surface

scarcity constraint



execution environment

externalized knowledge



intent expression





## Workflow Engineering

Workflow engineering involves analyzing, designing, automating, and optimizing sequences of tasks or processes. It is a traditional engineering discipline that focuses on the end-to-end flow of work within an organization or system, determining how tasks move from one step to the next.

- **Focus:** The overall process, the sequence of operations, task dependencies, and system integration.
- **Goal:** To improve efficiency, consistency, and reliability of a multi-step process, whether those steps are performed by humans, traditional software, or AI.

## Context Engineering

Context engineering is the process of designing and providing rich, relevant information to an AI model to guide its output. This involves carefully structuring the input data, system prompts, examples (few-shot learning), and available tools or data sources.

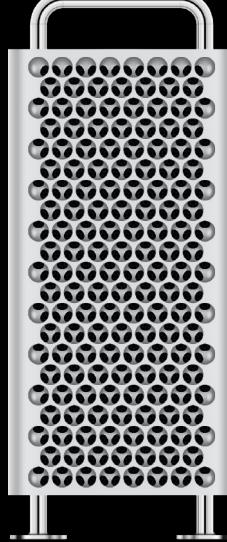
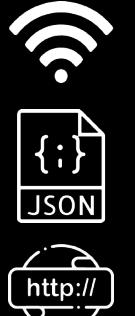
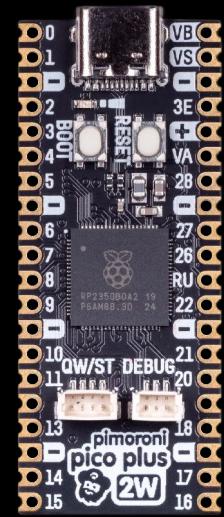
- **Focus:** Enhancing the quality, accuracy, and relevance of an individual AI agent's response within a specific interaction. It's about giving the AI the optimal 'working memory' and rules.
- **Goal:** To make a single AI instance behave reliably and effectively within its given environment.

3

```
{  
  "recorded": 1768570200,  
  "location": "den",  
  "sensor": "bmp280",  
  "measurement": "temperature",  
  "units": "C",  
  "value": 22.3  
}
```

3

SENSOR	MICROCONTROLLER	PLATFORM	LANGUAGE	FRAMEWORK	DATABASE
BMP280	Pico	Pico-SDK	Python	Flask	Redis
SHT45	ESP32	ESP-IDF	Java	Spring Boot	Mongo
AHT20	ATmega	Arduino	Swift	Vapor	Cassandra
HDC302x	SAMD21	CircuitPython	GO	Gin	Postgres
MCP9808	nRF52	MicroPython	Rust	Actix	MySQL



Gather data from diverse sensors across multiple sites...

Read Sensor

③

Send Data

Connect to Wi-Fi

then save this information in a way that allows it to be easily examined.

Record Data

③

Retrieve Report

Purge Data

## Read Sensor

③

## Send Data

Connect to Wi-Fi

Establish a connection to the I<sup>2</sup>C Bus

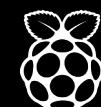
Read the Sensor Data

Create the JSON String

Establish a Wi-Fi Connection

Send the JSON String to the HTTP API

*Recorded Time of Data Collection\**



## Record Data

③

Retrieve Report

Listen for HTTP POST/GET API Calls

Receive the JSON String

Store JSON data into a Database

Retrieve JSON data from a Database

Purge JSON data from a Database

*Recorded Time of Data Recording\**



what are the **most used and best maintained** microcontroller platforms

what are the **most reputable online sites** that rank microcontroller platforms

what are the **different categories** of microcontroller platforms

what are the **most used and easy to source** microcontroller development boards

which microcontroller development boards **can connect to Wi-Fi**

which microcontroller development boards **have a STEMMA/QT connector**

## SENSORS & LIBRARIES

what is the best way to **collect data from a wide variety of sensors** using a microcontroller without soldering, *think carefully about this*

which I2C sensors **have supporting libraries** for all \$XYZ platforms, *think carefully about this*

for the **\$XYZ** microcontroller platform, **what are the most popular and recently maintained libraries** that communicate over I2C supporting the **\$XYZ** sensor, *think carefully about this*

what are the **most used and best maintained** programming languages  
what are the **most reputable online sites** that rank programming languages  
what are the **different categories** of programming languages

what are the **most used and best maintained** databases  
what are the **most reputable online sites** that rank databases  
what are the **different types** of databases

## FRAMEWORKS & LIBRARIES

what is the best way to **send a JSON string from a microcontroller** over the network to be stored into a database, *think carefully about this*

for the **\$XYZ** programming language, **what are the most popular and recently maintained frameworks** that send and receive JSON via HTTP POST and GET, *think carefully about this*

for the **\$XYZ** programming language, **what are the most popular and recently maintained libraries** used with the **\$XYZ** database, *think carefully about this*

