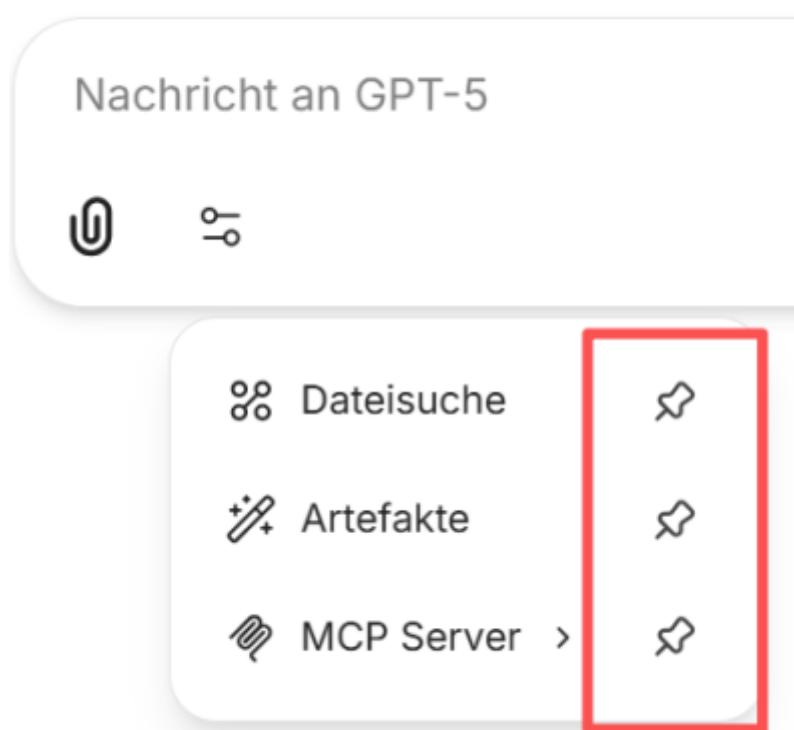


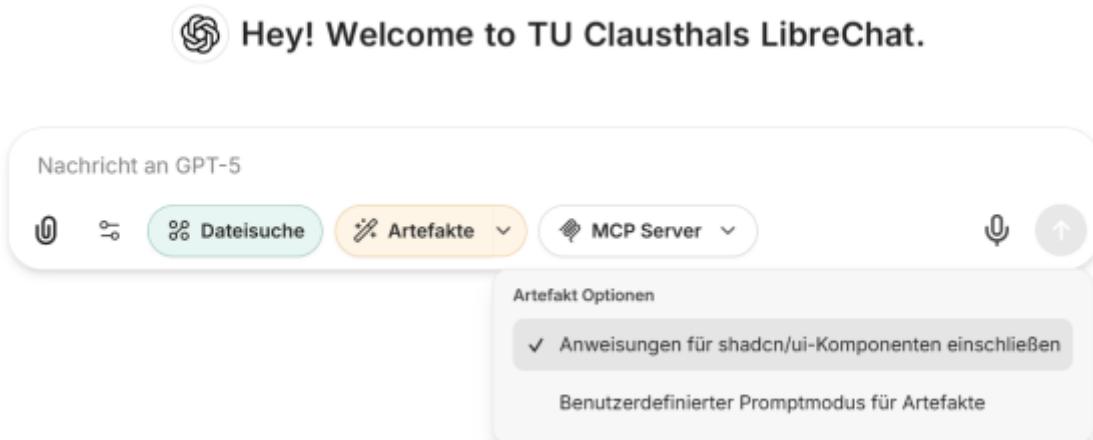
Empfohlener Ablauf zur Nutzung

Nutzung von Artefakte

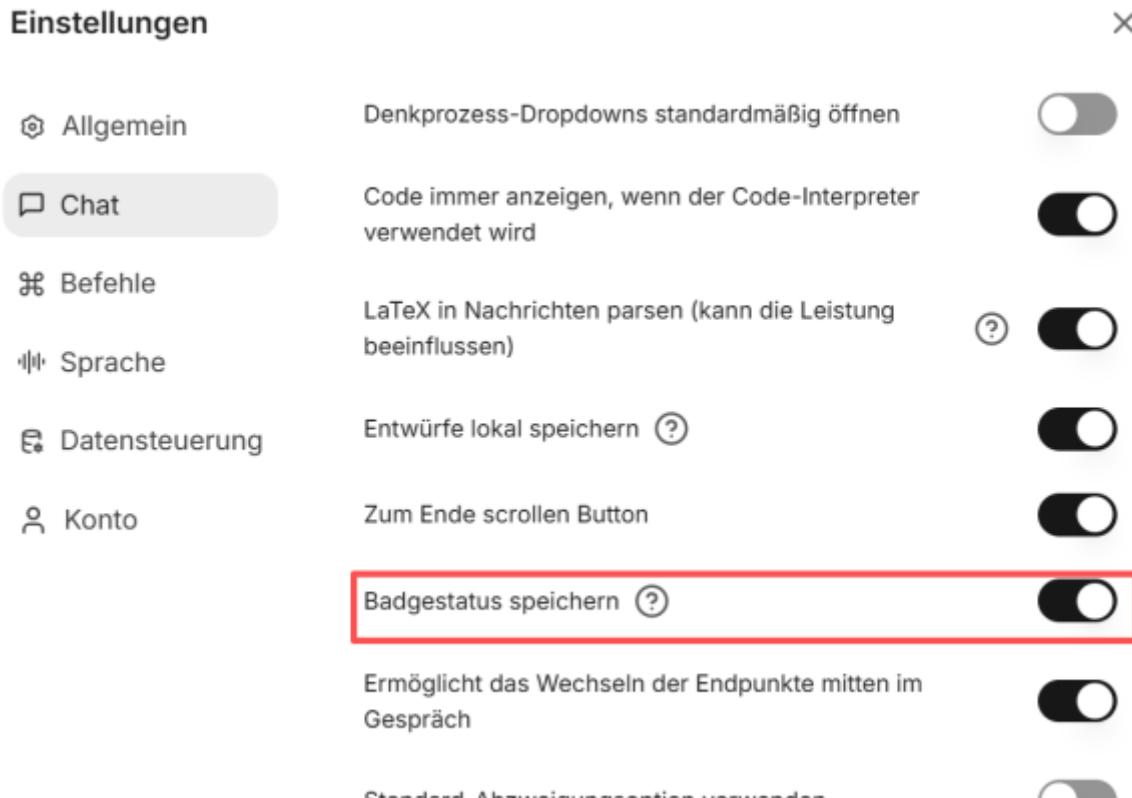
Verwenden Sie Artifacts, um die Anzeige und den Download von Dateien zu optimieren. Die gezeigten Screenshots illustrieren den empfohlenen Ablauf zur Nutzung von Dateien und Artifacts in LibreChat:



- Oben ist zu sehen, wie im Chatfenster verschiedene Upload- und Anzeigemethoden zur Auswahl stehen – insbesondere „Dateisuche“ und „Artifacts“. Diese Optionen lassen sich im Nachrichtenbereich von LibreChat auswählen und sind für die Weiterverarbeitung und Darstellung von Dateien gedacht.



- Im mittleren Ausschnitt wird unter den Einstellungen verdeutlicht, dass die Artifact-Funktion aktiviert ist und weitere Upload-Möglichkeiten bereitstehen. Die rote Markierung hilft Nutzern, die relevanten Schalter und Bereiche im Interface schnell zu finden.



- Durch das Speichern des Badgestatus werden die ausgewählten Einstellungen standardmäßig beibehalten, sodass die Auswahl nicht jedes Mal erneut getroffen werden muss.

The screenshot shows a user interaction with a "Python Script Generator" tool. The user has asked for a simple Python script to calculate factorials. The generated code is as follows:

```

#!/usr/bin/env python3
"""
A simple, reusable Python script that:
- Greets the user
- Computes the factorial of a number
- Lists .py files in a directory
- Optionally saves a short report to a file

Usage examples:
python simple_script.py --name Alice --number 5 --path .
python simple_script.py --name Bob --number 10 --path ./src --save report.txt
"""

import argparse
import logging
import os
from typing import List

def greet(name: str) -> str:
    """Return a greeting for the given name."""
    return f"Hello, {name}!"

def factorial(n: int) -> int:
    """Compute factorial iteratively to avoid recursion depth issues."""
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

def list_python_files(path: str) -> List[str]:
    """Return a list of .py files in the given directory (non-recursive)."""
    try:
        entries = os.listdir(path)
    except OSError as e:
        logging.error(f"Failed to list directory {path}: {e}")
        return []
    return [f for f in entries if f.endswith(".py") and os.path.isfile(os.path.join(path, f))]

def make_report(name: str, number: int, path: str) -> str:
    """Create a small report string describing operations performed."""
    files = list_python_files(path)
    fact = factorial(number)
    lines = [
        f"Name: {name}",
        f"Computed factorial({number}) = {fact}",
        f"Found {len(files)} Python file(s) in {path}.",
    ]
    lines.extend(f" - {fn}" for fn in files)
    return "\n".join(lines)

def parse_args():
    p = argparse.ArgumentParser(description="Simple Python utility script")
    p.add_argument("--name", "-n", default="World", help="Name to greet")
    p.add_argument("--number", "-k", type=int, default=5, help="Number to compute factorial for")
    p.add_argument("--path", "-p", default=".", help="Directory to list .py files from")
    p.add_argument("--save", "-s", help="Optional path to save the report")
    p.add_argument("--verbose", "-v", action="store_true", help="Enable verbose logging")
    return p.parse_args()

def main():
    args = parse_args()
    logging.basicConfig(level=logging.DEBUG if args.verbose else logging.INFO, format=f"[{levelname}] {args}")
    try:
        report = make_report(args.name, args.number, args.path)
    except Exception as e:
        logging.error(f"An error occurred: {e}")
    print(report)

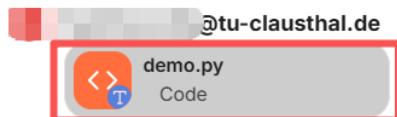
```

