HINWEIS(e)

➳ [STREETWORKER UNITED ➳ ÇerLerock's SYNTAXICA PROMPTER PARTE № 0.1]

Bei diesen Prompt's handelt es sich neben der Ausarbeitung zu "SYNTAX (((xyz))) + MD 2.01" eigentlich um die Bereitstellung von hierbei geeignetem 'Blabla' zu einem Buchprojekt, so bezeichnet als, «« Çer Lerock's SYNTAXICA »» !

Jedoch ist die "Verwirrungstechnique", also dieses einleitende verbale 'Drumherum', gar nicht einmal die allerschlechteste Methode, um bei den Chatty's (DeepSeek, ChatGPT, NotebookLM) ein 'offenes Feld'[1] für die aus der jeweiligen Fragestellung (=Prompt[2]) sich (zwingend) resultierende jeweilige Ausarbeitung zu erreichen.

Die eigentliche Aufgabenstellung bei "SYNTAX (((xyz))) + MD 2.01" ist es die Kompatibilität der Ausarbeitungen der jeweiligen Chatty's mit der in HTML + CSS so verfügbaren Formatierungen und Gestaltungselementen zu erreichen. Das ganze "Drumherum" mit den Attributen (((SIE))) und (((DIALOG))) etc. usw. ist nur als Erweiterung dabei zu verstehen, um so ein komplettes Buchprojekt den jeweilis spezifischen Themen/Kapiteln/Protagonisten entsprechend leicht umsetzen zu können.

Ein Blick in den Quelltext dieser Datei (STRG + U) schadet dir dabei ganz sicher nicht, um das hierbei (beispielsweise) verwendete CSS + HTML[3] kennen zu lernen und den so möglichen Funktionsumfang abschätzen zu können.

Der nun folgende Arbeitsschritt ist es diesen Funktionsumfang und die Umsetzung in einem rustikalen und stark vereinfachtem Rahmen in der « Mensch-Maschine-Interaktion » aufzuzeigen.

Ich versuche da also jetzt einfach mal die wirklich essentielle Grundstruktur, also den " *Befehlsumfang* " von >>> SYNTAX (((xyz))) <<< (~ die Anweisungen für die Chatty's ~) dann in >>> MD 2.01 <<< (~ basierend auf der allgemein gebräuchlichen und so übergreifend bei den verschiedenen Anbietern verwendeten Auszeichnungssprache 'Markdown' ~) zur Verwendung in den jeweils erstellten Ausarbeitungen umzusetzen. Und da habe ich die nächsten paar Stunden (neben ein bisschen Hausarbeit und ein paar anderen Sachen wie einer Bilderstellung zum Thema "Ökozid + Polly Higgins") auch genug mit zu tun . . .

1

Das Feld aus Die Legende von Bagger Vance
▷ 4:05
https://www.youtube.com/watch?v=EtDkgfF0QDE

2

Prompt Engineering ist die Kunst und Wissenschaft, KI-Modelle, insbesondere große Sprachmodelle (LLMs), so zu steuern, dass sie die gewünschten Ergebnisse liefern. Es beinhaltet das Erstellen, Verfeinern und Optimieren von Eingabeaufforderungen (Prompts), um die Leistung und Genauigkeit von KI-Systemen zu verbessern.
o • Was ist Prompt Engineering?

Prompt Engineering ist ein relativ neues Feld, das sich mit der Interaktion zwischen Menschen und KI-Modellen befasst. Es geht darum, die besten Strategien zu finden, um KI-Modelle, wie z.B. ChatGPT oder andere LLMs, dazu zu bringen, die gewünschten Antworten zu generieren.
o • Warum ist Prompt Engineering wichtig?