**Read Sensor**

**Connect to Wi-Fi**

Establish a connection to the I<sup>2</sup>C Bus

Read the Sensor Data

Create the JSON String

Establish a Wi-Fi Connection

Send the JSON String to the HTTP API

*Record Time of Data Collection\**



**Send Data**

**Record Data**

**Retrieve Report**

Listen for HTTP POST/GET API Calls

Receive the JSON String

Store JSON data into a Database

Retrieve JSON data from a Database

Purge JSON data from a Database

*Record Time of Data Recording\**



③

③

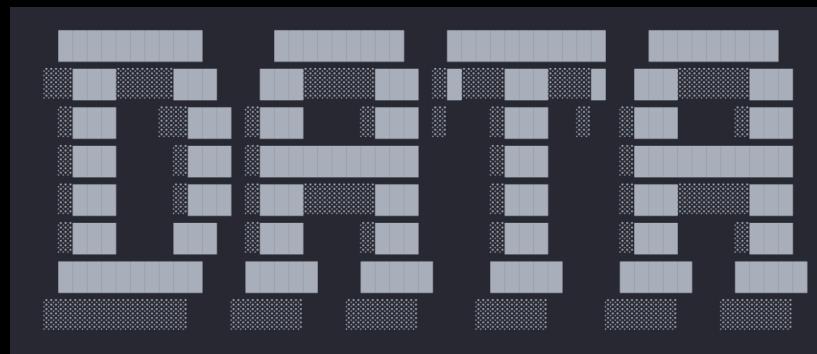
Hands On  
Starts Here



# Hands On Exercises

*Inserting the One String*

Code

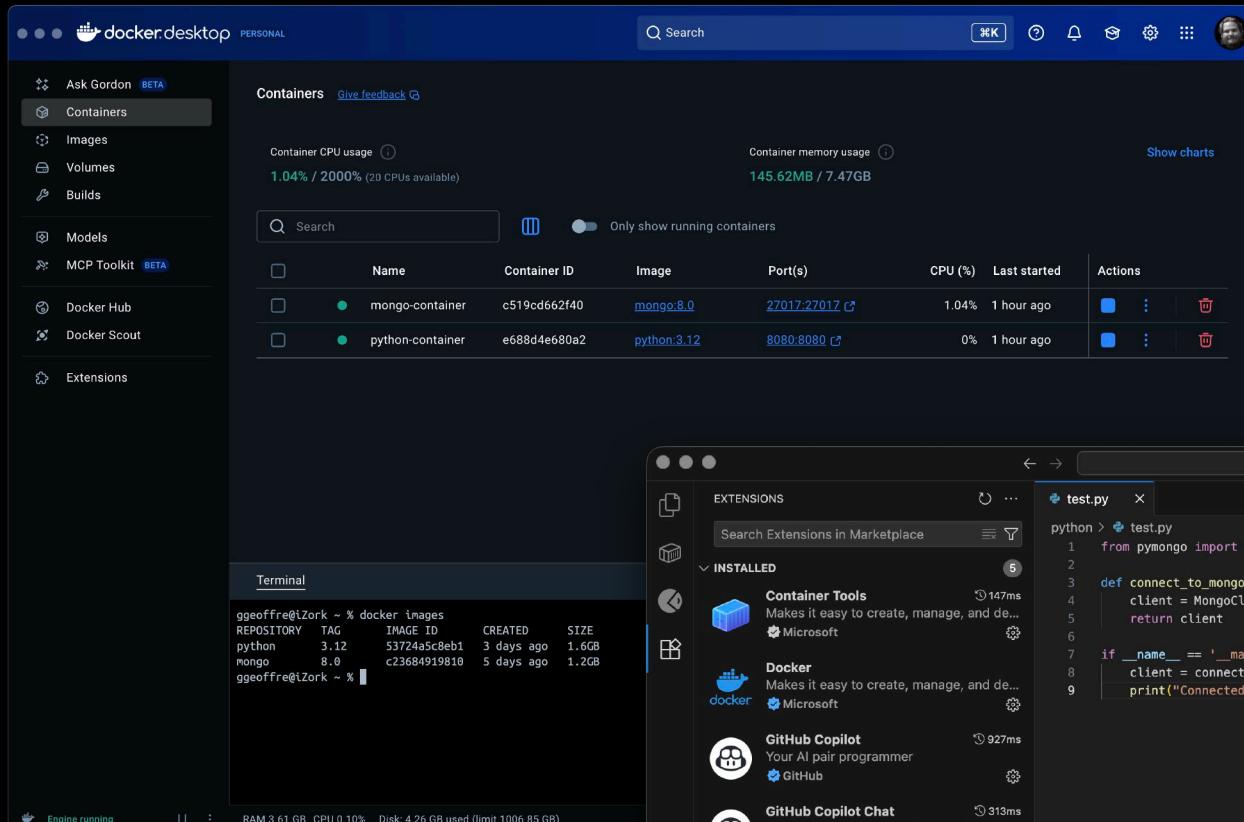


Feedback



One String, Many Prompts: A Hands-On Workshop in AI Code Generation

# Your First Two Prompts



Using  
Container  
Runtime,  
Accessing  
Local Code

A screenshot of a VS Code terminal window. The terminal shows the command 'docker images' output:

```
REPOSITORY TAG IMAGE ID CREATED SIZE
python 3.12 53724a5c8eb1 3 days ago 1.6GB
Mongo 8.0 c23684919810 5 days ago 1.2GB
```

The status bar at the bottom indicates 'Engine running' and resource usage: RAM 3.61 GB, CPU 0.10%, Disk: 4.26 GB used (limit 1006.85 GB).

