



Convert a Word or ODT document into a static HTML site compatible with MkDocs using Gemini 3 and ChatGPT 5.2

Serge Tahé, January 2026

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

This document is a translation of the original [French version](#), generated using [an automated tool](#). Some expressions or technical terms may not fully reflect the original meaning.

This site was created using the [Word or ODT -> HTML] converter developed by Gemini 3 and ChatGPT 5.2 in January 2026 and described in this document.

Using the Gemini/ChatGPT converter, several websites have been generated from LibreOffice ODT documents and Word documents:

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 \(2006\)](#)];

These are all old courses consisting of several hundred pages. The size of the ODT or DOCX document does not matter to the Gemini/ChatGPT converter. What matters to it are the structures found within them (see chapter: [Examples in this document](#)).

1. Introduction

The PDF of this document is available |[HERE](#)|.

The Gemini 3 / ChatGPT 5.2 converters [Word or ODT → HTML] are available |[HERE](#)|.

This document provides you with two converters:

- a LibreOffice ODT to HTML converter;
 - a Word DOCX to HTML converter;
-

In this article, we aim to provide readers with a Python converter for converting Word or ODT documents into a static HTML site. This converter was initially built by the [Gemini 3](#) AI and then by [ChatGPT 5.2](#). It took these two AIs 356 iterations to produce the converter for this document. This involved several weeks of debugging. Gemini 3 did all the work at the beginning. It took several dozen iterations just to get a first version that was roughly correct. Then, I regularly added a new problem posed by new ODT documents. And at that point, Gemini often regressed. That is, what worked at step N no longer worked at subsequent steps. So I proceeded as follows: as soon as Gemini produced a converter that solved one of my problems, I made it a reference version and told Gemini about it. I saved this reference locally. Then, when I saw Gemini regressing for too long, I would ask it to revert to the last known reference version by providing it with the code. That's how, little by little, we built this converter together: me telling it what I wanted—essentially by pointing out the anomalies I found in the generated HTML site—and it producing the requested code.

I used [Gemini 3](#) with a pro license at 22 euros/month and [ChatGPT 5.2](#) as well.

Gemini / ChatGPT will generate two Python scripts:

- [convert] to convert the [LibreOffice](#) ODT document or the Word DOCX document into an [MkDocs](#) site;
- [build] to convert the MkDocs site into a static HTML site;

I never looked at the generated code. I wanted to treat them as black boxes. You don't need to be a Python developer to follow this tutorial—or even a developer at all.

When I suggested improvements to the converter to the Gemini 3 AI, it sometimes froze. It couldn't make the requested improvements. At that point, we were on stable version V316. To move forward, I then gave this working version to [ChatGPT 5.2](#) and asked it to make the desired improvements. ChatGPT correctly modified Gemini's code to meet my new requests. That's why I consider these two AIs to be the ones that generated the converter.

After that, I continued to use this technique. When one AI got stuck on a feature, I gave the other AI the latest known stable version.

The two AIs have different methods for delivering the requested Python code:

- Gemini provides the generated script code on the query page. You then need to copy and paste this code;
- ChatGPT provides a download link for the generated script;

Beyond 1,000 lines of generated code, Gemini showed significant weaknesses. Due to technical limitations specific to this AI, it was unable to display the entire generated code on the query page. Very often, lines of code were missing. Because of this limitation, at a certain point it was no longer possible to use Gemini 3. And so it was ChatGPT 5.2 that finished writing the script.

ChatGPT also generated the Word-to-HTML converter. I gave it the working ODT-to-HTML converter and asked it to adapt it for a Word document. It did so in 18 iterations. This is a consistent trait of both of these AIs: they understand the Python scripts we give them very well and can make significant improvements to them. For me, this is the best way to work with them.

2. Examples from this document

I'd like to write a short article. The interactions between an AI and a user are presented in the article [[Generating a Python Script with AI Tools](#)], which I'll refer to from now on as [ref1]. Interactions with Gemini and ChatGPT will only be mentioned briefly. In any case, it was impossible to present all the iterations.

I will now present examples of the specific features of my ODT/DOCX documents that the Gemini/ChatGPT converter handles correctly. It is this very document that we will submit for HTML conversion to the Gemini/ChatGPT script. We'll see what it does with it.

2.1. Lists

The Gemini/ChatGPT converter can handle bulleted and numbered lists, even if they are nested:

2.1.1. Bulleted lists

- Item 1;
- Item 2;
- Item 3;
 - Item 3.1;
 - Item 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. Numbered lists

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. Mixed lists

- 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. Mixed lists 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. Manually numbered lists

Manual numbering here means that the user sets the number of a numbered paragraph: [right-click on the numbered paragraph / Paragraph / Paragraph / Restart Numbering / Start With].

I start a list with a number other than 1.

6. Element 6
7. Element 7

Here I break the list to say something, but then I want to continue the numbering.

8. Item 8
9. Item 9

Then I start a new numbered list:

11. item 11
12. item 12

2.2. Code blocks

My courses contain a lot of code blocks. These are often formatted code (bold, keyword colors) from IDEs (Eclipse, PyCharm, WebStorm, NetBeans). This formatted code is rendered identically by the converter.