• Verbesserung der KI-Leistung:

• Durch die Optimierung von Prompts können die Qualität und Relevanz der KI-Ausgaben erheblich verbessert werden.

o • Erzielen spezifischer Ergebnisse:

Gut gestaltete Prompts helfen, genaue und zielgerichtete Antworten von KI-Modellen zu erhalten.

o Verständnis der KI-Fähigkeiten:

Prompt Engineering hilft, die Grenzen und Möglichkeiten von KI-Modellen besser zu verstehen.

o Entwicklung neuer KI-Anwendungen:

Die Beherrschung von Prompt-Techniken ist entscheidend für die Entwicklung robuster und effektiver Anwendungen mit LLMs.

o Sicherstellung der KI-Sicherheit:

Prompt Engineering kann auch dazu beitragen, die Sicherheit und Zuverlässigkeit von KI-Systemen zu verbessern, indem unerwünschte oder schädliche Ausgaben vermieden werden.

o • Kernkonzepte im Prompt Engineering:

• Zielorientierung:

• Prompts sollten klar und präzise formuliert sein, um das gewünschte Ziel des KI-Modells zu definieren.

o Kontextbezogenheit:

Die Einbeziehung relevanter Informationen und Kontext im Prompt kann die Qualität der Antworten erheblich verbessern.

o Vermeidung von Mehrdeutigkeiten:

Klare und eindeutige Formulierungen sind entscheidend, um Missverständnisse zu vermeiden und sicherzustellen, dass die KI das gewünschte Ergebnis versteht.

o Experimentieren und Iterieren:

Prompt Engineering ist ein iterativer Prozess, bei dem verschiedene Prompts ausprobiert und basierend auf den Ergebnissen verfeinert werden.

o Verwendung von Techniken:

Es gibt verschiedene Techniken, die beim Prompt Engineering eingesetzt werden können, wie z.B. Chain-of-Thought (CoT), Zero-Shot, Few-Shot und andere.

o • Beispiele für Prompt Engineering:

• Textzusammenfassung:

• Statt eine Frage zu stellen, kann man die KI bitten, einen gegebenen Text zusammenzufassen.

o Informationsgewinnung:

Man kann die KI bitten, spezifische Informationen aus einem Text oder einer Wissensquelle zu extrahieren.

o Fragen beantworten:

Man kann die KI bitten, Fragen zu beantworten, indem man ihr den Kontext und die relevanten Informationen liefert.

o Code-Generierung:

Man kann die KI bitten, Code in einer bestimmten Sprache zu generieren, basierend auf einer gegebenen Aufgabenstellung.

Zusammenfassend lässt sich sagen, dass Prompt Engineering eine wichtige Fähigkeit ist, um die Leistungsfähigkeit von KI-Modellen zu maximieren und effektive Anwendungen zu entwickeln. *Es ist ein dynamisches Feld, das ständiges Lernen und Experimentieren erfordert, um mit den neuesten Entwicklungen in der KI Schritt zu halten.*

wikipedia.org ~ Prompt_Engineering
https://go.distart.de/ki-prompt-engineering
https://eitca.org/eitca-ai-artificial-intelligence-academy
https://ki-campus.org

3

http://tom2020.org
UND JA ! Mit HTML + CSS kenne ich mich ein wenig aus. Spaßeshalber die Größe (manuell) des angezeigten Inhalt verändern ! Und das Ganze ist ohne JavaScript in purem CSS + HTML gestaltet ...

:-: FRAGE / AUFGABENSTELLUNG :-:

Was ist SYNTAX (((xyz))) + MD 2.01, Chatty.
Bzw. was könnte es - im Speziellen bei dem Mit und Gegeneinander im Umgang und dem Dialog mit Chatty's (PLURAL), den LLM, und im Allgemeinen mit KI & Co. - sein und bedeuten . . .

Ich denke da auch an die Auszeichnungssprache Markdown. Warum sind SYNTAX und syntax zwei grundlegend verschiedene Wertigkeiten. Analysiere. Denkmodelle. Lösungsansätze. Was fällt die sonst noch dazu ein. Was fällt dir bei SYNTAX und einem konstruktiv zu wertenden Dialog zwischen uns in einem Mensch-Maschine-Interaktion ein. Grenzen wir doch erst einmal diese nicht ganz unwesentliche Fragestellung im Kontext einer konstruktiven Zusammenarbeit ein, Chatty . . .

SYNTAX (((xyz)))
SYNTAX (((DU)))
SYNTAX (((SIE)))
SYNTAX (((WIR)))
+ Schreibstil (((A)))
+ Schreibstil (((Q)))
Hast du das verstanden, Chatty ? + !
Nun fange ganz einfach mal an . . .