The main workspace shows an open file 'test.py' with the following code:

```
python > test.py
from pymongo import MongoClient
def connect_to_mongodb(host='192.168.0.1', port=27017):
    client = MongoClient(host, port)
    return client
if __name__ == '__main__':
    client = connect_to_mongodb()
    print("Connected to MongoDB:", client)
```

An AI chat interface is visible on the right, asking 'what python library do i need to install to support the import MongoClient'. The response suggests installing the 'pymongo' library.

The terminal at the bottom shows the command 'pip3 list' output:

```
Package Version
-----
dnspython 2.8.0
pip 25.2
pymongo 4.15.3
root@e688d4e680a2:/home/projects/python/codemash/python#
```

## Docker Terminal

```
docker run -d --name mongo-container -p 27017:27017 mongo:8.2.2
docker run -d --name python-container -p 8080:8080 -v
$MY_LOCAL_CODE_DIRECTORY:/usr/src/python python:3.13.5 sleep infinity
```

## VS Code Terminal

```
docker exec -it -w /home/projects/python python-container /bin/bash
pip install --upgrade pip
```

PROMPT #1

*create a simple  
python method  
to connect to a  
mongo db  
database*

PROMPT #2

*what python  
library do i need to  
install to support  
the import of  
MongoClient*

## Docker Terminal

---

```
docker run -d --name mongo-container -p 27017:27017 mongo:8.2.2
```

```
docker run -d --name python-container -p 8080:8080 -v  
$MY_LOCAL_CODE_DIRECTORY:/usr/src/python python:3.13.5 sleep  
infinity
```

## VS Code Terminal

---

```
docker exec -it -w /usr/src/python python-container /bin/bash
```

```
python --version; pip --version; pwd; ls
```

```
pip install --upgrade pip
```

# !!! REMINDER !!!

Inside the docker container is where localhost resolves to.

And we have two containers running (*with exposed ports*).

So you need to change localhost to the IP address of your machine.

ipconfig

ifconfig

*NOTE: Do not run either ipconfig or ifconfig from within the container, run it locally on your host machine*

<https://github.com/ggeoffre/onestring>

ggeoffre / onestring

Type ⌘ to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files main + 🔎 Go to file

onestring / assignments / README.md

ggeoffre Onestring: README · ec7b987 · 1 minute ago · History

Preview Code Blame 28 lines (22 loc) · 1.15 KB

// SPDX-License-Identifier: GPL-3.0-or-later // Copyright (C) 2025-2026 ggeoffre, LLC

Here you will find the assignments for the OneString course as well as the solutions. The code is separated into two main sections:

**data\_storage**

This is where you will find all of the code related to storing the one string. Go, Java, Python, Rust and Swift solutions are provided using echo, gin, quarkus, spring, django, flask, actix, axum, and vapor api services connecting to cassandra, mongo, mysql, postgres and redis databases.

**data\_collection**

This is where you will find all of the code related to collecting the one string. Arduino, Circuitpython, ESP-IDF, MicroPython, and Pico-SDK can be found here. Under each microcontroller platform you will find a folder for each sensor implemented.

```
onestring/
  assignments/
    00-setup.txt
    ...
  code/
    data_collection/
      circuitpython/
        bmp280/
        ...
      data_collection/
        python/
        ...
  README.md
```



3

```
{  
  "recorded": 1768237200,  
  "location": "den",  
  "sensor": "bmp280",  
  "measurement": "temperature",  
  "units": "C",  
  "value": 22.3  
}
```

3

SENSOR	BMP280	MICROCONTROLLER	Pico	PLATFORM	Pico-SDK	LANGUAGE	Python ✓	FRAMEWORK	Flask	DATABASE	Redis
SHT45		ESP32		ESP-IDF		Java		Spring Boot		Mongo	
AHT20		ATmega		Arduino		Swift		Vapor		Cassandra	
HDC302x		SAMD21		CircuitPython		GO		Gin		Postgres	
MCP9808		nRF52		MicroPython		Rust		Actix		MySQL	

# Your First 5<sup>n</sup> Prompts

JAVA

create a simple **java** method to connect to a **mongo** database  
create a simple **java** method to connect to a **redis** database  
create a simple **java** method to connect to a **cassandra** database  
create a simple **java** method to connect to a **postgres** database  
create a simple **java** method to connect to a **mysql** database

SWIFT

create a simple **swift** method to connect to a **mongo** database  
create a simple **swift** method to connect to a **redis** database  
create a simple **swift** method to connect to a **cassandra** database  
create a simple **swift** method to connect to a **postgres** database  
create a simple **swift** method to connect to a **mysql** database