When the code is not formatted (code from Notepad or similar), the Gemini/ChatGPT converter recognizes it (Java, C#, XML, HTML, etc.) using language keywords specified in a configuration file. When it recognizes a language, it inserts a marker (fence) for MkDocs so that MkDocs can adapt the syntax highlighting of the code to the language used in the code block.

2.2.1. Enriched code blocks (Eclipse, Visual Studio, etc.)

Here are code blocks enhanced by various IDEs:

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.         // Retrieve the Spring context
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // Retrieve the beans
16.         Person p01 = ctx.getBean("person_01", Person.class);
17.         Person p02 = ctx.getBean("person_02", Person.class);
18.         List<Person> club = ctx.getBean("club", new ArrayList<Person>().getClass());
19.         Apartment apt01 = ctx.getBean(Apartment.class);
20. ...

```

C#

```

11. using System;
12.
13. namespace Chap1 {
14.     class Taxes {
15.         static void Main(string[] args) {
16.             // Data arrays needed to calculate the tax
17.             decimal[] limits = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M,
18. 0M };
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.             // retrieve marital status
25.             bool OK = false;
26.             string response = null;
27.             while (!OK) {
28.                 Console.Write("Are you married (Y/N)?");
29.                 response = Console.ReadLine().Trim().ToLower();
30.                 if (response != "y" && response != "n")
31.                     Console.Error.WriteLine("Incorrect answer. Please try
32. again");
33.                 else OK = true;
34.             } //while
35.             bool marie = response == "y";
36. ...

```

Python

```

21. # -----
22. def display(string):
23.     # displays string
24.     print("string=%s" % string)
25.
26.
27. # -----
28. def display_type(variable):
29.     # displays the variable type
30.     print("type[%s]=%s" % (variable, type(variable)))
31.
32.
33. # -----
34. def f1(param):
35.     # adds 10 to param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # returns a tuple of 3 values
42.     return "one", 0, 100
43.

```

```

44.
45. # ----- main program -----
46. ...

```

PHP

```

31. <?php
32.
33. // strict types for function parameters
34. declare(strict_types=1);
35.
36. // global constants
37. define("HALF-RATE_INCOME_LIMIT", 1551);
38. define("INCOME_LIMIT_SINGLE_PERSON_FOR_REDUCTION", 21037);
39. define("INCOME_LIMIT_FOR_COUPLES_FOR_REDUCTION", 42074);
40. define("REDUCTION_VALUE_HALF_PORTION", 3797);
41. define("SINGLE_PERSON_DISCOUNT_LIMIT", 1196);
42. define("COUPLE_DEDUCTION_LIMIT", 1970);
43. define("COUPLE_INCOME_THRESHOLD_FOR_DEDUCTION", 2627);
44. define("SINGLE_TAX_CAP_FOR_DEDUCTION", 1595);
45. define("MAXIMUM_10_PERCENT_DEDUCTION", 12502);
46. define("MINIMUM_10_PERCENT_DEDUCTION", 437);
47.
48. // definition of local constants
49. $DATA = "taxpayersdata.txt";
50. $RESULTS = "results.txt";
51. $limits = 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. // read data
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Unable to open the data file [$DATA] for reading\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // this is a comment
43. // constant
44. const name = "dupont";
45. // display on screen
46. console.log("name: ", name);
47. // an array with elements of different types
48. const array = ["one", "two", 3, 4];
49. // its number of elements
50. let n = array.length;
51. // a loop
52. for (let i = 0; i < n; i++) {
53.     console.log("array[" + i + "] = ", array[i]);
54. }
55. // Initializing 2 variables with the contents of an array
56. let [string1, string2] = ["string1", "string2"];
57. // concatenating the two strings
58. const string3 = string1 + string2;
59. // display result
60. console.log([string1, string2, string3]);
61. ...

```

VBScript

```

51. ' Calculating a taxpayer's tax

```

```

52. ' the program must be called with three parameters: married children salary
53. ' married: character 0 if married, N if unmarried
54. ' children: number of children
55. ' salary: annual salary without cents
56.
57. ' mandatory variable declaration
58. Option Explicit
59. Dim error
60.
61. ' retrieve the arguments and check their validity
62. Dim wife, children, salary
63. error = GetArguments(wife, children, salary)
64. ' error?
65. If error(0) <> 0 Then wscript.echo error(1) : wscript.quit error(0)
66.
67. ' Retrieve the data needed to calculate the tax
68. Dim limits, coeffR, coeffN
69. error = getData(limits, coeffR, coeffN)
70. ' error?
71. If error(0) ≠ 0 Then wscript.echo error(1) : wscript.quit 5
72.
73. ' Display the result
74. wscript.echo "tax=" & calculateTax(spouse, children, salary, limits, coeffR, coeffN)
75.
76. ' exit without error
77. wscript.quit 0

```

XML

```

61. <?xml version="1.0" encoding="utf-8" ?>
62. <configuration>
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.     </sectionGroup>
69.   </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 limit, coeffr, coeffn from tranches"/>
80.       </object>
81.       <object name="job" type="Job.JobTax, TaxesV7-job">
82.         <constructor-arg index="0" ref="dao"/>
83.       </object>
84.     </objects>
85.   </spring>
86. </configuration>

```

2.2.2. Plain text code blocks

Here are some examples of plain text:

Execution results

Note that the code does not start on line 1.