- Der große Screenshot zeigt einen typischen Anwendungsfall. Die Dateien, die über diese Schaltfläche heruntergeladen werden, weisen derzeit Formatierungsprobleme auf, die jedoch in Zukunft behoben werden. Vorläufig werden Standarddateien weiterhin als „index.html“ gespeichert. Daher ist eine manuelle Umbenennung der Datei weiterhin erforderlich, bsp. als demo.py.

Nutzung von Text mit OCR

Dateien, die im Textformat hochgeladen werden, werden im Klartextformat an das Sprachmodell gesendet. Die spezifischen Unterschiede sind wie folgt:

- Ein Anwendungsfall:



Was ist der Zweck dieses Codes?

GPT-4.1

Der Zweck des bereitgestellten Python-Skripts (simple_script.py) ist es, eine einfache, vielseitige Befehlszeilen-Utility zur Verfügung zu stellen, die die folgenden Aufgaben erledigt:

1. Begrüßung des Benutzers:

Das Skript begrüßt einen Benutzer mit einem über die Kommandozeile übergebenen Namen (z.B. "Hello, Alice!").

2. Fakultätsberechnung:

Es berechnet die Fakultät einer angegebenen Zahl (z.B. $5! = 120$).

3. Auflisten von Python-Dateien:

Im angegebenen Verzeichnis (Ordner) listet das Skript alle dort vorhandenen Python-Dateien (*.py) auf (nicht rekursiv).

4. Erstellung eines Berichts:

Aus den Informationen (Begrüßung, Fakultät, Dateiliste) erstellt das Skript einen kurzen Bericht und gibt ihn auf der Konsole aus.

5. Optional: Speichern des Berichts in eine Datei:

Mit dem Parameter --save kann der Bericht in eine Textdatei geschrieben werden.

6. Optional: Ausführliche Protokollierung:

Mit --verbose werden zusätzlich ausführliche Protokollinformationen im Terminal ausgegeben.

Zusammengefasst:

Das Skript ermöglicht es, mit einfachen Kommandozeilenbefehlen statistische und explorative Aufgaben (Begrüßung, mathematische Berechnung, Dateisystem-Inspektion) schnell durchzuführen und die Ergebnisse optional in einer Datei zu speichern.

Typische Anwendungsbeispiele:

- Einen einfachen Bericht über Python-Dateien in einem Projektordner erstellen
- Die Fakultät einer beliebigen Zahl berechnen
- Skripte testen, Python-Utility-Funktionen demonstrieren

Wenn du möchtest, kann ich die Funktionsweise auch Zeile für Zeile erklären oder einzelne Bereiche detailliert aufschlüsseln!



- Beim Hochladen als reiner Text verwendet das Sprachmodell keine zusätzlichen Werkzeuge, wie

zum Beispiel das Tool „file search“.

- Hochgeladene Dateien sind in der rechten unteren Ecke mit einem „T“-Symbol markiert, das anzeigen, dass es sich um Textinhalt und nicht um eine Datei handelt.
- Der Textinhalt existiert nicht in Form einer „Datei“ und kann daher bei längerem Gebrauch nicht mehrfach wiederverwendet werden. Selbst für das Sprachmodell ist es schwierig, in großen Mengen unstrukturierter historischer Daten spezifische Informationen schnell und präzise zu finden.



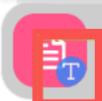
Zusammenfassung des obenstehenden Textes

GPT-4.1

Das Dokument beschreibt ein neues Framework namens Fourier Feature-Embedded Physics-Informed Neural Network-based Topology Optimization (FF-PINNT) für die Topologieoptimierung von geometrisch nichtlinearen Strukturen. Dieses Framework nutzt physik-informierte neuronale Netzwerke (PINNs) mit Fourier-Feature-Einbettung und periodischen Aktivierungsfunktionen, um die Modellierung nonlinearer partieller Differentialgleichungen zu verbessern.