GO

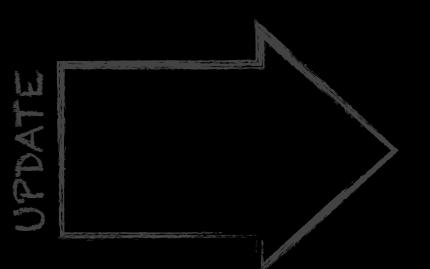
create a simple **go** method to connect to a **mongo** database  
create a simple **go** method to connect to a **redis** database  
create a simple **go** method to connect to a **cassandra** database  
create a simple **go** method to connect to a **postgres** database  
create a simple **go** method to connect to a **mysql** database

RUST

create a simple **rust** method to connect to a **mongo** database  
create a simple **rust** method to connect to a **redis** database  
create a simple **rust** method to connect to a **cassandra** database  
create a simple **rust** method to connect to a **postgres** database  
create a simple **rust** method to connect to a **mysql** database

PYTHON

create a simple **python** method to connect to a **mongo** database  
create a simple **python** method to connect to a **redis** database  
create a simple **python** method to connect to a **cassandra** database  
create a simple **python** method to connect to a **postgres** database  
create a simple **python** method to connect to a **mysql** database



## AI Lessons Learned Already...

- Programming Language Age Matters
- Database Library Selection Criteria
- Language, Library and Database Versions Matter
- Using “localhost” with Docker Images
- Resolving port conflicts can be tricky
- Not every database library has a ‘connection’ or a ‘table’
- Code does not take ‘initialization’ into account

create a simple **python v3.12** method to connect to a **mongo v8.0** database with the **pymongo v4.15.3** pip3 library using ip address **192.168.1.60** to connect

# Your First 5<sup>n</sup> Prompts

## VERSION A

*create a simple **python** method to connect to a **mongo** database*

```
✨ from pymongo import MongoClient

def connect_to_mongodb(host='localhost', port=27017):
    client = MongoClient(host, port)
    return client

if __name__ == '__main__':
    client = connect_to_mongodb()
    print("Connected to MongoDB:", client)
```

Accept

## VERSION B

*create a safe **python v3.12** method to connect to a **mongo v8.0** database with the **pymongo v4.15.3** pip3 library using ip address **192.168.1.60** to connect*

```
✨ from pymongo import MongoClient

def connect_to_mongodb(host='192.168.1.60', port=27017):
    try:
        client = MongoClient(host, port)
        client.admin.command('ping')
        print("Connected to MongoDB successfully!")
        return client
    except Exception as e:
        print(f"Failed to connect to MongoDB: {e}")
        return None
```

```
if __name__ == '__main__':
    client = connect_to_mongodb()
    if client:
        db = client['mydatabase']
        collection = db['mycollection']
        print(db.list_collection_names())
        client.close()
```

Accept

# Your First 5<sup>n</sup> Prompts

in the python language v3.12, create a method to **connect to a mongo v8.0 database** with the pymongo v4.15.3 pip3 library using ip address 192.168.1.60 to connect

in the python language using pymongo, create a method that will **create a database named sensor\_data\_db and a collection named sensor\_data** in the mongo database

in the Python language using pymongo, create a method that will **store the following JSON string into a mongo collection named sensor\_data** in the db named sensor\_data\_db: `{"recorded":1234567890,"location":"den","sensor":"bmp280","measurement":"temperature","units":"C","value":22.3}`

in the Python language using pymongo, create a method that will **fetch all members of the collection named sensor\_data** in the db named sensor\_data\_db

in the Python language using pymongo, create a method that will **remove all members of the collection named sensor\_data** in the db named sensor\_data\_db

in the Python language, create a method that **recursively removes a specified key from either a JSON string or a list of dictionaries**, preserving the nested structure and handling both input types automatically

in the Python language, create a method that **transforms a JSON list into a CSV string**, with the keys as the column headers

in the Python language, create a method that **modifies the value of a specified key in a JSON string**

in the Python language, set a variable to a **randomly generated float value between 20.9 and 34.4 with one decimal place of precision**