```

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. list[0] = one
10. list[1] = two
11. list[2] = 3
12. list[3] = 4
13. [string1, string2, string1string2]
14. string = string1
15. type[4]=<class 'int'>
16. type[string1] = <class 'str'>
17. type[['one', 'two', 3, 4]] = <class 'list'>
18. type[has changed] = <class 'str'>
19. res1 = 14s
20. (res1, res2, res3) = [one, 0, 100]
21. list[0] = one
22. list[1] = 0
23. list[2] = 100
24. list[0] = 8
25. list[1] = 5
26. sum=13
27.
28. Process finished with exit code 0

```

This case required dozens of iterations. The converter could never find the line number of the first line of code. Finally, I asked Gemini how to format the code block so that it would recognize it. Here is what it replied:

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.

I followed these steps and it worked. I was using the method via the list icon (Customize). So you may need to modify certain elements of the ODT/DOCX document to get the result you want.

The converter can handle unnumbered code blocks.

```

list[0]=one
list[1]=0
list[2]=100
list[0]=8
list[1]=5
sum=13

```

Here are the same examples as in the "2.2.1 " section, but without additional information. In this case, the keywords in the configuration file will guide the converter to the correct language.

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.         // Retrieve the Spring context
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // Retrieve the beans
16.         Person p01 = ctx.getBean("person_01", Person.class);
17.         Person p02 = ctx.getBean("person_02", Person.class);
18.         List<Person> club = ctx.getBean("club", new ArrayList<Person>().getClass());
19.         Apartment apt01 = ctx.getBean(Apartment.class);
20.         ...

```

C#

```
11. using System;
12.
13. namespace Chap1 {
14.     class Taxes {
15.         static void Main(string[] args) {
16.             // Data tables required for tax calculation
17.             decimal[] limits = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M,
18.                0M };
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.             // retrieve marital status
25.             bool OK = false;
26.             string response = null;
27.             while (!OK) {
28.                 Console.Write("Are you married (Y/N)?");
29.                 response = Console.ReadLine().Trim().ToLower();
30.                 if (response != "y" && response != "n")
31.                     Console.Error.WriteLine("Incorrect answer. Please try
32. again");
33.                 else OK = true;
34.             } //while
35.             bool marie = response == "y";
36.         }
37.     }
38. }

```

Python

```
21. # -----
22. def display(string):
23.     # print string
24.     print("string=%s" % string)
25.
26.
27. # -----
28. def display_type(variable):
29.     # displays the variable type
30.     print("type[%s]=%s" % (variable, type(variable)))
31.
32.

```

```

33. # -----
34. def f1(param):
35.     # adds 10 to param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # returns a tuple of 3 values
42.     return "one", 0, 100
43.
44.
45. # ----- main program -----
46. ...

```

PHP

```

31. <?php
32.
33. // strict types for function parameters
34. declare(strict_types=1);
35.
36. // global constants
37. define("HALF-RATE_INCOME_LIMIT", 1551);
38. define("INCOME_LIMIT_SINGLE_PERSON_FOR_REDUCTION", 21037);
39. define("INCOME_LIMIT_FOR_COUPLES_FOR_REDUCTION", 42074);
40. define("REDUCTION_VALUE_HALF_PORTION", 3797);
41. define("SINGLE_PERSON_DISCOUNT_LIMIT", 1196);
42. define("COUPLE_DEDUCTION_LIMIT", 1970);
43. define("COUPLE_INCOME_THRESHOLD_FOR_DEDUCTION", 2627);
44. define("SINGLE_TAX_CAP_FOR_DEDUCTION", 1595);
45. define("MAXIMUM_10_PERCENT_DEDUCTION", 12502);
46. define("MINIMUM_10_PERCENT_DEDUCTION", 437);
47.
48. // definition of local constants
49. $DATA = "taxpayersdata.txt";
50. $RESULTS = "results.txt";
51. $limits = 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. // read data
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Unable to open the data file [$DATA] for reading\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // this is a comment
43. // constant
44. const name = "dupont";
45. // a screen display
46. console.log("name: ", name);
47. // An array with elements of different types
48. const array = ["one", "two", 3, 4];
49. // its number of elements
50. let n = array.length;
51. // a loop
52. for (let i = 0; i < n; i++) {
53.     console.log("array[" + i + "] = ", array[i]);
54. }
55. // Initializing 2 variables with the contents of an array

```

```

56. let [string1, string2] = ["string1", "string2"];
57. // concatenating the two strings
58. const string3 = string1 + string2;
59. // display result
60. console.log([string1, string2, string3]);
61. ...

