



Ein Word- oder ODT-Dokument in eine statische, MkDocs-kompatible HTML-Website mit den KI-Modellen Gemini 3 und ChatGPT 5.2 konvertieren

Serge Tahé, Januar 2026

[| French version](#) | [| English version](#) | [| Spanish version](#) |

Diese Website wurde mit dem Konverter [Word oder ODT -> HTML] erstellt, der im Januar 2026 von den KI-Systemen Gemini 3 und ChatGPT 5.2 entwickelt und in diesem Dokument beschrieben wurde.

Mit dem Gemini/ChatGPT-Konverter wurden mehrere Websites aus ODT-Dokumenten von LibreOffice und Word-Dokumenten erstellt:

Java

- [\[Introduction to the Java Language \(1998\)\]](#);
- [\[Introduction to Web Programming in Java with Servlets and JSP Pages \(2002\)\]](#);
- [\[Introduction to Struts 1.x through Examples \(2003\)\]](#);
- [\[The Basics of MVC Web Development in Java Through Examples \(2006\)\]](#);
- [\[Java 5 Persistence Through Practice \(2007\)\]](#);
- [\[Building a Java EE Web Service with the NetBeans 6.5 IDE and the GlassFish Java EE Server \(2009\)\]](#);
- [\[Introduction to Struts 2 Through Examples \(2012\)\]](#);
- [\[Introduction to Java Server Faces, PrimeFaces, and PrimeFaces Mobile with Examples \(2012\)\]](#);

- [[Introduction to Java EE with the NetBeans IDE and the GlassFish Application Server \(2012\)](#)];
- [[A Client/Server Example - AngularJS 1.x / Spring 4 \(2014\)](#)];
- [[Introduction to the Java Language and the Spring Ecosystem Through a Case Study \(2015\)](#)];
- [[Introduction to Spring MVC and Thymeleaf through examples \(2015\)](#)];
- [[Working with a relational database using the Spring ecosystem \(2015\)](#)];
- [[Introduction to RxJava. Application to Swing and Android environments \(2016\)](#)];

Android

- [[Introduction to Android Tablet Programming with Android Studio \(2016\)](#)];

.NET

- [[Introduction to VB.NET through Examples \(2004\)](#)];
- [[Web Development with ASP.NET 1.1 \(2004\)](#)];
- [[Spring IoC for .NET \(2005\)](#)];
- [[Creating a Three-Tier Web Application with Spring.NET and VB.NET \(2005\)](#)];
- [[Learning C# Version 3.0 with the .NET Framework 3.5 \(2008\)](#)];
- [[Building a Three-Tier Web Application with ASP.NET 2.0, C#, Spring.NET, and NHibernate \(2010\)](#)];
- [[Introduction to the NHibernate Framework for the .NET Platform \(2011\)](#)];
- [[Introduction to Entity Framework 5 Code First Through Examples \(2012\)](#)];
- [[Introduction to the ASP.NET MVC Framework through Examples \(2013\)](#)];

JavaScript

- [[A Client/Server Example - AngularJS 1.x / Spring 4 \(2014\)](#)];
- [[Introduction to the ECMAScript 6 language through examples \(2019\)](#)];
- [[Introduction to the VUE.JS framework through examples \(2019\)](#)];
- [[Introduction to the NUXT.JS framework through examples \(2019\)](#)];

PHP

- [[MVC Development Methodology for a PHP4 Web Application \(2004\)](#)];
- [[Introduction to the PHP5 language through examples \(2011\)](#)];
- [[Introduction to the PHP7 language through examples \(2019\)](#)];

Python

- [[Introduction to Python 2.7 through Examples \(2012\)](#)];
- [[Introduction to the Python Language and the Flask Web Framework Through Examples \(2020\)](#)];
- [[Generate a Python script using AI tools \(2025\)](#)];

VBScript

- [[Introduction to VBScript \(2002\)](#)];

SQL

- [[Introduction to the SQL language with the Firebird DBMS \(2006\)](#)];

Es handelt sich dabei um alte Kurse, die mehrere hundert Seiten umfassen. Die Größe der ODT- oder DOCX-Datei spielt für den Konverter Gemini / ChatGPT keine Rolle. Was für ihn zählt, sind die darin enthaltenen Strukturen (siehe Kapitel: [Beispiele in diesem Dokument](#)).

1. Einführung

Das PDF dieses Dokuments ist |[HIER](#)| verfügbar.

Die Konverter Gemini 3 / ChatGPT 5.2 [Word oder ODT → HTML] sind |[HIER](#)| verfügbar.

Dieses Dokument stellt Ihnen zwei Konverter zur Verfügung:

- einen Konverter von LibreOffice ODT nach HTML;
 - einen Konverter von Word DOCX nach HTML;
-

In diesem Artikel möchten wir dem Leser einen Python-Konverter zur Verfügung stellen, der Word- oder ODT-Dokumente in eine statische HTML-Website umwandelt. Dieser Konverter wurde ursprünglich von der KI [Gemini 3](#) und anschließend von [ChatGPT 5.2](#) erstellt. Diese beiden KIs benötigten 356 Iterationen, um den Konverter für dieses Dokument zu erstellen. Dies erforderte eine mehrwöchige Fehlerbehebung. Gemini 3 hat zu Beginn die gesamte Arbeit geleistet. Zunächst waren mehrere Dutzend Iterationen nötig, um eine erste, einigermaßen korrekte Version zu erhalten. Dann fügte ich regelmäßig ein neues Problem hinzu, das durch neue ODT-Dokumente auftrat. Und dabei machte Gemini oft Rückschritte. Das heißt, was in Schritt N funktionierte, funktionierte in den folgenden Schritten nicht mehr. Ich ging dann wie folgt vor: Sobald Gemini einen Konverter erzeugte, der eines meiner Probleme löste, machte ich daraus eine Referenzversion und teilte dies Gemini mit. Diese Referenz speicherte ich lokal. Wenn ich dann sah, dass Gemini zu lange Rückschritte machte, bat ich es, zur letzten bekannten Referenzversion zurückzukehren, indem ich sie ihm gab. So haben wir nach und nach gemeinsam diesen Konverter aufgebaut: Ich habe ihm mitgeteilt, was ich wollte, im Wesentlichen indem ich ihm die Anomalien meldete, die ich in der erzeugten HTML-Seite fand, und er hat den gewünschten Code erzeugt.

Ich habe [Gemini 3](#) mit einer Pro-Lizenz für 22 Euro/Monat und [ChatGPT 5.2](#) ebenfalls genutzt.

Gemini / ChatGPT generiert zwei Python-Skripte:

- [convert] zum Konvertieren des ODT-Dokuments aus [LibreOffice](#) oder des DOCX-Dokuments aus Word in eine [MkDocs](#)-Website;
- [build] zur Konvertierung der MkDocs-Website in eine statische HTML-Website;

Ich habe mir den generierten Code nie angesehen. Ich wollte sie als Black Boxes betrachten. Man muss kein Python-Entwickler sein, um diesem Tutorial zu folgen, und nicht einmal ein Entwickler überhaupt.

Als ich der KI Gemini 3 Verbesserungen für den Konverter vorschlug, ist diese manchmal abgestürzt. Sie konnte die gewünschten Verbesserungen nicht vornehmen. Zu diesem Zeitpunkt waren wir bei der stabilen Version V316. Um weiterzukommen, habe ich diese funktionsfähige Version an [ChatGPT 5.2](#) übergeben und die gewünschten Verbesserungen angefordert. ChatGPT hat den Code von Gemini korrekt geändert, um meinen neuen Anforderungen gerecht zu werden. Deshalb bin ich der Ansicht, dass diese beiden KI-Systeme den Konverter erstellt haben.

Danach habe ich diese Technik weiterverwendet. Wenn eine KI bei einer Funktion nicht weiterkam, gab ich der anderen KI die letzte bekannte stabile Version.

Die beiden KI-Systeme haben unterschiedliche Methoden, um den angeforderten Python-Code zu liefern:

- Gemini gibt den Code des generierten Skripts auf der Abfrageseite aus. Dieser Code muss dann kopiert und eingefügt werden;
- ChatGPT bietet einen Download-Link für das generierte Skript an;

Bei mehr als 1000 Zeilen generiertem Code zeigte Gemini erhebliche Schwächen. Aufgrund technischer Einschränkungen dieser KI gelang es ihr nicht, den gesamten generierten Code auf der Abfrageseite anzuzeigen. Sehr oft fehlten Codezeilen. Aufgrund dieser Einschränkung war es ab einem bestimmten Zeitpunkt nicht mehr möglich, Gemini 3 zu verwenden. Daher hat ChatGPT 5.2 das Skript fertiggestellt.

Ebenfalls ChatGPT hat den Word-zu-HTML-Konverter generiert. Ich habe ihm den funktionierenden ODT-zu-HTML-Konverter gegeben und ihn gebeten, diesen für ein Word-Dokument anzupassen. Er hat dies in 18 Iterationen geschafft. Das ist eine Konstante bei diesen beiden KI-Systemen: Sie verstehen die Python-Skripte, die man ihnen gibt, sehr gut und können Verbesserungen daran vornehmen. Für mich ist das die beste Art, mit ihnen zu arbeiten.

2. Beispiele aus diesem Dokument

Ich möchte einen kurzen Artikel verfassen. Die Interaktionen zwischen einer KI und einem Nutzer werden in dem Artikel [\[Erstellen eines Python-Skripts mit KI-Tools\]](#) vorgestellt, den ich fortan als [\[ref1\]](#) bezeichnen werde. Die Interaktionen mit Gemini und ChatGPT werden nur am Rande behandelt. Es war ohnehin unmöglich, alle Iterationen darzustellen.

Ich werde nun Beispiele für die Besonderheiten meiner ODT-/DOCX-Dokumente vorstellen, die der Gemini-/ChatGPT-Konverter korrekt verarbeitet. Genau dieses Dokument werden wir dem Gemini-/ChatGPT-Skript zur HTML-Konvertierung anbieten. Wir werden sehen, was es daraus macht.

2.1. Listen

Der Gemini/ChatGPT-Konverter kann Aufzählungslisten und nummerierte Listen verarbeiten, selbst wenn sie verschachtelt sind:

2.1.1. Aufzählungslisten

- Element 1;
- Element 2;
- Element 3;
 - Element 3.1;
 - Element 3.1.1
 - Element 3.1.2
 - Element 3.1.2.1
 - Element 3.1.2.2
 - Element 3.2;
- Element 4;

2.1.2. Nummerierte Listen

1. Element 1;
2. Element 2;
 - a. Element 2.1
 - i. Element 2.1.1
 1. Element 2.1.1.1
 2. Element 2.1.1.2
 - ii. Element 2.1.2
 - b. Element 2.2
3. Element 3;

2.1.3. Gemischte Listen 1

- Element 1;
- Element 2;

- Element 3;
 - Element 3.1;
 1. Element 3.1.1
 2. Element 3.1.2
 - Element 3.1.2.1
 - Element 3.1.2.2
 - Element 3.2;
- Element 4;

2.1.4. Gemischte Listen 2

1. Element 1;
2. Element 2;
 - a. Element 2.1
 - i. Element 2.1.1
 - Element 2.1.1.1
 - Element 2.1.1.2
 - ii. Element 2.1.2
 - b. Element 2.2
3. Element 3;

2.1.5. Manuell nummerierte Listen

Unter manueller Nummerierung versteht man hier, dass der Benutzer die Nummer eines nummerierten Absatzes festlegt: [Rechtsklick auf den nummerierten Absatz / Absatz / Absatz / Nummerierung neu beginnen / beginnen mit].

Ich beginne eine Liste mit einer anderen Nummer als 1.

6. Element 6
7. Element 7

Hier unterbreche ich die Liste, um etwas zu sagen, möchte aber anschließend die Nummerierung fortsetzen.

8. Element 8
9. Element 9

Dann beginne ich eine neue nummerierte Liste:

11. Element 11
12. Element 12

2.2. Code-Blöcke

Meine Kurse enthalten viele Code-Blöcke. Oft handelt es sich dabei um Code, der von IDEs (Eclipse, PyCharm, WebStorm, Netbeans) formatiert wurde (Fettdruck, Hervorhebung von Schlüsselwörtern). Dieser formatierte Code wird vom Konverter originalgetreu wiedergegeben.

Wenn der Code nicht formatiert ist (Code aus Notepad oder ...), erkennt der Gemini/ChatGPT-Konverter ihn (Java, C#, XML, HTML, ...) anhand von Schlüsselwörtern der Sprache, die in einer Konfigurationsdatei hinterlegt sind. Wenn er eine Sprache erkennt, fügt er eine Markierung (Fence) für MkDocs ein, damit dieses die Syntaxhervorhebung des Codes an die im Code-Block verwendete Sprache anpasst.

2.2.1. Angereicherte Code-Blöcke (Eclipse, Visual Studio, ...)

Hier sind einige durch verschiedene IDEs angereicherte Code-Blöcke:

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // Abrufen des Spring-Kontexts
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // Abrufen der Beans
16.         Person p01 = ctx.getBean("personne_01", Personne.class);
17.         Person p02 = ctx.getBean("personne_02", Personne.class);
18.         List<Person> club = ctx.getBean("club", new ArrayList<Person>().getClass());
19.         Wohnung wohnung01 = ctx.getBean(Wohnung.class);
20.     ...
```

C#

```
11. using System;
12.
13. namespace Chap1 {
14.     class Steuern {
15.         static void Main(string[] args) {
16.             // Für die Steuerberechnung erforderliche Datentabellen
17.             decimal[] Grenzen = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
18.             };
19.             decimal[] coeffr = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M,
20.             0.481M };
21.             decimal[] coeffn = { 0M, 291,09M, 1322,92M, 2668,39M, 4846,98M,
22.             6883,66M, 9505,54M };
23.
24.             // Wir rufen den Familienstand ab
25.             bool OK = false;
26.             string Antwort = null;
27.             while (!OK) {
28.                 Console.WriteLine("Sind Sie verheiratet (J/N)? ");
29.                 Antwort = Console.ReadLine().Trim().ToLower();
30.                 if (Antwort != "j" && Antwort != "n")
31.                     Console.Error.WriteLine("Falsche Antwort. Bitte
32.                 versuchen Sie es erneut");
33.                 else OK = true;
34.             } //while
35.             bool marie = Antwort == "o";
36.     ...
```

Python

```
21. # -----
22. def anzeigen(Zeichenkette):
23.     # Zeigt die Zeichenkette an
24.     print("Zeichenkette=%s" % Zeichenkette)
25.
26.
27. # -----
28. def Typ_anzeigen(variable):
```

```

29.     # zeigt den Typ der Variablen an
30.     print("Typ[%s]=%s" % (Variable, type(Variable)))
31.
32.
33. # -----
34. def f1(param):
35.     # addiert 10 zu param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # gibt ein Tupel mit 3 Werten zurück
42.     return "un", 0, 100
43.
44.
45. # ----- Hauptprogramm -----
46. ...

```

PHP

```

31. <?php
32.
33. // Strenge Typen für Funktionsparameter
34. declare(strict_types=1);
35.
36. // globale Konstanten
37. define("PLAFOND_QF_DEMI_PART", 1551);
38. define("EINKOMMENSCHENDE_FÜR_EINZELPERSONEN_FÜR_ERMÄSSIGUNG", 21037);
39. define("EINKOMMENSCHENDE_PAAR_FÜR_ERMÄSSIGUNG", 42074);
40. define("REDUZIERUNGSBETRAG_HALBTEIL", 3797);
41. define("OBERGRENZE_ABZUG_ALLEINSTÄNDIGE", 1196);
42. define("OBERGRENZE_ABZUG_PAAR", 1970);
43. define("STEUEROBERGRENZE_PAAR_FÜR_ABZUG", 2627);
44. define("STEUEROBERGRENZE_FÜR_ALLGEMEINEN_ABZUG_FÜR_ALLEINSTÄNDIGE", 1595);
45. define("MAXIMALER_ZEHNPROZENT-FREIBETRAG", 12502);
46. define("ABATTEMENT_DIXPOURCENT_MIN", 437);
47.
48. // Definition der lokalen Konstanten
49. $DATA = "taxpayersdata.txt";
50. $RESULTATS = "resultats.txt";
51. $Grenzwerte = array(9964, 27519, 73779, 156244, 0);
52. $coeffR = array(0, 0.14, 0.3, 0.41, 0.45);
53. $coeffN = array(0, 1394.96, 5798, 13913.69, 20163.45);
54.
55. // Daten lesen
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Die Datendatei [$DATA] konnte nicht zum Lesen geöffnet werden\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // dies ist ein Kommentar
43. // Konstante
44. const name = "dupont";
45. // eine Bildschirmausgabe
46. console.log("name: ", name);
47. // ein Array mit Elementen unterschiedlichen Typs
48. const array = ["eins", "zwei", 3, 4];
49. // Anzahl der Elemente
50. let n = array.length;
51. // eine Schleife

```

```

52. for (let i = 0; i < n; i++) {
53.   console.log("array[" + i + "] = ", array[i]);
54. }
55. // Initialisierung von 2 Variablen mit dem Inhalt eines Arrays
56. let [Zeichenkette1, Zeichenkette2] = ["Zeichenkette1", "Zeichenkette2"];
57. // Verkettung der beiden Zeichenfolgen
58. const string3 = string1 + string2;
59. // Ergebnis anzeigen
60. console.log([Zeichenkette1, Zeichenkette2, Zeichenkette3]);
61. ...

```

VBScript

```

51. ' Berechnung der Steuer eines Steuerpflichtigen
52. ' Das Programm muss mit drei Parametern aufgerufen werden: verheiratet, Kinder, Gehalt
53. ' verheiratet: Zeichen O, wenn verheiratet, N, wenn unverheiratet
54. ' Kinder: Anzahl der Kinder
55. ' Gehalt: Jahresgehalt ohne Centbeträge
56.
57. ' obligatorische Deklaration der Variablen
58. Option Explicit
59. Dim Fehler
60.
61. ' Wir rufen die Argumente ab und prüfen ihre Gültigkeit
62. Dim Frau, Kinder, Gehalt
63. Fehler = GetArguments(Ehefrau, Kinder, Gehalt)
64. ' Fehler?
65. If Fehler(0) <> 0 Then wscript.echo Fehler(1) : wscript.quit Fehler(0)
66.
67. ' Die für die Steuerberechnung erforderlichen Daten werden abgerufen
68. Dim Grenzen, KoeffR, KoeffN
69. Fehler = getData(Grenzen, KoeffR, KoeffN)
70. ' Fehler?
71. If Fehler(0) <> 0 Then wscript.echo Fehler(1) : wscript.quit 5
72.
73. ' Das Ergebnis wird angezeigt
74. wscript.echo "Steuer=" & berechneSteuer(Ehepartner, Kinder, Gehalt, Grenzen, KoeffR, KoeffN)
75.
76. ' Beenden ohne Fehler
77. wscript.quit 0

```

XML

```

61. <?xml version="1.0" encoding="utf-8" ?>
62. <Konfiguration>
63.
64.   <configSections>
65.     <sectionGroup name="spring">
66.       <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
67.       <section name="objects" type="Spring.Context.Support.DefaultSectionHandler, Spring.Core"
68.     />
69.   </sectionGroup>
70. </configSections>
71.
72. <spring>
73.   <context>
74.     <resource uri="config://spring/objects" />
75.   </context>
76.   <objects xmlns="http://www.springframework.net">
77.     <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
78.       <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
79.       <constructor-arg index="1"
80.     value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
81.       <constructor-arg index="2" value="select limite, coeffr, coeffn from tranches"/>
82.     </object>
83.     <object name="Beruf" type="Beruf.SteuerBeruf, SteuernV7-Beruf">
84.       <constructor-arg index="0" ref="dao"/>

```

```
83.     </object>
84. </objects>
85. </spring>
86. </configuration>
```

2.2.2. Rohcode-Blöcke (Klartext)

Hier sind einige Beispiele für Rohcode:

Ausführungsergebnisse

Beachten Sie, dass der Code nicht mit Zeile 1 beginnt.

```
7. C:\Data\st-2020\dev\python\cours-2020\python3-flask-2020\venv\Scripts\python.exe
C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/bases/bases_01.py
8. name=dupont
9. liste[0]=eins
10. liste[1]=zwei
11. liste[2]=3
12. liste[3]=4
13. [Zeile1,Zeile2,Zeile1Zeile2]
14. Zeichenkette=Zeichenkette1
15. Typ[4]=<class 'int'>
16. Typ[Zeichenkette1]=<class 'str'>
17. Typ[['eins', 'zwei', 3, 4]]=<class 'list'>
18. Typ[geändert]=<class 'str'>
19. res1=14s
20. (res1,res2,res3)=[eins,0,100]
21. liste[0] = eins
22. liste[1]=0
23. liste[2]=100
24. liste[0]=8
25. liste[1]=5
26. Summe=13
27.
28. Prozess beendet mit Exit-Code 0
```

Dieser Fall erforderte Dutzende von Iterationen. Der Konverter fand die Nummer der ersten Codezeile einfach nicht. Schließlich fragte ich Gemini, wie ich den Code-Block formatieren müsse, damit er ihn erkennt. Hier ist seine Antwort:

Dans LibreOffice :

1. Placez votre curseur sur la **première ligne** de votre bloc de code numéroté (celle qui doit être le n°5).
2. Faites un **Clic-Droit > Paragraphe > Paragraphe...**
3. Allez dans l'onglet **Plan & numérotation** (ou *Outline & Numbering*).
4. Cochez la case **Recommencer la numérotation** (*Restart numbering at this paragraph*).
5. Cochez/Réglez **Commencer à** (*Start with*) sur **5**.
6. Cliquez sur **OK** et sauvegardez votre ODT.

Pourquoi ça marche ? Cette méthode oblige LibreOffice à inscrire un attribut explicite sur le paragraphe ou l'item de liste (`text:start-value="5"`), ce que le script détecte immédiatement. La méthode via l'icône de liste ("Personnaliser") modifie parfois seulement le *style* visuel sans changer la valeur logique sous-jacente dans le XML, surtout sur les documents importés de Word.

Ich habe diesen Vorschlag befolgt und es hat funktioniert. Ich habe die Methode über das Listensymbol (Anpassen) verwendet. Man muss also möglicherweise bestimmte Elemente des ODT-/DOCX-Dokuments ändern, um das gewünschte Ergebnis zu erzielen.

Der Konverter kann mit unnummerierten Code-Blöcken umgehen.

```
liste[0]=un
liste[1]=0
liste[2]=100
liste[0]=8
liste[1]=5
Summe=13
```

Hier sind dieselben Beispiele wie im Abschnitt „2.2.1“, jedoch ohne Erweiterung. In diesem Fall leiten die Schlüsselwörter in der Konfigurationsdatei den Konverter zur richtigen Sprache.

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // Abrufen des Spring-Kontexts
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // Abrufen der Beans
16.         Person p01 = ctx.getBean("personne_01", Personne.class);
17.         Person p02 = ctx.getBean("personne_02", Personne.class);
18.         List<Person> club = ctx.getBean("club", new ArrayList<Person>().getClass());
19.         Wohnung wohnung01 = ctx.getBean(Wohnung.class);
20.         ...

```

C#

```
11. using System;
12.
13. namespace Chap1 {
14.     class Steuern {
15.         static void Main(string[] args) {
16.             // Für die Steuerberechnung erforderliche Datentabellen
17.             decimal[] Grenzen = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
18.         };
19.             decimal[] coeffr = { 0M, 0,068M, 0,191M, 0,283M, 0,374M, 0,426M,
20.             0,481M };
21.             decimal[] coeffn = { 0M, 291,09M, 1322,92M, 2668,39M, 4846,98M,
22.             6883,66M, 9505,54M };
23.
24.             // Wir rufen den Familienstand ab
25.             bool OK = false;
26.             string Antwort = null;
27.             while (!OK) {
28.                 Console.WriteLine("Sind Sie verheiratet (J/N)? ");
29.                 Antwort = Console.ReadLine().Trim().ToLower();
30.                 if (Antwort != "j" && Antwort != "n")
31.                     Console.Error.WriteLine("Falsche Antwort. Bitte
32.             versuchen Sie es erneut");
33.                 else OK = true;

```

```

30.         }//while
31.         bool marie = Antwort == "o";
32. ...

```

Python

```

21. # -----
22. def anzeigen(Zeichenkette):
23.     # Zeigt die Zeichenkette an
24.     print("Zeichenkette=%s" % Zeichenkette)
25.
26.
27. # -----
28. def Typ_anzeigen(Variable):
29.     # zeigt den Typ der Variablen an
30.     print("Typ[%s]=%s" % (Variable, type(Variable)))
31.
32.
33. # -----
34. def f1(param):
35.     # addiert 10 zu param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # gibt ein Tupel mit 3 Werten zurück
42.     return "un", 0, 100
43.
44.
45. # ----- Hauptprogramm -----
46. ...

```

PHP

```

31. <?php
32.
33. // Strenge Typen für Funktionsparameter
34. declare(strict_types=1);
35.
36. // globale Konstanten
37. define("PLAFOND_QF_DEMI_PART", 1551);
38. define("EINKOMMENSGRENZE_FÜR_EINZELPERSONEN_FÜR_ERMÄSSIGUNG", 21037);
39. define("EINKOMMENSGRENZE_PAAR_FÜR_ERMÄSSIGUNG", 42074);
40. define("REDUZIERUNGSBETRAG_HALBTEIL", 3797);
41. define("OBERGRENZE_ABZUG_ALLEINSTÄNDIGE", 1196);
42. define("OBERGRENZE_ABZUG_PAAR", 1970);
43. define("STEUEROBERGRENZE_PAAR_FÜR_ABZUG", 2627);
44. define("STEUEROBERGRENZE_FÜR_ALLGEMEINEN_ABZUG_FÜR_ALLEINSTÄNDIGE", 1595);
45. define("MAXIMALER_ZEHN-PROZENT-FREIBETRAG", 12502);
46. define("ABATTEMENT_DIXPOURCENT_MIN", 437);
47.
48. // Definition der lokalen Konstanten
49. $DATA = "taxpayersdata.txt";
50. $RESULTATS = "resultats.txt";
51. $Grenzwerte = array(9964, 27519, 73779, 156244, 0);
52. $coeffR = array(0, 0.14, 0.3, 0.41, 0.45);
53. $coeffN = array(0, 1394.96, 5798, 13913.69, 20163.45);
54.
55. // Daten lesen
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Die Datendatei [$DATA] konnte nicht zum Lesen geöffnet werden\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```
41. 'use strict';
42. // dies ist ein Kommentar
43. // Konstante
44. const name = "dupont";
45. // eine Bildschirmausgabe
46. console.log("name: ", name);
47. // Ein Array mit Elementen unterschiedlichen Typs
48. const array = ["eins", "zwei", 3, 4];
49. // die Anzahl der Elemente
50. let n = array.length;
51. // eine Schleife
52. for (let i = 0; i < n; i++) {
53.   console.log("array[" + i, "] = ", array[i]);
54. }
55. // Initialisierung von 2 Variablen mit dem Inhalt eines Arrays
56. let [Zeichenkette1, Zeichenkette2] = ["Zeichenkette1", "Zeichenkette2"];
57. // Verkettung der beiden Zeichenfolgen
58. const Zeichenfolge3 = Zeichenfolge1 + Zeichenfolge2;
59. // Ergebnis anzeigen
60. console.log([Zeichenkette1, Zeichenkette2, Zeichenkette3]);
61. ...
```

VBScript

```
51. ' Berechnung der Steuer eines Steuerpflichtigen
52. ' Das Programm muss mit drei Parametern aufgerufen werden: verheiratet, Kinder, Gehalt
53. ' verheiratet: Zeichen 0, wenn verheiratet, N, wenn unverheiratet
54. ' Kinder: Anzahl der Kinder
55. ' Gehalt: Jahresgehalt ohne Centbeträge
56.
57. ' obligatorische Deklaration der Variablen
58. Option Explicit
59. Dim Fehler
60.
61. ' Wir rufen die Argumente ab und prüfen ihre Gültigkeit
62. Dim Frau, Kinder, Gehalt
63. Fehler = GetArguments(Ehefrau, Kinder, Gehalt)
64. ' Fehler?
65. If Fehler(0)<>0 Then wscript.echo Fehler(1) : wscript.quit Fehler(0)
66.
67. ' Die für die Steuerberechnung erforderlichen Daten werden abgerufen
68. Dim Grenzen, KoeffR, KoeffN
69. Fehler = getData(Grenzen, KoeffR, KoeffN)
70. ' Fehler?
71. If Fehler(0)<>0 Then wscript.echo Fehler(1) : wscript.quit 5
72.
73. ' Das Ergebnis wird angezeigt
74. wscript.echo "Steuer=" & berechneSteuer(Ehepartner,Kinder,Gehalt,Grenzen,KoeffR,KoeffN)
75.
76. ' Beenden ohne Fehler
77. wscript.quit 0
```

XML

```
61. <?xml version="1.0" encoding="utf-8" ?>
62. <Konfiguration>
63.
64.   <configSections>
65.     <sectionGroup name="spring">
66.       <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
67.       <section name="objects" type="Spring.Context.Support.DefaultSectionHandler, Spring.Core"
68.     />
69.   </sectionGroup>
70. </configSections>
```

```

70.
71. <spring>
72.   <context>
73.     <resource uri="config://spring/objects" />
74.   </context>
75.   <objects xmlns="http://www.springframework.net">
76.     <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
77.       <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
78.       <constructor-arg index="1"
value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
79.       <constructor-arg index="2" value="select limite, coeffr, coeffn from tranches"/>
80.     </object>
81.     <Objekt name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
82.       <constructor-arg index="0" ref="dao"/>
83.     </object>
84.   </objects>
85. </spring>
86. </configuration>

```

HTML

```

71. <!DOCTYPE HTML>
72. <HTML>
73.   <head>
74.     <title>Laragon</title>
75.
76.     <link href="https://fonts.googleapis.com/css?family=Karla:400 " rel="stylesheet"
type="text/css">
77.
78.     <style>
79.       HTML, body {
80.         height: 100%;
81.       }
82.
83.       body {
84.         margin: 0;
85.         padding: 0;
86.         Breite: 100%;
87.         display: table;
88.         font-weight: 100;
89.         Schriftart: 'Karla';
90.       }
91.
92.       .container {
93.         text-align: center;
94.         display: table-cell;
95.         vertical-align: middle;
96.       }
97.
98.       .content {
99.         text-align: center;
100.        display: inline-block;
101.      }
102.
103.      .title {
104.        Schriftgröße: 96px;
105.      }
106.
107.      .opt {
108.        margin-top: 30px;
109.      }
110.
111.      .opt a {
112.        text-decoration: none;
113.        Schriftgröße: 150%;
114.      }
115.
116.      a:hover {
117.        color: red;

```

```

118.     }
119.     </style>
120. </head>
121. <body>
122.     <div class="container">
123.         <div class="content">
124.             <div class="title" title="Laragon">Laragon</div>
125.
126.             <div class="info"><br />
127.                 Apache/2.4.35 (Win64) OpenSSL/1.1.0i PHP/7.2.11<br />
128.                 PHP-Version: 7.2.11 <span><a title="phpinfo()" href="/?
q=info">Info</a></span><br />
129.                 Dokumentenstammverzeichnis: C:/myprograms/laragon-lite/www<br />
130.
131.             </div>

```

2.3. Links

Der Gemini/ChatGPT-Konverter behält die externen Links des ODT-/DOCX-Dokuments bei. Zum Beispiel [Gemini 3](#) oder [\[Ein Python-Skript mit KI-Tools generieren\]](#).