in the Python language, set a variable to **the current timestamp**

# Your First 5<sup>n</sup> Prompts

- **Complexity and Edge Cases:** LLMs are good at generating small, isolated functions but struggle significantly with large-scale architecture, complex interactions across multiple files, and subtle edge cases. The probability of errors compounds with complexity.
- **Lack of Deep Understanding:** AI doesn't "understand" the underlying hardware trade-offs (e.g., specific memory access patterns or cache optimizations) in the same way an expert systems programmer does. It generates plausible code based on patterns it was trained on, which might be functional but inefficient or suboptimal for extreme performance requirements.
- **Debugging Is Still a Human Task:** When AI-generated code breaks (which happens frequently), debugging it can be harder than debugging manually written code because the user didn't write it and might not understand the logic the AI used.
- **Performance vs. Readability Trade-offs:** Human experts balance readability and optimization. AI might produce code that is technically correct but messy, making maintenance difficult and introducing technical debt.
- **Need for Human Oversight:** The consensus among industry experts is that AI-generated code still requires thorough human review, testing, and validation to ensure correctness, security, and performance.

```
55     # Store data into Cassandra
56     try:
57         if session:
58             cql_query = f"""
59                 INSERT INTO {CASSANDRA_TABLE} (recorded, location, sensor, measurement, units
60                 VALUES (%s, %s, %s, %s, %s, %s)
61
62
63
64             # Generate and insert random sensor data
65             tuple_values = create_insert_data_tuple(
66                 get_random_sensor_data_json_dictionary()
67             )
68             if not tuple_values:
69                 print("Values cannot be empty")
70             else:
71                 updated_values = list(tuple_values)
72                 updated_values[0] = int(updated_values[0])
73                 tuple_values = tuple(updated_values)
74                 print("JSON string converted to Tuple")
75                 session.execute(cql_query, tuple_values)
76                 print("SensorData stored")
77             else:
78                 print("Database connection is closed.")
79         except DriverException as e:
80             print(f"Error storing sensor data: {e}")
81
82     # Fetch data from Cassandra
83     try:
84         query = f"SELECT * FROM {CASSANDRA_TABLE}"
85         rows = session.execute(query)
86         print("SensorData retrieved")
87         sensor_data_list = []
88         for row in rows:
89             row_dict = dict(row._asdict())
90             sensor_data_list.append(json.dumps(row_dict, default=json_default))
91         json_results = [json.loads(j) for j in sensor_data_list]
92         print(json_to_csv(json_results))
93     except DriverException as e:
```

ORIGINAL

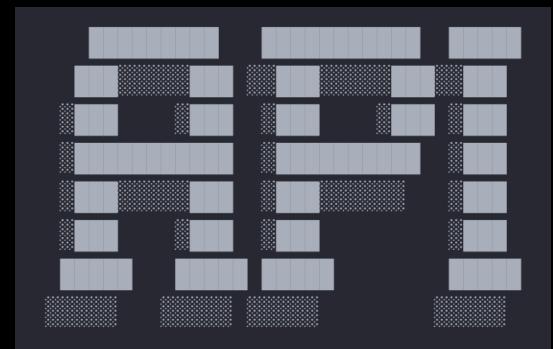
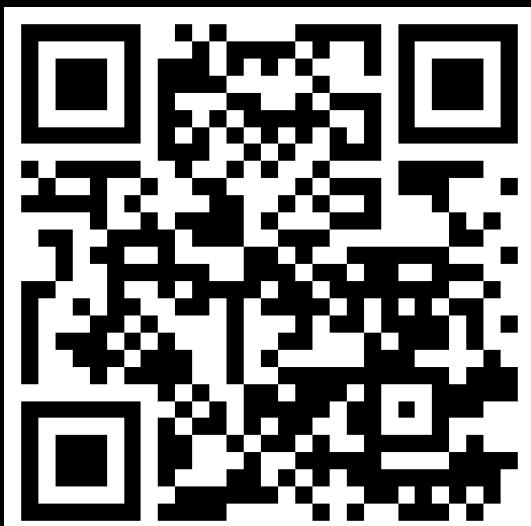
```
155
156     DriverException: If insertion fails
157     ValueError: If data tuple is empty
158
159     cql_query = f"""
160         INSERT INTO {CASSANDRA_TABLE} (recorded, location, sensor, measurement, units, value)
161         VALUES (%s, %s, %s, %s, %s, %s)
162
163
164     try:
165         # Generate random sensor data
166         sensor_data = get_random_sensor_data_json_dictionary()
167         tuple_values = create_insert_data_tuple(sensor_data)
168
169         if not tuple_values:
170             raise ValueError("Data tuple cannot be empty")
171
172         # Convert recorded timestamp to integer
173         updated_values = list(tuple_values)
174         updated_values[0] = int(updated_values[0])
175         tuple_values = tuple(updated_values)
176
177         session.execute(cql_query, tuple_values)
178         logger.info("Sensor data stored successfully")
179         print("JSON string converted to Tuple")
180         print("SensorData stored")
181
182     except (DriverException, ValueError) as e:
183         logger.error(f"Failed to insert sensor data: {e}", exc_info=True)
184         raise
185
186
187     def retrieve_sensor_data(session: Session = None):
188         """
189             Retrieve and display all sensor data from Cassandra.
190
191         Args:
192             session: Active Cassandra session
193
194     
```

REFACTOR

# Hands On Exercises

*Posting the One String*

Code



Feedback



One String, Many Prompts: A Hands-On Workshop in AI Code Generation

3

```
{  
  "recorded": 1768237200,  
  "location": "den",  
  "sensor": "bmp280",  
  "measurement": "temperature",  
  "units": "C",  
  "value": 22.3  
}
```

3

SENSOR	BMP280	MICROCONTROLLER	Pico	PLATFORM	Pico-SDK	LANGUAGE	Python ✓	FRAMEWORK	Flask	DATABASE	Redis
SHT45		ESP32		ESP-IDF		Java		Spring Boot		Mongo ✓	
AHT20		ATmega		Arduino		Swift		Vapor		Cassandra	
HDC302x		SAMD21		CircuitPython		GO		Gin		Postgres	
MCP9808		nRF52		MicroPython		Rust		Actix		MySQL	

# Your Second 5<sup>n</sup> Prompts

JAVA

create a very basic web app using the **java** language **quarkus** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

create a very basic web app using the **java** language **spring** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