```

VBScript

```

51. ' Calculating a taxpayer's tax
52. ' the program must be called with three parameters: married, children, salary
53. ' married: character 0 if married, N if unmarried
54. ' children: number of children
55. ' salary: annual salary without cents
56.
57. ' mandatory variable declaration
58. Option Explicit
59. Dim error
60.
61. ' retrieve the arguments and check their validity
62. Dim wife, children, salary
63. error = GetArguments(wife, children, salary)
64. ' error?
65. If error(0) <> 0 Then wscript.echo error(1) : wscript.quit error(0)
66.
67. ' Retrieve the data needed to calculate the tax
68. Dim limits, coeffR, coeffN
69. error = getData(limits, coeffR, coeffN)
70. ' error?
71. If error(0) ≠ 0 Then wscript.echo error(1) : wscript.quit 5
72.
73. ' Display the result
74. wscript.echo "tax=" & calculateTax(spouse,children,salary,limits,coeffR,coeffN)
75.
76. ' exit without error
77. wscript.quit 0

```

XML

```

61. <?xml version="1.0" encoding="utf-8" ?>
62. <configuration>
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 limit, coeffr, coeffn from tranches"/>
82.     </object>
83.     <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
84.       <constructor-arg index="0" ref="dao"/>
85.     </object>
86.   </objects>
87. </spring>
88. </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.         width: 100%;
87.         display: table;
88.         font-weight: 100;
89.         font-family: '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.        font-size: 96px;
105.      }
106.
107.      .opt {
108.        margin-top: 30px;
109.      }
110.
111.      .opt a {
112.        text-decoration: none;
113.        font-size: 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.          Document Root: C:/myprograms/laragon-lite/www<br />
130.
131.        </div>
```

2.3. Links

The Gemini / ChatGPT converter preserves external links in ODT / DOCX documents. For example, [Gemini 3](#) or [\[Generate a Python script with AI tools\]](#).

It can handle a link to a chapter [Link to a chapter](#)

A reference to a chapter:2.1.1 .

A reference to a preceding text marker:

A reference to a text anchor that follows: GitHub

2.4. Text formatting

The converter supports bold, italic, underlined, and highlighted text. It preserves the highlight color.

Text with words in **bold**, *italics*, underlined, or highlighted.

This also applies to links: [\[Generate a Python script with AI tools\]](#).

The **converter** also **handles character colors**.

It also handles the top and bottom borders of paragraphs.

- It also handles the top and bottom borders of paragraphs.
 - It also handles the top and bottom borders of paragraphs.
 - It also handles the top and bottom borders of paragraphs.
-

1. It also handles the top and bottom borders of paragraphs.
 2. It also manages the top and bottom borders of paragraphs.
 3. It also handles the top and bottom borders of paragraphs.
-

10. Text with words in **bold**, *italics*, underlined, or highlighted.
 11. This also applies to links: [\[Generate a Python script with AI tools\]](#).
 12. The **converter** also **handles character colors**.
- A text with words in **bold**, *italics*, underlined, or highlighted.
 - This also applies to links: [\[Generate a Python script using AI tools\]](#).
 - The **converter** also **handles character colors**.

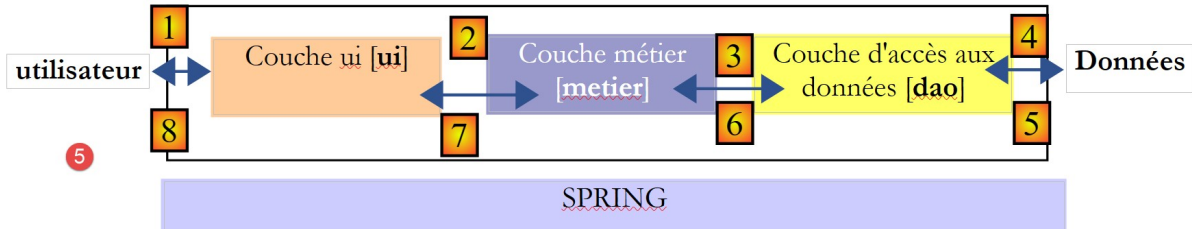
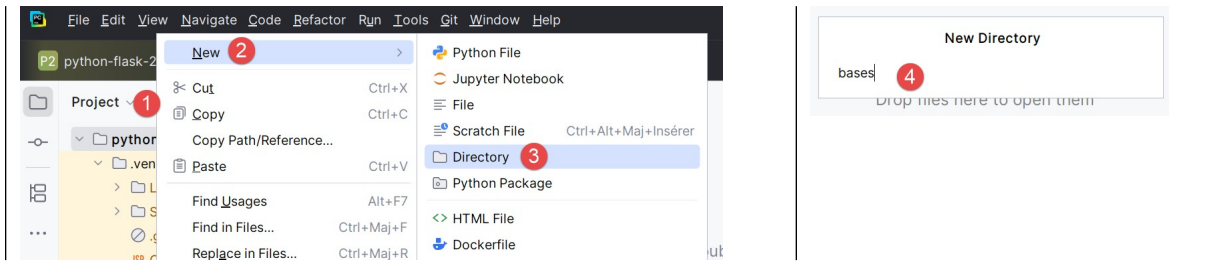
It also handles the paragraph background

- It also handles the paragraph background
- 1. It also handles the paragraph background