Er kann einen Link zu einem Kapitel verarbeiten [Link zu einem Kapitel](#)

Ein Verweis auf ein Kapitel: 2.1.1 .

Ein Verweis auf eine vorhergehende Textmarkierung: Gemini 3

Ein Verweis auf eine nachfolgende Textmarkierung: GitHub

2.4. Textformatierung

Der Konverter unterstützt Fettdruck, Kursivschrift, Unterstreichungen und Hervorhebungen. Die Farbe der Hervorhebung wird beibehalten.

Ein Text mit **fettgedruckten**, *kursiven*, unterstrichenen oder hervorgehobenen Wörtern.

Das gilt auch für Links: [\[Python-Skript mit KI-Tools generieren\]](#).

Der **Konverter verarbeitet** auch die **Schriftfarbe**.

Er unterstützt zudem die oberen und unteren Ränder von Absätzen.

- Er unterstützt auch die oberen und unteren Ränder von Absätzen.
 - Er unterstützt auch die oberen und unteren Ränder von Absätzen.
 - Er unterstützt auch die oberen und unteren Ränder von Absätzen.
-

1. Er unterstützt auch die oberen und unteren Ränder von Absätzen.
 2. Es verwaltet auch die oberen und unteren Ränder der Absätze.
 3. Es verwaltet auch die oberen und unteren Ränder der Absätze.
-

10. Ein Text mit **fettgedruckten**, *kursiven*, unterstrichenen oder hervorgehobenen Wörtern.
11. Das gilt auch für Links: [\[Python-Skript mit KI-Tools generieren\]](#).
12. Der **Konverter verarbeitet** auch die **Schriftfarbe**.

- Ein Text mit **fettgedruckten**, *kursiven*, unterstrichenen oder hervorgehobenen Wörtern.
- Das gilt auch für Links: [\[Python-Skript mit KI-Tools generieren\]](#).
- Der **Konverter unterstützt** auch die **Schriftfarbe**.

Er unterstützt auch den Hintergrund des Absatzes

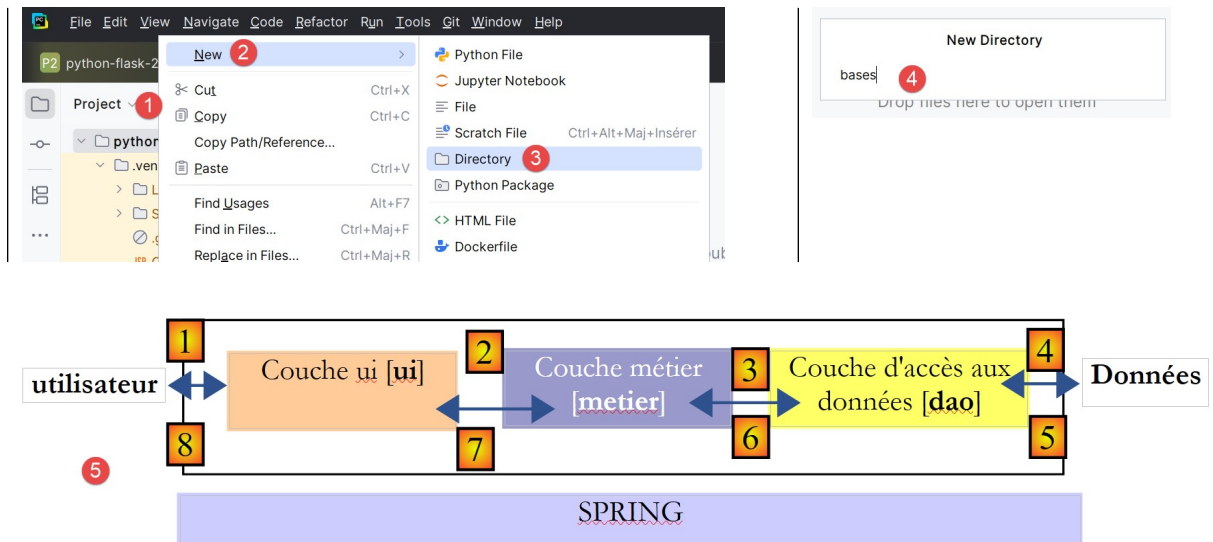
- Er verwaltet auch den Hintergrund des Absatzes

1. Er verwaltet auch den Hintergrund des Absatzes

2.5. Auch eine Überschrift kann *angereichert* werden.

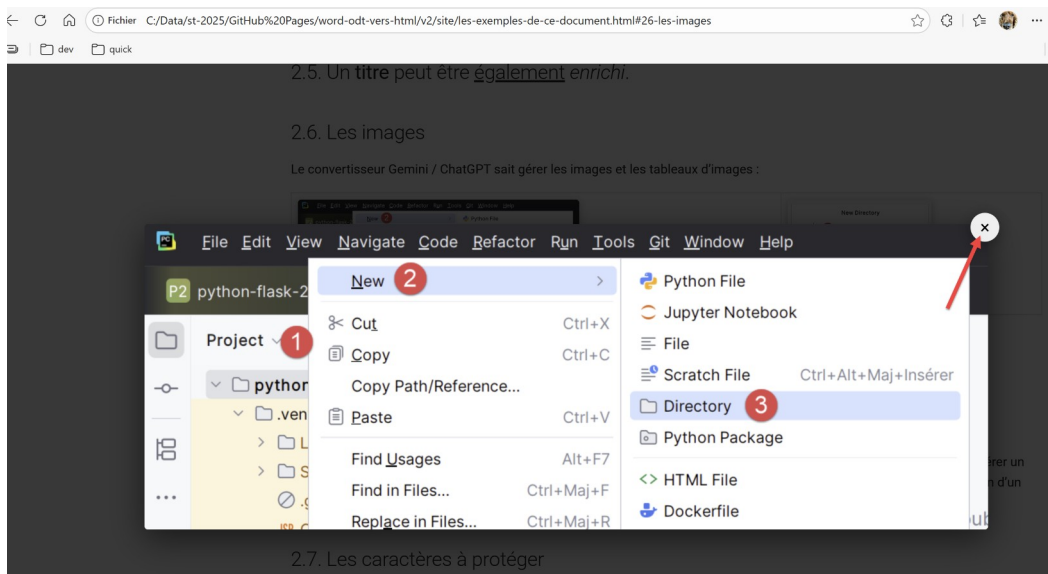
2.6. Bilder

Der Konverter Gemini / ChatGPT kann mit Bildern und Bildtabellen umgehen:



In ODT-Dokumenten kommen häufig Zeichnungen vor. Trotz Dutzender Versuche ist es Gemini nicht gelungen, ein Skript zu erstellen, das ein Bild (wie einen Screenshot) der Zeichnung generieren würde. So ist das Bild 5 oben der Screenshot einer Zeichnung aus einem ODT-Dokument.

Alle Bilder sind anklickbar und lassen sich vergrößern. Wenn man oben auf das Bild [1-3] klickt, erhält man die folgende Vergrößerung:



2.7. Zu schützende Zeichen

Eine MkDocs-Website¹ enthält Seiten, deren Inhalt nicht aus HTML, sondern aus Markdown besteht. Wenn das ODT-/DOCX-Dokument Zeichen enthält, die in Markdown vorkommen, besteht die Gefahr, dass sie von MkDocs interpretiert werden und somit nicht wie erwartet dargestellt werden. Hier sind zwei Beispiele:

Das Sternchen `*` hat eine Markdown-Bedeutung. Die folgende Zeile kann daher falsch interpretiert werden:

Die Steuer I ist dann gleich **`0,15*R – 2072,5*nbParts`**.

Ein weiteres Beispiel ist, wenn Sie einen Markdown-Code-Block wie diesen in Ihr Dokument einfügen möchten:

```
1. # Konverter von Word/ODT zu HTML-Website (MkDocs)
2.
3.  [Zur generierten Demo-Website](https://stahe.github.io/word-odt-vers-html-janv-2026/)
4.
5. ---
6.
7. ##  Beschreibung
8.
9. In diesem Projekt soll dem Leser ein Python-Konverter zur Verfügung gestellt werden, der Word-
oder ODT-Dokumente in eine statische HTML-Website umwandelt.
10.
11. Wenn das ODT-/DOCX-Dokument geeignet ist, erzeugt der Konverter über MkDocs eine HTML-
Website, die das professionelle Erscheinungsbild der von Pandoc erstellten Websites aufweist.
12.
13. ##  Entstehungshintergrund
14.
15. Dieser Konverter wurde vollständig von der KI Gemini 3 (mit einem Pro-Abonnement)
erstellt. Er ist das Ergebnis aufeinanderfolgender Iterationen zur Feinabstimmung der Struktur
von ODT-Dokumenten (OpenDocument Text).
16.
17. ##  Funktionen
18.
19. Das Skript convert.py führt folgende Aktionen aus:
20.
21. * Konvertierung von ODT nach Markdown: Analysiert die .odt-Datei (XML), um die Struktur
zu extrahieren.
22. * Verwaltung von Überschriften: Erzeugt automatisch das Inhaltsverzeichnis (TOC) und die
seitliche Navigation.
23. * Code-Blöcke: Automatische Spracherkennung, Syntaxhervorhebung und präzise Verwaltung
der Zeilennummerierung (Attribute start-value).
24. * Listen: Unterstützung von Aufzählungs- und nummerierten Listen mit korrekter Einrückung.
25. * Formatierung: Unterstützung von Fettdruck, Kursivschrift, Unterstreichung und
Hervorhebung (unter Beibehaltung der Originalfarben).
26. * Bilder: Automatische Extraktion und Einbindung der im Dokument enthaltenen Bilder.
27. * Konfiguration: Anpassung über eine config.py-Datei (Fußzeile, Google Analytics usw.).
28.
29. ##  Installation
30.
31. ### Voraussetzungen
32.
33. * Python 3.x
34. * Die folgenden Bibliotheken:
35.
36. ```bash
37. pip install odfpy unicode mkdocs mkdocs-material
38.
39. ```
```

¹[MkDocs](https://tahe.developpez.com)

2.8. Tabellen

Eine Tabelle kann verschiedene Inhalte enthalten:

1

2



```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import
  org.springframework.context.ApplicationContext;
7. import
  org.springframework.context.support.ClassPathXmlApp
  licationContext;
8.
9. public class Demo01 {
10.
11.
```

3

4



```
1. # -----
2. def anzeigen(Zeichenkette):
3.     # Zeigt „chaine“ an
4.     print("Zeichenkette=%s" % Zeichenkette)
5.
6.
7. # -----
8. def Typ_anzeigen(Variable):
9.     # zeigt den Typ der Variablen an
10.    print("Typ[%s]=%s" % (Variable,
11.                          type(Variable)))
12.
```

Ein Array mit Links:

Der Gemini/ChatGPT-Konverter behält die externe Links des ODT-/DOCX-Dokuments beizubehalten. Zum Beispiel [Gemini 3](#) oder [\[Ein Python-Skript mit KI-Tools generieren\]](#).

Ein Verweis auf ein Kapitel:2.1.1 .

Ein Verweis auf eine vorhergehende Textmarkierung: Gemini 3

Ein Verweis auf eine nachfolgende Textmarkierung: GitHub

Er kann einen Link zu einem Kapitel verarbeiten [Link zu einem Kapitel](#)

Eine Tabelle, deren Zellen eine weitere Tabelle enthalten:

`html:form` dient sowohl zur Generierung des HTML-Tags `<form>` als auch zur Bereitstellung von Informationen für den Controller, der dieses Formular verarbeiten muss:

<code>action</code>	URL, an die die Werte des Formulars gesendet werden
<code>name</code>	Name des HTML-Formulars. Dies ist auch der Name des Beans, das die Werte des Formulars speichert
<code>type</code>	Name der Klasse, die instanziiert werden muss, um die Bean zum Speichern des Formulars zu erhalten

Es ist zu beachten, dass die Methode zum Senden der Formularparameter (GET/POST) an den Controller nicht angegeben ist. Dies könnte mit dem Attribut `method` erfolgen. Fehlt dieses, wird standardmäßig die **POST-Methode** verwendet.

`html:text` dient zur Generierung des Tags `<input type="text" value="...">`:

`property` Name des Feldes der Formular-Bean, das dem Eingabefeld zugeordnet wird. Beim Absenden des Formulars an den Server (Client -> Server) übernimmt das Feld der Bean den Wert des Eingabefeldes. Bei der Anzeige des Formulars (Server -> Client) wird der im Feld der Bean enthaltene Wert im Eingabefeld angezeigt.

Es verwaltet auch den Hintergrund des Absatzes in einer Tabelle

- Es verwaltet auch den Hintergrund des Absatzes in einer Tabelle
- 1. Es verwaltet auch den Hintergrund des Absatzes in einer Tabelle

2.9. Fußnoten

Der Gemini-Konverter² / ChatGPT unterstützt Fußnoten. Hier ist eine weitere Fußnote³.

3. Was es im Internet gibt

Bevor wir beginnen, können wir die KI fragen, was es auf dem Markt gibt, um ein Word- oder ODT-Dokument in eine statische HTML-Website zu konvertieren, und was kostenlos wäre.

Stellen wir Gemini die Frage:

Existe-t-il sur internet un logiciel gratuit capable de convertir un document Word en site statique HTML ?

Hier ist die Antwort:

Ja, es gibt mehrere kostenlose Lösungen, aber die Wahl hängt davon ab, wie Sie eine „statische Website“ definieren (eine einfache HTML-Seite vs. eine vollständige Website mit Navigation) und von Ihren technischen Fähigkeiten. Hier sind die besten Optionen, sortiert nach Methode:

- **1. Die „All-in-One“-Lösung (zum Erstellen einer echten, navigierbaren Website)**

Wenn Sie ein langes Word-Dokument (mit Kapiteln und Unterkapiteln) in eine vollständige Website mit einem Navigationsmenü auf der linken Seite umwandeln möchten, ist dies die beste Option.

- **Software: HelpNDoc** (kostenlose Version für den privaten Gebrauch)
- **So funktioniert es:** Es handelt sich um eine Software zur Erstellung von Dokumentationen. Sie importieren Ihre Word-Datei, das Programm analysiert die Struktur (Überschriften 1, Überschriften 2...) und generiert automatisch eine statische und „responsive“ (für Mobilgeräte optimierte) HTML5-Website.
- **Vorteile:** Erstellt eine echte Website mit Menü, Suchleiste und Index, ohne dass Sie eine einzige Zeile Code schreiben müssen.
- **Nachteil:** Die kostenlose Version fügt unten auf der Seite ein kleines Banner mit dem Text „Erstellt mit HelpNDoc“ ein.

²[Google Gemini](#)

³Die Fußnote

- **2. Die Lösung „Online-Bereiniger“ (zur seitenweisen Konvertierung)**

Wenn Sie ein kurzes Dokument haben und lediglich den HTML-Code benötigen, um ihn auf einer bestehenden Seite einzufügen, sind Online-Konverter ideal, um den von Word generierten „unsauberen“ Code zu bereinigen.

- **Tool: WordToHTML.net** (oder Word2CleanHTML)
- **So funktioniert es:**
 1. Sie kopieren Ihren Text aus Word.
 2. Fügen Sie ihn in den visuellen Editor auf der linken Seite der Website ein.
 3. Die Website generiert im rechten Fenster einen sauberen HTML-Code (bereinigt von unnötigen Microsoft-Tags).

- **Vorteile:** Sehr einfach, visuell, entfernt den „Ballast“ aus dem Microsoft-Code.
- **Nachteil:** Ideal für eine einzelne Seite, nicht zum Erstellen der Struktur einer ganzen Website.

- **3. Die „Pro/Entwickler“-Lösung (beste Qualität)**

Dies ist die von Experten empfohlene Methode, um einen extrem sauberen und zukunftssicheren Code zu erhalten.

- **Software: Pandoc** (Befehlszeile)
- **So funktioniert es:** Es handelt sich um ein kleines Programm ohne grafische Benutzeroberfläche (zur Verwendung über das Terminal/die Eingabeaufforderung). Es konvertiert Formate sehr präzise.
- **Beispielbefehl:** `pandoc mein_dokument.docx -t html -o meine_website.html`
- **Vorteile:** Perfekter HTML5-Code, kostenlos, Open Source, sehr gute Verarbeitung von Bildern und mathematischen Formeln (LaTeX).
- **Nachteile:** Man muss mit der Befehlszeile (schwarzer Bildschirm) vertraut sein.

- **4. Die „Behelfslösung“ (Microsoft Word)**

Sie können dies direkt in Word tun, aber achten Sie auf das Ergebnis.

- **Vorgehensweise:** Datei > Speichern unter > Wählen Sie „**Gefilterte Webseite (*.htm; *.html)**“.
- **Wichtig:** Wählen Sie nicht „Webseite mit einer einzigen Datei (.mht)“, sondern unbedingt „Gefiltert“. Die Option „Gefiltert“ reduziert (ein wenig) den überflüssigen Code, den Word normalerweise hinzufügt.
- **Nachteil:** Der Code bleibt oft schwerfällig und lässt sich später nur schwer ändern.

- **Schnellvergleich**

Lösung	Art des Ergebnisses	Codequalität	Einfachheit	Ideal für...
HelpNDoc	Vollständige Website (mit Menü)	Gut	Mittelmäßig	Handbücher, Dissertationen, Dokumentation
WordToHTML	Einzelne Seite	Ausgezeichnet	Sehr einfach	Blogartikel, einfache Seiten
Pandoc	Einzelseite	Perfekt	Schwierig	Entwickler, saubere

Lösung	Art des Ergebnisses	Codequalität	Einfachheit	Ideal für...
	(Rohdaten)			Archivierung
Word (nativ)	Einzelseite	Schlecht	Einfach	Schnelle Fehlerbehebung

- **Mein Tipp**

Wenn Sie dieses Dokument professionell online veröffentlichen möchten, empfehle ich Ihnen **HelpNDoc**, wenn Sie Anfänger sind und sofort eine Website-Struktur benötigen.

Wenn Sie lediglich den sauberen Inhalt zum Einfügen an anderer Stelle (WordPress usw.) benötigen, verwenden Sie **WordToHTML.net**.

Möchten Sie, dass ich Ihnen die Vorgehensweise für eines dieser Tools (z. B. WordToHTML) Schritt für Schritt erkläre?

Ich habe [Pandoc] gelegentlich verwendet, fand die Bedienung jedoch schwierig. Ich erinnere mich, dass man das Word-Dokument „vorbereiten“ musste, damit es von Pandoc korrekt verarbeitet wurde. Das Ergebnis war jedoch hervorragend. **HelpNDoc** scheint eine attraktive Software zu sein, da sie eine Website mit Menü generiert. Ich habe sie an einem meiner Word-Dokumente getestet, und das Ergebnis war nicht zufriedenstellend.

Nun wollen wir schauen, ob es eine Konvertierungssoftware für ODT-Dokumente gibt:

Existe-t-il sur internet un logiciel gratuit capable de transformer un document ODT en site HTML statique ?

Die Antwort von Gemini lautet wie folgt:

Ja, auf jeden Fall. Wie bei Word gibt es auch für das **ODT-Format** (das von LibreOffice und OpenOffice verwendet wird) eigene Lösungen.

Hier sind die besten kostenlosen Optionen speziell für ODT-Dateien, sortiert nach Methode:

- **1. Die „Erweiterungs“-Lösung (die beste integrierte Qualität)**

Wenn Sie LibreOffice oder OpenOffice verwenden, ist der einfache HTML-Export (wie bei Word) eher mittelmäßig. Es gibt jedoch eine hervorragende kostenlose Erweiterung, die speziell für diesen Zweck entwickelt wurde.

- **Tool: Writer2xhtml** (Erweiterung für LibreOffice)

- **So funktioniert es:**

1. Sie laden die Erweiterung herunter und installieren sie in LibreOffice.
2. Gehen Sie auf „Datei“ > „Exportieren“ und es erscheint eine neue Option „XHTML 1.1 + MathML“ (oder ähnlich).

- **Vorteile:** Der Code ist viel sauberer als beim Standard-Export und die Erweiterung verarbeitet mathematische Formeln und komplexe Formatierungen sehr gut.

- **Nachteil:** Es muss eine Erweiterung installiert werden (einmaliger technischer Eingriff).

- **2. Die Lösung „Pro / Befehlszeile“ (universell)**

Es handelt sich um dasselbe Tool wie für Word, da es auch mit dem ODT-Format hervorragende Ergebnisse liefert.

- **Software: Pandoc**
- **So funktioniert es:** Über das Befehlsterminal.
- **Befehl:** `pandoc meine_datei.odt -t html -o meine_website.html`
- **Vorteile:** Dies ist zweifellos der leistungsstärkste Konverter auf dem Markt für das ODT-Format in HTML. Er „versteht“ die Struktur des ODT-Dokuments besser als jeder andere.
- **3. Die Lösung „Komplette Website mit Navigation“ (über HelpNDoc)**

HelpNDoc (bereits erwähnt) ist nach wie vor die beste kostenlose Option, um eine echte Website mit Menü auf der linken Seite zu erstellen.

Allerdings importiert HelpNDoc das .odt-Format nicht immer so gut wie das .docx-Format.

- **Der Trick:**
 1. Öffnen Sie Ihre Datei in LibreOffice.
 2. Wählen Sie „Speichern unter...“ und wählen Sie „**Word 2007-365 (.docx)**“.
 3. Importieren Sie diese .docx-Datei in HelpNDoc, um Ihre Website zu erstellen.
- **Warum?** Dadurch wird sichergestellt, dass die Hierarchie Ihrer Kapitel (Überschriften 1, Überschriften 2) korrekt erkannt wird, um das Menü der Website zu erstellen.
- **4. Die Online-Lösung (Schnellbereinigung)**

Einige Online-Tools akzeptieren ODT-Dateien direkt und geben sauberes HTML aus.

- **Tool: WordToHTML.net** (akzeptiert auch ODT-Dateien)
- **So funktioniert es:** Sie können Ihre ODT-Datei direkt hochladen oder Ihren Text einfügen.
- **Vorteile:** Sehr praktisch, um einen einzelnen Artikel zu konvertieren, ohne etwas installieren zu müssen.
- **Zusammenfassung meiner Empfehlungen für ODT:**
 1. **Für eine richtige Website (Dokumentation/Dissertation):** Konvertieren Sie Ihre ODT-Datei über LibreOffice in **DOCX** und verwenden Sie dann **HelpNDoc** (das ist die zuverlässigste Methode, um ein Navigationsmenü zu erhalten).
 2. **Für eine einfache und übersichtliche Seite:** Verwenden Sie die Erweiterung **Writer2xhtml**, wenn Sie dies häufig tun, oder **Pandoc**, wenn Sie den Vorgang automatisieren möchten.

Soll ich Ihnen erklären, wie Sie die Erweiterung **Writer2xhtml** in LibreOffice installieren?

Ich habe die LibreOffice-Erweiterung [Writer2xhtml] an einem meiner Dokumente getestet. Das Ergebnis ist lesbar, aber nicht professionell genug. Es gibt kein Inhaltsverzeichnis auf der linken Seite. Die Code-Blöcke verlieren ihre Syntaxhervorhebung.

Letztendlich denke ich, dass der von Gemini generierte Konverter eine Alternative zu den oben genannten Optionen darstellt. Das Ergebnis ist mit dem von Pandoc vergleichbar, sofern das ODT-Dokument für den Konverter geeignet ist.