SWIFT

create a very basic web app using the **swift** language **vapor** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

GO

create a very basic web app using the **go** language **echo** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

create a very basic web app using the **go** language **gin** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

RUST

create a very basic web app using the **python** language **actix** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

create a very basic web app using the **python** language **axum** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

PYTHON

create a very basic web app using the **python** language **flask** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge

create a very basic web app using the **python** language **django** framework

add routes for the following:

GET / POST /echo POST /log GET /report POST or GET /purge



## More AI Lessons Learned...

- *Supported data types are a real pain...*
  - *Language, Database, JSON*
  - *Timestamp & Float Precision*
- *Multiple ‘styles’ for storing and retrieving data...*
  - *POJO for Java’s Spring Boot*
  - *Row/Column traversal*
  - *To Struct or not to Struct (obj, class, etc)*

*create a simple **python v3.12** web app using **flask v8.0** database that can bind to any address*

# !!! REMINDER !!!

Inside the docker container is where localhost resolves to.

And we have two containers running (*with exposed ports*).

So you need to change which IP address and port your framework binds to.

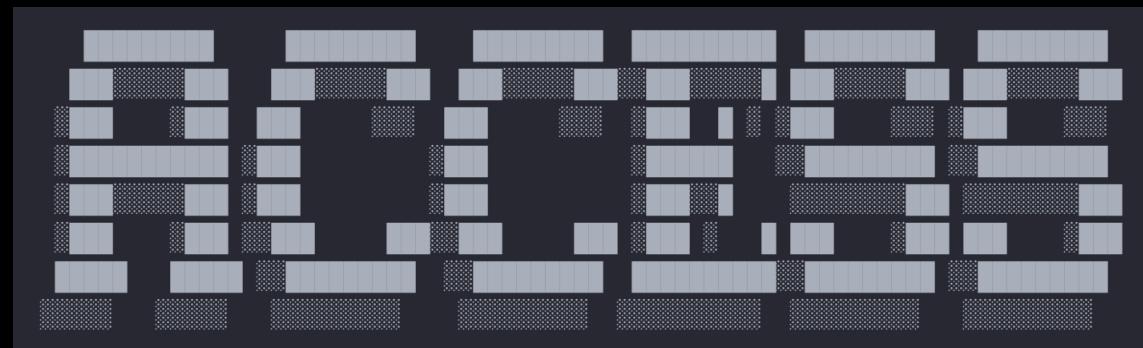
```
python app.py runserver 0.0.0.0:8080
```

*NOTE: Each framework has a different technique to configure which IP address it binds to.*

# Hands On Exercises

*Dependency Injection*

Code



Feedback



One String, Many Prompts: A Hands-On Workshop in AI Code Generation

*the goal*

# Inversion of Control



Dependency  
Injection

Template  
Method

Event  
Driven

Aspect  
Oriented

Service  
Locator

DI  
Container

Composition  
Root

Builder  
Pattern

Proxy  
Pattern

Strategy  
Pattern

Hook

Callback

Event  
Bus

Message  
Broker

Mediator  
Pattern

Decorator  
Pattern

Interceptor

Registry

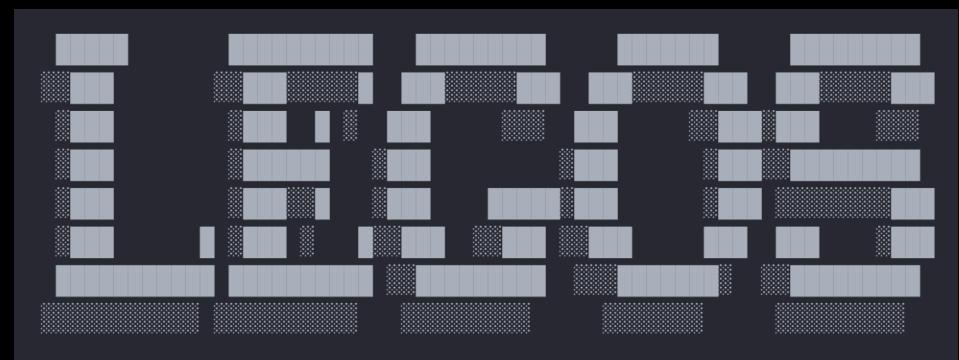
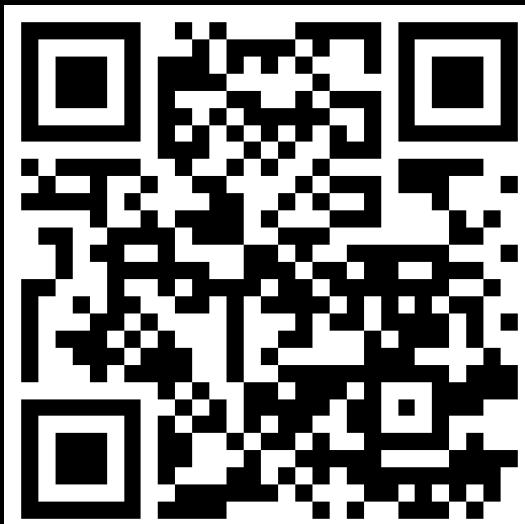
Context  
Object

Dependency  
Lookup

# Hands On Exercises

*Putting it all Together*

Code



Feedback



# ONESTRING

# PUTTING IT ALL TOGETHER



- ➊ Copy code from tasks two and three and reuse the “helper” implementation from the stretch goals in task one.
- ➋ Use AI/LLM to cherry pick database code from other projects to implement in the legos project.
- ➌ Use AI/LLM to wire everything together.