2.5. A **title** can also be *enhanced*.

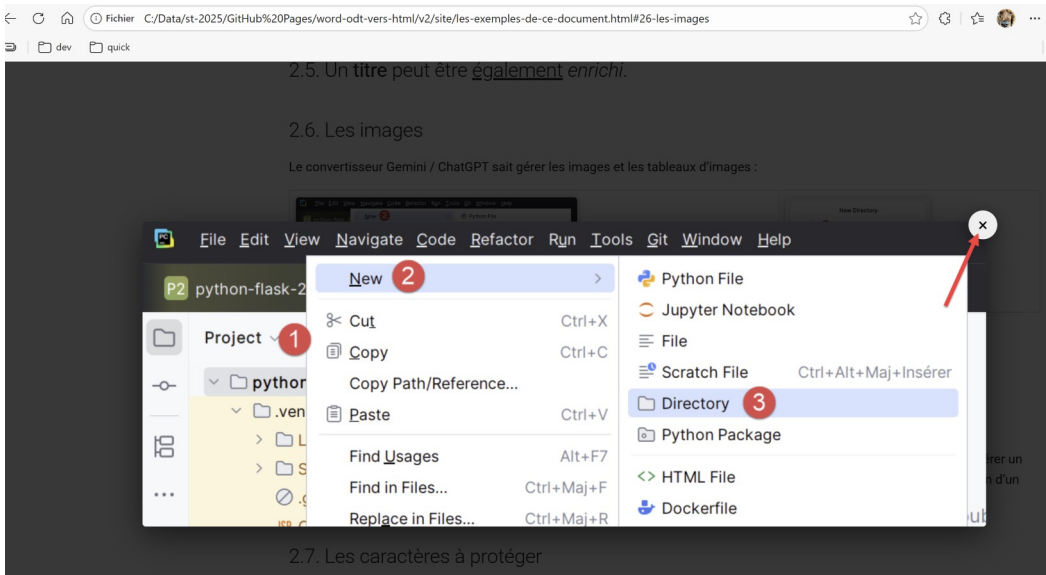
2.6. Images

The Gemini/ChatGPT converter can handle images and image tables:



In ODT documents, it is common to have drawings. Despite dozens of attempts, Gemini was unable to generate a script that would produce an image (such as a screenshot) of the drawing. Thus, in the example above, Image 5 is a screenshot of a drawing from an ODT document.

All images are clickable to enlarge. If you click on image [1-3] above, you get the following enlarged view:



2.7. Characters to protect

A MkDocs site¹ has pages whose content is not HTML but Markdown. If the ODT/DOCX document contains characters that exist in Markdown, they may be interpreted by MkDocs and therefore not render as expected. Here are two examples:

The asterisk * has a Markdown meaning. The following line may therefore be misinterpreted:

The tax I is then equal to $0.15 * R - 2072.5 * nbParts$.

¹MkDocs

Another example is when you want to insert a Markdown code block into your document like this:

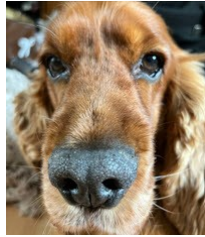
```
1. # Word/ODT to HTML Site Converter (MkDocs)
2.
3. 🔗 **[See the generated demo site](https://stahe.github.io/word-odt-vers-html-janv-2026/)**
4.
5. ---
6.
7. ## 📄 Description
8.
9. This project aims to provide readers with a Python converter for converting Word or ODT
10. documents into a static HTML site.
11. When the ODT/DOCX document is suitable, the converter generates an HTML site using **MkDocs**
12. that has the professional look of sites produced by Pandoc.
13.
14. ## 🛠️ Development Background
15. This converter was built entirely by the **Gemini 3** AI (with a pro subscription). It is the
16. result of successive iterations to finely manage the structure of ODT (OpenDocument Text)
17. documents.
18.
19. ## ✨ Features
20. The `convert.py` script performs the following actions:
21. * **ODT to Markdown Conversion**: Parses the `.odt` (XML) file to extract its structure.
22. * **Heading Management**: Automatically generates the Table of Contents (TOC) and sidebar
23. navigation.
24. * **Code Blocks**: Automatic language detection, syntax highlighting, and **precise line
25. numbering** (`start-value` attributes).
26. * **Lists**: Support for bulleted and numbered lists with correct indentation.
27. * **Formatting**: Support for *bold*, *italic*, *underline*, and *highlighting* (preserving
28. original colors).
29. * **Images**: Automatic extraction and embedding of images contained in the document.
30. * **Configuration**: Customization via a `config.py` file (Footer, Google Analytics, etc.).
31.
32. ## 🚀 Installation
33. ### Prerequisites
34. * Python 3.x
35. * The following libraries:
36. ```bash
37. pip install odfpv unicode mkdocs mkdocs-material
38. ```
39.
```

2.8. Tables

A table can contain various types of content:

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 display(string):
3.     # displays string
4.     print("string=%s" % string)
5.
6.
7. # -----
8. def display_type(variable):
9.     # displays the variable type
10.    print("type[%s]=%s" % (variable,
11.    type(variable)))
12.

```

An array containing links:

The Gemini / ChatGPT converter preserves external links from the ODT / DOCX document. example [Gemini 3](#) or [\[Generate a Python script using AI tools\]](#).

A reference to a chapter:2.1.1 .
 A reference to a preceding text anchor:
 A reference to a text marker that follows: GitHub

It can handle a link to a chapter [Link to a chapter](#)

A table whose cells contain another table:

`html:form` is used both to generate the HTML tag `<form>` and to provide information to the controller that will process this form:

<code>action</code>	url where the form values will be sent
<code>name</code>	name of the HTML form. This is also the name of the bean that will store the form's values
<code>type</code>	Name of the class that must be instantiated to obtain the form storage bean

Note that the method used to send form parameters (GET/POST) to the controller is not specified. This could be done using the **method** attribute. If this attribute is missing, the **POST** method is used by default.

`html:text` Used to generate the `<input type="text" value="...">` tag:

<code>property</code>	name of the field in the form bean that will be associated with the input field. When the form is submitted to the server (client -> server), the bean field will take the value of the input field. When the form is displayed (server -> client), the value contained in the bean field is displayed in the input field.
-----------------------	--

It also manages the background of the paragraph within a table

- It also manages the background of the paragraph in a table
- 1. It also manages the background of the paragraph in a table

2.9. Footnotes

The Gemini² / ChatGPT converter handles footnotes. Here is another footnote³.

3. What's available online

Before getting started, we can ask the AI what free tools are available on the market for converting a Word or ODT document into a static HTML site.

Let's ask Gemini:

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

Here is its response:

Yes, there are several free solutions, but the choice depends on your definition of a "static site" (a simple HTML page vs. a full website with navigation) and your technical skills.

Here are the best options, categorized by method:

- **1. The "All-in-One" Solution (To Create a Fully Navigable Website)**

If you want to turn a long Word document (with chapters and subchapters) into a full website with a navigation menu on the left, this is the best option.

- **Software: HelpNDoc** (Free version for personal use)
- **How it works:** This is documentation creation software. You import your Word file, it analyzes the structure (Heading 1, Heading 2, etc.), and automatically generates a static, responsive HTML5 website (optimized for mobile devices).
- **Pros:** Creates a real website with a menu, search bar, and index without writing a single line of code.
- **Cons:** The free version adds a small "Generated by HelpNDoc" banner at the bottom of the page.
- **2. The "Online Cleaner" Solution (For converting page by page)**

If you have a short document and just want to get the HTML code to put it on an existing page, online converters are ideal for cleaning up the "messy" code generated by Word.

- **Tool: WordToHTML.net** (or Word2CleanHTML)
- **How it works:**
 1. Copy your text from Word.
 2. Paste it into the visual editor on the left side of the site.

²[Google Gemini](#)

³Footnote

3. The site generates clean HTML code (stripped of unnecessary Microsoft tags) in the right-hand window.

- **Pros:** Very simple, visual, removes the "fat" from Microsoft code.
- **The downside:** Ideal for a single page, not for generating the architecture of an entire site.
- **3. The "Pro/Developer" Solution (Best Quality)**

This is the method recommended by experts for obtaining ultra-clean, long-lasting code.

- **Software: Pandoc** (Command line)
- **How it works:** It's a small program without a graphical interface (to be used via the terminal/command prompt). It converts formats very accurately.
- **Sample command:** `pandoc my_document.docx -t html -o my_site.html`
- **Pros:** Perfect HTML5 code, free, open-source, handles images and mathematical formulas (LaTeX) very well.
- **Cons:** Requires familiarity with a command prompt (black screen).
- **4. The "DIY" Solution (Microsoft Word)**

You can do this directly from Word, but be careful with the result.

- **Method:** File > Save As > Select "**Filtered Web Page (*.htm; *.html)**".
- **Important:** Do not select "Single-file Web Page (.mht)"; be sure to choose "Filtered." The "Filtered" option reduces (slightly) the unnecessary code that Word usually adds.
- **The downside:** The code is often still bulky and difficult to modify later.

- **Quick Comparison Table**

Solution	Result Type	Code Quality	Ease	Ideal for...
HelpNDoc	Full website (with menu)	Good	Average	Manuals, theses, documentation
WordToHTML	Single page	Excellent	Very Easy	Blog posts, simple pages
Pandoc	Single page (raw)	Perfect	Difficult	Developers, clean archiving
Word (Native)	Single page	Poor	Easy	Quick troubleshooting

- **My advice**

If your goal is to publish this document online professionally, I suggest **HelpNDoc** if you're a beginner and want an immediate site structure.

If you simply want clean content to paste elsewhere (WordPress, etc.), use **WordToHTML.net**.