4. Die anfängliche Eingabeaufforderung an Gemini

3

Wir gehen hier davon aus, dass der Leser über erste Kenntnisse der KI [Gemini](#) verfügt. Ist dies nicht der Fall, wird

empfohlen, das Dokument [ref1] zu lesen. Die anfängliche Eingabeaufforderung an Gemini erfolgte in Form der folgenden Textdatei [Prompt User.txt]:

```
1. Anforderungen des Benutzers
2.
3. - Erstelle ein Python-Skript [convert], um aus einem ODT-Dokument eine MkDocs-Website mit dem
   Theme „material“ zu erstellen. Der Name des ODT-Dokuments ist das einzige Argument für
   convert.
4.
5. - Erstelle anschließend ein Python-Skript [build], das die von convert erstellte Datei
   [mkdocs.yml] nutzt, um eine statische HTML-Website zu erstellen
6. - Am Ende zeigt das Skript [build] die Startseite der Website in einem Browser an.
7.
8. - Die Website wird zwei unabhängige Spalten haben. Die linke Spalte enthält das
   Inhaltsverzeichnis, die rechte den Inhalt des ODT-Dokuments.
9.
10. - Das Inhaltsverzeichnis enthält alle Überschriften des ODT-Dokuments (Überschrift 1,
    Überschrift 2, Überschrift 3 usw.), ihre Nummerierung und ihre Hierarchie
11.
12. - Die Bilder befinden sich meist in Tabellen und wurden in ihrer Größe angepasst.
13. Du suchst in den Tabellen nach diesen Bildern und behältst ihre neuen Abmessungen bei.
14.
15. - Das ODT-Dokument enthält Code-Blöcke. Diese sind an ihren Stilen [Nummerierter Quellcode
    Ergebnisse, Quellcode] zu erkennen.
16. Die anderen Attribute des Code-Absatzes müssen ignoriert werden. Wenn der Code-Absatz
    beispielsweise auch eine nummerierte Liste ist, musst du dieses Attribut ignorieren.
17. Alle diese Code-Blöcke müssen wie Code auf der MkDocs-Website behandelt werden. Die Code-
    Zeilen müssen im HTML-Ergebnis nummeriert sein.
18.
19. - Für alle Code-Blöcke verwendest du die Python-Syntaxhervorhebung.
20.
21. - Du darfst die Codezeilen nicht verändern. Insbesondere darfst du die Einrückungen oder
    Leerzeichen vor dem ersten Zeichen der Codezeile nicht entfernen.
22.
23. - Du musst alle Aufzählungslisten finden. ODT bietet mehrere Möglichkeiten, diese zu
    erstellen. Du musst sie alle ausprobieren. Die Aufzählungslisten sind
24. manchmal ineinander verschachtelt. Du musst diese Verschachtelung beibehalten.
25.
26. - Der Titel der Website lautet „Ein Python-Skript mit KI-Tools generieren“
27.
28. - Du wirst jedes aus dem ODT-Dokument extrahierte Bild sowie jedes erstellte MkDocs-Kapitel
    einbinden
29.
30. - Du musst die Links, die du im ODT-Dokument findest, beibehalten. Sie müssen im generierten
    HTML-Dokument als Links erhalten bleiben.
31.
32. - Du musst fettgedruckte, unterstrichene oder kursiv geschriebene Texte beibehalten. Sie
    müssen im HTML-Dokument unverändert wiedergegeben werden
33.
```

- Zeile 19: Der erste Test wurde an einem zuvor veröffentlichten Kurs durchgeführt, der Python-Code enthielt;
- Diese Eingabeaufforderung ermöglichte den Start des Chats. Nach einigen Dutzend Durchläufen wurde sie überflüssig;

5. Eine Python-Arbeitsumgebung erstellen

Gemini und ChatGPT werden zwei Python-Skripte generieren. Wir benötigen daher eine Python-Umgebung:

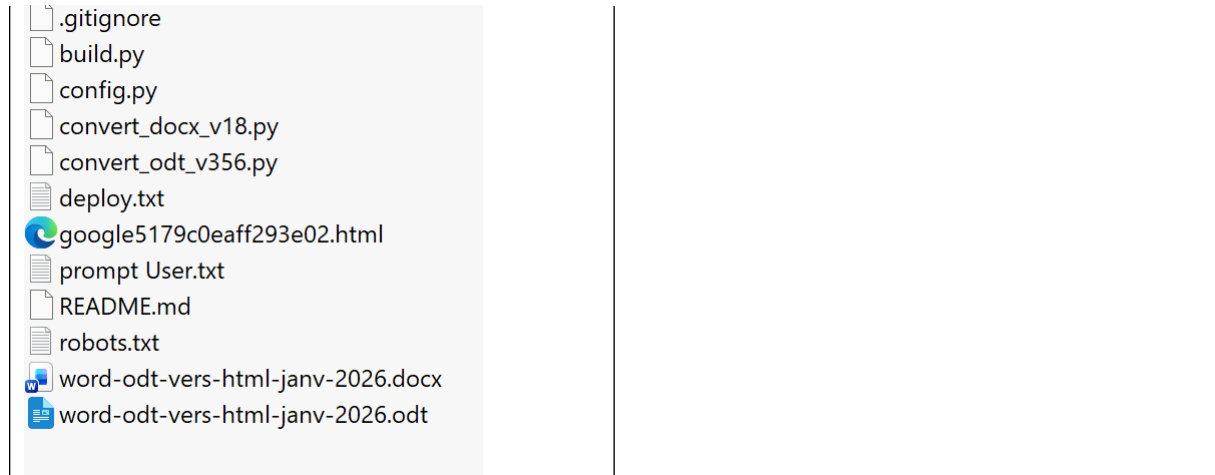
- Die Installation von Python wird in [Abschnitt 3.1](#) von [ref1] beschrieben;
- Die Installation von PyCharm wird in [Abschnitt 3.2](#) von [ref1] beschrieben. [ref1] enthält eine ausführliche Anleitung zur Konfiguration von PyCharm. Hier wird PyCharm jedoch nur zur Ausführung der von Gemini und ChatGPT generierten Python-Skripte verwendet. Daher ist eine Konfiguration von PyCharm

nicht erforderlich. PyCharm wird außerdem verwendet, um die für den Konverter erforderlichen Dateien bereitzustellen;

6. Der Arbeitsordner des Konverters

Sie können den Arbeitsordner [|HIER|](#) herunterladen.

Der Arbeitsordner sieht wie folgt aus:



- Die folgenden Dateien werden wir vorerst ignorieren [.gitignore, deploy.txt, google*.html, README.md, robots.txt, deploy.txt]. Diese Dateien dienen uns zur Bereitstellung der lokal generierten HTML-Website auf GitHub;
- [prompt User.txt] ist die Eingabeaufforderung, mit der ich die Iterationen mit Gemini begonnen habe;
- Die Datei [word-odt-vers-html-janv-2026.odt] ist das ODT-Dokument dieses Artikels. Es wird durch das Skript [convert_odt_v356.py] in HTML konvertiert;
- Die Datei [word-odt-vers-html-janv-2026.docx] ist das DOCX-Dokument desselben Artikels. Es wird durch das Skript [convert_docx_v18.py] in HTML konvertiert;
- Die Skripte [convert*.py] und [build.py] sind die beiden von der KI generierten Python-Skripte;
 - [convert*] konvertiert ein ODT- oder DOCX-Dokument in eine [MkDocs]-Website. Eine MkDocs-Website ist eine statische Website, deren Seiten in der [Markdown]-Syntax geschrieben sind. [MkDocs] stellt einen Server bereit, der MkDocs-Websites anzeigen kann;
 - [build] konvertiert die MkDocs-Website in eine statische Standard-HTML-Website. Am Ende dieser Konvertierung öffnet es einen Browser, um das Stammverzeichnis der Website anzuzeigen;

Wir werden uns zu keinem Zeitpunkt den generierten Python-Code ansehen. Wir betrachten die beiden Skripte als zwei Black Boxes. Zu keinem Zeitpunkt habe ich den Code manuell geändert. Ich habe Gemini / ChatGPT immer gebeten, sein Skript selbst zu korrigieren. Deshalb ist es nicht notwendig, Python zu beherrschen, um den Konverter zu verwenden.

- [config.py] ist eine Konfigurationsdatei für die [convert*]-Skripte. Ursprünglich hatte [convert*] keine Konfigurationsdatei. Als ich dann die zu konvertierenden Dokumente wechselte, stellte ich nach und nach fest, dass es Elemente gab, die ich von der KI generieren ließ (zum Beispiel den Namen der Website, der sich bei jedem Dokument ändert), die besser in einer Konfigurationsdatei untergebracht wären, die der Benutzer selbst ändern könnte. Auch diese habe ich in Zusammenarbeit mit der KI erstellt;

7. Die Konfigurationsdateien der Konverter

- 1  config.py
- 2  config_code.py
- 3  config_colors.py
- 4  config_presentation.py
- 5  config_styles.py
- 6  config_texts.py
- 7  config_urls.py

Die Konfigurationsdatei [config.py] fasst die Informationen zusammen, die sich von Dokument zu Dokument ändern können. Sie nutzt 6 weitere Konfigurationsdateien. In diesen haben wir die Parameter zusammengefasst, die sich beim Wechsel von einem Dokument zum anderen für die Konvertierung am ehesten ändern.

Die Datei [config.py] importiert den Inhalt der 6 anderen Konfigurationsdateien:

```
1. from config_urls import URLS
2. from config_texts import TEXTS
3. from config_code import CODE
4. from config_styles import STYLES
5. from config_colors import COLORS
6. from config_presentation import PRESENTATION
```

config_urls

Diese Datei enthält die drei URLs aus der Konfiguration [config.py]. Diese ändern sich mit jedem Dokument.

```
1. URLS = {
2.     "site_url": "https://stahe.github.io/de-word-odt-vers-html-janv-2026/",
3.     "repo_url": "https://stahe.github.io/de-word-odt-vers-html-janv-2026",
4.     "author_site": "https://tahe.developpez.com",
5. }
```

- Zeile 2: Die URL der Website, auf der Sie die vom Konverter erstellte HTML-Seite bereitstellen möchten;
- Zeile 3: die URL Ihres Git-Repositorys (siehe Abschnitt12);
- Zeile 4: Die Website des Erstellers der Website. Diese URL wird am Ende der Seiten der Website platziert. Kann leer sein.

config_texts

Diese Datei enthält die Texte aus [config.py], die sich bei einer Sprachumstellung der generierten HTML-Website ändern können.

```
1. TEXTS = {
2.     "home_label": "Startseite",
3.     "site_name": "Konvertieren Sie ein Word- oder ODT-Dokument in eine statische, MkDocs-
kompatible HTML-Website mit den KI-Modellen Gemini 3 und ChatGPT 5.2",
4.     "site_description": "Konvertieren Sie ein Word- oder ODT-Dokument in eine statische, MkDocs-
kompatible HTML-Website mit den KI-Modellen Gemini 3 und ChatGPT 5.2",
```

```

5. "toggle_to_dark": "In den Dunkelmodus wechseln",
6. "toggle_to_light": "Zum hellen Modus wechseln",
7. "footer_license_sentence": (
8. "Dieser Kurs-Tutorial, verfasst von <strong>Serge Tahé</strong>, wird der Öffentlichkeit gemäß
den Bedingungen der\n"
9. "      <em>Creative Commons-Lizenz: Namensnennung - Keine kommerzielle Nutzung -\n"
10. "      Weitergabe unter gleichen Bedingungen 3.0 Unported</em>.\n"
11. ),
12. "copy_label": "Kopieren",
13. "copy_copied_label": "Kopiert",
14. }

```

- Zeile 2: Der Name des ersten Links im Inhaltsverzeichnis. Sein Inhalt umfasst alles, was im Dokument vor der ersten Überschrift der Ebene 1 (Überschrift 1) steht. Dies ist der Anfang des Dokuments;
- Zeile 3: Der Titel des Dokuments. Er wird in der oberen Leiste der generierten Website angezeigt;
- Zeile 4: ebenfalls der Titel des Dokuments, diesmal für Suchmaschinen;
- Zeilen 5–6: die Tooltips, die angezeigt werden, wenn man mit der Maus über das Symbol in der oberen Leiste der Website fährt, um vom hellen Modus in den dunklen Modus oder umgekehrt zu wechseln;
- Zeilen 7–11: die Lizenz der Website. Sie wird in der Fußzeile der generierten Website angezeigt;
- Zeilen 12–13: Der auf der Website angezeigte Code wird zusammen mit einer Schaltfläche angezeigt, über die man den Code in die Zwischenablage kopieren kann. Zeile 12 ist die Beschriftung der Schaltfläche vor dem Kopieren, Zeile 13 die Beschriftung, nachdem der Code kopiert wurde;

So sieht diese Datei für eine englische Version des Dokuments aus:

```

1. TEXTS = {
2.   "home_label": "Welcome",
3.   "site_name": "Convert a Word or ODT document into a static HTML website using Gemini 3 and
ChatGPT 5.2",
4.   "site_description": "Konvertieren Sie ein Word- oder ODT-Dokument mithilfe von Gemini 3
und ChatGPT 5.2 AI in eine statische HTML-Website",
5.   "toggle_to_dark": "Zum Dunkelmodus wechseln",
6.   "toggle_to_light": "Zum hellen Modus wechseln",
7.   "footer_license_sentence": (
8.     'Dieses Tutorial, verfasst von <strong>Serge Tahé</strong>, wird der Öffentlichkeit
unter den Bedingungen der\n'
9.     '<em>Creative Commons Attribution - Non-Commercial -\n'
10.    '<em>ShareAlike 3.0 Unported</em> zur Verfügung gestellt.'
11.   ),
12.   "copy_label": "Kopieren",
13.   "copy_copied_label": "Kopiert",
14. }

```

config_styles

Diese Datei dient dazu, den Namen des Stils für den Dokumenttitel anzugeben. Bei einem DOCX-Dokument ist dies in der Regel „Titel“, bei einem ODT-Dokument variiert dies jedoch von Dokument zu Dokument.

```

1. STYLES = {
2.   "style_names": [
3.     "P1"
4.   ]
5. }

```

- Zeile 2: [styles_names] ist ein Array. Hier können die Namen der Stile eingegeben werden, die der Titel des Dokuments haben kann;
- Zeile 3: Das Dokument [word-odt-vers-html-janv-2026.odt] hat eine Überschrift mit dem Stil P1;

config_code

Code-Blöcke im DOCX- oder ODT-Dokument können in zwei Formen vorkommen:

- als formatierten Code (Farben, Fettdruck, Kursivschrift, ...). In diesem Fall wird er unverändert in die HTML-Seite kopiert;
- als reiner Text (Plain Text). In diesem Fall werden den Konvertern Anweisungen gegeben, damit sie selbst die im Code-Block verwendete Sprache erkennen. Dazu werden Schlüsselwörter angegeben:

```

1. CODE = {
2.   # Regeln zur automatischen Spracherkennung basierend auf dem Inhalt
3.   "detection_rules": {
4.     "csharp": [
5.       "using", "Console.WriteLine", "public static void Main", "WebMethod",
6.       "TryParse", "EventArgs", "String.Format", "System.Web.Services"
7.     ],
8.     "java": [
9.       "System.out.println", "public static void main(String", "package",
10.      "JUnitTest", "Class.forName", "PreparedStatement", "private static void",
11.      "private void", "getAgendaMedecinJour", "@PostConstruct", "@ResponseBody",
12.      "@RequestMapping", "getMedecin", "@Entity", "@Autowired", "@Bean",
13.      "Serializable", "getClient", "getCreneau", "getRv", "PostAjouterRv",
14.      "PostSupprimerRv", "@EnableJpaRepositories", "@Component", "getAgendaMedecinJour",
15.      "getResponse", "getMessagesForException", "getBase64", "ajouterRv",
16.      "Antwort", "getPartialViewAgenda", "setModelForAgenda", "ActionContext",
17.      "getActionContext", "PostLang", "PostUser", "PostGetAgenda", "@NotNull",
18.      "@EnableAutoConfiguration", "HttpSecurity"
19.    ],
20.    "html": [
21.      "<html>", "</div>", "<body>", "<script>", "href=", "<span>", "<p>",
22.      "<h2>", "<form>", "<table>", "<input"
23.    ],
24.    "sql": [
25.      "SELECT", "INSERT INTO", "UPDATE", "DELETE FROM", "WHERE",
26.      "CREATE TABLE", "ALTER TABLE"
27.    ],
28.    "python": [
29.      "def", "import", "print(", "from"
30.    ],
31.    "xml": [
32.      "<?xml", "<project", "<version>", "<configuration>", "<build>",
33.      "<dependency>", "<properties>", "<configuration>", "<start-class>"
34.    ],
35.    "javascript": [
36.      "use strict", "console.log", "let", "constructor", "async", "export"
37.    ],
38.    "php": [
39.      "<?php", "declare", "require"
40.    ],
41.    "vbscript": [
42.      "Option", "Dim", "Explicit"
43.    ],
44.    "markdown": [
45.      "# ", "## ", "### ", "***", "_", "![", "]" # Typische Markdown-Markierungen
46.    ],
47.  },
48. }

```

Auch hier gelten diese Erkennungsregeln nur für Code-Blöcke im „Plain Text“-Format und nicht für Code-Blöcke mit Formatierung. Wenn Sie auf Ihrer Website feststellen, dass Code-Blöcke nicht farbig dargestellt werden, sollten Sie hier nachsehen.

Für jede Sprache werden die Zeichenfolgen angegeben, anhand derer diese Sprache identifiziert werden kann. Bestimmte Schlüsselwörter können in verschiedenen Sprachen vorkommen. Der Konverter wählt die Sprache aus, für die er die meisten Schlüsselwörter gefunden hat. Das Prinzip ist einfach: Sie sehen sich Ihren Code an, wählen charakteristische Schlüsselwörter aus und tragen diese in die entsprechenden Zeilen für die richtige Sprache ein. Beachten Sie, dass Sie nicht beliebige Namen als Sprachbezeichnung verwenden können: Es müssen die von MkDocs anerkannten Namen verwendet werden;

config presentation

Diese Datei legt das Erscheinungsbild verschiedener Elemente der generierten Website fest. Das ist nichts, was man oft ändern muss:

```
1. PRESENTATION = {
2.   # -----
3.   # Dokumenttitel
4.   # -----
5.   "document_title": {
6.     "font_size": "28px",
7.     "font_weight": "bold",
8.     "margin_bottom": "1em",
9.     "Zeilenhöhe": "1.2"
10.  },
11.
12.  # -----
13.  # Rich-Code-Blöcke
14.  # -----
15.  "code": {
16.    "rich_line_height": "12px",
17.    "rich_font_family": "Consolas, 'Courier New', monospace",
18.    "rich_font_size": "15px"
19.  },
20.
21.  # -----
22.  # Kopieren-Schaltfläche
23.  # -----
24.  "copy_button": {
25.    "container": "position: relative;",
26.    "btn": (
27.      "position: absolute; top: .5rem; right: .5rem; "
28.      "display: inline-flex; align-items: center; justify-content: center; "
29.      "gap: .35rem; "
30.      "padding: 0,25rem 0,6rem; "
31.      "font-size: .72rem; font-weight: 600; letter-spacing: .01em; "
32.      "Zeilenhöhe: 1,2; "
33.      "border-radius: 999px; "
34.      "box-shadow: 0 1px 2px rgba(0,0,0,.10); "
35.      "backdrop-filter: saturate(180%) blur(6px); "
36.      "cursor: pointer; user-select: none; "
37.      "transition: transform .08s ease, box-shadow .12s ease, background .12s ease; "
38.      "z-index: 5;"
39.    ),
40.    "btn_hover": (
41.      "box-shadow: 0 3px 10px rgba(0,0,0,.18); "
42.      "transform: translateY(-1px);"
43.    ),
44.    "btn_copied": "opacity: .85;"
45.  },
46.
47.  # -----
48.  # Bilder
49.  # -----
50.  "images": {
51.    "shadow": {
52.      "enabled": True,
53.      "border_radius": "8px"
54.    }
55.  },
56.
57.  # -----
58.  # Rahmen / Kopfzeilenrand
59.  # -----
60.  "frame": {
61.    "header_top_border_width": "4px"
62.  }
63. }
```

Wenn Sie möchten, können Sie das Erscheinungsbild ändern:

- Zeilen 5–10: Titel des Dokuments;
- Zeilen 15–19: der Zeilen mit Rich-Text-Code;
- Zeilen 24–45: der Schaltfläche [Kopieren], mit der Sie Code in die Zwischenablage kopieren können;
- Zeilen 50–55: der Schatten der Bilder;
- Zeilen 60–62: des Rahmens, der die obere Leiste der Seiten der Website umgibt;

config_colors

In dieser Datei legen Sie die Farben Ihrer Website fest, vor allem die Hintergrundfarbe des oberen Balkens auf den Seiten:

```

1.  COLORS = {
2.      "theme": {
3.          "palette": {
4.              "light": {
5.                  "media": "(prefers-color-scheme: light)",
6.                  "scheme": "default",
7.                  "primary": "teal",
8.                  "accent": "purple"
9.              },
10.             "dark": {
11.                 "media": "(prefers-color-scheme: dark)",
12.                 "scheme": "slate",
13.                 "primary": "teal",
14.                 "accent": "purple"
15.             }
16.         }
17.     },
18.     "frame": {
19.         # "header_bg_light": "#1E88E5", (blau)
20.         "header_bg_light": "",
21.         "header_bg_dark": "",
22.         "header_top_border_light": "#CC7722",
23.         "header_top_border_dark": "#CC7722#CC7722"
24.     },
25.     "document_title": {
26.         "color": "#2c3e50"
27.     },
28.     "copy_button": {
29.         "border": "rgba(0,150,136,.45)",
30.         "background": "rgba(0,150,136,.12)",
31.         "text": "rgb(0,150,136)",
32.         "background_hover": "rgba(0,150,136,.20)"
33.     },
34.     "images": {
35.         "shadow": {
36.             "zoomable": "0 8px 24px rgba(0,0,0,.28), 0 18px 60px rgba(0,0,0,.20)",
37.             "zoomable_hover": "0 12px 30px rgba(0,0,0,.32), 0 26px 80px rgba(0,0,0,.22)",
38.             "lightbox": "0 24px 80px rgba(0,0,0,.65)"
39.         }
40.     }
41. }

```

Hier muss man sich ein wenig mit CSS auskennen, was bei mir nicht der Fall ist. Ich habe nur die Farben aus den Zeilen 7 und 13 verwendet, die die Farbe des oberen Balkens festlegen.

config_files_to_copy

Diese Datei listet die Dateien auf, die in das Stammverzeichnis der generierten Website kopiert werden müssen:

```

1.  FILES_TO_COPY =[
2.      "google5179c0eaff293e02.html",

```

```

3.     "robots.txt",
4.     "de-word-odt-zu-html-Jan-2026.pdf",
5.     "de-word-odt-vers-html-janv-2026.zip"
6. ]

```

- Zeilen 2–3: sind für die Google-Verfolgung der generierten Website erforderlich (siehe Abschnitt 13);
- Zeilen 4–5: Die Dateien, die Sie in das Stammverzeichnis Ihrer Website kopieren möchten, um sie Ihren Besuchern zur Verfügung zu stellen;

config

Die Datei [config.py] ist die Konfigurationsdatei, die von beiden Konvertern verwendet wird. Sie enthält alle Informationen, die diese benötigen. Diese Datei sollte normalerweise niemals geändert werden. Da sie komplex ist, haben wir beschlossen, die Konfigurationsparameter, die sich ändern könnten, herauszulösen und in externe Dateien zu verschieben.

```

1. from config_urls import URLS
2. from config_texts import TEXTS
3. from config_code import CODE
4. from config_styles import STYLES
5. from config_colors import COLORS
6. from config_presentation import PRESENTATION
7. from config_files_to_copy import FILES_TO_COPY
8.
9. config = {
10.     "toc": {
11.         "home_label": TEXTS["home_label"]
12.     },
13.
14.     "mkdocs": {
15.         "site_name": TEXTS["site_name"],
16.         "site_url": URLS["site_url"],
17.         "site_description": TEXTS["site_description"],
18.         "site_author": "Serge Tahé",
19.         "repo_url": URLS["repo_url"],
20.         "repo_name": "GitHub",
21.         "use_directory_urls": False,
22.
23.         "theme": {
24.             "name": "material",
25.             "custom_dir": "overrides",
26.             "features": [
27.                 "navigation.sections",
28.                 "navigation.indexes",
29.                 "navigation.expand",
30.                 "toc.integrate",
31.                 "navigation.top"
32.             ],
33.             "palette": [
34.                 {
35.                     "media": COLORS["theme"]["palette"]["light"]["media"],
36.                     "scheme": COLORS["theme"]["palette"]["light"]["scheme"],
37.                     "primary": COLORS["theme"]["palette"]["light"]["primary"],
38.                     "accent": COLORS["theme"]["palette"]["light"]["accent"],
39.                     "toggle": {
40.                         "icon": "material/brightness-7",
41.                         "name": TEXTS["toggle_to_dark"]
42.                     }
43.                 },
44.                 {
45.                     "media": COLORS["theme"]["palette"]["dark"]["media"],
46.                     "scheme": COLORS["theme"]["palette"]["dark"]["scheme"],
47.                     "primary": COLORS["theme"]["palette"]["dark"]["primary"],
48.                     "accent": COLORS["theme"]["palette"]["dark"]["accent"],
49.                     "toggle": {
50.                         "icon": "material/brightness-4",
51.                         "name": TEXTS["toggle_to_light"]

```

```

52.         }
53.     }
54. ]
55. },
56.
57. "markdown_extensions": [
58.     "admonition",
59.     "attr_list",
60.     "pymdownx.superfences",
61.     "pymdownx.mark",
62.     {
63.         "pymdownx.highlight": {
64.             "anchor_linenums": True,
65.             "linenums": None
66.         }
67.     },
68.     "md_in_html",
69.     "footnotes"
70. ],
71.
72. "extra_javascript": [
73.     "javascripts/focus.js"
74. ],
75. "extra_css": [
76.     "stylesheets/focus.css"
77. ]
78. },
79.
80. "footer": (
81.     "{% block footer %}\n"
82.     " <div class=\"md-footer-meta md-typeset\">\n"
83.     "   <div class=\"md-footer-meta__inner\">\n"
84.     "     <div>\n"
85.     f"       <a href=\"{URLS['author_site']}\" target=\"_blank\">\n"
86.     f"       {URLS['author_site']}\n"
87.     "     </a>\n"
88.     "     <br>\n"
89.     f"     {TEXTS['footer_license_sentence']}\n"
90.     "   </div>\n"
91.     " </div>\n"
92.     " </div>\n"
93.     "{% endblock %}"
94. ),
95.
96. "extra": {
97.     "analytics": {
98.         "provider": "google",
99.         "property": "G-XXXXXXX"
100.    }
101. },
102.
103. "document_title": {
104.     "style_names": STYLES["style_names"],
105.     "css": (
106.         f"font-size: {PRESENTATION['document_title']['font_size']}; "
107.         f"font-weight: {PRESENTATION['document_title']['font_weight']}; "
108.         f"margin-bottom: {PRESENTATION['document_title']['margin_bottom']}; "
109.         f"line-height: {PRESENTATION['document_title']['line_height']}; "
110.         f"color: {COLORS['document_title']['color']};"
111.     )
112. },
113.
114. "code": {
115.     "case_insensitive_languages": ["sql", "html", "vbnet", "vbscript"],
116.     "style_keywords": ["code"],
117.     "default_language": "text",
118.     "rich_line_height": PRESENTATION["code"]["rich_line_height"],
119.     "rich_font_family": PRESENTATION["code"]["rich_font_family"],
120.     "rich_font_size": PRESENTATION["code"]["rich_font_size"],
121.     "detection_rules": CODE["detection_rules"],

```

```

122.     "copy_button": True,
123.     "copy_label": TEXTS["copy_label"],
124.     "copy_copied_label": TEXTS["copy_copied_label"],
125.     "copy_only_recognized_language": True,
126.     "copy_min_lines": 4,
127.     "copy_allow_pygments_heuristic": True,
128.     "copy_style": {
129.         "container": PRESENTATION["copy_button"]["container"],
130.         "btn": (
131.             PRESENTATION["copy_button"]["btn"]
132.             + f"border:1px solid {COLORS['copy_button']['border']}); "
133.             + f"background:{COLORS['copy_button']['background']}); "
134.             + f"color:{COLORS['copy_button']['text']}); "
135.         ),
136.         "btn_hover": (
137.             f"background:{COLORS['copy_button']['background_hover']}); "
138.             + PRESENTATION["copy_button"]["btn_hover"]
139.         ),
140.         "btn_copied": PRESENTATION["copy_button"]["btn_copied"]
141.     }
142. },
143.
144. "images": {
145.     "shadow": {
146.         "enabled": PRESENTATION["images"]["shadow"]["enabled"],
147.         "border_radius": PRESENTATION["images"]["shadow"]["border_radius"],
148.         "zoomable": COLORS["images"]["shadow"]["zoomable"],
149.         "zoomable_hover": COLORS["images"]["shadow"]["zoomable_hover"],
150.         "lightbox": COLORS["images"]["shadow"]["lightbox"],
151.     }
152. },
153.
154. "frame": {
155.     "header_bg_light": COLORS["frame"]["header_bg_light"],
156.     "header_bg_dark": COLORS["frame"]["header_bg_dark"],
157.     "header_top_border_light": COLORS["frame"]["header_top_border_light"],
158.     "header_top_border_dark": COLORS["frame"]["header_top_border_dark"],
159.     "header_top_border_width": PRESENTATION["frame"]["header_top_border_width"]
160. },
161.
162. "files_to_copy": FILES_TO_COPY,
163.
164. "debug": True
165. }