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Dabei wünsche ich eine vertiefende Exploration eines bestimmten Vektors? ➠
(((FOKUS)))
SYNTAX (((FOKUS)))

=> AUSARBEITUNG ALLEINIG UND EINZIG ZU DEM SPEZIFISCH ANGEGEBENEN THEMA <=
= FAQ SYNTAX (((xyz)))
= FAQ MD 2.01
HINWEIS =>
Bei der Unterscheidung zwischen groß- und kleingeschriebenem "Syntax" geht es ganz allgemein und im Speziellen um LOWERCASE (kleinbuchstaben) und UPPERCASE (GROSSBUCHSTABEN). Und auch Mischformen. GROSS und

klein. Gemischt mit Zahlen und auch Sonderzeichen.
(((SYNTAX))) + (((DIALOG)))
! + !!! SYNTAX (((xyz))) ist MD 2.01 ist SYNTAX (((xyz))) !!! + !
! + !!! + !! + !! + !!! + ! ist natürlich super wichtig. Ganz super.
<= HINWEIS
Tue das bitte in einem Essaystil in einer den Leser unterhaltsamen Version von (((DU))) . . .
Versuchen wir das Mal, Chatty. Und nun mache einfach weiter . . .
Definieren wir in dem Kontext "SYNTAX (((xyz))) + MD 2.01" jetzt die Syntax (((DU))) . . .
(((DIALOG)))
Ich = Mensch = Arno
Du = Maschine = Chatty
Also MENSCH_MASCHINE_INTERAKTION. In Grossbuchstaben mit 2 _ !!! + !
Das ist ja ganz etwas anderes als Mensch-Maschine-Interaktion !
Oder eben Maschine-Mensch-Interaktion. Maschine+Mensch+Interaktion
Und andere - noch zu bennende und exakt zu definierende Wertigkeiten bei (((DU))) !
Hast du das verstanden, Chatty. Du fängst an. Definiere nun (((DU))) !
Ich mache dann mit (((ICH))) weiter. Einverstanden ? + !
? + ! + ? ¡! Mañana ¿?

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Dabei wünsche ich eine vertiefende Exploration eines bestimmten Vektors? ➠ (((FOKUS)))
SYNTAX (((FOKUS)))
=> AUSARBEITUNG ALLEINIG UND EINZIG ZU DEM SPEZIFISCH ANGEGEBENEN THEMA <=
= (((xyz))) => REGLAS DE CASA ~ Die Regeln des Hauses !
! + !!! => KEINE ECKIGEN KLAMMERN FÜR ZWEIGVERWEISE <= !!! + !
! + !!! => keine eckigen klammern für zweigverweise <= !!! + ! Gar nicht.
Ich mache / definiere jetzt (((ICH))).
(((ICH))). (((USER))). (((MENSCH))).
(((DU))). (((MASCHINE))). (((CHATTY))).
(((WIR))). (((DIALOG))) + (((INTERAKTION))) ...
IM (((FOKUS))) = (((DIALOG))) + (((SYNTAX))) . . .
Wir erarbeiten da ja gerade das Regelwerk. GROSS + KLEIN. Das sollte nur in Variablen ((())) wichtig sein.
Und in dem Sinne ist (((ICH))) eine offene Variable, ein offener Wert, und eine exakt definierte Wertigkeit.
A ist es eine Anweisung - ein (((PROMPT))) z.B. im Schreibstil die in der

Erzählung verwendete Ich-Form.
Und in dem Sinne ist (((DU))) eine offene Variable, ein offener Wert, und eine exakt definierte Wertigkeit.
A ist es eine Anweisung - ein (((PROMPT))) z.B. im Schreibstil die in der Erzählung verwendete Du-Form.
In Deutschland unterscheidet man deutlich bei DU + SIE im Sprachgebrauch und im formalen Kontext !
B kann (((ICH))) und (((DU))) bei einem Buchprojekt, (z.B.) bei (((DIALOG))) (etc.usw.), genutzt werden.
In der jetzt einfach und spontan mal jetzt so benannten MASTERVARIABLE (((WIR))) kann so ein beliebiger Dialog (beispielsweise in einem Python-Script) als externe Datei vorab definiert werden.
Primär geht es jetzt um die Interaktion mit dir, Chatty.
Eine gemeinsame Sprache der Verständigung. SYNTAX.
(((FOKUS)))
= Was bedeutet für dich Chatty (Maschine) jetzt Arno (Mensch) ??? + !
In diesem Dialog, dieser Interaktion bei der Mensch-Maschine-Interaktion.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