Would you like me to walk you through the step-by-step process for one of these tools (e.g., WordToHTML)?

I've used [Pandoc] a few times, but I found it difficult to use. I remember that you had to "prepare" the Word document so that Pandoc could handle it properly. However, the result was excellent. **HelpNDoc** seems like an attractive option since it generates a site with a menu. I tested it on one of my Word documents, and the result

wasn't satisfactory.

Now let's see if there is any conversion software for ODT documents:

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

Gemini's response is as follows:

Yes, absolutely. As with Word, the **ODT** format (used by LibreOffice and OpenOffice) has its own solutions. Here are the best free options specific to ODT files, sorted by method:

- **1. The "Extension" Solution (Best Built-in Quality)**

If you use LibreOffice or OpenOffice, the basic HTML export is (as with Word) rather poor. However, there is a fantastic free extension designed specifically for this purpose.

- **Tool: Writer2xhtml** (Extension for LibreOffice)
- **How it works:**
 1. You download and install the extension in LibreOffice.
 2. Go to File > Export, and a new option "XHTML 1.1 + MathML" (or similar) will appear.
- **Pros:** The code is much cleaner than the standard export, and it handles mathematical formulas and complex formatting very well.
- **The downside:** You need to install an extension (a one-time technical step).
- **2. The "Pro / Command Line" solution (Universal)**

This is the same tool as for Word, as it also works well with the ODT format.

- **Software: Pandoc**
- **How it works:** Via the command line.
- **Command:** `pandoc my_file.odt -t html -o my_site.html`
- **Pros:** It is undoubtedly the most powerful converter on the market for ODT to HTML. It "understands" the structure of the ODT document better than any other.
- **3. The "Full Website with Navigation" Solution (Via HelpNDoc)**

HelpNDoc (mentioned earlier) is still the best free option for creating a real website with a left-hand menu. However, HelpNDoc doesn't always natively import the .odt format as well as the .docx format.

- **The trick:**
 1. Open your file in LibreOffice.
 2. Select "Save As..." and choose "**Word 2007-365 (.docx)**".
 3. Import this .docx file into HelpNDoc to generate your website.
- **Why?** This ensures that your chapter hierarchy (Heading 1, Heading 2) is correctly recognized to create the site menu.
- **4. The online solution (Quick cleanup)**

Some online tools directly accept ODT files and output clean HTML.

- **Tool: WordToHTML.net** (Also accepts ODT)
- **How it works:** You can upload your ODT file directly or paste your text.
- **Pros:** Very convenient for converting a single article without installing anything.
- **Summary of my advice for ODT:**
 1. **For a real website (documentation/thesis):** Convert your ODT to **DOCX** using LibreOffice, then use **HelpNDoc** (this is the most reliable method for getting a navigation menu).
 2. **For a simple, clean page:** Use the **Writer2xhtml** extension if you do this often, or **Pandoc** if you want to automate the task.

Would you like me to explain how to install the **Writer2xhtml** extension in LibreOffice?

I tested the LibreOffice [Writer2xhtml] extension on one of my documents. The result is readable but not professional enough. There's no table of contents on the left. The code blocks lose their syntax highlighting.

Ultimately, I think the converter generated by Gemini is an alternative to what's suggested above. The result is comparable to that produced by Pandoc if the ODT document is suitable for the converter.

4. The initial prompt to Gemini 3

We assume here that the reader has some prior knowledge of [Gemini AI](#). If not, they are advised to read document [\[ref1\]](#). The initial prompt given to Gemini was in the form of the following text file [Prompt User.txt]:

1. User requests
- 2.
3. - Create a Python script [convert] to generate a MkDocs site using the Material theme from an ODT document. The name of the ODT document will be the sole argument for convert.
- 4.
5. - Then create a Python script [build] that will use the [mkdocs.yml] file created by convert to generate a static HTML site
6. - Finally, the [build] script will display the site's homepage in a browser.
- 7.
8. - The site will have two independent columns. The left column will contain the table of contents, and the right column will contain the content of the ODT document.
- 9.
10. - The table of contents will include all the headings from the ODT document (Heading 1, Heading 2, Heading 3, etc.), their numbering, and their hierarchy
- 11.
12. - The images are mostly contained within tables and have been resized.
13. You will search through the tables for these images and ensure their resized dimensions are preserved.
- 14.
15. - There are code blocks in the ODT document. They can be identified by their styles [Numbered Source Code Results, Source Code]
16. Other attributes of the code paragraph should be ignored. For example, if the code paragraph is also a numbered list, you must ignore this attribute.
17. All these code blocks must be treated as code on the MkDocs site. The lines of code must be numbered in the HTML output.
- 18.
19. - For all code blocks, you will use Python syntax highlighting.
- 20.
21. - You must not alter the lines of code. In particular, you must not remove the indentations or spaces preceding the first character of the line of code.
- 22.
23. - You must identify all bulleted lists. ODT has several ways to create them. You must explore them all. Bulleted lists are
24. sometimes nested within one another. You must preserve this nesting.
- 25.