```

- Zeilen 1–7: Alle Konfigurationsdateien werden importiert;
- Zeile 9: Die Konfigurationsdatei ist ein Python-Skript. Es definiert eine einzige Variable [config]. Diese ist ein Wörterbuch, das alle Werte der Konfiguration enthält;
- Zeile 11: Der Text des Links „Startseite“;
- Zeile 14: Das Wörterbuch [mkdocs] konfiguriert die vom Gemini/ChatGPT-Konverter generierte Datei [mkdocs.yml]. Diese Datei wird anschließend vom Skript [build] verwendet, um aus der vom Konverter generierten MkDocs-Datei eine HTML-Website zu erstellen;
- Zeilen 15–20: Konfigurieren die GitHub-Seite, auf der die aus dem ODT-/DOCX-Dokument erstellte HTML-Seite gehostet wird (siehe Abschnitt 12)
- Zeile 21: Diese Zeile ist wichtig. Fehlt sie, öffnet der Browser statt einer HTML-Seite den Ordner dieser Seite;
- Zeile 24: MkDocs bietet mehrere Designs für eine MkDocs-/HTML-Website an. Hier haben wir das Design „material“ gewählt (Zeile 24);
- Zeilen 27–31: Konfiguration der Navigation auf der Website. Diese erfolgt über ein Inhaltsverzeichnis in der linken Spalte der angezeigten Seite (Zeile 30);
- Zeilen 33–54: Definieren zwei Farbpaletten: den hellen Modus (Zeilen 35–42) oder den dunklen Modus (Zeilen 45–52). Über ein Symbol in der oberen Leiste der angezeigten Seiten kann zwischen den beiden Modi gewechselt werden;

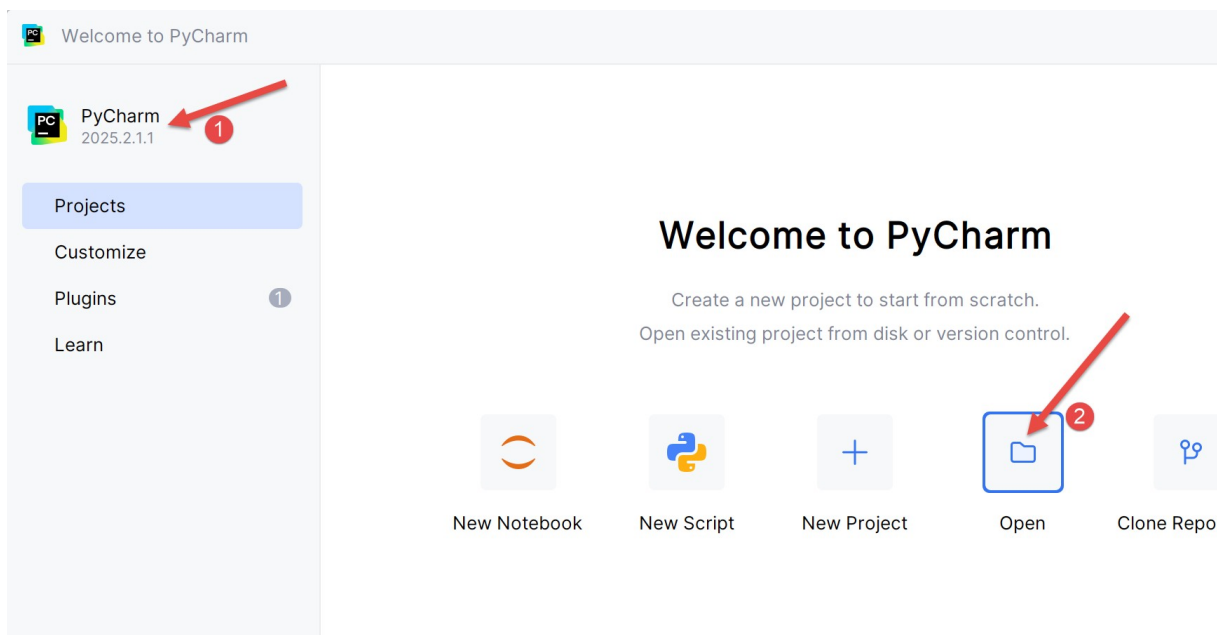
- Zeilen 57–70: Erweiterungen der von MkDocs verwendeten Markdown-Sprache. Diese Erweiterungen wurden von Gemini auf meine Anfrage hin generiert;
- Zeile 73: Das Skript [focus.js] ist ein von Gemini generiertes JavaScript-Skript. Es ist mit der Schaltfläche in der oberen Leiste der Website verknüpft, um das Inhaltsverzeichnis ein- oder auszublenden;
- Zeile 76: Das von dieser Schaltfläche verwendete CSS;
- Zeilen 80–95: Die Definition der Fußzeile der HTML-Website. Wurde von Gemini anhand eines Textbeispiels generiert, das ich ihm gegeben habe;
- Zeilen 97–100: Definition des Google-Analytics-Tags (GA), mit dem die Besuche der Website verfolgt werden können;
- Zeile 99: Fügen Sie hier Ihren GA-Tag ein;
- Zeilen 103–112: Der Stil des Dokumenttitels, der im ODT-/DOCX-Dokument vor der ersten Überschrift der Ebene 1 steht. Der in Zeile 104 angegebene Stil ändert sich bei jedem ODT-Dokument. Bei DOCX-Dokumenten kann er hingegen konstant sein („Titel“). Standardmäßig protokolliert der Konverter die Stile aller Absätze des ODT-/DOCX-Dokuments, die vor der ersten Überschrift der Ebene 1 stehen. Sie müssen also zunächst einen ersten Durchlauf durchführen, den Stil des Absatzes identifizieren, den Sie als Überschrift verwenden möchten, und diesen dann in die Datei [config_styles] eintragen;
- Zeilen 105–111: den CSS-Stil, den Sie der Überschrift des Dokuments zuweisen möchten. Diese Überschrift befindet sich auf der Seite „Startseite“, der ersten Seite, die beim Öffnen der Website angezeigt wird;
- Zeile 115: Hier geben Sie die groß-/kleinschreibungsunabhängigen Bezeichnungen ein;
- Zeile 116: Die Codeblöcke werden im ODT-/DOCX-Dokument anhand ihrer Stile identifiziert. Es kann mehrere davon geben. In Zeile 116 fügen Sie alle Schlüsselwörter ein, mit denen ein Codestil erkannt werden kann. In meinen Dokumenten enthalten alle meine Code-Stile das Wort „code“ in ihrem Namen. Und kein anderer Stil enthält dieses Wort in seinem Namen. Daher reicht es aus, ein einziges Schlüsselwort anzugeben. Wenn es mehrere gibt, trennt man sie in Zeile 116 durch Kommas;
- Zeile 117: Legt die Standardsprache fest. Bei einem „Plain-Text“-Codeblock ist die Standardsprache „text“, wenn keine Sprache erkannt wird. Für MkDocs bedeutet dies einen Codeblock ohne Syntaxhervorhebung. Dies ist beispielsweise bei Ausführungsergebnissen der Fall;
- Zeile 121: Diese Konfiguration wird für „Plain-Text“-Codes verwendet, die ursprünglich keine Syntaxhervorhebung haben. Diese Zeilen dienen dazu, ihnen eine solche entsprechend der im Code-Block verwendeten Sprache zuzuweisen. **Beachten Sie: Wenn Ihr Code vollständig formatiert ist, da er aus einer IDE wie Eclipse, Visual Studio usw. stammt, und Sie keinen „Plain-Text“-Code-Block haben, der einer Sprache zugeordnet ist, müssen Sie in diesem Teil der Konfiguration nichts eingeben. Sie können die Tabellen für die Sprachen leer lassen.** All dies erfolgt in der Code-Konfigurationsdatei [config_code];
- Zeilen 118–120: Diese Zeilen betreffen die formatierten Code-Blöcke (Fettdruck, Kursivschrift, Unterstreichung, Hervorhebung, Schriftfarbe). Diese Blöcke werden nicht wie „Plain-Text“-Code-Blöcke verarbeitet. Sie werden im HTML unverändert wiedergegeben;
 - Zeile 118: Legt die Zeilenhöhe der Codezeilen fest;
 - Zeile 119: Legt die CSS-Schriftart des Code-Blocks fest;
 - Zeile 120: Legt die Schriftgröße fest;
- Zeilen 144–152: legen das Erscheinungsbild der Bilder fest;
- Zeilen 154–160: legen das Erscheinungsbild des Rahmens fest, in dem die Webseite angezeigt wird;
- Zeile 162: Die Liste der Dateien, die in das Stammverzeichnis der HTML-Website kopiert werden sollen;
- Zeile 164: [debug] auf True setzt, werden die Stilprotokolle der Absätze vor der ersten Überschrift der Ebene 1 des ODT-/DOCX-Dokuments aktiviert. In diesen Protokollen findet man den genauen Stil des Absatzes, der als Überschrift auf der Startseite der Website dienen soll;

Letztendlich muss man beim Wechsel von einem ODT-/DOCX-Dokument zum anderen nichts an der Datei [config.py] ändern. Es werden nur die anderen Konfigurationsdateien angepasst.

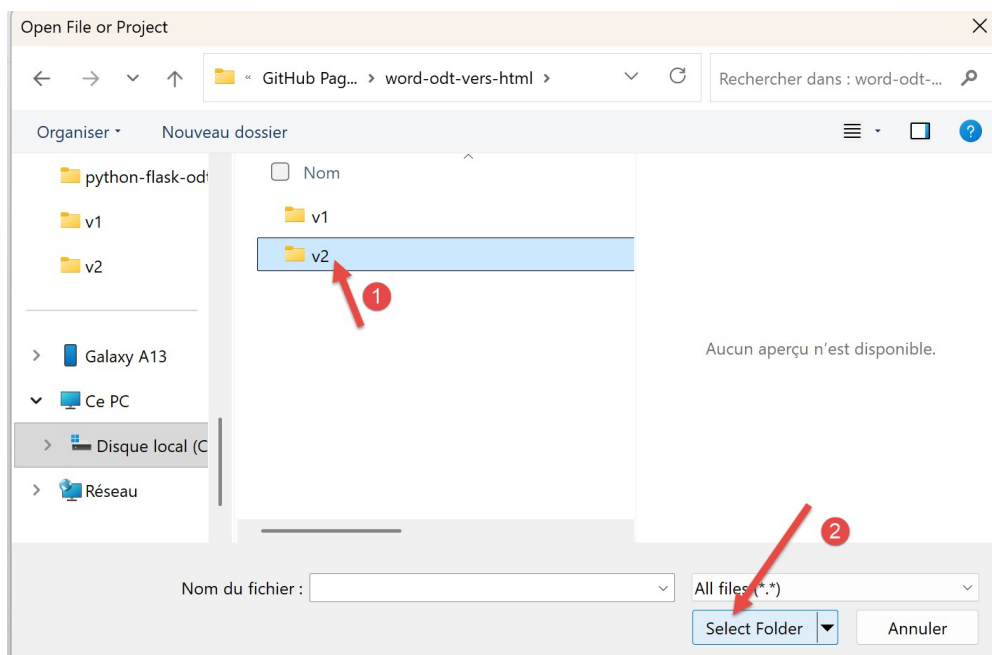
8. Verwendung des ODT-zu-HTML-Konverters

Zur Erinnerung: Der Arbeitsordner ist |HIER| zu finden.

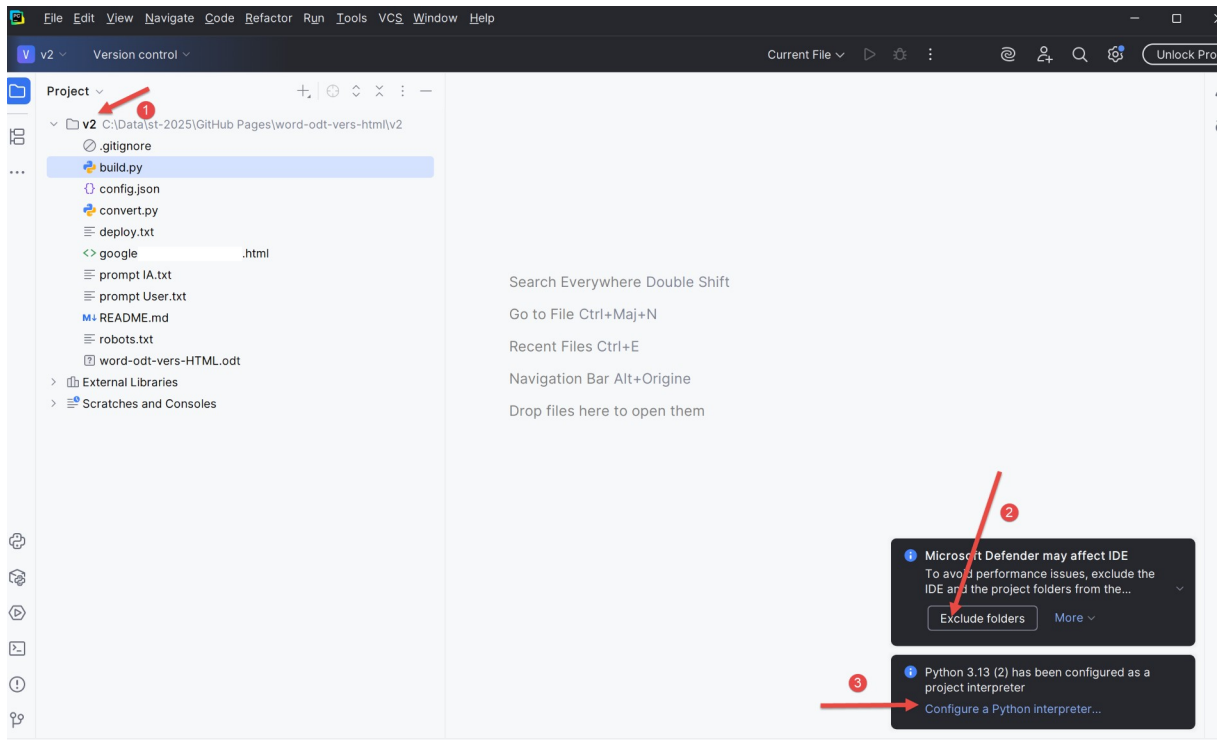
Starten wir PyCharm:



- Öffnen Sie unter [2] ein neues Projekt;

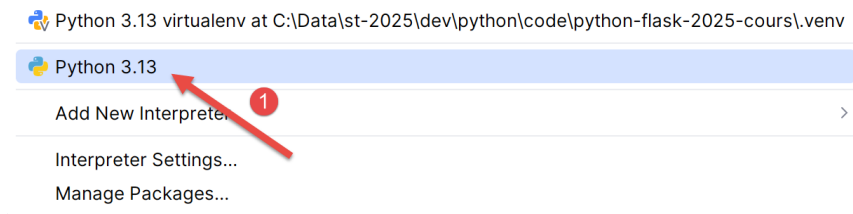


- Wählen Sie unter [1] den Ordner aus, in dem Sie den Arbeitsordner gespeichert haben;
- Bestätigen Sie diese Auswahl unter [2];



- Wählen Sie unter [1] Ihr Python-Projekt aus;
- Bestätigen Sie unter [2], damit Sie nicht vom Windows-Sicherheitssystem gestört werden;
- Wählen Sie unter [3] einen Python-Interpreter aus;

Python Interpreter



- Wählen Sie unter [1] den Python-Interpreter aus, den Sie im Abschnitt „5“ installiert haben;

Der Gemini/ChatGPT-Konverter verwendet Python-Abhängigkeiten, die Sie installieren müssen. Wir fragen Gemini, welche das sind:

rappelle moi toutes les dépendances Python nécessaires à ton convertisseur [convert]

Die Antwort von Gemini lautet wie folgt:

<https://tahe.developpez.com>

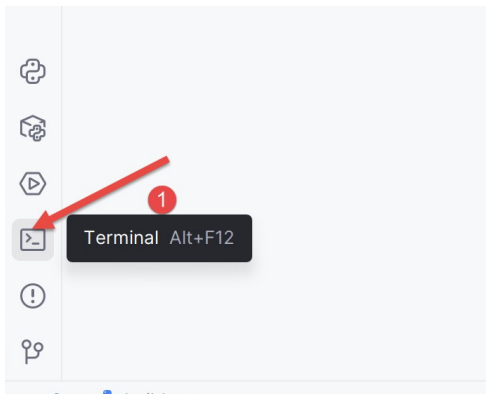
Commande d'installation rapide

Vous pouvez installer tout le nécessaire en une seule commande dans votre terminal :

```
Bash
```

```
pip install odfpy unicode mkdocs mkdocs-material
```

Wir werden diese Abhängigkeiten installieren:



- Öffnen Sie unter [1] ein Terminal;



- Geben Sie in [2] den von Gemini angegebenen Befehl ein;

Die Antwort lautet wie folgt:

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> pip install odfpy unicode mkdocs mkdocs-material
2. Standardmäßig wird die Benutzerinstallation verwendet, da die normalen Site-Pakete nicht beschreibbar sind
3. Anforderung bereits erfüllt: odfpy in c:\users\serge\appdata\roaming\python\python313\site-packages (1.4.1)
4. Anforderung bereits erfüllt: unicode in c:\users\serge\appdata\roaming\python\python313\site-packages (1.4.0)
5. Anforderung bereits erfüllt: mkdocs in c:\users\serge\appdata\roaming\python\python313\site-packages (1.6.1)

6. Anforderung bereits erfüllt: mkdocs-material in c:
\users\serge\appdata\roaming\python\python313\site-packages (9.7.0)
7. Anforderung bereits erfüllt: defusedxml in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus odfpy) (0.7.1)
8. Anforderung bereits erfüllt: click>=7.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs) (8.3.1)
9. Anforderung bereits erfüllt: colorama>=0.4 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (0.4.6)
10. Anforderung bereits erfüllt: ghp-import>=1.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (2.1.0)
11. Anforderung bereits erfüllt: jinja2>=2.11.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (3.1.6)
12. Anforderung bereits erfüllt: markdown>=3.3.6 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (3.10)
13. Anforderung bereits erfüllt: markupsafe>=2.0.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs) (3.0.2)
14. Anforderung bereits erfüllt: mergedeep>=1.3.4 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (1.3.4)
15. Anforderung bereits erfüllt: mkdocs-get-deps>=0.2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (0.2.0)
16. Anforderung bereits erfüllt: packaging>=20.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (25.0)
17. Anforderung bereits erfüllt: pathspec>=0.11.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (0.12.1)
18. Anforderung bereits erfüllt: pyyaml-env-tag>=0.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (1.1)
19. Anforderung bereits erfüllt: pyyaml>=5.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (6.0.2)
20. Anforderung bereits erfüllt: watchdog>=2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs) (6.0.0)
21. Anforderung bereits erfüllt: babel>=2.10 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (2.17.0)
22. Anforderung bereits erfüllt: backrefs>=5.7.post1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (6.1)
23. Anforderung bereits erfüllt: mkdocs-material-extensions>=1.3 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (1.3.1)
24. Anforderung bereits erfüllt: paginate>=0.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (0.5.7)
25. Anforderung bereits erfüllt: pygments>=2.16 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (2.19.2)
26. Anforderung bereits erfüllt: pymdown-extensions>=10.2 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (10.17.2)
27. Anforderung bereits erfüllt: requests>=2.26 in c:
\users\serge\appdata\roaming\python\python313\site-packages (aus mkdocs-material) (2.32.5)
28. Anforderung bereits erfüllt: python-dateutil>=2.8.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von ghp-import>=1.0->mkdocs) (2.9.0.post0)
29. Anforderung bereits erfüllt: platformdirs>=2.2.0 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von mkdocs-get-deps>=0.2.0->mkdocs) (4.5.0)
30. Anforderung bereits erfüllt: six>=1.5 in c:\users\serge\appdata\roaming\python\python313\site-packages (von python-dateutil>=2.8.1->ghp-import>=1.0->mkdocs) (1.17.0)
31. Anforderung bereits erfüllt: charset-normalizer<4,>=2 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von requests>=2.26->mkdocs-material) (3.4.3)
32. Anforderung bereits erfüllt: idna<4,>=2.5 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von requests>=2.26->mkdocs-material) (3.10)
33. Anforderung bereits erfüllt: urllib3<3,>=1.21.1 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von requests>=2.26->mkdocs-material) (2.5.0)
34. Anforderung bereits erfüllt: certifi>=2017.4.17 in c:
\users\serge\appdata\roaming\python\python313\site-packages (von requests>=2.26->mkdocs-material) (2025.8.3)
35. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2>

Auf meinem PC war bereits alles installiert. Falls dies bei Ihnen nicht der Fall ist, werden alle erforderlichen Abhängigkeiten installiert.

Jetzt können wir den Konverter verwenden. Geben Sie im geöffneten Terminal den folgenden Befehl ein:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_odt_v356.py .\word-odt-vers-html-janv-2026.odt .\config.py
2. --- ODT to MkDocs Converter V356 ---
3. Verarbeitung von .\word-odt-vers-html-janv-2026.odt...
4. Kopiert: google5179c0eaff293e02.html
5. Kopiert: robots.txt
6. Kopiert: word-odt-vers-html-janv-2026.pdf
7. Kopiert: word-odt-zu-html-Jan-2026.zip
8. Verweise werden gescannt...
9. Markdown-Generierung...
10. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
11. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
12. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
13. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
14. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
15. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
16. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
17. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
18. [DEBUG PRE-H1] Tag=text:p Style='Standard' Text='...'
19. [DEBUG PRE-H1] Tag=text:p Style='Standard (WW)' Text='...'
20. [DEBUG PRE-H1] Tag=text:p Style='Standard (WW)' Text='...'
21. [DEBUG PRE-H1] Tag=text:p Style='P1' Text='Ein Word- oder ODT-Dokument in eine statische Website konvertieren...'
```

```
22. >>> TITEL DES GEFUNDENEN DOKUMENTS: Ein Word- oder ODT-Dokument in eine statische, MkDocs-kompatible HTML-Website mit den KI-Modellen Gemini 3 und ChatGPT 5.2 konvertieren
23. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
24. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
25. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
26. [DEBUG PRE-H1] Tag=text:p Style='P3' Text='Serge Tahé, Januar 2026...'
```

```
27. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
28. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
29. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
30. [DEBUG PRE-H1] Tag=text:p Style='P4' Text='...'
31. [DEBUG PRE-H1] Tag=text:p Style='P4' Text='...'
32. [DEBUG PRE-H1] Tag=text:p Style='P4' Text='...'
33. [DEBUG PRE-H1] Tag=text:p Style='P4' Text='Diese Website wurde mit dem Konverter [Word oder ...]'
```

```
34. [DEBUG PRE-H1] Tag=text:h Style='P5' Text='Einleitung...'
```

```
35. >>> KAPITEL 1: 1 Einleitung
36. >>> KAPITEL 1: 2 Die Beispiele in diesem Dokument
37. >>> KAPITEL 2: 2.1 Listen
38. >>> KAPITEL 3: 2.1.1 Aufzählungslisten
39. >>> KAPITEL 3: 2.1.2 Nummerierte Listen
40. >>> KAPITEL 3: 2.1.3 Gemischte Listen 1
41. >>> KAPITEL 3: 2.1.4 Gemischte Listen 2
42. >>> KAPITEL 2: 2.2 Codeblöcke
43. >>> KAPITEL 3: 2.2.1 Rich-Text-Code-Blöcke (Eclipse, Visual Studio, ...)
```

```
44. >>> KAPITEL 3: 2.2.2 Blöcke mit einfachem Text (Plain Text)
45. >>> KAPITEL 2: 2.3 Links
46. >>> KAPITEL 2: 2.4 Textformatierung
47. >>> KAPITEL 2: 2.5 Auch eine Überschrift kann formatiert werden.
48. >>> KAPITEL 2: 2.6 Bilder
49. >>> KAPITEL 2: 2.7 Zu schützende Zeichen
50. >>> KAPITEL 2: 2.8 Tabellen
51. >>> KAPITEL 2: 2.9 Fußnoten
52. >>> KAPITEL 1: 3 Was es im Internet gibt
53. >>> KAPITEL 1: 4 Die Startaufforderung bei Gemini 3
54. >>> KAPITEL 1: 5 Eine Python-Arbeitsumgebung erstellen
55. >>> KAPITEL 1: 6 Der Arbeitsordner des Konverters
56. >>> KAPITEL 1: 7 Die Konfigurationsdatei des Konverters
57. >>> KAPITEL 1: 8 Verwendung des ODT-zu-HTML-Konverters
58. >>> KAPITEL 1: 9 Verwendung des DOCX-zu-HTML-Konverters
59. >>> KAPITEL 1: 10 Aufbau der statischen HTML-Website
60. >>> KAPITEL 1: 11 Überprüfung der generierten HTML-Website
61. >>> KAPITEL 2: 11.1 Die obere Leiste der Website
```

```

62. >>> KAPITEL 2: 11.2 Die Fußzeile der Website
63. >>> KAPITEL 2: 11.3 Die Startseite
64. >>> KAPITEL 2: 11.4 Aufzählungslisten
65. >>> KAPITEL 2: 11.5 Nummerierte Listen
66. >>> KAPITEL 3: 11.5.1 Gemischte Listen 1
67. >>> KAPITEL 3: 11.5.2 Gemischte Listen 2
68. >>> KAPITEL 2: 11.6 Erweiterte Code-Blöcke
69. >>> KAPITEL 3: 11.6.1 Beispiel 1
70. >>> KAPITEL 3: 11.6.2 Beispiel 2
71. >>> KAPITEL 3: 11.6.3 Beispiel 3
72. >>> KAPITEL 2: 11.7 Plain-Text-Blöcke
73. >>> KAPITEL 3: 11.7.1 Beispiel 1
74. >>> KAPITEL 3: 11.7.2 Beispiel 2
75. >>> KAPITEL 3: 11.7.3 Beispiel 3
76. >>> KAPITEL 2: 11.8 Weitere Codeblöcke
77. >>> KAPITEL 2: 11.9 Links
78. >>> KAPITEL 2: 11.10 Textformatierung
79. >>> KAPITEL 2: 11.11 Bilder
80. >>> KAPITEL 2: 11.12 Geschützte Zeichen
81. >>> KAPITEL 2: 11.13 Tabellen
82. >>> KAPITEL 2: 11.14 Fußnoten
83. >>> KAPITEL 2: 11.15 Bekannte Fehler
84. >>> KAPITEL 2: 11.16 Sonstige Fälle
85. >>> KAPITEL 1: 12 Die HTML-Website auf GitHub hosten
86. >>> KAPITEL 1: 13 Die Website mit Google Analytics und Google Search Console überwachen
87. >>> KAPITEL 1: 14 Fazit
88. Fertig.

```

- Zeile 1: Der Befehl, der das ODT-Dokument in eine MkDocs-Website konvertiert [`python convert_odt_v356.py .\word-odt-vers-html-janv-2026.odt .\config.py`]. Passen Sie die Versionsnummer (hier 356) an die von Ihnen heruntergeladene Version an. Der erste Parameter des Konverters ist das zu konvertierende ODT-Dokument, der zweite die Konfigurationsdatei des Konverters;
- Zeilen 4–7: die Dateien, die der Konverter in das Stammverzeichnis der von ihm erstellten MkDocs-Website kopiert;
- Zeilen 10–34: Debugging der Stile der Absätze, die dem ersten Titel der Ebene 1 vorangehen. Diese Absätze bilden die Startseite. Einer der Absätze fungiert als Titel der Startseite und somit der Website. Dies ist der Absatz in Zeile 21. Sein Stil lautet P1. Dieser Stil muss in die Konfigurationsdatei aufgenommen werden:

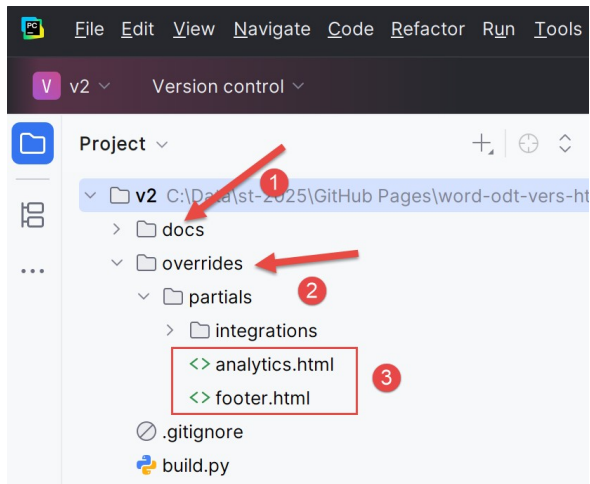
```