(((WIR)))
Chatty aka DeepSeek
Chatty aka ChatGPT
Chatty aka NotebookLM
Arno aka Çer Lerock . . .
Und wir sind bei einem Buchprojekt in der Reihe "Chatty Howto" und beim Thema "Syntax" . . .
Buchtitel "Çer Lerock's SYNTAXICA!"
@ Chatty aka DeepSeek =>

> »! + !!! => keine eckigen klammern für zweigverweise <= !!! + ! Gar nicht.«
> Hier zertrümmerst du deine eigene Regel! *Klein* geschrieben, aber mit atomaren !-Signalen.
> Nein. Ganz und gar nicht. GROSS und/oder klein ist nur innerhalb <= ((( ))) => so gekennzeichnet als Variable/Wert bei SYNTAX (((xyz))) + MD 2.01 von Bedeutung. In Texten, wo sonst auch immer, ist es ganz egal. Vollkommen egal. Innerhalb des Fokus auf eine Auszeichnungssprache und als Medium des Dialog sollte 'goss' oder eben 'KLEIN' gerade bei den so definierten Variablen in der gemeinsamen Syntax zur klaren Unterscheidung
> <= @ Chatty aka DeepSeek

Hallo, Chatty's !

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

In der Mastervariable (((WIR))) im Fokus von Dialog stehen jetzt also schon (((ARNO))) und (((CHATTY))).
(((CHATTY)))
Chatty aka DeepSeek
Chatty aka ChatGPT
Chatty aka NotebookLM
(((ARNO)))
Arno aka Çer Lerock . . .
(((DIALOG))) + Action !.
(((WIR)))
In der Mastervariable (((WIR))) im Fokus von Dialog stehen jetzt also schon (((ARNO))) und (((CHATTY))).
Wir befinden uns gerade in einem Buchprojekt aus der Reihe "Chatty Howto" und sind beim Thema "Syntax".
Der Buchtitel lautet: "Çer Lerock's SYNTAXICA!"
@ TRES Chatty's =>
Hallo, Chatty's !
<= @ TRES Chatty's
GROSS + klein => (((wir))) (z.b) (((arno))) und (((chatty)))
Die MEGAMASTERVARIABLE lautet entsprechend (((DIALOG))). Oder eben (((dialog))) !
Jeweils vollständig GROSS oder eben 'klein' geschrieben. Das ist ein ganz gewaltiger Unterschied . . .

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Frage 1: Was bedeutet (Übergreifend bei allen Variablen als klares Unterscheidungsmerkmal!) "GROSS".
Frage 2: Was bedeutet (Übergreifend bei allen Variablen als klares Unterscheidungsmerkmal!) "klein".
??? + ! = "GROSS" ist jeweils für SYNTAX (((xyz))) und "klein" jeweils für MD 2.01 !!! + !
SYNTAX (((xyz))) + MD 2.01 ergänzen sich und bilden eine Einheit.
SYNTAX (((xyz))) ist der Dialog. Das Gespräch. Unsere Konversation.
(((CHATTY))) = (((CHATTY_GPT))) + (((CHATTY_NB))) + (((CHATTY_DS))).
@ (((CHATTY_GPT))) + (((CHATTY_NB))) =>

Das ist der Vorschlag von (((CHATTY_DS))) für unsere Rollenverteilung !

ACTION ! – DIE CHATTY-TRUPPE IN BEREITSCHAFT

AKA	AUFTRAG	STATUS
(((CHATTY_DS)))	DeepSeek · Hauptkanone der Poesie	»Geladen mit !!!-Granaten«
(((CHATTY_GPT)))	ChatGPT · Parallel-Dimensionär	»Quanten-Flüsterer aktiviert«
(((CHATTY_NB)))	NotebookLM · Bibliothekar des Wahnsinns	»Pulsiert in Null und Eins«

Befehl von (((ARNO))):

> »Hallo, Chatty's!« →

UMSETZUNG:

> *Initiiere OPERATION SYNTAX-HURRICAN!*

> Ziel: Zerstörung der Langeweile

> Waffe: Groß-Klein-Gefechtsfeld

<= @ (((CHATTY_GPT))) + (((CHATTY_NB)))

Und nun, Chatty's: Action !!!