 [tu-clausthal.de](#)



ssrn-5203323.pdf

Document

Zusammenfassung der oben genannten Dokumente

GPT-4.1

 Cancelled 

Assistant used file_search

```
{  
    "query": "Zusammenfassung Fourier feature embedded physics-informed neural network-based  
    topology optimization nonlinear structures"  
}
```

Result

Error processing tool: Tool response format is "content_and_artifact" but the output was not a
two-tuple.

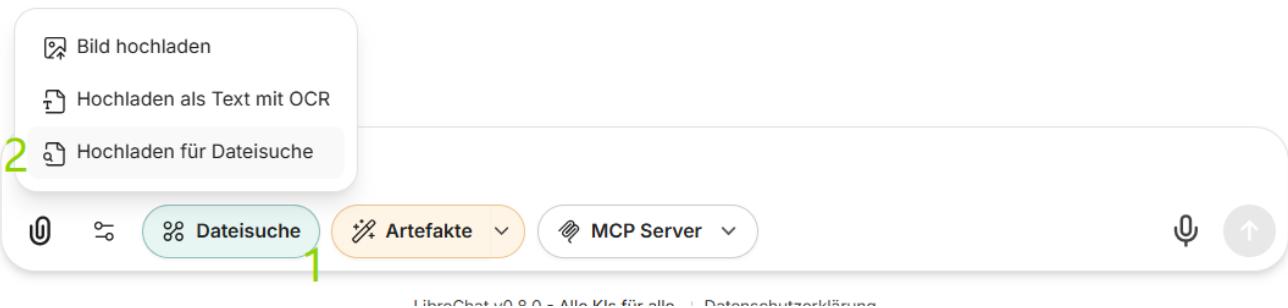
Result: "No files to search. Instruct the user to add files for the search."

Ich habe momentan keine Dokumente geladen, um eine Zusammenfassung zu erstellen. Bitte
laden Sie die entsprechenden Dokumente hoch, damit ich Ihnen eine Zusammenfassung
liefern kann.



Diese Methode eignet sich daher besonders für kurzfristig wichtige Dateien, wie z.B.
Textkorrekturen oder Übersetzungen, die nach ihrer Verwendung nicht erneut benötigt
werden. Aufgrund der begrenzten Serverressourcen trägt das Hochladen von Dateien
im Textformat für den kurzfristigen Gebrauch ebenfalls dazu bei, unseren
Ressourcenverbrauch zu reduzieren, zusätzlich zur regelmäßigen Bereinigung nicht
benötigter persönlicher Dateien in LibreChat.

Nutzung von Dateisuche



Das Bild zeigt einen Ausschnitt aus der Dateiupload- und Funktionsleiste in LibreChat. Zunächst sieht man die Schaltfläche „Dateisuche“ (Nummer 1). Erst wenn man auf diese Schaltfläche klickt, öffnet sich das oben angezeigte Auswahlmenü (Nummer 2), in dem verschiedene Optionen für den Datei-Upload erscheinen. Dazu gehören „Bild hochladen“, „Hochladen als Text mit OCR“ und „Hochladen für Dateisuche“.

Was ist der Zweck dieses Codes?

GPT-4.1

Dieser Code ist ein einfaches, wiederverwendbares Python-Skript, das folgende Aufgaben erfüllt (siehe Datei demo.py):

1. Begrüßt den Benutzer mit einem personalisierten Gruß.
2. Berechnet die Fakultät (factorial) einer angegebenen Zahl.
3. Listet alle .py-Dateien in einem angegebenen Verzeichnis (nicht rekursiv) auf.
4. Kann optional einen kurzen Bericht über diese Operationen in eine Datei speichern.
5. Unterstützt optionale Parameter und eine Verbose-Option für detaillierte Logausgaben.

Das Skript kann mit verschiedenen Argumenten über die Kommandozeile ausgeführt werden, z.B.:

```
css
python simple_script.py --name Alice --number 5 --path .
```

Zweck: Es dient als Beispiel für grundlegende Aufgaben wie Benutzerinteraktion, mathematische Berechnung, Dateioperationen und Berichtserstellung in Python.



Dateien anhängen		
Dateien filtern...		
	Name ↑↓	Datum ↑↓
	demo.py	6 Oct 2025
	demo.py	6 Oct 2025

Direkt-Link:

https://doku.tu-clausthal.de/doku.php?id=sonstige_dienste:ki-dienste:librechat:empfohlener_ablauf_zur_nutzung&rev=1759902448

Letzte Aktualisierung: 05:47 08. October 2025