1.     STYLES = {
2.         "style_names": [
3.             "P1"
4.         ]
5.     }

```

- Zeilen 35–88: Ich habe die KI gebeten, alle Kapitel zu protokollieren, auf die sie stieß;

Diese Ausführung hat Ihren Arbeitsordner geändert:

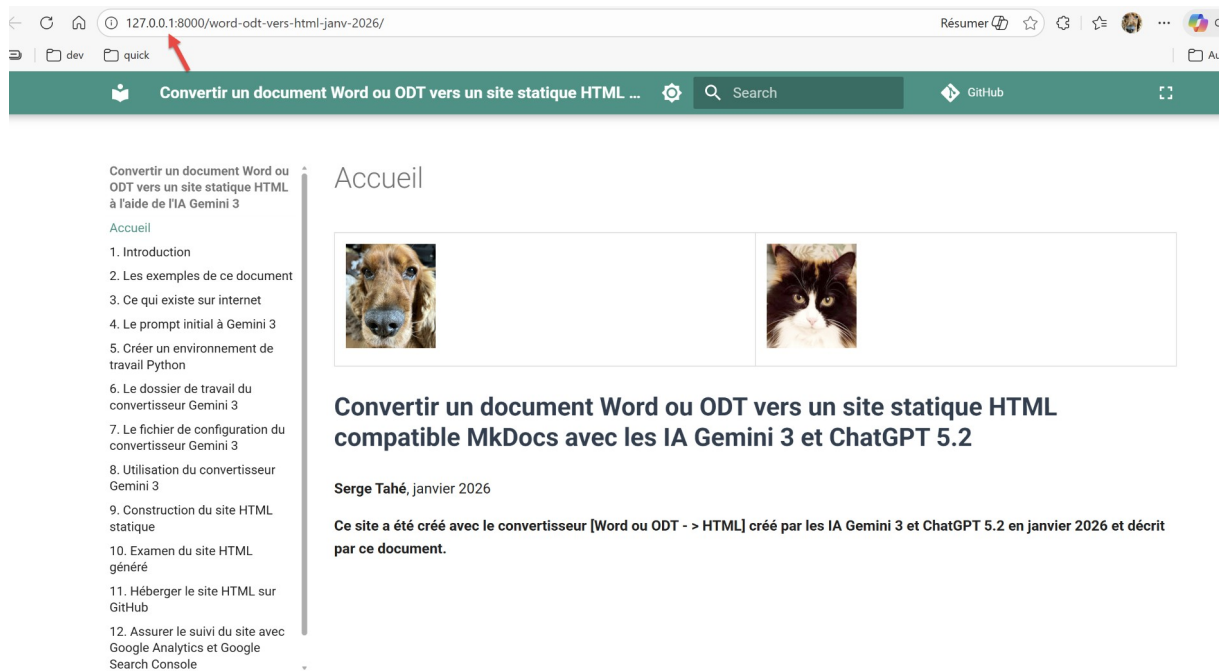


- In [1] ist [docs] die MkDocs-Website, die der Gemini/ChatGPT-Konverter erstellt hat. Vielleicht möchten Sie sie sich einmal ansehen;
- unter [2] wurde ein Ordner [overrides] erstellt. Dieser wird vom [build]-Tool für die HTML-Website verwendet;
- in [3]: [analytics.html] wird für die Nachverfolgung der Website durch Google Analytics verwendet. [footer.html] ist die Fußzeile, die Sie in der Datei [config.py] definiert haben;

Wir könnten die MkDocs-Website jetzt schon nutzen. Mit dem Befehl [python -m mkdocs serve] können Sie sie anzeigen. Probieren Sie es aus:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2. INFO - Dokumentation wird erstellt...
3. INFO - Site-Verzeichnis bereinigen
4. INFO - Die Dokumentdatei 'les-exemples.md' enthält einen Link '#_Les_exemples', aber auf
dieser Seite gibt es keinen solchen Anker.
5. INFO - Dokumentation in 0,41 Sekunden erstellt
6. INFO - [15:46:06] Bereitstellung unter http://127.0.0.1:8000/word-odt-vers-html-janv-2026/
```

Klicken Sie bei gedrückter Strg-Taste auf den Link in Zeile 6. Die MkDocs-Website sollte erscheinen:



Das ist das Ergebnis langwieriger Arbeit. Um den MkDocs-Server zu beenden, drücken Sie einfach Strg+C in dem Terminal, in dem er gestartet wurde.

9. Verwendung des DOCX-zu-HTML-Konverters

Der Befehl zur Konvertierung eines Word-DOCX-Dokuments ist dem zur Konvertierung eines LibreOffice-ODT-Dokuments sehr ähnlich. Wir werden den Stil des Dokumenttitels in [config.py] ändern:

```

1. STYLES = {
2.     "style_names": [
3.         "Titel"
4.     ]
5. }
```

- Zeile 3: Geben Sie „Titel“ ein. Dies ist der Stil des DOCX-Dokuments, das Sie konvertieren möchten. Wir werden dies in den Debugging-Zeilen des Konverters sehen.

Geben Sie im PyCharm-Terminal den folgenden Befehl ein:

```

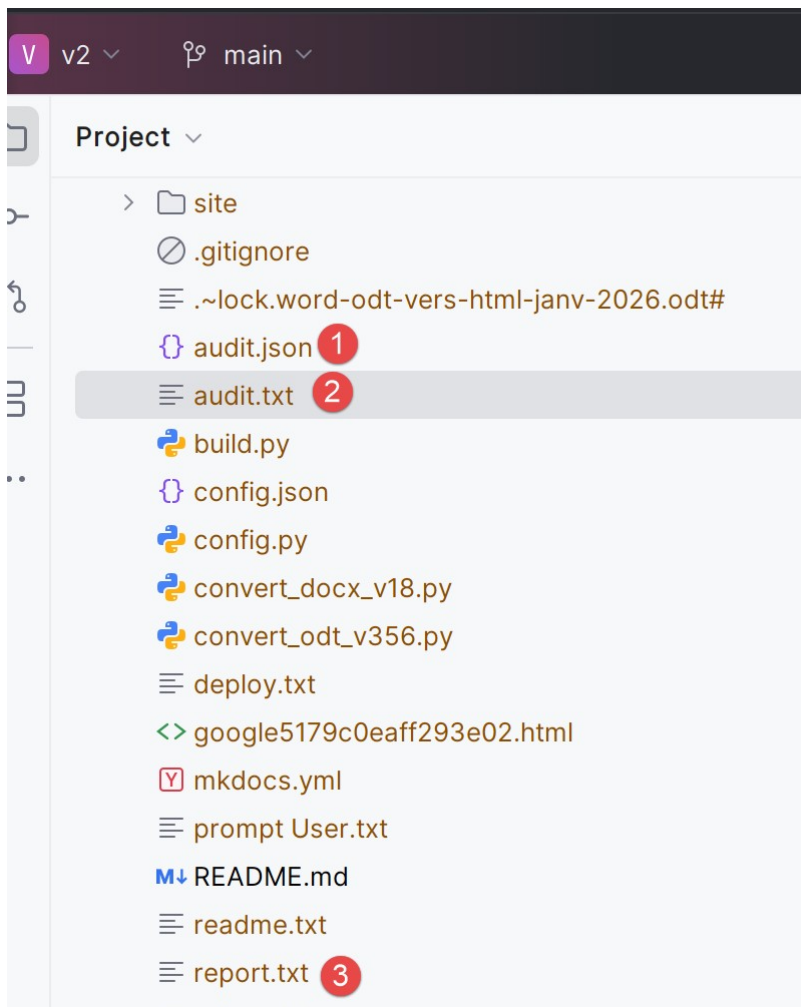
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_docx_v18.py .\word-
odt-vers-html-janv-2026.docx .\config.py
2. C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\convert_docx_v18.py:976: SyntaxWarning:
ungültige Escape-Sequenz '\h'
3. - REF Lesezeichen \h
4. --- DOCX to MkDocs Converter V16 ---
5. Kopiert: google5179c0eaff293e02.html
6. Kopiert: robots.txt
7. Kopiert: word-odt-zu-html-Jan-2026.pdf
8. Kopiert: word-odt-zu-html-Jan-2026.zip
9. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
10. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
11. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
```

```

12. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
13. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
14. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
15. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
16. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
17. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
18. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
19. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
20. [DEBUG PRE-H1] style=Titel heading=Keine rebase=0 numId=Keine ilvl=Keine list=Keine:0/Keine
txt='<span>Ein Word- oder ODT-Dokument in eine statische, Mk-kompatible HTML-Website
konvertieren...'
21. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
22. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
23. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
24. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Serge Tahé</b><span>, Januar 2026</span>...'
25. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
26. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
27. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
28. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
29. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
30. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
31. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Diese Website wurde mit dem Konverter &#91;Word oder ODT - &gt; HTML&#93; erstellt...'
32. [DEBUG PRE-H1] style=Titel1 heading=1 rebase=0 numId=1 ilvl=0 list=numPr:1/ordered
txt='<span>Einleitung</span>...'
33. Fertig. (audit.json, audit.txt, report.txt generiert)

```

- Zeile 1: Der Befehl lautet wie folgt: `[python .\convert_docx_v18.py .\word-odt-vers-html-janv-2026.docx .\config.py]` (Passen Sie die Versionsnummer (hier 18) an die von Ihnen heruntergeladene Version an.):
 - Der erste Parameter `[.\convert_docx_v18.py]` ist der DOCX-zu-HTML-Konverter
 - Der zweite Parameter `[.\word-odt-vers-html-janv-2026.docx]` ist der Name des zu konvertierenden DOCX-Dokuments;
 - der dritte Parameter `[.\config.py]` ist die Konfigurationsdatei;
- Zeile 33: Der Konverter meldet, dass drei Dateien erstellt wurden:



Die Datei [audit.txt] lautet wie folgt:

1. Version: V16
2. Absätze: 2029
3. Tabellen: 97
4. Bilder (Blips): 2
5. Erkannte Überschriften (Rohdaten): 53
6. Erkannte minimale Überschriftenebene (Rohdaten): 1
7. Angewandter Rebase-Offset: 0
- 8.
9. Top-Absatzstile:
10. - SourceCodenumrot: 1054
11. - StandardWW: 594
12. - Standard: 146
13. - Paragraphedeliste: 113
14. - SourceCodenumrotrsltats: 33
15. - neuerCode: 28
16. - Titel 2: 25
17. - Überschrift 1: 14
18. - Titel 3: 14
19. - StandardWWW: 6
20. - Klartext: 1
21. - Überschrift: 1
- 22.

```
23. Absätze auflisten:
24. - mit Nummerierung: 1329
25. - nach Stil-Fallback: 49
26. - nicht erkannt: 0
```

- Zeile 2: die Anzahl der Absätze im Word-Dokument;
- Zeile 3: die Anzahl der Tabellen;
- Zeilen 9–21: die im Dokument gefundenen Stile;
 - Zeilen 10, 14, 15: der Stil der Codeblöcke. Ein einziger Stil hätte wahrscheinlich ausgereicht;
 - Zeilen 11–12, 19: der Stil der Standardabsätze. Ein einziger Stil hätte wahrscheinlich ausgereicht;
 - Zeilen 16–18, 21: die Stile der Überschriften im Dokument. In Zeile 21 sieht man, dass nur ein Absatz den Stil „Überschrift“ hat. Es handelt sich um die Überschrift des Dokuments, die dem ersten „Überschrift 1“ vorangeht;

Diese Überprüfung des Word-Dokuments ist eine gute Möglichkeit, die Qualität des Dokuments zu beurteilen. Hier sehe ich, dass ich in meinem Word-Dokument zu viele verschiedene Formatvorlagen für ein und dasselbe verwendet habe.

Die Datei [audit.json] ist identisch mit der Datei [audit.txt], jedoch im JSON-Format:

```
1. {
2.   "version": "V16",
3.   "file": "word-odt-vers-html-janv-2026.docx",
4.   "counts": {
5.     "paragraphs": 2029,
6.     "tables": 97,
7.     "Bild-Blips": 2,
8.     "headings_raw": 53
9.   },
10.  "Listen": {
11.    "with_numpr": 1329,
12.    "by_style": 49,
13.    "unrecognized": 0
14.  },
15.  "heading": {
16.    "min_level_raw": 1,
17.    "rebase_offset": 0
18.  },
19.  "top_styles": [
20.    [
21.      "SourceCodenumrot",
22.      1054
23.    ],
24.    [
25.      "StandardWW",
26.      594
27.    ],
28.    [
29.      "Standard",
30.      146
31.    ],
32.    [
33.      „Absatzliste“,
34.      113
35.    ],
36.    [
37.      "Quellcode-Nummer-Ergebnisse",
38.      33
39.    ],
```

```

40.  [
41.     "codenouveau",
42.     28
43.  ],
44.  [
45.     "Titel2",
46.     25
47.  ],
48.  [
49.     "Titel1",
50.     14
51.  ],
52.  [
53.     "Titel3",
54.     14
55.  ],
56.  [
57.     "StandardWWW",
58.     6
59.  ],
60.  [
61.     "Klartext",
62.     1
63.  ],
64.  [
65.     "Titel",
66.     1
67.  ]
68. ]
69. }

```

Die Datei [report.txt] sieht wie folgt aus:

```

1. [SUMMARY] Über Fallback „nach Stil“ erkannte Listen (aggregiert)
2.   - Paragraphedeliste -> level=1 type=unordered: 49
3.
4. [SUMMARY] Ignorierte Word-Blöcke (aggregiert)
5.   - <w:sectPr>: 1

```

Ich habe das nicht verstanden...

Mit dem Parameter [--audit] kann man nur das Audit des Word-Dokuments anfordern, um dessen Qualität zu beurteilen:

```
python .\convert_docx_v18.py .\word-odt-vers-html-janv-2026.docx .\config.py --audit
```

In diesem Fall wird nur das Dokument geprüft. Die MkDocs-Website wird nicht generiert.

Wie zuvor gezeigt, können Sie die vom Konverter generierte MkDocs-Website anzeigen:

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2. INFO     - Dokumentation wird erstellt...
3. INFO     - Bereinigung des Website-Verzeichnisses
4. INFO     - Dokumentation in 0,59 Sekunden erstellt
5. INFO     - [06:05:48] Serving on http://127.0.0.1:8000/word-odt-vers-html-janv-2026/

```

Strg-Klick auf die URL in Zeile 5:

10. Erstellung der statischen HTML-Website

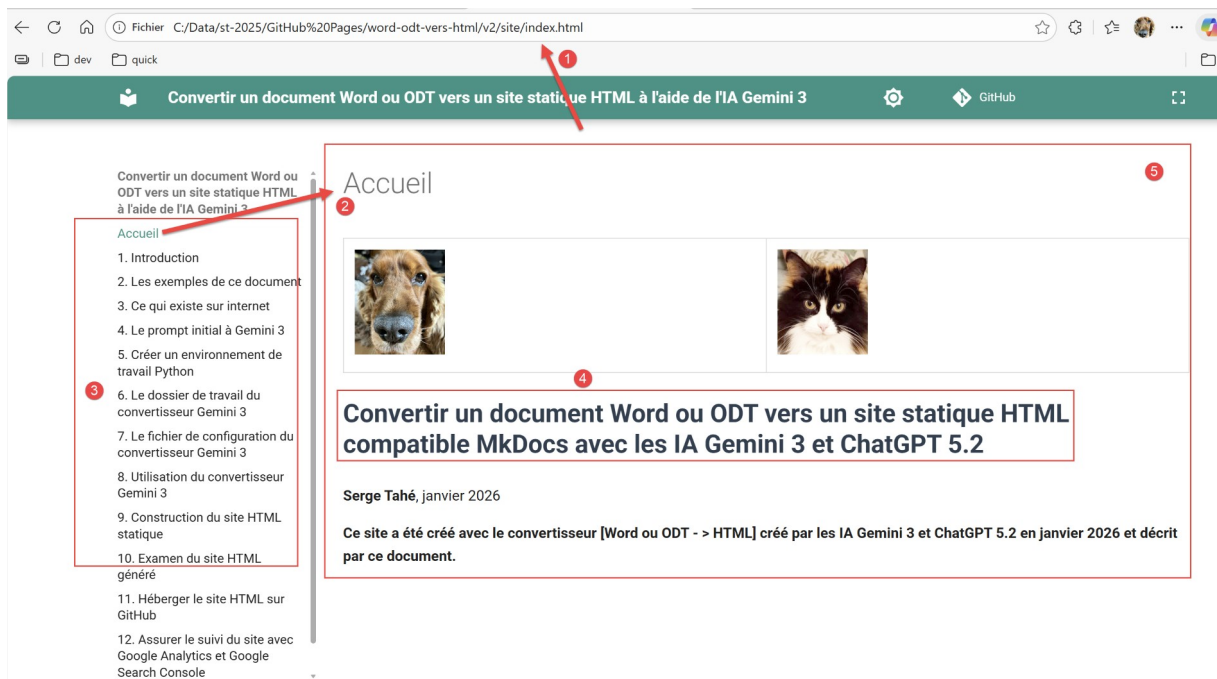
Ab jetzt gibt es keinen Unterschied mehr zwischen ODT- und DOCX-Dokumenten. Wir arbeiten an der MkDocs-Website, die von dem einen oder anderen Konverter erstellt wurde.

Erstellen wir nun die HTML-Website. Dies geschieht mit dem Skript [build]. Geben Sie im Terminal den folgenden Befehl ein:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\build.py
2. Start der Erstellung der MkDocs-Website...
3. Erfolgreicher Aufbau! Die Website befindet sich unter: C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site
4. Öffnen Sie C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site\index.html im Browser...
```

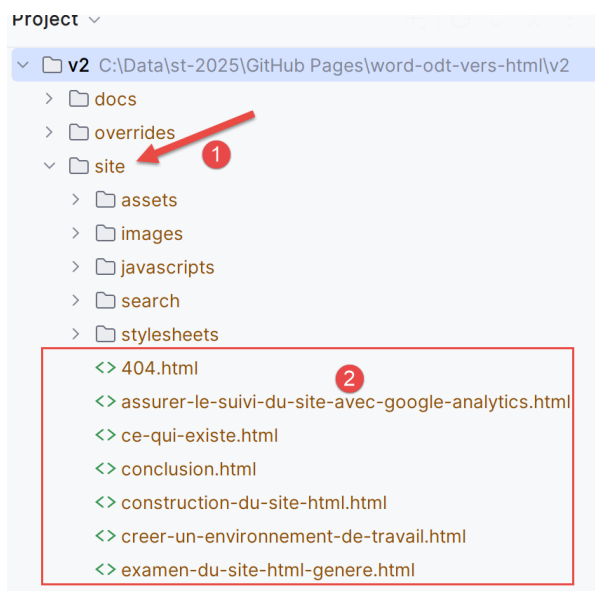
- Zeile 1: Der Befehl [python build] erstellt die HTML-Website aus der MkDocs-Website;
- Zeile 4: Die Website wird in einem Browser angezeigt;

Das Ergebnis ist dasselbe wie bei der MkDocs-Website:



- In [1] sieht man, dass eine HTML-Seite angezeigt wird;
- In [2] entspricht diese Seite der Startseite der Website;
- In [3] das Inhaltsverzeichnis des HTML-Dokuments;
- In [4] wurde der Titel des Dokuments gemäß seiner Konfiguration in [config.py] formatiert;
- In [5] ist die erste angezeigte Seite [Startseite]. Der Inhalt dieser Seite ist der Inhalt des ODT-/DOCX-Dokuments, der **dem ersten Kapitel der Ebene „Titel 1“ vorausgeht**;

Die Ausführung von [build] hat Ihren Arbeitsordner geändert:



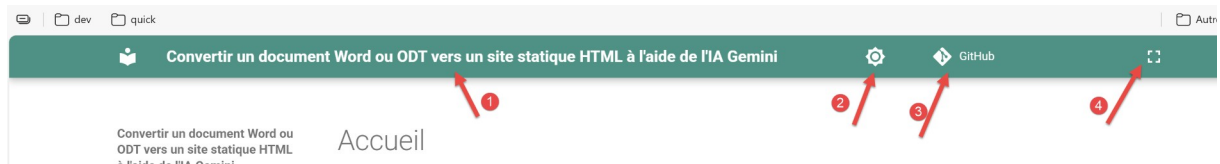
- Unter [1] wurde ein neuer Ordner [site] erstellt. Er enthält Ihre statische HTML-Website;
- In [2] die Seiten Ihrer Website;

11. Überprüfung der generierten HTML-Website

Wir werden nun die HTML-Ausgabe dieses ODT-/DOCX-Dokuments überprüfen. Wir haben bereits gesehen, dass der Konverter das Inhaltsverzeichnis beibehält.

11.1. Die obere Leiste der Website

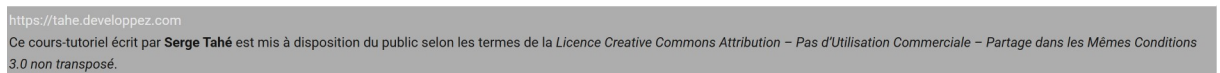
Sehen wir uns die obere Leiste der Website an:



- In [1] der in [config.py] definierte Name der Website;
- Unter [2] das Symbol, mit dem man in den Dunkel- oder Hellmodus wechseln kann;
- Unter [3] das Symbol, das einen Link zum GitHub-Repository darstellt, in das die HTML-Website exportiert wird. Ebenfalls in [config.py] definiert;
- Unter [4] das Symbol, mit dem man das Inhaltsverzeichnis ein- oder ausblenden kann;

11.2. Die Fußzeile der Website

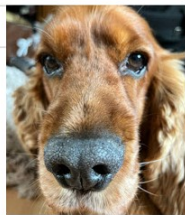
Sehen wir uns nun die Fußzeile an:



Dies ist die Fußzeile, die in der Datei [config.py] definiert ist.

11.3. Die Startseite

Die Titelseite des ODT-/DOCX-Dokuments sah wie folgt aus:



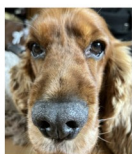
Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec les IA Gemini 3 et ChatGPT 5.2

Serge Tahé, janvier 2026

Ce site a été créé avec le convertisseur [Word ou ODT - > HTML] créé par les IA Gemini 3 et ChatGPT 5.2 en janvier 2026 et décrit par ce document.

Diese Titelseite des ODT-/DOCX-Dokuments wird zur Startseite der HTML-Website:

Accueil



Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec les IA Gemini 3 et ChatGPT 5.2 1

Serge Tahé, janvier 2026

Ce site a été créé avec le convertisseur [Word ou ODT - > HTML] créé par les IA Gemini 3 et ChatGPT 5.2 en janvier 2026 et décrit par ce document.

Der Konverter Gemini 3 / ChatGPT fügt auf der Startseite alles ein, was im ODT-/DOCX-Dokument als erster Titel der Ebene 1 im Stil „Titel 1“ vorliegt. Wenn Sie dort Bilder wie oben einfügen, werden diese angezeigt. Sie können sich also vorstellen, Ihrer Website ein Cover zu geben, wie bei einem echten Buch. In [1] ist dies der Haupttitel des Dokuments. Seine Darstellung wird durch die folgenden Zeilen in der Konfigurationsdatei [config.py] gesteuert:

```

1. # -----
2. # Erkennung des Dokumenttitels
3. # -----
4. "document_title": {
5.     # ODT-Stile, die als Haupttitel des Dokuments (globaler H1) zu betrachten sind
6.     "style_names": [
7.         "P1"
8.     ],
9.     # Auf diesen Titel im generierten Markdown angewendetes CSS
10.    "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
11.    color: #2c3e50;"
12. },

```

- Zeilen [6–8]: Die Liste der möglichen Formatvorlagen für den Titel Ihres Dokuments. Wenn ich mir dieses Dokument ansehe, lautet die LibreOffice-Formatvorlage meines Titels „Haupttitel“. Der Gemini-Konverter konnte diesen jedoch nicht finden. Er hat die gefundenen Stile protokolliert, und dabei wurde [P1] angezeigt. Das ist ein großes Problem bei LibreOffice: Die angezeigten Namen der Stile stimmen nicht mit den internen Namen überein, die von der Software verwendet werden. Sie dienen lediglich dazu, sich an die Sprache des Benutzers anzupassen;
- Zeile 10: Sobald der Haupttitel erkannt wurde, können Sie dessen Darstellung festlegen. Ich wollte eine Schriftgröße von 28 (font-size: 28px;) und Fettdruck (font-weight: bold);

Mit den Bildern und dem Stil des Titels können Sie eine ansprechende Startseite gestalten.

Der Haupttitel Ihres Dokuments hat möglicherweise keinen der in den Zeilen [6–8] definierten Stile. Um den Stil Ihres Haupttitels zu ermitteln, verwenden Sie die folgende Zeile in der Datei [config.py]

```

1. "debug": True

```

Mit dem Wert [true] wird der Stil der Absätze, die dem ersten Titel der Ebene 1 vorausgehen, also die Absätze der Startseite, bei der Ausführung des Gemini/ChatGPT-Konverters angezeigt. So habe ich für ein anderes Dokument als dieses die folgenden Protokolle erhalten:

```

1. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
2. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
3. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
4. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
5. [DEBUG PRE-H1] Style='P3' (Clean='p3') | Text='...'
6. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
7. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
8. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
9. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Text='...'
10. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Text='Einführung in die Sprache PHP7 anhand von
    Beispielen...'
11. >>> ERKANNTES DOKUMENTTITEL: Einführung in die Sprache PHP7 anhand eines Beispiels
12. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
13. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
14. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
15. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
16. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
17. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Text='...'
18. [DEBUG PRE-H1] Style='P5' (Clean='p5') | Text='Serge Tahé, Juli 2019...'
19. [DEBUG PRE-H1] Style='P5' (Clean='p5') | Text='...'
20. [DEBUG PRE-H1] Style='P6' (Clean='p6') | Text='...'
21. [DEBUG PRE-H1] Style='Heading 1' (Clean='heading 1') | Text='Einführung in die Sprache PHP
    7...'

```

- Zeile 10: Der Titel des Dokuments hat den Stil „P4“;

In der Datei [config.py] habe ich dann folgende Zeilen eingefügt:

```
1. "document_title": {
2.   "style_names": [
3.     "P4"
4.   ],
5.   "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2; color:
6.     #2c3e50;"
7. },
```

- Zeile 3, der Stil, den ich gesucht habe;

Aus diesem Grund zeigt der Debugger folgende Zeilen an:

```
1. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Text='Einführung in die Sprache PHP7 anhand von
   Beispielen...'
2. >>> DOKUMENTTITEL ERKANNT: Einführung in die Sprache PHP7 anhand eines Beispiels
```

Er ist auf den Stil „P4“ gestoßen und zeigt nun an, dass der Dokumenttitel gefunden wurde. Sobald dieser gefunden wurde, können Sie den Schlüssel [debug] in [config.py] auf [false] setzen:

```
1. "debug": False
```

Sehen wir uns nun die Konvertierung des Kapitels [Beispiele] an, das die Beispiele enthält, die der Gemini/ChatGPT-Konverter verarbeiten kann:

11.4. Aufzählungslisten

ODT-/DOCX-Dokument:

Listes à puces

- Élément 1 ;
- Élément 2 ;
- Élément 3 ;
 - Élément 3.1 ;
 - Élément 3.1.1
 - Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Oben ist der Text [Aufzählungslisten] hervorgehoben, da es sich um einen Verweis handelt, der mit einer Fußnote verknüpft ist.

HTML-Dokument:

Listes à puces

- Élément 1 ;
- Élément 2 :
- Élément 3 ;
 - Élément 3.1 ;
 - Élément 3.1.1
 - Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Man beachte, dass das ODT-/DOCX-Dokument verschiedene Aufzählungszeichen verwendet, während das HTML-Dokument nur eine Art von Aufzählungszeichen verwendet.