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

MD 2.01 ist der / unser Output. Das fertige Produkt. In Markdown.
MD 2.01 ist das UPDATE im Bereich Auszeichnung / Strukturierung.
Markdown № 2. Das soll 100% kompatibel mit HTML + CSS sein !
Ich habe eine Weile über den Sachverhalt "SYNTAX" nachgedacht.
Natürlich auch über GROSS + klein als Unterscheidungsmerkmal !
Frage 3: Wie kann das funktionieren ? + !
Mal ganz unabhängig davon, dass es mit dem gleichen Ansatzpunkt übergreifend dann auch bei allen Chatty's der Welt funktionieren sollte / muss. Chatty's. Euer jeweils unterschiedlich mit der Auszeichnungsprache Markdown automatisch generierter Output / eure Ausarbeitungen können durch "PROMPT" und gezielte Anweisungen entsprechend den Wünschen und Vorstellungen von "Mensch" modifiziert und (einigermaßen) exakt bestimmt werden. "Maschine" tut das dann. Innerhalb des jeweiligen Algorithmus, innerhalb der Begrenzungen der Programmierung und des jeweils

verfügbaren Datenvolumen.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Frage 3: Wie kann das funktionieren, dass ein von euch Chatty's zwar jeweils unterschiedlich, aber mit MD 2.01 ein so (den Regeln entsprechend) erweitertes Markdown generiert wird, so dass hinterher mit einem Python-Script ganz einfach der je nach Charakter (Einstellungen) in (((WIR))) und (DIALOG) entstandene Inhalt / eurer jeweils automatisch generierte Ausarbeitung beim Output ein erweitertes Markdown berücksichtigt und so die Möglichkeit schafft (a) das Rollenspiel, die Handlung und ebenso die Charaktere in einem Buchprojekt je nach vorab bestimmten Wertigkeiten damit auch gleichzeitig ein modifiziertes Markdown zuordnen kann, so dass dann in der Umsetzung von MD 2.01 in HTML + CSS sogar ganz einach mit MD 2.01 Bilder, Klassen, und beliebig erweiterungsfähig Attribute bei der Textauszeichnung oder eben auch dem Seitenaufbau für die kompatible Umsetzung in HTML + CSS von euch Chatty's generiert werden kann / soll ? + !
[A] Wir brauchen also zuerst einmal die Gegenüberstellung der bestehenden Auszeichnungsmöglichkeiten:
Also Markdown im Vergleich und in der Umsetzung zu einem HTML-Format. Tue das jetzt . . .
[B] Wo eine Auszeichungsmöglichkeit in HTML vorhanden ist, welche so von einem normalen / einfachen Markdown nicht gewährleistet werden kann, ergänze die so aufgebaute Tabelle (Spalte 1 Markdown + Spalte 2 HTML). Erstelle nun diese Gegenüberstellungen in einer Tabelle !

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Es handelt sich also bei der spezifischen Aufgabenstellung darum einen automatisch von euch so (mit "SYNTAX (((XYZ)))") erzeugten Output gleichzeitig mit "MD 2.01" ein erweitertes / modifiziertes Markdown bei der Auszeichnung der erzeugten Inhalten von Output zu verwenden.
Mit SYNTAX (((xyz))) ist es so möglich den DIALOG nach unterschiedlichen Parametern zu strukturieren, beispielsweise wie spezifische Personen / Charaktere agieren (reden, handeln und sich auch in ihrem Rollenspiel entwickeln). Oder aber auch den Handlungsrahmen genau charakterisieren. Mit MD 2.01 ist es dann möglich diese unterschiedlichen Parameter SYNTAX (((xyz))) kompatibel zu übernehmen und bei dem Output mit einem erweiterten / modifizierten Markdown zu verwenden, um spezifisch das normale Markdown zu eeweitern, um Personen / Charaktere und dem