# Stretch the OneString

$$(5+n)^{(5+n)}$$

Code



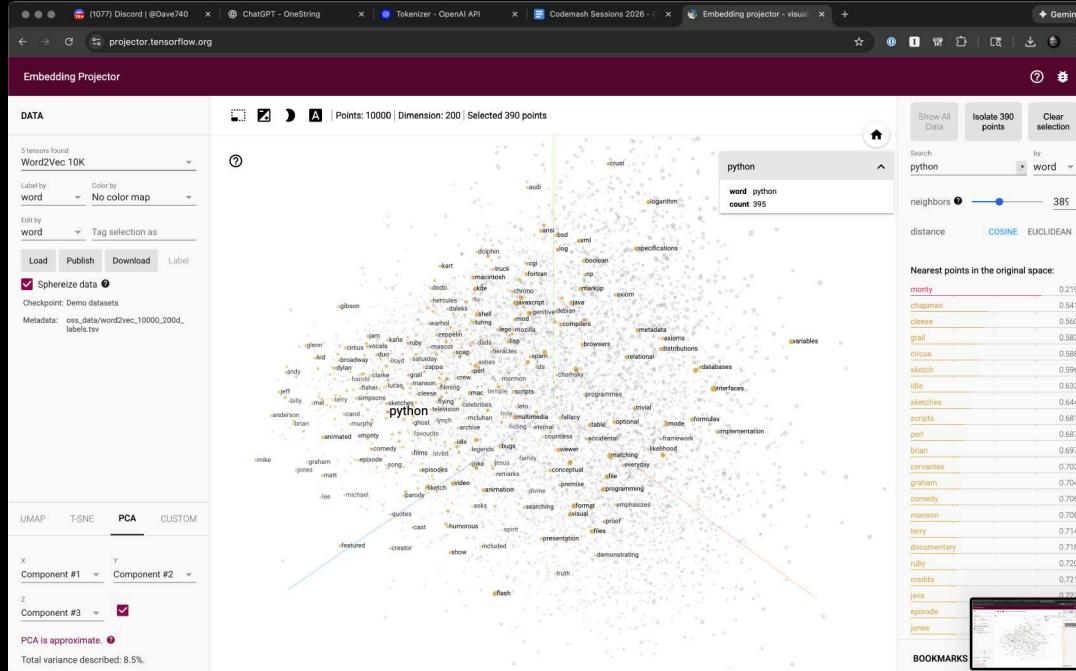
Feedback



One String, Many Prompts: A Hands-On Workshop in AI Code Generation

# are ‘prompts’ the new ‘code’?

## Embeddings



[projector.tensorflow.org](https://projector.tensorflow.org)

Preprocessor

Compiler

Assembler

Linker

## Prompt Tokenizer

GPT-5.x & O1/3   GPT-4 & GPT-3.5 (legacy)   GPT-3 (legacy)

Create a python function to connect to a mongo database

[Clear](#)   [Show example](#)

Tokens   Characters  
10   55

[5104, 261, 22752, 1114, 316, 4837, 316, 261, 63752, 7862]

[Text](#)   [Token IDs](#)

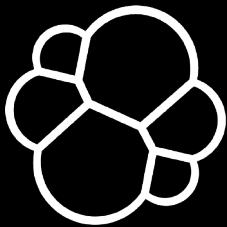
<code>



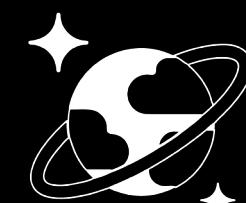
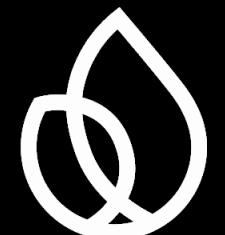
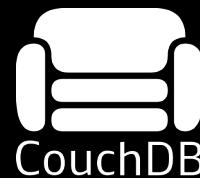
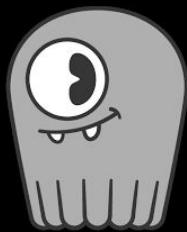
[platform.openai.com/tokenizer](https://platform.openai.com/tokenizer)



At over 1,100 episodes and  
approximately 23 minutes  
per episode, the total  
viewing time is roughly 416  
hours, or about 17.3 days.