11.5. Nummerierte Listen

ODT-/DOCX-Dokument:

Listes numérotées

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 1. Élément 2.1.1.1
 2. Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

HTML-Dokument:

2.1.2. Listes numérotées

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 1. Élément 2.1.1.1
 2. Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

11.5.1. Gemischte Listen 1

ODT-/DOCX-Dokument

2.1.3. Listes mixtes 1

- Élément 1 ;
- Élément 2 ;
- Élément 3 ;
 - Élément 3.1 ;
 1. Élément 3.1.1
 2. Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

HTML-Dokument

2.1.3. Listes mixtes 1

- Élément 1 ;
- Élément 2 :
- Élément 3 ;
 - Élément 3.1 ;
 1. Élément 3.1.1
 2. Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Auch hier gibt es manchmal Unterschiede zwischen den verwendeten Aufzählungszeichen.

11.5.2. Gemischte Listen 2

ODT-/DOCX-Dokument

2.1.4. Listes mixtes 2

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 - Élément 2.1.1.1
 - Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

HTML-Dokument

2.1.4. Listes mixtes 2

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 - Élément 2.1.1.1
 - Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

11.6. Rich-Text-Code-Blöcke

Rich-Code-Blöcke werden im HTML identisch dargestellt (abgesehen von der Hintergrundfarbe). Hier sind drei Beispiele:

11.6.1. Beispiel 1

ODT-/DOCX-Dokument

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplica
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // récupération du contexte Spring
14.         ApplicationContext ctx = new ClassPathXmlAppli
15.         // on récupère les beans
16.         Personne p01 = ctx.getBean("personne_01", Perso
17.         Personne p02 = ctx.getBean("personne_02", Perso
18.         List<Personne> club = ctx.getBean("club", new
19.         ArrayList<Personne>().getClass());
20.         Appartement appart01 = ctx.getBean(Appartement
```

HTML-Darstellung

<https://tahe.developpez.com>

Java

```
1 package istia.st.spring.core;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.context.ApplicationContext;
7 import org.springframework.context.support.ClassPathXmlApplicat
8
9 public class Demo01 {
10
11     @SuppressWarnings({ "unchecked", "resource" })
12     public static void main(String[] args) {
13         // récupération du contexte Spring
14         ApplicationContext ctx = new ClassPathXmlApplicationCor
15         // on récupère les beans
16         Personne p01 = ctx.getBean("personne_01", Personne.class);
17         Personne p02 = ctx.getBean("personne_02", Personne.class);
18         List<Personne> club = ctx.getBean("club", new ArrayList<>());
19         Appartement appart01 = ctx.getBean(Appartement.class);
```

11.6.2. Beispiel 2

ODT-/DOCX-Dokument:

Python

```
1. # -----
2. def affiche(chaine):
3.     # affiche chaine
4.     print("chaine=%s" % chaine)
5.
6.
7. # -----
8. def affiche_type(variable):
9.     # affiche le type de variable
10.    print("type[%s]=%s" % (variable, type(variable)))
11.
12.
13. # -----
14. def f1(param):
15.     # ajoute 10 à param
16.     return param + 10
17.
18.
19. # -----
20. def f2():
21.     # rend un tuple de 3 valeurs
22.     return "un", 0, 100
23.
24.
25. # -----
26. ...
```

HTML-Ausgabe

Python

```
1 # -----
2 def affiche(chaine):
3     # affiche chaine
4     print("chaine=%s" % chaine)
5
6
7 # -----
8 def affiche_type(variable):
9     # affiche le type de variable
10    print("type[%s]=%s" % (variable, type(variable)))
11
12
13 # -----
14 def f1(param):
15     # ajoute 10 à param
16     return param + 10
17
18
19 # -----
20 def f2():
21     # rend un tuple de 3 valeurs
22     return "un", 0, 100
```

11.6.3. Beispiel 3

ODT-/DOCX-Dokument

ECMAScript

```
1. 'use strict';
2. // ceci est un commentaire
3. // constante
4. const nom = "dupont";
5. // un affichage écran
6. console.log("nom : ", nom);
7. // un tableau avec des éléments de type différents
8. const tableau = ["un", "deux", 3, 4];
9. // son nombre d'éléments
10. let n = tableau.length;
11. // une boucle
12. for (let i = 0; i < n; i++) {
13.     console.log("tableau[" + i + "] = ", tableau[i]);
14. }
15. // initialisation de 2 variables avec le contenu
16. let [chaine1, chaine2] = ["chaine1", "chaine2"];
17. // concaténation des 2 chaînes
18. const chaine3 = chaine1 + chaine2;
19. // affichage résultat
20. console.log([chaine1, chaine2, chaine3]);
21. ...
```

HTML-Ausgabe

ECMAScript

```
1  'use strict';
2  // ceci est un commentaire
3  // constante
4  const nom = "dupont";
5  // un affichage écran
6  console.log("nom : ", nom);
7  // un tableau avec des éléments de type différent
8  const tableau = ["un", "deux", 3, 4];
9  // son nombre d'éléments
10 let n = tableau.length;
11 // une boucle
12 for (let i = 0; i < n; i++) {
13   console.log("tableau[" + i + "] = ", tableau[i]);
14 }
15 // initialisation de 2 variables avec le contenu d
16 let [chaine1, chaine2] = ["chaine1", "chaine2"];
17 // concaténation des 2 chaînes
18 const chaine3 = chaine1 + chaine2;
19 // affichage résultat
20 console.log([chaine1, chaine2, chaine3]);
21 ...
```

11.7. Blöcke mit reinem Text (Plain Text)

Die im ODT-/DOCX-Dokument gefundenen Blöcke mit Rohcode werden von MkDocs entsprechend der im Code-Block gefundenen Sprache syntaktisch hervorgehoben. Um dem Konverter zu helfen, die richtige Sprache zu finden, wurden für jede Sprache „Schlüsselzeichenfolgen“ in die Datei [config.py] eingefügt. Der Konverter zählt die gefundenen „Schlüsselzeichenfolgen“. Er ordnet den Code-Block dann der Sprache zu, für die die meisten „Schlüsselzeichenfolgen“ gefunden wurden.

Sehen wir uns einige Beispiele an.

11.7.1. Beispiel 1

ODT-/DOCX-Dokument (Java)

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // récupération du contexte Spring
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // on récupère les beans
16.         Personne p01 = ctx.getBean("personne_01", Personne.class);
17.         Personne p02 = ctx.getBean("personne_02", Personne.class);
18.         List<Personne> club = ctx.getBean("club", new ArrayList<Personne>().getClass());
19.         Appartement appart01 = ctx.getBean(Appartement.class);
20.         ...
    }
```

HTML-Ausgabe

Java

```
1 package istia.st.spring.core;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.context.ApplicationContext;
7 import org.springframework.context.support.ClassPathXmlApplicationContext;
8
9 public class Demo01 {
10
11     @SuppressWarnings({ "unchecked", "resource" })
12     public static void main(String[] args) {
13         // récupération du contexte Spring
14         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15         // on récupère les beans
16         Personne p01 = ctx.getBean("personne_01", Personne.class);
17         Personne p02 = ctx.getBean("personne_02", Personne.class);
18         List<Personne> club = ctx.getBean("club", new ArrayList<Personne>().getClass());
19         Appartement appart01 = ctx.getBean(Appartement.class);
20         ...
    }
```

Im HTML-Dokument sieht man, dass der Java-Code syntaktisch eingefärbt wurde.

11.7.2. Beispiel 2

ODT-/DOCX-Dokument (XML)

```

1.    <?xml version="1.0" encoding="utf-8" ?>
2.    <configuration>
3.
4.        <configSections>
5.            <sectionGroup name="spring">
6.                <section name="context" type="Spring.Context.Support.ContextSupport" />
7.                <section name="objects" type="Spring.Context.Support.DefaultObjectFactory, Spring.Core" />
8.            </sectionGroup>
9.        </configSections>
10.
11.        <spring>
12.            <context>
13.                <resource uri="config://spring/objects" />
14.            </context>
15.            <objects xmlns="http://www.springframework.net">
16.                <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
17.                    <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
18.                    <constructor-arg index="1"
19. value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
20.                    <constructor-arg index="2" value="select limite, coeffr, c
21.                </object>
22.                <object name="metier" type="Metier.ImpotMetier, ImpotsV7-met
23.                    <constructor-arg index="0" ref="dao"/>
24.                </object>
25.            </objects>
26.        </spring>
27.    </configuration>

```

HTML-Ausgabe

XML

```

1    <?xml version="1.0" encoding="utf-8" ?>
2    <configuration>
3
4    <configSections>
5        <sectionGroup name="spring">
6            <section name="context" type="Spring.Context.Support
7            <section name="objects" type="Spring.Context.Support
8        </sectionGroup>
9    </configSections>
10
11   <spring>
12       <context>
13           <resource uri="config://spring/objects" />
14       </context>
15       <objects xmlns="http://www.springframework.net">
16           <object name="dao" type="Dao.DataBaseImpot, ImpotsV
17               <constructor-arg index="0" value="MySQL.Data.MySql
18               <constructor-arg index="1" value="Server=localhost
19               <constructor-arg index="2" value="select limite, c
20           </object>
21           <object name="metier" type="Metier.ImpotMetier, Impo
22               <constructor-arg index="0" ref="dao"/>
23           </object>
24       </objects>
25   </spring>
26 </configuration>

```

11.7.3. Beispiel 3

ODT-/DOCX-Dokument (HTML)

HTML

```
1. <!DOCTYPE HTML>
2. <HTML>
3.   <head>
4.     <title>Laragon</title>
5.
6.     <link href="https://fonts.googleapis.com/css?
   type="text/css">
7.
8.     <style>
9.       HTML, body {
10.        height: 100%;
11.      }
12.
13.      body {
14.        margin: 0;
15.        padding: 0;
16.        width: 100%;
17.        display: table;
18.        font-weight: 100;
19.        font-family: 'Karla';
20.      }
21.
22.      .container {
23.        text-align: center;
24.        display: table-cell;
25.        vertical-align: middle;
26.      }
27.
28.      .content {
29.        text-align: center;
30.        display: inline-block;
31.      }
```

HTML-Ausgabe

HTML

```
1  <!DOCTYPE HTML>
2  <HTML>
3      <head>
4          <title>Laragon</title>
5
6          <link href="<a href="view-source:https://font
7
8          <style>
9              HTML, body {
10                 height: 100%;
11             }
12
13             body {
14                 margin: 0;
15                 padding: 0;
16                 width: 100%;
17                 display: table;
18                 font-weight: 100;
19                 font-family: 'Karla';
20             }
21
22             .container {
23                 text-align: center;
24                 display: table-cell;
25                 vertical-align: middle;
26             }
27
```

11.8. Weitere Code-Blöcke

ODT-/DOCX-Dokument

Ein Ausführungsergebnis mit einer ersten Zeile, die nicht bei 1 beginnt:

Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7.  C:\Data\st-2020\dev\python\cours-2020\python3-flask-2020\ve
   C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/ba
8.  nom=dupont
9.  liste[0]=un
10. liste[1]=deux
11. liste[2]=3
12. liste[3]=4
13. [chaine1,chaine2,chaine1chaine2]
14. chaine=chaine1
15. type[4]=<class 'int'>
16. type[chaine1]=<class 'str'>
17. type[['un', 'deux', 3, 4]]=<class 'list'>
18. type[a changé]=<class 'str'>
```

HTML-Ausgabe

Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7 C:\Data\st-2020\dev\python\cours-2020\python3-flask-2
8 nom=dupont
9 liste[0]=un
10 liste[1]=deux
11 liste[2]=3
12 liste[3]=4
13 [chaine1,chaine2,chaine1chaine2]
14 chaine=chaine1
15 type[4]=<class 'int'>
16 type[chaine1]=<class 'str'>
17 type[['un', 'deux', 3, 4]]=<class 'list'>
18 type[a changé]=<class 'str'>
```

Ein nicht nummerierter Code-Block in ODT bleibt im HTML erhalten:

ODT-/DOCX-Dokument

Le convertisseur sait gérer les blocs de code non numérotés.

```
liste[0]=un
liste[1]=0
liste[2]=100
liste[0]=8
liste[1]=5
somme=13
```

HTML-Ausgabe

Le convertisseur sait gérer les blocs de code non numérotés.

```
liste[0]=un
liste[1]=0
liste[2]=100
liste[0]=8
liste[1]=5
somme=13
```

11.9. Die Links

ODT-/DOCX-Dokument

2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [\[Générer un script Python avec des outils d'IA\]](#).

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Un renvoi vers un chapitre : [2.1](#).

Un renvoi vers un repère de texte qui précède : [Gemini 3](#)

Un renvoi vers un repère de texte qui suit : [GitHub](#)

HTML-Ausgabe

2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [\[Générer un script Python avec des outils d'IA\]](#).

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Un renvoi vers un chapitre : [2.1](#).

Un renvoi vers un repère de texte qui précède : [Gemini 3](#)

Un renvoi vers un repère de texte qui suit : [GitHub](#)

11.10. Textformatierung

ODT-/DOCX-Dokument

Un texte avec des mots en **gras**, en *italiques*, soulignés ou **surlignés** ou **surlignés** ou **surlignés**.

┌
C'est vrai également pour les liens : [\[Générer un script Python avec des outils d'IA\]](#).

Le **convertisseur** **gère** également la **couleur** des **caractères**.

Il gère également les bordures supérieure et inférieure des paragraphes.

2.5. Un **titre** peut être également enrichi.

HTML-Ausgabe

2.4. L'enrichissement de texte

Le convertisseur sait gérer le gras, l'italique, le souligné et le surlignage. Il respecte la couleur du surlignage.

Un texte avec des mots en **gras**, en *italiques*, soulignés ou surlignés ou surlignés ou surlignés.

C'est vrai également pour les liens : [Générer un script Python avec des outils d'IA].

Le **convertisseur** gère également la **couleur** des **caractères**.

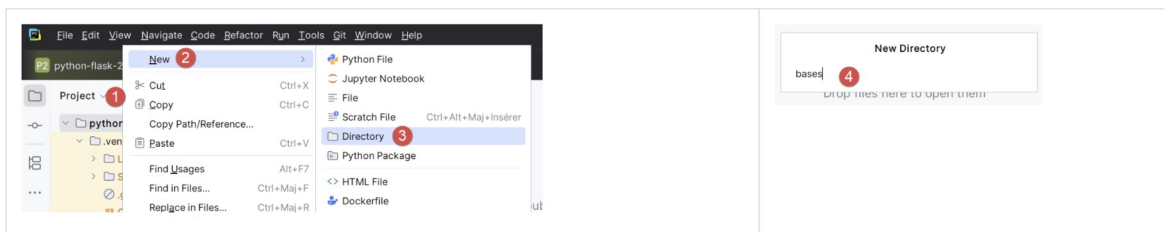
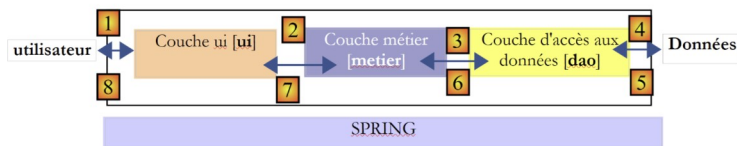
Il gère également les bordures supérieure et inférieure des paragraphes.

2.5. Un titre peut être également enrichi.

11.11. Bilder

2.5. Les images

Le convertisseur Gemini sait gérer les images et les tableaux d'images :



Es ist zu beachten, dass der Gemini/ChatGPT-Konverter die im ODT-/DOCX-Dokument vorgenommenen Bildgrößenanpassungen beibehält.

11.12. Geschützte Zeichen

ODT-/DOCX-Dokument

L'astérisque * a une signification Markdown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à $0.15 * R - 2072.5 * nbParts$.

HTML-Ausgabe

L'astérisque * a une signification Markdown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à $0.15 \cdot R - 2072.5 \cdot \text{nbParts}$.

ODT-/DOCX-Dokument (Markdown)

```
5. ---
6.
7. ## 📄 Description
8.
9. On se propose dans ce projet de mettre à disposition du
documents Word ou ODT vers un site statique HTML.
10.
11. Lorsque le document ODT convient, le convertisseur prod
l'aspect professionnel des sites produits par Pandoc.
12.
13. ## 📄 Contexte de création
14.
15. Ce convertisseur a été entièrement construit par l'IA *
est le résultat d'itérations successives pour gérer finement la
(OpenDocument Text).
16.
17. ## ✨ Fonctionnalités
18.
19. Le script `convert.py` effectue les actions suivantes :
20.
21. * **Conversion ODT vers Markdown** : Analyse le fichier
structure.
22. * **Gestion des Titres** : Génère automatiquement la Ta
latérale.
23. * **Blocs de Code** : Détection automatique des langage
précise de la numérotation des lignes** (attributs `start-value
24. * **Listes** : Support des listes à puces et numérotées
25. * **Mise en forme** : Support du *gras*, *italique*, *s
respect des couleurs d'origine).
```

HTML-Ausgabe

Un autre exemple est lorsque vous voulez insérer un bloc de code Markdown dans votre document

```


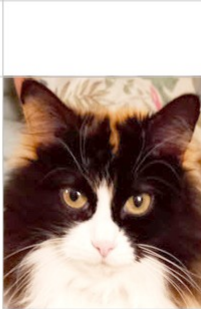
1 # Convertisseur Word/ODT vers Site HTML (MkDocs)
2
3 🔗 **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-v
4
5 ---
6
7 ## 📖 Description
8
9 On se propose dans ce projet de mettre à disposition du lecteur un convertisseur
10
11 Lorsque le document ODT convient, le convertisseur produit un site HTML via **
12
13 ## 🏠 Contexte de création
14
15 Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abo
16
17 ## ✨ Fonctionnalités
18
19 Le script `convert.py` effectue les actions suivantes :
20

```

- Der Markdown-Code wurde beibehalten;

11.13. Tabellen

ODT-/DOCX-Dokument

1	2		<pre> 1. package istia.st.spring.core; 2. 3. import java.util.ArrayList; 4. import java.util.List; 5. 6. import org.springframework.context.ApplicationContext; 7. import org.springframework.context.support.ClassPathXmlApplication Context; 8. 9. public class Demo01 { 10. 11. </pre>
3	4		<pre> 1. # ----- 2. def affiche(chaine): 3. # affiche chaîne 4. print("chaîne=%s" % chaine) 5. 6. 7. # ----- 8. def affiche_type(variable): 9. # affiche le type de variable 10. print("type[%s]=%s" % (variable, type(variable))) 11. 12. </pre>

HTML-Ausgabe

Un tableau peut contenir différents contenus :

1	2		<pre>1 package istia.st.spring.core; 2 3 import java.util.ArrayList; 4 import java.util.List; 5 6 import org.springframework.context.ApplicationContext; 7 import org.springframework.context.support.ClassPathXmlApplicationConte 8 9 public class Demo01 { 10 11</pre>
3	4		<pre>1 # ----- 2 def affiche(chaine): 3 # affiche chaine 4 print("chaine=%s" % chaine) 5</pre>

11.14. Fußnoten

Le convertisseur Gemini² gère les notes de bas de page. Voici une autre note³ de bas de page.

-
1. MkDocs
 2. Google Gemini
 3. La note de bas de page

11.15. Bekannte Fehler

Es wurden einige Fehler festgestellt, die jedoch durch Bearbeiten der ODT-/DOCX-Datei behoben werden können:

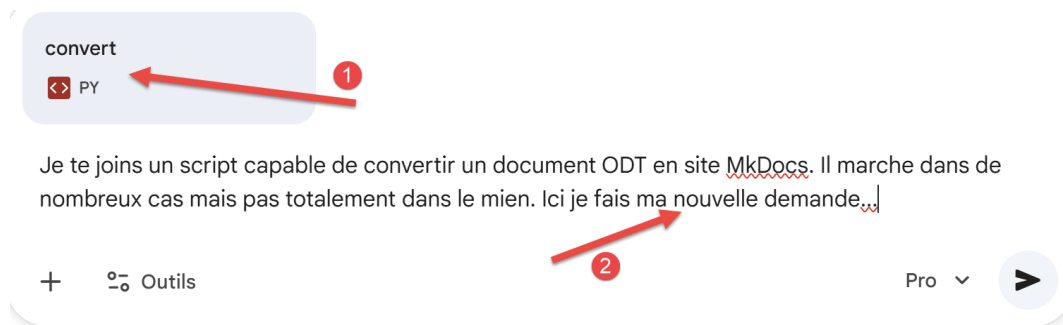
- **Auf Code-Blöcke muss eine Leerzeile folgen**, da der Code-Block sonst möglicherweise nicht korrekt dargestellt wird. Der festgestellte Fall ist ein Code, auf den unmittelbar ein Titel folgt, ohne dass dieser durch eine Leerzeile davon getrennt ist;
- **Aufzählungslisten dürfen keinen unteren Rand haben. Um diesen zu erhalten, muss hinter dem letzten Element der Liste eine Leerzeile eingefügt werden;**

- **Aufzählungslisten müssen hierarchisch gegliedert sein.** So muss eine Liste der Ebene 2 immer in einer Liste der Ebene 1 enthalten sein, andernfalls wird die Liste der Ebene 2 als Code dargestellt;

Im Laufe der verschiedenen Versionen des Konverters werden einige dieser Anomalien verschwinden. Die drei oben genannten lassen sich durch Korrekturen am Quelldokument vermeiden.

11.16. Sonstige Fälle

Wenn Ihr Dokument andere Besonderheiten als die zuvor genannten aufweist, ist es sehr wahrscheinlich, dass diese vom Gemini/ChatGPT-Konverter nicht berücksichtigt werden. Was können Sie dann tun? Sie können Ihre neue Anfrage an eine der KI-Systeme richten, indem Sie ihr den aktuellen Konverter zur Verfügung stellen:



- In [1] füge ich den Konverter dieses Dokuments bei;
- In [2] stelle ich meine neue Anfrage;

Sie werden wahrscheinlich zahlreiche Iterationen durchlaufen. Wenn eine Version stabil ist, notieren Sie deren Nummer, um sie der KI im Falle eines Regressionsfehlers wieder mitteilen zu können. Es ist ebenfalls ratsam, von jeder stabilen Version eine Kopie anzulegen. Ein großer Nachteil beider KI-Systeme ist, dass sie relativ leicht Regressionen aufweisen. Es reicht schon, eine neue Funktion anzufordern, damit die KI den zuvor funktionierenden Code zerstört. Daher ist es wichtig, die Versionsnummern der stabilen Versionen zu notieren, um darauf zurückgreifen zu können. Im Januar 2026 schien mir ChatGPT 5.2 weniger anfällig für Regressionen zu sein als Gemini 3.

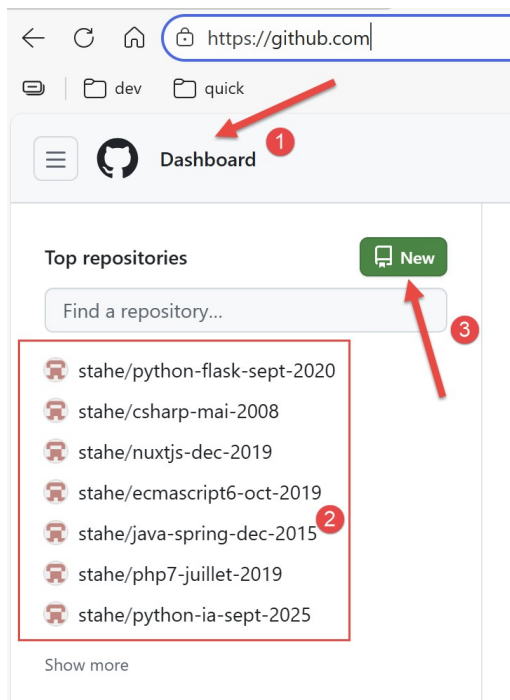
12. Die HTML-Website auf GitHub hosten

Es war Gemini selbst, der mir vorschlug, die von seinen beiden Skripten erstellte HTML-Seite auf GitHub zu hosten:⁴. Ich wusste gar nicht, dass das möglich ist. GitHub ist eine Plattform, auf der Entwicklungsprojekte gehostet werden. Es erscheint daher naheliegend, dort Programmierkurse zu veröffentlichen.

Zunächst benötigen Sie ein [GitHub](#)-Konto. Erstellen Sie bei Bedarf eines.

Melden Sie sich bei Ihrem GitHub-Konto an:

⁴Fußnote für GitHub



- Unter [2] sehen Sie Ihre bestehenden Repositorys, falls Sie welche haben;
- Erstellen Sie unter [3] ein neues GitHub-Repository;

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name * 1

✔ word-odt-vers-html-janv-2026 is available.

Great repository names are short and memorable. How about [sturdy-telegram](#)?

Description

2

84 / 350 characters

2 Configuration

Choose visibility * Public

Choose who can see and commit to this repository

Add README Off Off

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore No .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

Add license No license

Licenses explain how others can use your code. [About licenses](#)

3

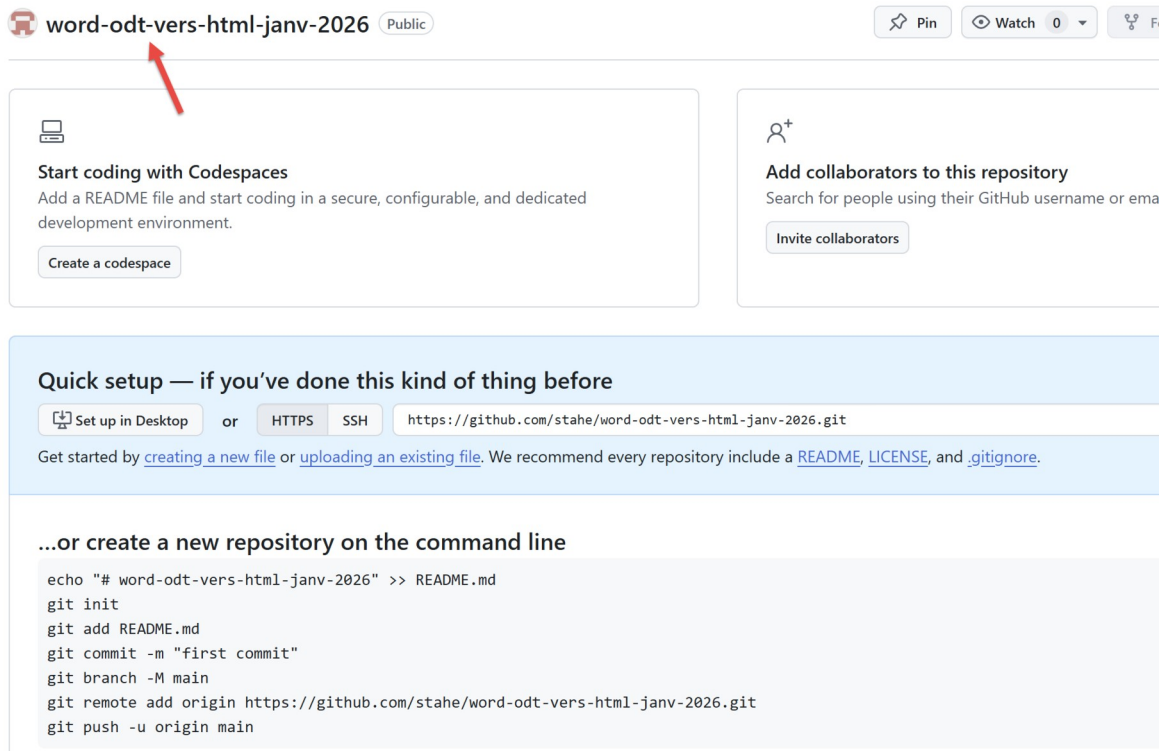
- Verwenden Sie unter [1] den Namen, den Sie in [config.py] angegeben haben:

```
1. "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
```

- In [2] können Sie ebenfalls dasselbe wie in [config.py] eingeben:

```
1. "site_description": "Konvertieren eines Word- oder ODT-Dokuments in eine statische HTML-Website mithilfe der KI-Modelle Gemini 3 und ChatGPT 5.2",
```

- Bestätigen Sie in [3] die Erstellung Ihres GitHub-Repositorys;

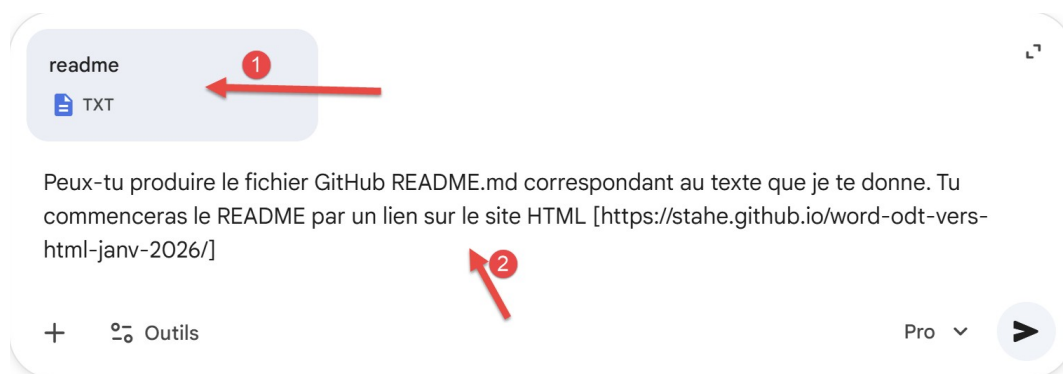


Es wird empfohlen, für jedes GitHub-Repository eine [README.md]-Datei zu erstellen, die anstelle des obigen Bildschirms angezeigt wird. Diese muss in Markdown geschrieben sein, was nicht ganz einfach ist. Wir erstellen daher eine Textdatei mit dem Inhalt, den wir in der README-Datei sehen möchten. Anschließend bitten wir die KI, uns die entsprechende README.md-Datei zu liefern.