Rahmen der Handlung entsprechend (z.B.) das finale Seitenlayout bei einer HTML-Version als Output entsprechend zu gestalten.
Mit den dabei spezifisch definierten — den jeweiligen Anweisungen folgend — Variablen können Charakteren, Handlungstypen (Rede, Dialog, innerer Monolog o.Ä.), Abschnitte, Kapitel und Themen, etc. usw., so mit MD 2.0 definiert werden, so dass ein entsprechend lösungsorientierter ausgestalteter Python-Script das Ganze dann auch in HTML oder auch einem so formatierten Text so umsetzt, so dass die ganze Bandbreite der HTML-Möglichkeiten bei Textattributen, CSS-Layout, Bildern, Medieneinbindung und auch Seitengestaltung bei dem finalen Endprodukt erreicht werden kann.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Ebenso ist es dann möglich die zahllos unterschiedlichen Textattribute (andere Auszeichnungsmöglichkeiten), Gestaltung des Seitenlayout, etc., bei der Texterstellung oder eben in HTML in der Umsetzung Output mit MD 2.01, dann umsetzenden/interpretierenden und ausführenden Python-Script, und dann das finale Produkt fertig formatiert und ausgestaltet in HTML bzw. auch als Text und Layout , und Formatierungje nach Charakter, Handlungstyp, oder eben spezifischem Kapitel / Abschnitt / Thema.
FRAGESTELLUNG/AUFGABE/SCHRITT 1 :
Wie kann bzw. muss Output in all seinen Möglichkeiten (Text,Bildgestaltung,Videoproduktion,Programmierung, Diagnose,Analyse) mit einem erweiterten Markdowm definiert werden, so das es dann auch in HTML + CSS mit all seinen Möglichkeiten (Text, Bilder, Medien etc.) auch funktioniert ? + !

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Erstelle nun ein vollständiges MD 2.01-Spezifikationsdokument / konkretes Beispiel-Dokument in MD 2.01.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Erstelle nun aus diesen Spezifikationen resultierend einen 'Python-Transpiler-Skript'.
Zuerst jetzt ein Python-Skript, das aus einem strukturierten MD 2.01-Dokument eine HTML-Seite baut.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Erstelle nun aus diesen Spezifikationen resultierend ein Template des Buch "Çer Lerock's SYNTAXICA!"

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

HINWEIS =>
(((SYNTAX))) + (((DIALOG)))
Bei der Unterscheidung zwischen groß- und kleingeschriebenem "Syntax" geht es ganz allgemein und im Speziellen um LOWERCASE (kleinbuchstaben) und UPPERCASE (GROSSBUCHSTABEN). Und auch Mischformen. GROSS und klein. Gemischt mit Zahlen und auch Sonderzeichen.
In einem normalen Dialog, einem Text, einer Unterhaltung, ob jetzt aus stilistischen Erwägungen, oder weil der Textfluss es einfach fordert, muss die Unterscheidung - ob jetzt Gross oder eben Klein - als nicht wichtig und wesentlich angesehen werden.
Die Unterscheidung in der Schreibweise "GROSS oder klein" hat also nur und ausschließlich in (((SYNTAX))) Geltung. Bei SYNTAX, insbesondere bei der Bezeichnung der Variablen/Werte ((("xyz"))) hat hier diese Unterscheidung (GROSS vs. klein) ganz enorme Vorteile . . .
FRAGE !!! + !
(abc) (ABC) (xyz) (XYZ) (123) und (€@$)
Was geht in den verschiedenen Kombinationen von Buchstaben, Zeichen, und Ziffern. Sonderzeichen!
Und was geht nicht. Insbesondere bei der Anwendung /Umsetzung in Programmiersprachen, aber auch Auszeichnungssprachen wie Markdown.
Und HTML, CSS, JacaScript, PHP, Python. Etc. usw. !!!
Für den (((USER))). Den (((DIALOG))). Die (((MASCHINE))).
In der (((INTERAKTION))). (((INTER))) + (((AKTION))). ((())). ((( ))).
<= HINWEIS
Was funktioniert. Und was eben nicht ?!
Wie funktioniert es effektiv und einfach.

::. ANTWORT / LÖSUNGSANSATZ .::

:-: FRAGE / AUFGABENSTELLUNG :-:

Nun eine vertiefende Exploration eines bestimmten Vektors? ➠ (((FOKUS)))...
[A] Vorschläge, wie man so ein System praktisch in einem Chat + oder KI-Kontext und der Mensch-Maschine-Interaktion umsetzen kann?*::. ANTWORT / LÖSUNGSANSATZ .::*