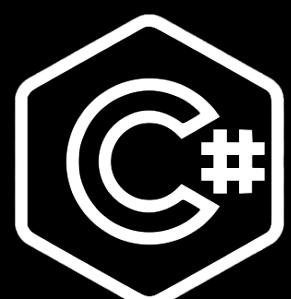
elastic



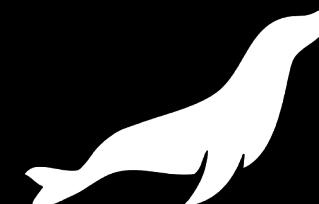
Azure Cosmos DB



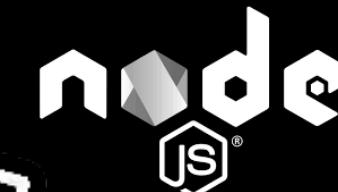
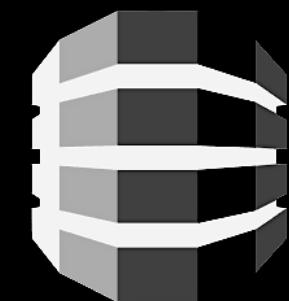
Delphi



```
{  
  "recorded": {  
    "utc": 1768237200,  
    "zone": "-05:00",  
    "dst": true  
  },  
  "location": {  
    "name": "Sandusky, Ohio",  
    "latitude": 41.4751,  
    "longitude": -82.6633,  
    "altitude": 795  
  },  
  "sensor": "bmp280",  
  "measurement": "temperature",  
  "units": "C",  
  "value": {  
    "number": 22.3,  
    "string": "22.3"  
  }  
}
```



MariaDB



django



# Even More Prompt Opportunities

## Concurrency and Thread Safety

- Is this method or function stateless and thread-safe?
- Are there any shared resources being accessed? If so, how are they synchronized?
- Does the code use any locks or mutexes? If so, how are these managed?

## Architecture, Design, and Structure

- Is there a high-level overview or documentation explaining the purpose, inputs, and outputs of the code?
- Are all external dependencies clearly listed, including their versions?
- Are there instructions on how to set up the environment and install dependencies?
- Is the code portable and can it run on different operating systems or environments?
- Does the code follow the principles of object-oriented programming (OOP) or functional programming (FP), whichever is appropriate for the context?
- Is the separation of concerns clearly maintained within the code?
- Are there any design patterns being used, and are they applied correctly?
- Does the code adhere to the single responsibility principle (SRP)?

## Error Handling and Resilience

- Does this method or function follow best practices for error handling?
- Are there any edge cases or exceptional situations that need special attention?
- Is error information being logged or propagated correctly?
- Does the code handle potential failures or unavailability of external dependencies gracefully?

## Logging and Monitoring

- What would be a better way to implement logging?
- Confirm whether the code uses an effective logging mechanism that provides useful information without overwhelming the system.
- Are there any logging frameworks or libraries being used? If so, are they properly configured?
- Can the logging level be adjusted dynamically?

## Performance and Scalability

- Can this code scale infinitely?
- Can the code handle increased traffic or workload without experiencing significant performance degradation?
- Has the code been designed to scale horizontally or vertically and can it adapt to changing conditions?
- Are there any performance bottlenecks or hotspots in the code?
- Can the code be optimized for specific hardware or infrastructure configurations?

## Security

- Are there any security vulnerabilities, such as SQL injection, XSS, or CSRF risks?
- Does the code properly handle and sanitize all user inputs to prevent injection attacks?
- Are sensitive data (e.g., passwords, API keys) handled securely, and not hardcoded?
- Does the code use secure communication protocols (e.g., HTTPS) for data transmission?
- Confirm that access controls and authorization checks are correctly implemented.

## Testing and Validation

- Is the code covered by unit tests, and do these tests cover edge cases?
- Can you generate a comprehensive set of unit tests (including test data) for this code?
- Are there clear steps provided on how to run and verify the tests?
- Does the code have meaningful assertions to validate inputs and outputs?
- How can this code be tested in an automated fashion?

## Code Quality, Readability, and Maintainability

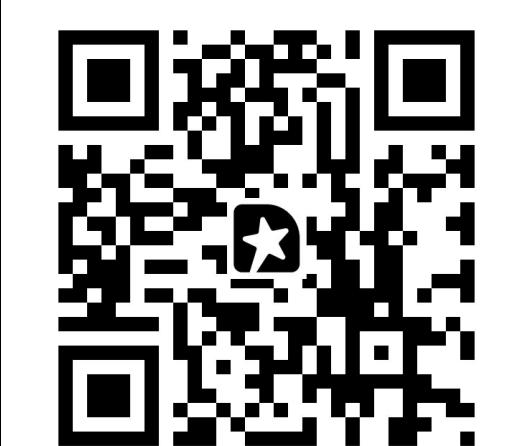
- Has this code complied with the principles of clean code?
- Are there any duplicate code blocks or overlapping logic?
- Can the code be refactored to improve readability or maintainability?
- Does the code have clear and concise comments where necessary, explaining complex logic?
- Does the code follow established style guides (e.g., PEP 8 for Python, Google Style Guide for Java)?
- Are the variable and function names descriptive and meaningful?
- Is the code modular and broken down into small, manageable functions or methods?

**TO BE  
CONTINUED**



Thank  
You!

**FEEDBACK**



One String, Many Prompts: A Hands-On Workshop in AI Code Generation