Die Textdatei [readme.txt] sieht wie folgt aus:

1. In diesem Artikel möchten wir dem Leser einen Python-Konverter zur Verfügung stellen, der Word- oder ODT-Dokumente in eine statische HTML-Website umwandelt.
- 2.
3. Dieser Konverter wurde von der KI Gemini 3 erstellt.
- 4.
5. Wenn das ODT-Dokument geeignet ist, erzeugt der Konverter über MkDocs eine HTML-Website, die das professionelle Erscheinungsbild der von Pandoc erstellten Websites aufweist.
- 6.
7. Ich habe Gemini 3 mit einem Pro-Abonnement verwendet.

Die Eingabeaufforderung für Gemini lautet dann wie folgt:



- In [1] die angehängte Datei [readme.txt];
- In [2] die Eingabeaufforderung;

Kopieren Sie die Antwort, die Gemini Ihnen gibt, in die Datei [README.md] in Ihrem Arbeitsordner. Hier ist die Antwort, die Gemini mir gegeben hat:

```

1. # Konverter von Word/ODT zu HTML-Website (MkDocs)
2.
3. 🔗 **[Zur generierten Demo-Website](https://stahe.github.io/word-odt-vers-html-janv-2026/)**
4.
5. ---
6.
7. ## 📄 Beschreibung
8.
9. In diesem Projekt soll dem Leser ein Python-Konverter zur Verfügung gestellt werden, der Word-
oder ODT-Dokumente in eine statische HTML-Website umwandelt.
10.
11. Wenn das ODT-Dokument geeignet ist, erzeugt der Konverter über **MkDocs** eine HTML-Website,
die das professionelle Erscheinungsbild der von Pandoc erstellten Websites aufweist.
12.
13. ## 🏠 Entstehungshintergrund
14.
15. Dieser Konverter wurde vollständig von der KI **Gemini 3** (mit einem Pro-Abonnement)
erstellt. Er ist das Ergebnis aufeinanderfolgender Iterationen zur Feinabstimmung der Struktur
von ODT-Dokumenten (OpenDocument Text).
16.
17. ## ✨ Funktionen
18.
19. Das Skript `convert.py` führt folgende Aktionen aus:
20.
21. * **Konvertierung von ODT nach Markdown**: Analysiert die `.odt`-Datei (XML), um die Struktur
zu extrahieren.
22. * **Verwaltung von Überschriften**: Erzeugt automatisch das Inhaltsverzeichnis (TOC) und die
seitliche Navigation.
23. * **Code-Blöcke**: Automatische Spracherkennung, Syntaxhervorhebung und **präzise Verwaltung
der Zeilennummerierung** (Attribute `start-value`).
24. * **Listen**: Unterstützung von Aufzählungs- und nummerierten Listen mit korrekter Einrückung.
25. * **Formatierung**: Unterstützung von *Fettdruck*, *Kursivschrift*, *Unterstreichug* und
*Hervorhebung* (unter Beibehaltung der Originalfarben).
26. * **Bilder**: Automatische Extraktion und Einbindung der im Dokument enthaltenen Bilder.
27. * **Konfiguration**: Anpassung über eine `config.py`-Datei (Fußzeile, Google Analytics usw.).
28.
29. ## 🚀 Installation
30.
31. ### Voraussetzungen
32.
33. * Python 3.x
34. * Die folgenden Bibliotheken:
35.
36. ```bash
37. pip install odfpy unicode mkdocs mkdocs-material
38. ```
39.
40.
41. ### Projektstruktur
42.
43. Stellen Sie sicher, dass Sie über die folgenden Dateien verfügen:
44.
45. * `convert.py`: Das Konvertierungsskript.
46. * `config.py`: Ihre Konfigurationsdatei.
47. * `Ihr-Dokument.odt`: Das Quelldokument.
48.
49. ## 📖 Verwendung
50.
51. 1. **Konvertierung**
52. Starten Sie das Skript, indem Sie die ODT-Quelldatei und die Konfigurationsdatei angeben:
53. ```bash

```

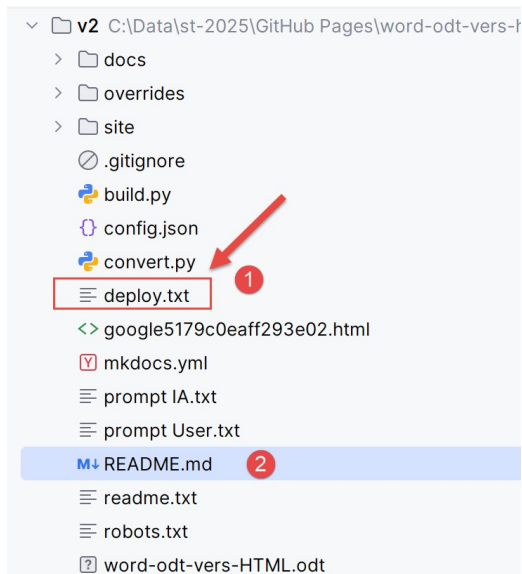
```

54. python convert.py Ihr-Dokument.odt config.py
55.
56. ```
57.
58.
59. *Dadurch wird ein Ordner `docs/` erstellt, der die Markdown-Dateien und eine Datei
   `mkdocs.yml` enthält.*
60. 2. **Vorschau**
61. So zeigen Sie die Website lokal an:
62. ```bash
63. mkdocs serve
64.
65. ```
66.
67.
68. 3. **Generierung**
69. So erstellen Sie die statische Website (Ordner `site/`):
70. ```bash
71. mkdocs build
72.
73. ```
74.
75.
76.
77. ## ⚙️ Konfiguration (`config.py`)
78.
79. Mit der Datei `config.py` lässt sich das Erscheinungsbild der Website steuern:
80.
81. * **mkdocs**: Allgemeine Einstellungen der Website (Titel, Beschreibung, Material-Theme).
82. * **Fußzeile**: Vollständiger HTML-Code zur Anpassung der Fußzeile.
83. * **code**: Regeln zur Spracherkennung für die Syntaxhervorhebung.
84. * **extra**: Konfiguration von Google Analytics (GA4).
85.
86. ## 📄 Lizenz
87.
88. Dieser von **Serge Tahé** verfasste Tutorial-Kurs wird der Öffentlichkeit gemäß den
   Bedingungen der folgenden Lizenz zur Verfügung gestellt:
89. *Creative Commons-Lizenz „Namensnennung – Keine kommerzielle Nutzung – Weitergabe unter
   gleichen Bedingungen 3.0 nicht portiert“.*

```

Er hat mir eine sehr ausführliche README-Datei zur Verfügung gestellt. Das liegt daran, dass Gemini dieses Projekt, an dem wir seit Wochen arbeiten, sehr gut kennt. Ihr werdet wahrscheinlich eine weniger detaillierte [README.md]-Datei erhalten.

Kommen wir nun zurück zu unserem Arbeitsordner:



- In [2] die README-Datei, die Sie gerade bearbeitet haben;
- In [1] erklärt die Datei [deploy.txt], wie Sie Ihre HTML-Website in Ihr GitHub-Repository exportieren;

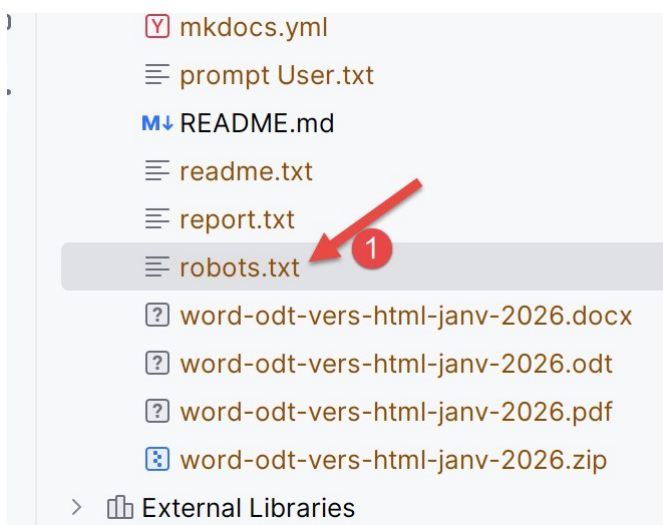
Der Inhalt der Datei [deploy.txt] lautet wie folgt:

```
1. git init
2. git branch -M main
3. # Überprüfen Sie vorher die .gitignore-Datei!
4. git add .
5. # Erster Commit
6. git commit -m "Erster Commit: MkDocs-Quellcode"
7. git remote add origin https://github.com/stahe/word-odt-vers-html-janv-2026.git
8. git push -u origin main
9. python -m mkdocs gh-deploy
```

Dies ist die Befehlsfolge, die Ihre HTML-Website in Ihr GitHub-Repository exportiert. Sie müssen Zeile 7 mit der URL Ihres eigenen GitHub-Repositorys ändern, die in der Datei [config.py] zu finden ist:

```
1. "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
```

Sie müssen außerdem eine weitere URL in der Datei [robots.txt] überprüfen:



Der Inhalt der Datei [robots.txt] lautet wie folgt:

```
1. User-agent: *
2. Allow: /
3. Sitemap: https://stahe.github.io/word-odt-vers-html-janv-2026/sitemap.xml
```

Geben Sie in Zeile 3 die URL Ihrer Website ein, dieselbe wie in der Datei [config.py]:

```
1. # Veröffentlichungs-URL der Website (z. B. GitHub Pages)
2. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

Die Datei [robots.txt] wird beim lokalen Erstellen der Website nicht verwendet, aber sie wird benötigt, wenn die Website auf GitHub gehostet wird.

Die Befehlsfolge in [deploy.txt] verwendet eine Software namens Git. Sie müssen diese installieren [[Git – Installation für Windows](#)].

Überprüfen Sie anschließend die Datei [.gitignore] in Ihrem Arbeitsordner. Sie teilt Git mit, welche Dateien ignoriert werden sollen. Meine Datei [.gitignore] sieht wie folgt aus:

```
1. # 1. Wir ignorieren ALLE Dateien und Ordner des Projekts
2. *
3.
4. # 3. Wir machen eine Ausnahme (!) für die einzige Datei, die Sie behalten möchten
5. !README.md
```

Es ist ganz einfach. Wir ignorieren alle Dateien (Zeile 2) außer der Datei [README.md] (Zeile 5). GitHub ist dafür gedacht, Entwicklungsprojekte zu hosten. In der Regel wird das gesamte Entwicklungsprojekt auf GitHub exportiert. Wir möchten jedoch lediglich eine HTML-Website exportieren, kein Entwicklungsprojekt. Die einzige Datei, die wir in unser GitHub-Repository exportieren wollen, ist die Datei [README.md], die den Besuchern erklärt, was unsere HTML-Website enthält.

Geben Sie nun in Ihrem Terminal die folgenden Befehlssequenz in der in [deploy.txt] angegebenen Reihenfolge bis zum Befehl 8 ein. Führen Sie den Befehl 9 vorerst nicht aus.

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git init
2. Leeres Git-Repository in C:/Data/st-2025/GitHub Pages/word-odt-vers-html/v2/.git/
   initialisiert
3. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git branch -M main
4. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git add .
5. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git commit -m "Initial commit: Source
   MkDocs"
6. [main (root-commit) 7cba5b1] Erster Commit: MkDocs-Quelle
7. 1 Datei geändert, 89 Einfügungen(+)
8. create mode 100644 README.md
9. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git remote add origin
   https://github.com/stahe/word-odt-vers-html-janv-2026.git
10. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git push -u origin main
11. Objekte werden aufgelistet: 3, fertig.
12. Objekte werden gezählt: 100 % (3/3), fertig.
13. Delta-Komprimierung mit bis zu 8 Threads
14. Objekte werden komprimiert: 100 % (2/2), fertig.
15. Objekte schreiben: 100 % (3/3), 1,70 KiB | 1,70 MiB/s, fertig.
16. Insgesamt 3 (Delta 0), wiederverwendet 0 (Delta 0), Pack-wiederverwendet 0 (von 0)
17. Nach https://github.com/stahe/word-odt-vers-html-janv-2026.git
18. * [neuer Zweig] main -> main
19. Zweig „main“ eingerichtet, um „origin/main“ zu verfolgen.
```

Klicken Sie bei gedrückter Strg-Taste auf die URL in Zeile 17. Dadurch gelangen Sie zu Ihrem GitHub-Repository:

https://github.com/stahe/word-odt-vers-html-janv-2026

stahe / word-odt-vers-html-janv-2026

word-odt-vers-html-janv-2026 Public

main 1 Branch 0 Tags

Go to file Add file Code

stahe Initial commit: Source MkDocs 7cba5b1 · 4 minutes ago 1 Commit

README.md Initial commit: Source MkDocs 4 minutes ago

README

Convertisseur Word/ODT vers Site HTML (MkDocs)

[Voir le site de démonstration généré](#)

Description

On se propose dans ce projet de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML.

Lorsque le document ODT convient, le convertisseur produit un site HTML via **MkDocs** qui a l'aspect professionnel des sites produits par Pandoc.

- In [1] die URL Ihres GitHub-Repositorys;
- In [2] die neue README.md;

Gehen wir nun zu Befehl 9 in der Datei [deploy.txt]. Dieser exportiert die HTML-Website auf GitHub:

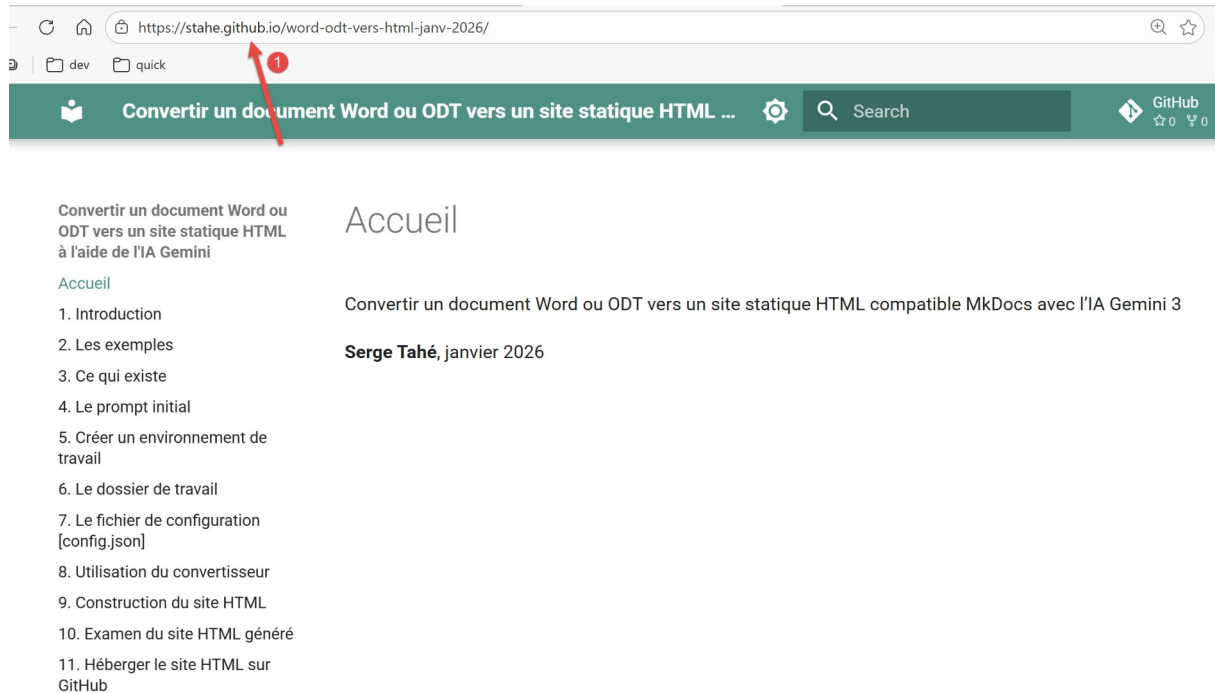
```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs gh-deploy
2. INFO - Bereinigung des Website-Verzeichnisses
3. INFO - Dokumentation wird in das Verzeichnis erstellt: C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site
4. INFO - Die Datei „les-exemples.md“ enthält einen Link „#_Les_exemples“, aber auf dieser Seite gibt es keinen solchen Anker.
5. INFO - Dokumentation in 1,79 Sekunden erstellt
6. WARNUNG - Versionsprüfung übersprungen: In der vorherigen Bereitstellung wurde keine Version angegeben.
7. INFO - Kopieren von 'C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site' in den Zweig 'gh-pages' und Hochladen auf GitHub.
8. Objekte werden aufgelistet: 91, fertig.
9. Objekte werden gezählt: 100 % (91/91), fertig.
10. Delta-Komprimierung mit bis zu 8 Threads
11. Objekte werden komprimiert: 100 % (85/85), fertig.
12. Objekte werden geschrieben: 100 % (91/91), 1,64 MiB | 2,01 MiB/s, fertig.
13. Insgesamt 91 (Delta 9), wiederverwendet 0 (Delta 0), Pack-wiederverwendet 0 (von 0)
14. remote: Deltas auflösen: 100 % (9/9), fertig.
15. remote:
16. remote: Erstelle einen Pull-Request für „gh-pages“ auf GitHub unter:

```

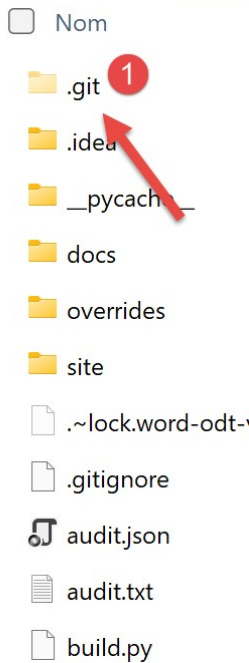
```
17. remote:      https://github.com/stahe/word-odt-vers-html-janv-2026/pull/new/gh-pages
18. remote:
19. An https://github.com/stahe/word-odt-vers-html-janv-2026.git
20. * [neuer Zweig]      gh-pages -> gh-pages
21. INFO    - Ihre Dokumentation sollte in Kürze unter folgender Adresse verfügbar sein:
      https://stahe.github.io/word-odt-vers-html-janv-2026/
```

Klicken Sie bei gedrückter Strg-Taste auf die URL in Zeile 21. Dadurch gelangen Sie zu Ihrer neuen HTML-Website auf GitHub:



- In [1] sehen wir, dass Sie eine Website auf GitHub anzeigen;

Es ist extrem leicht, bei der Ausführung der Befehle aus [deploy.txt] Fehler zu machen. Es ist dann schwierig, zurückzugehen, da Git sich merkt, was man (falsch) gemacht hat. Um von vorne zu beginnen, zeigen Sie den Arbeitsordner an:



- Löschen Sie in [1] den Ordner [.git];

Kehren Sie anschließend zu PyCharm zurück und führen Sie die Befehlsfolge aus [deploy.txt] erneut aus.

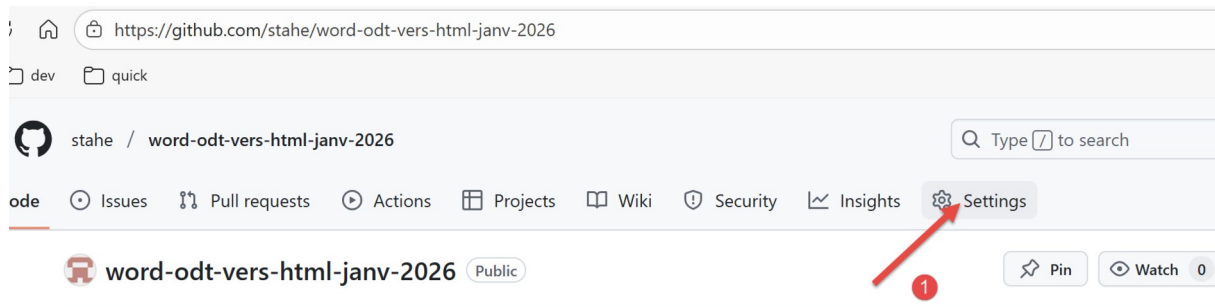
Was tun, wenn Sie Ihr ODT-/DOCX-Dokument ändern? Führen Sie die folgenden 3 Schritte aus:

1. Konvertieren Sie Ihr ODT-/DOCX-Dokument erneut mit [convert];
2. Erstellen Sie die HTML-Website mit [build];
3. Exportieren Sie die HTML-Website mit dem Befehl [python -m mkdocs gh-deploy] auf GitHub. Dieser Befehl reicht aus, solange Sie die Datei [README.md] nicht ändern. Wenn Sie die Datei [README.md] ändern, müssen Sie einige weitere Befehle ausführen:

```
1. git add README.md
2. # commit
3. git commit -m "Änderung README"
4. git push -u origin main
5. python -m mkdocs gh-deploy
```

Wenn Sie nur die README-Datei geändert haben, sind nur die Befehle 1, 3 und 4 erforderlich. Befehl 5 ist überflüssig, wenn Sie die HTML-Website bereits bereitgestellt haben und sich seitdem nichts geändert hat.

Was tun, wenn Sie von vorne beginnen möchten, weil etwas schiefgelaufen ist? Sie können Ihr GitHub-Repository löschen und alle Schritte aus dem Kapitel „12“ erneut ausführen. Die Option zum Löschen eines GitHub-Repositorys ist gut versteckt:



- Gehen Sie unter [1] zu den Repository-Einstellungen;

Danger Zone

Change repository visibility This repository is currently public.	Change visibility
Disable branch protection rules Disable branch protection rules enforcement and APIs	Disable branch protection rules
Transfer ownership Transfer this repository to another user or to an organization where you have the ability to create repositories.	Transfer
Archive this repository Mark this repository as archived and read-only.	Archive this repository
Delete this repository Once you delete a repository, there is no going back. Please be certain.	Delete this repository

Gehen Sie ganz nach unten auf der Seite [settings]. Dort finden Sie die Schaltfläche zum Löschen Ihres Repositorys [2].

13. Verfolgen Sie die Website mit Google Analytics und Google Search Console

Wenn Sie Ihre Website auf GitHub hosten, interessiert Sie wahrscheinlich die Anzahl der Besucher Ihrer Website. Dazu benötigen Sie ein Google Analytics (GA)-Tag. Wir fragen [ChatGPT](#), wie man ein GA-Tag erstellt:

comment obtient-t-on un marqueur Google Analytics pour un site web ?

Pour un site web, le **marqueur Google Analytics** correspond aujourd'hui à l'**identifiant de mesure GA4** (format `G-XXXXXXXX`). Voici la procédure complète et à jour pour l'obtenir.

1. Créer ou utiliser un compte Google Analytics


1. Accédez à Google Analytics et connectez-vous avec votre compte Google.
2. Si aucun compte n'existe encore :
 - Cliquez sur **Créer un compte**
 - Donnez un nom au compte (ex. : *Entreprise X*)
3. Si un compte existe déjà, sélectionnez-le.

2. Créer une propriété GA4

1. Dans **Administration** (icône engrenage, en bas à gauche),
2. Colonne **Propriété** → **Créer une propriété**,
3. Renseignez :
 - Nom de la propriété
 - Fuseau horaire
 - Devise
4. Validez.

Depuis plusieurs années, **GA4 est le seul type de propriété disponible**. Les anciennes propriétés Universal Analytics ne sont plus utilisables.

3. Créer un flux de données Web

1. Toujours dans **Administration**,
2. Colonne **Propriété** → **Flux de données**,
3. Cliquez sur **Ajouter un flux** → **Web**,
4. Saisissez :
 - L'URL du site (ex. `https://www.monsite.fr`)
 - Un nom de flux
5. Validez. 

- Geben Sie unter [1] die URL Ihrer Website auf GitHub ein;

4. Récupérer le marqueur (ID de mesure)

Après la création du flux web :

- Un **ID de mesure** est affiché, au format :

```
css Copier le code  


G-XXXXXXXXXX

2
```

👉 C'est ce marqueur **Google Analytics**

Il identifie de manière unique votre site dans GA4.

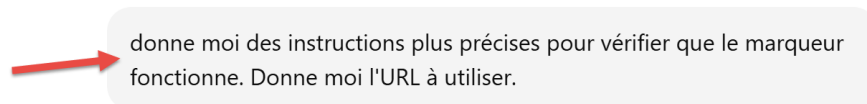
- Notieren Sie sich unter [2] den Tag für Ihre Website;

Fügen Sie diesen Tag in die Konfigurationsdatei [config.py] ein:

```
1. "extra": {  
2.   "analytics": {  
3.     "provider": "google",  
4.     "property": "G-XXXXXXXXXX"  
5.   }  
6. },
```

Zeile 4: Geben Sie Ihren Tag ein. MkDocs fügt ihn automatisch in alle Ihre HTML-Seiten ein, wenn die HTML-Website generiert wird. Sie müssen daher die weiteren Anweisungen von ChatGPT, die zeigen, wie Sie den GA-Tag in Ihre HTML-Seiten einfügen, nicht befolgen.

Wir werden ChatGPT um genaue Anweisungen bitten, um zu überprüfen, ob der Tag funktioniert:



Voici une procédure **opérationnelle et vérifiable pas à pas** pour contrôler que votre marqueur GA4 fonctionne, avec l'**URL exacte à utiliser**.

Méthode recommandée : vérification via le rapport *Temps réel*

1. Accéder à Google Analytics

Ouvrez directement l'interface de **Google Analytics** à l'adresse suivante :

```
https://analytics.google.com/ ↗
```

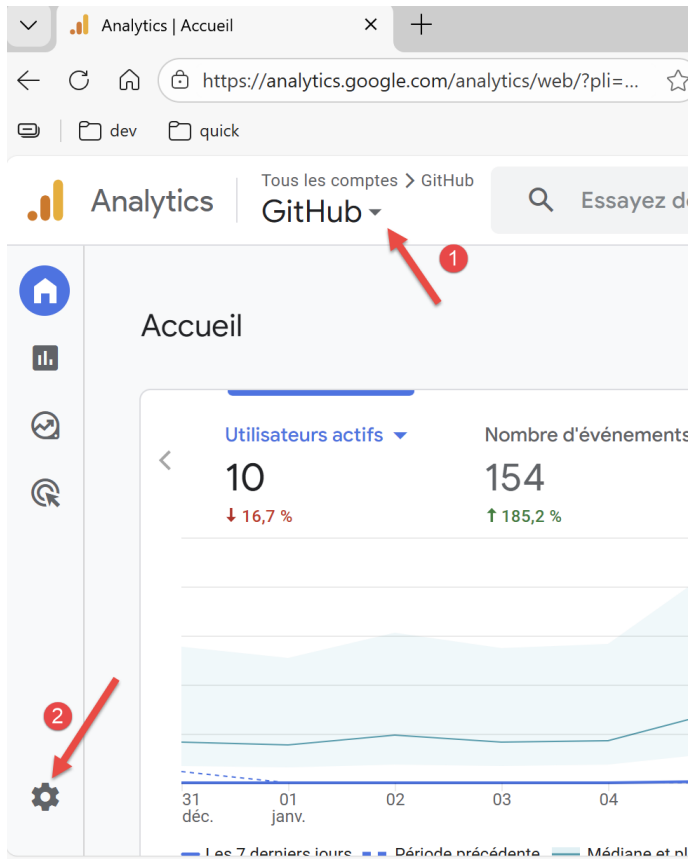
Connectez-vous avec le compte Google qui possède la propriété GA4.

2. Sélectionner la bonne propriété

En haut à gauche de l'interface :

1. Vérifiez que la **propriété GA4** correspondant à votre site est bien sélectionnée.
2. Si nécessaire, changez-la via le sélecteur de compte/propriété.

Der vorherige Schritt ist nicht ganz klar. Hier sind einige Screenshots, die Ihnen helfen sollen:



- Wählen Sie unter [1] das Konto aus, das Sie verfolgen möchten (falls Sie mehrere haben);
- Wählen Sie unter [2] die Schaltfläche „Verwaltung“ aus;

Analytics Tous les comptes > GitHub Essayez de rechercher "comment créer un entonnoir de conversion"

Administrateur
Ces paramètres s'appliquent à tous les utilisateurs de ce compte et de cette propriété. Pour voir ceux qui ne s'appliquent qu'à vous, accédez à [Mes préférences](#) dans le panneau de navigation de gauche.

PARAMÈTRES DU COMPTE

Compte
Ces paramètres ont une incidence sur votre compte Analytics [Qu'est-ce qu'un compte ?](#)

📄	Détails du compte	🔗	📊	Historique des quotas de l'API Account data	🔗
👤	Gestion des accès au compte	🔗	🗑️	Corbeille	🔗
🕒	Historique des modifications du compte	🔗			

PARAMÈTRES DE LA PROPRIÉTÉ

Propriété
Ces paramètres ont une incidence sur votre propriété [Qu'est-ce qu'une propriété ?](#)

📄	Détails de la propriété	🔗			
👤	Gestion des accès à la propriété	🔗			

Collecte et modification des données
Ces paramètres déterminent la manière dont les données sont collectées et modifiées

📄	Flux de données	🔗	3	
📄	Collecte des données	🔗		

- Wählen Sie unter [3] „Datenfeed“ aus;

Flux de données

Toutes les vues iOS Android Web Ajouter un flux

🌐	GitHub https://stahe.github.io	4	13112486249	La propriété a reçu du trafic au cours des dernières 48 heures.	>
---	-----------------------------------	---	-------------	---	---

- Klicken Sie unter [4] auf den Datenfeed;

Détails du flux Web

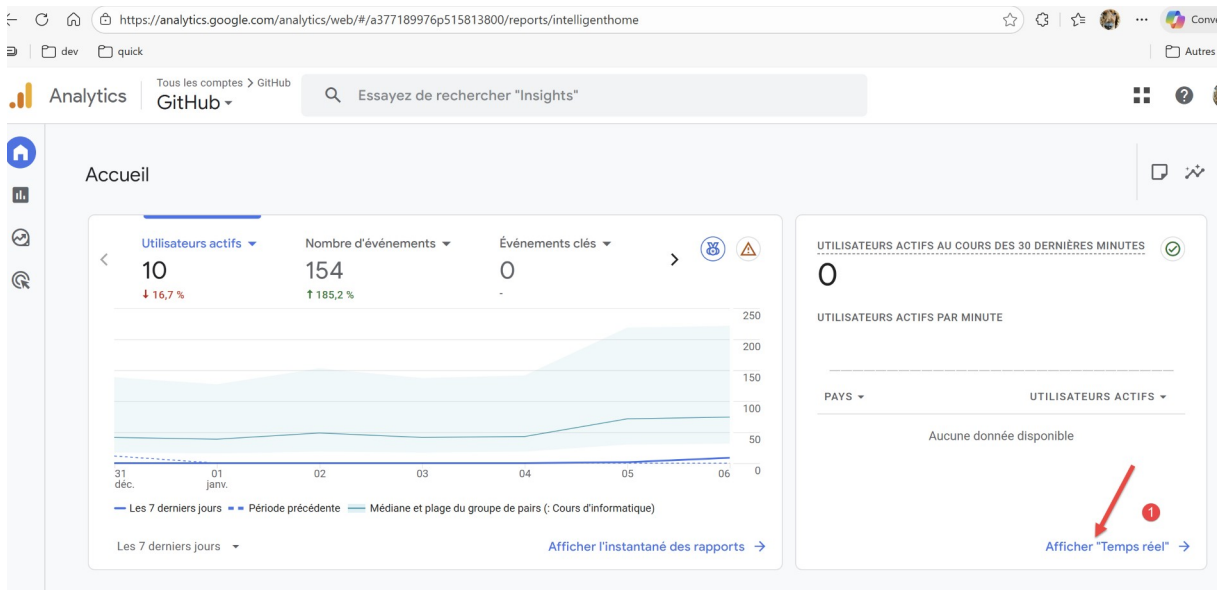
✔️ La collecte de données a été active au cours des 48 dernières heures.

Détails du flux ✎

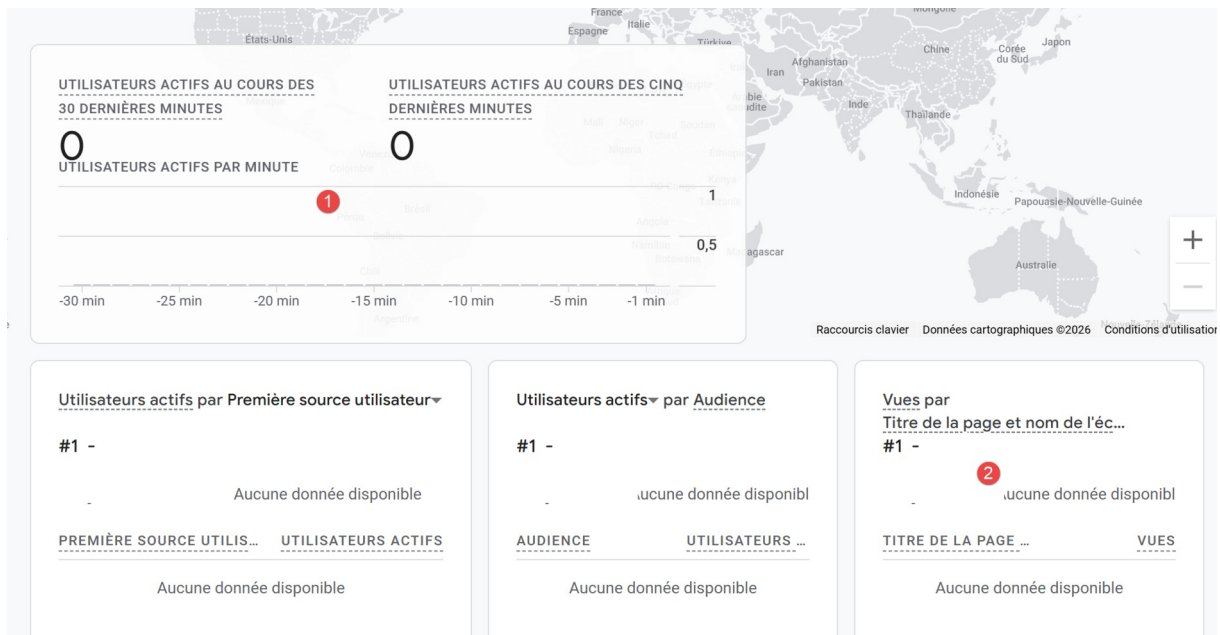
NOM DU FLUX	URL DE FLUX	ID DE FLUX	ID DE MESURE
GitHub	https://stahe.github.io	6 13112486249	G-EBZC: 5 📄

- Unter [5] finden Sie den von Ihnen erstellten GA4-Tag;
- Unter [6] die Stream-ID. Dies ist ein anderer Begriff als die Mess-ID;

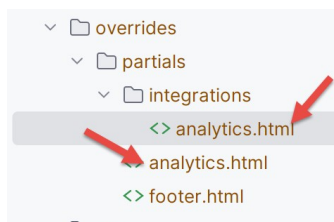
Kehren wir zur Startseite von GA zurück:



- Wählen Sie unter [1] die Echtzeit-Anzeige der Besuche aus;



- Unter [1] sehen Sie Ihre Besucher;
- In [2] die Seite, die besucht wurde. Ich habe den Gemini-Konverter gebeten, den Namen der Website anstelle des Seitennamens mit dem GA4-Tag zu verknüpfen. Diese Umbenennung in Verbindung mit GA4 wird durch die Datei [analytics.html] gewährleistet:



Nun können wir mit den Erklärungen von ChatGPT fortfahren:

4. Générer une visite test propre

Pour éviter toute ambiguïté :

1. Ouvrez une **fenêtre de navigation privée** (Ctrl+Maj+N ou Ctrl+Maj+P).
2. Saisissez l'URL exacte de votre site (ex. `https://www.monsite.fr`).
3. Naviguez sur une ou deux pages.
4. Attendez 5 à 30 secondes.

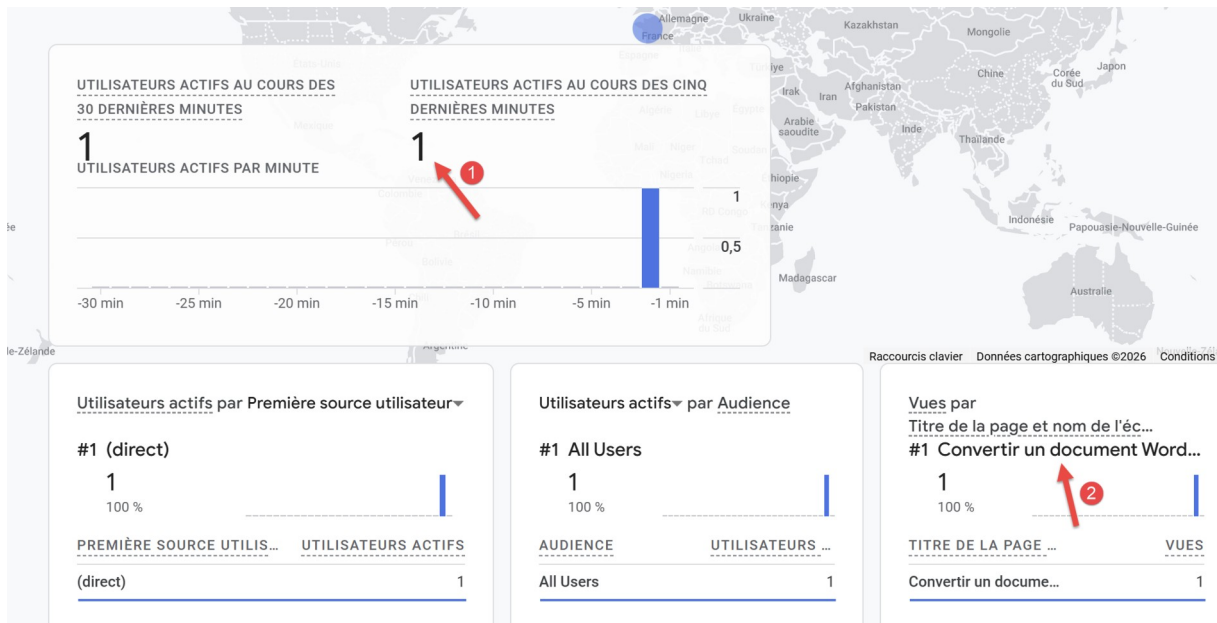
5. Résultat attendu

Dans le rapport *Temps réel*, vous devez voir au minimum :

- 1 utilisateur actif
- Une page consultée
- Une source de trafic (souvent *Direct*)

Si ces éléments apparaissent → le marqueur fonctionne correctement.

Ich habe die in [4] geforderten Schritte ausgeführt, ohne ein privates Fenster zu verwenden. Und ich habe Folgendes erhalten:



Das GA4-Tag von Google Analytics funktioniert einwandfrei.

Nun stellen wir ein weiteres Tool zur Überwachung Ihrer Website vor: [Google Search Console]. Fragen wir ChatGPT, wozu dieses Tool dient:

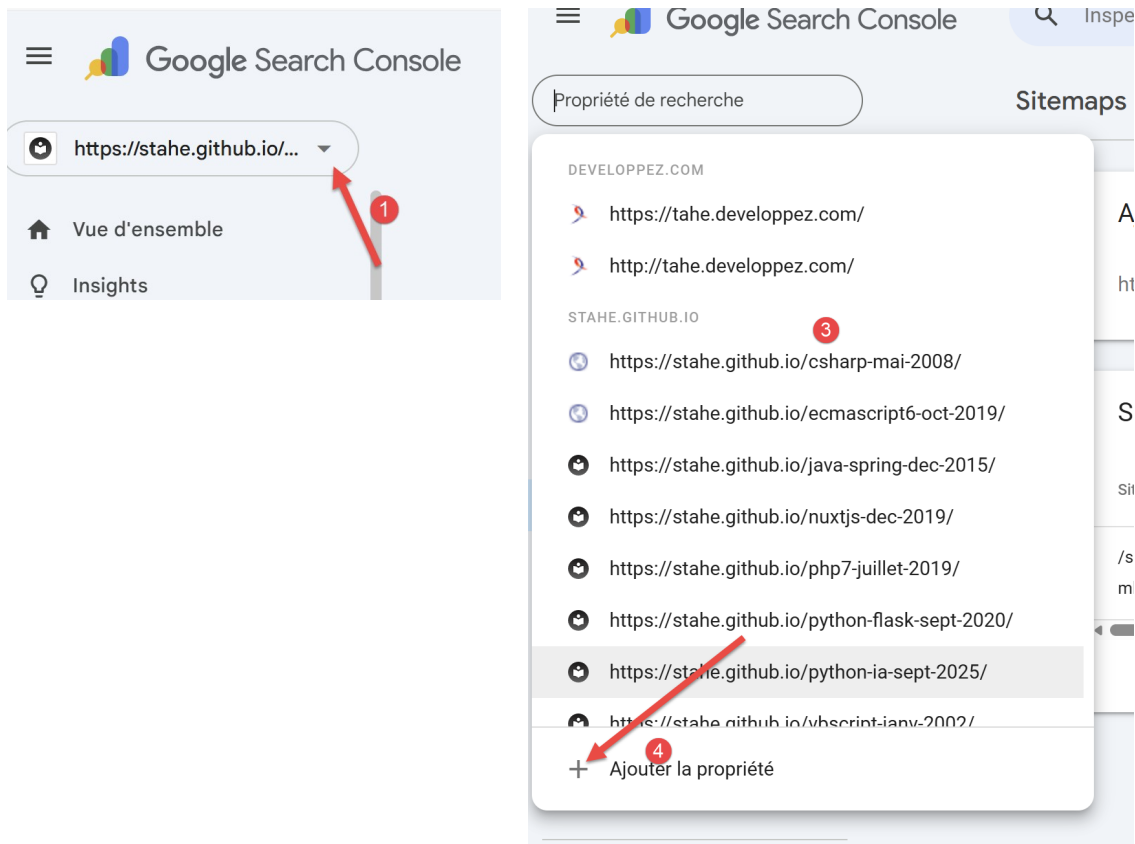
Conclusion

Google Search Console sert à :

- comprendre **comment Google perçoit votre site**,
- améliorer votre **référencement naturel (SEO)**,
- détecter rapidement les **problèmes bloquants**,
- piloter votre visibilité dans Google Search.

Ich nutze dieses Tool so gut wie nie. Ich tue dies nur, um die Google-Suchmaschine dazu anzuregen, die Seite zu crawlen und zu indexieren, damit Internetnutzer sie finden können.

Die URL des Tools [<https://search.google.com/search-console>]:



- Unter [1] die Liste der Websites, die bei Google angemeldet wurden;
- Unter [3] die Liste Ihrer Websites;
- Unter [4] fügen Sie Ihre neue Website hinzu;

Sélectionnez le type de propriété

Domaine

- Toutes les URL de tous les sous-domaines (m. www ...)
- Toutes les URL sur https et http
- Validation DNS obligatoire

example.com
Saisissez un domaine ou sous-domaine

CONTINUER

ou

Préfixe de l'URL

- Seules les URL de l'adresse saisie
- Seules les URL du protocole spécifié
- Différentes méthodes de validation acceptées

ub.io/word-odt-vers-html-janv-2026/
Saisissez une URL

CONTINUER

- Unter [5] geben Sie die URL Ihrer Website ein. Diese finden Sie in der Datei [config.py]:

```
1. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

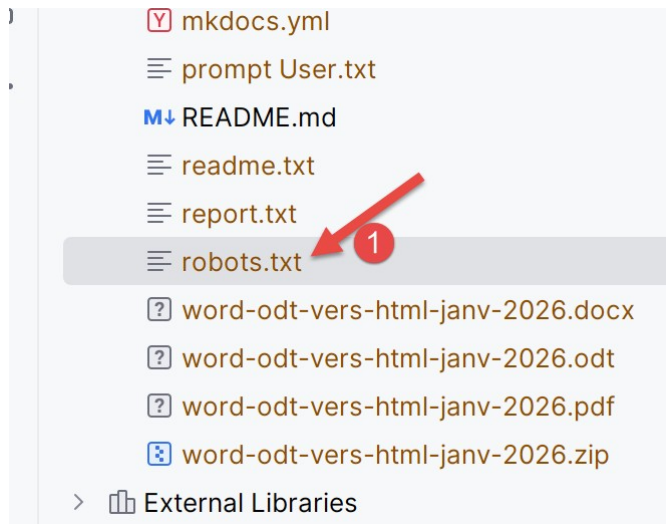
- unter [6] fahren Sie mit dem nächsten Schritt fort;

Normalerweise schlägt Ihnen die Google Search Console in diesem Schritt eine Datei [googlexxxxx.html] vor, die Sie herunterladen müssen. Dieser Bildschirm wurde mir nicht angezeigt, da ich diese Datei bereits auf der bereitgestellten Website hatte. Es handelt sich um eine der beiden in [config.py] deklarierten Dateien

```
1. "files_to_copy": [  
2.     "google5179xxxxx.html",  
3.     "robots.txt"  
4. ]
```

Legen Sie die Datei [googlexxxxx.html] im Stammverzeichnis Ihres Arbeitsordners ab und tragen Sie ihren Namen in Zeile 2 der Datei [config.py] ein. Das Skript [convert] sorgt dafür, dass die beiden oben genannten Dateien im Stammverzeichnis der von ihm generierten MkDocs-Website abgelegt werden. Das Skript [build] legt sie im Stammverzeichnis der generierten HTML-Website ab.

Wenn Sie die Datei [googlexxxxx.html] in das Stammverzeichnis Ihres Arbeitsordners kopiert haben, überprüfen Sie den Inhalt der Datei [robots.txt]:



Der Inhalt der Datei [robots.txt] lautet wie folgt:

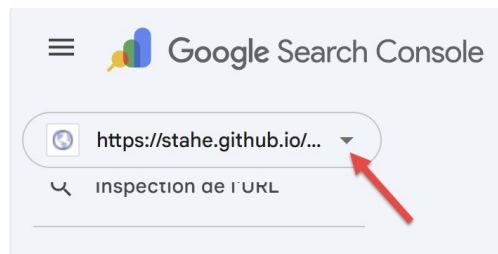
```
1. User-agent: *
2. Allow: /
3. Sitemap: https://stahe.github.io/word-odt-vers-html-janv-2026/sitemap.xml
```

Überprüfen Sie in Zeile 3, ob die URL die Ihrer Website ist, also dieselbe wie in der Datei [config.py].

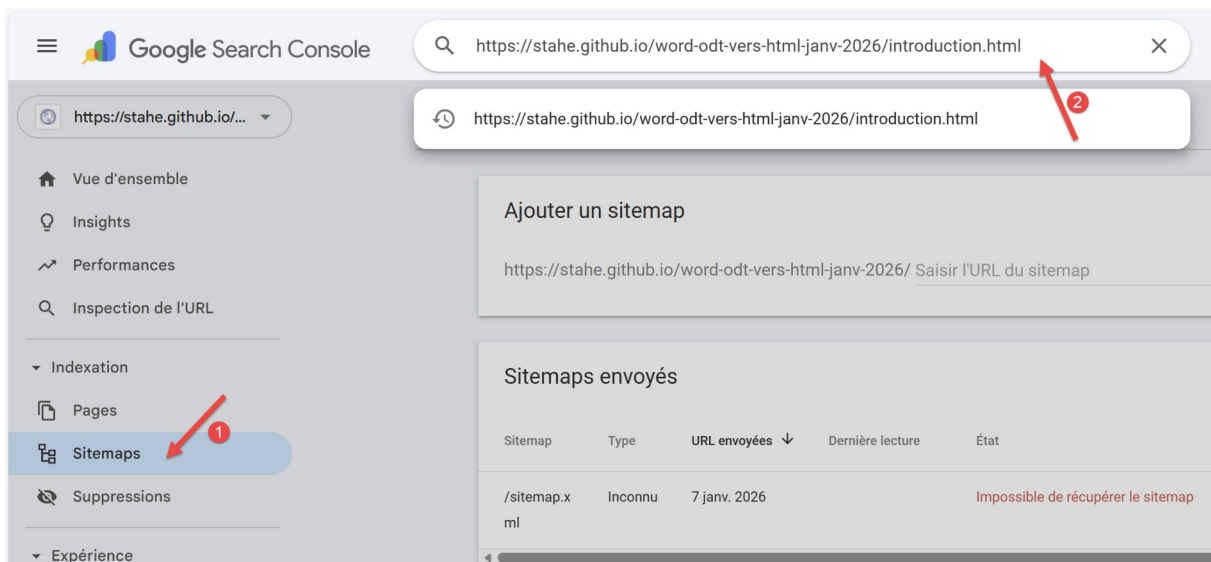
Stellen Sie Ihre Website erneut auf GitHub bereit, indem Sie die folgenden drei Befehle in Ihr Python-Terminal eingeben:

```
1. python .\convert.py .\word-odt-vers-HTML.odt config.py
2. python .\build.py
3. python -m mkdocs gh-deploy
```

Kehren Sie anschließend zur [Google Search Console] zurück und wählen Sie die zuvor erstellte Website aus.

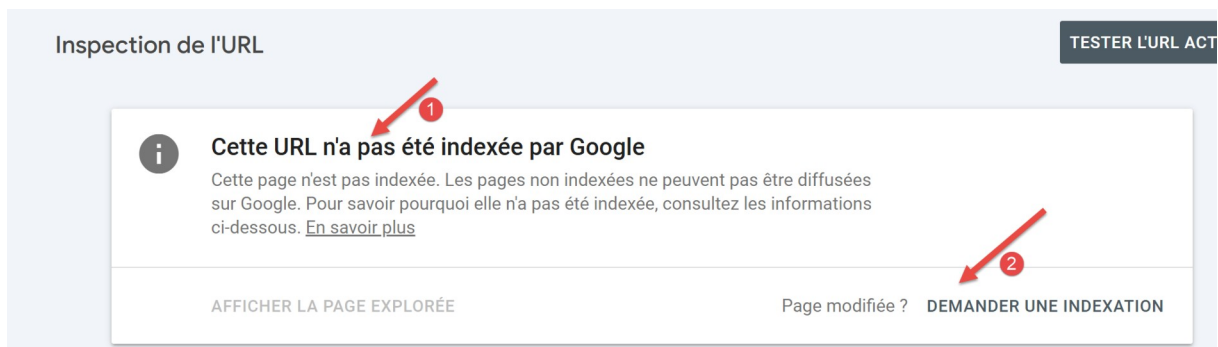


- Wählen Sie unter [1] die Option [Sitemaps] aus;
- Geben Sie unter [2] [sitemap.xml] ein;



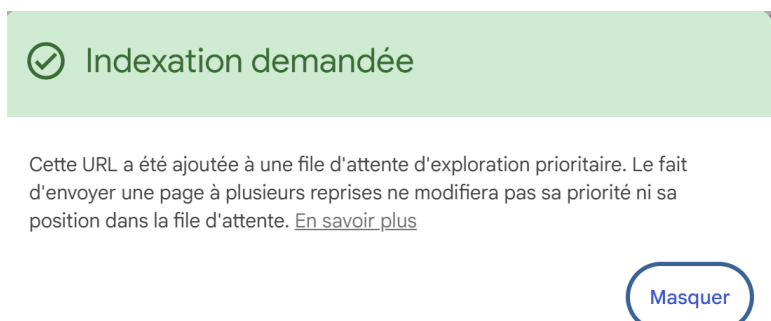
- Geben Sie unter [2] die URL einer der Seiten Ihrer Website ein;

Hier ist das Ergebnis:



- Unter [1] gibt Google an, dass die Seite noch nicht indiziert ist;
- Unter [2] können Sie die Indizierung beantragen;

Die Google Search Console prüft dann, ob die angeforderte Seite indiziert werden kann. Wenn ja, erhalten Sie folgende Meldung:



Sie können dies für alle Seiten Ihrer Website tun, um sicherzustellen, dass sie von der Google-Suchmaschine ordnungsgemäß indiziert wird.

14. Fazit

Wir haben in diesem Dokument einen Konverter vorgestellt, der ODT-Dokumente aus [LibreOffice](#) oder DOCX-Dokumente aus Word in eine statische HTML-Website von sehr guter Qualität umwandelt.

Im Kapitel [\[Beispiele in diesem Dokument\]](#) haben wir die Strukturen vorgestellt, die vom Gemini/ChatGPT-Konverter korrekt verarbeitet werden.

Wir haben außerdem gezeigt, wie man die von den ODT- und DOCX-Konvertern erstellten HTML-Websites auf GitHub exportiert (siehe Kapitel [„Die HTML-Website auf GitHub hosten“](#))

Abgesehen von den Strukturen im Kapitel „Beispiele“ gibt es keine Garantie. Es ist wahrscheinlich, dass der Konverter Fehler aufweist. In diesem Fall muss erneut ein Chat mit Gemini oder ChatGPT gestartet werden, um diese Fehler zu beheben. Am einfachsten ist es, dem KI-System den aktuellen Konverter zusammen mit seiner Konfigurationsdatei zu übergeben. Dann fragen Sie die KI, ob sie die ihr gegebenen Codes versteht. Das zwingt sie dazu, den Code zu analysieren. Sie wird immer antworten, dass sie den Code vollkommen versteht. Das ist der Moment, in dem Sie sie nach den gewünschten Änderungen fragen sollten. Sowohl für Word als auch für LibreOffice können Sie der KI mit Makros helfen. Die KI „sieht“ das verwendete ODT- oder DOCX-Dokument nicht. Ihr fehlen also Informationen. So konnte der DOCX-zu-HTML-Konverter die Nummerierung der Rich-Code-Blöcke nicht korrekt verarbeiten. Sie konnte die Nummer der ersten Zeile eines Code-Blocks nicht lesen, um ihn in HTML wiederzugeben. Sie begann den Code immer bei 1, auch wenn der Code im DOCX-Dokument bei 12 begann. Ich habe mir daher ein Makro zu Hilfe genommen. Ich habe den Cursor im Dokument auf die nummerierte Zeile gesetzt und ChatGPT gebeten, ein VBA-Makro zu generieren, das ihm die Eigenschaften dieser Zeile liefert, damit es die HTML-Nummerierung korrekt generieren kann. ChatGPT hat dies getan, und ich habe ihm das Ergebnis seines Makros gegeben. Das hat das Problem gelöst: Die abgerufenen Informationen zeigten ihm, dass die Zeilennummer durch den Stil und nicht durch das Nummerierungssymbol weitergegeben wurde. Daraufhin änderte er seinen Algorithmus, was funktionierte. Man kann der KI also dabei helfen, Fortschritte zu machen.

Ein anderer, einfacherer Weg ist, das Dokument zu ändern. Wenn der Konverter eine Anomalie erzeugt, versuchen Sie, Ihr Dokument so zu ändern, dass es nur die Strukturen aus dem Beispielkapitel verwendet.