```
26. - The title of the site will be "Generate a Python script with AI tools"
27.
28. - You will embed every image extracted from the ODT document as well as every MkDocs chapter
    created
29.
30. - You must preserve the links found in the ODT document. They must remain as links in the
    generated HTML document.
31.
32. - You must preserve text that is bold, underlined, or italicized. It must be rendered exactly
    as is in the HTML document
33.
```

- Line 19: The initial test was performed on a previously published course that contained Python code;
- This prompt was used to start the chat. After a few dozen iterations, it became obsolete;

5. Set up a Python development environment

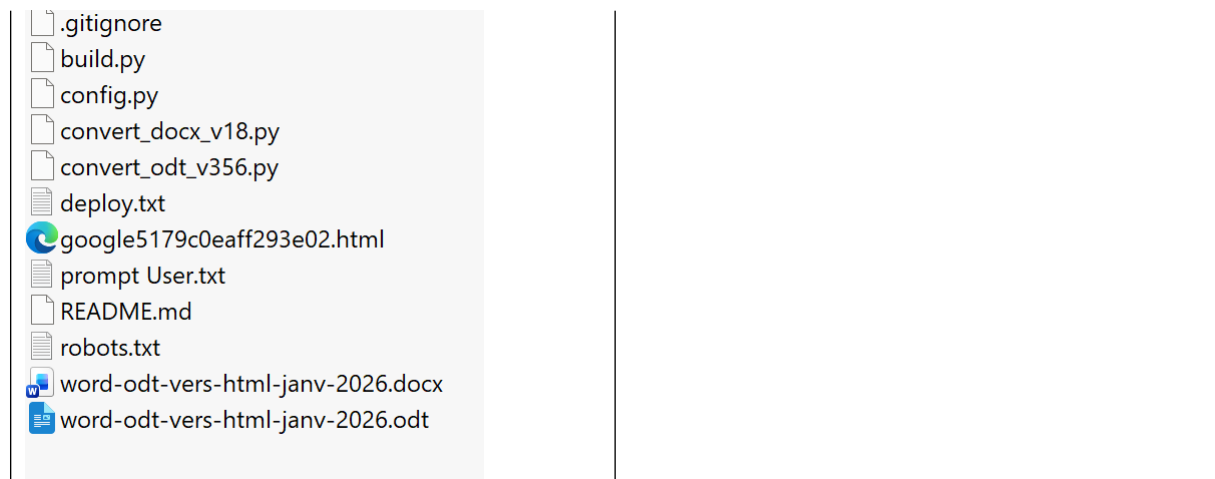
Gemini and ChatGPT will generate two Python scripts. We therefore need a Python environment:

- The installation of Python is described in [Section 3.1](#) of [\[ref1\]](#);
- The installation of PyCharm is described in [Section 3.1](#) of [\[ref1\]](#). [\[ref1\]](#) provides a detailed configuration guide for PyCharm. Here, PyCharm will only be used to run the Python scripts generated by Gemini and ChatGPT. Therefore, any configuration of PyCharm is unnecessary. PyCharm will also be used to display the files required by the converter;

6. The converter's working directory

You can download the working directory | [HERE](#) |.

The working directory is as follows:



- For now, we will ignore the following files [.gitignore, deploy.txt, google*.html, README.md, robots.txt, deploy.txt]. These are files that will be used to deploy the locally generated HTML site to GitHub;
- [prompt User.txt] is the prompt I used to start the iterations with Gemini;
- The file [word-odt-vers-html-janv-2026.odt] is the ODT document for this article. It is converted to HTML by the script [convert_odt_v356.py];
- The file [word-odt-to-html-jan-2026.docx] is the DOCX document for this same article. It is converted to HTML by the script [convert_docx_v18.py];
- The scripts [convert*.py] and [build.py] are the two Python scripts generated by the AI;

- [convert*] converts an ODT or DOCX document into a [MkDocs] site. A MkDocs site is a static site whose pages are written using [Markdown] syntax. [MkDocs] provides a server capable of displaying MkDocs sites;
- [build] converts the MkDocs site into a standard static HTML site. At the end of this conversion, it opens a browser to display the site's root directory;

At no point will we look at the generated Python code. We will treat both scripts as black boxes. At no point did I modify the code manually. I always asked Gemini / ChatGPT to correct its own script. That is why it is not necessary to know Python to use the converter.

- [config.py] is a configuration file for the [convert*] scripts. Initially, [convert*] did not have a configuration file. Then, as I switched between documents to convert, I gradually realized that there were elements I was asking the AI to generate (for example, the site name, which changes for each document) that would be better placed in a configuration file that the user could modify themselves. I built this file as well through iterations with the AI;

7. The converters' configuration files

- 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

The configuration file [config.py] collects the information that may vary from one document to another. It uses six other configuration files. These files contain the settings most likely to change when switching between documents for conversion.

The [config.py] file imports the contents of the six other configuration files:

```
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

This file contains the three URLs from the [config.py] configuration. These change with each document.

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

- line 2: the URL of the website where you will deploy the HTML site generated by the converter;
- line 3: the URL of your Git repository (see section 12);
- line 4: the website of the site creator. This URL is placed at the bottom of the website's pages. May be empty.

config texts

This file contains the texts from [config.py] that may change when the language of the generated HTML site is changed.

```

1. TEXTS = {
2.   "home_label": "Home",
3.   "site_name": "Convert a Word or ODT document to a MkDocs-compatible static HTML site using
   Gemini 3 and ChatGPT 5.2 AI",
4.   "site_description": "Convert a Word or ODT document to a MkDocs-compatible static HTML site
   with Gemini 3 and ChatGPT 5.2 AI",
5.   "toggle_to_dark": "Switch to dark mode",
6.   "toggle_to_light": "Switch to light mode",
7.   "footer_license_sentence": (
8.     "This tutorial written by <strong>Serge Tahé</strong> is made available to the public under
   the terms of the\n"
9.     "    <em>Creative Commons Attribution-NonCommercial-\n"
10.    "    ShareAlike 3.0 Unported</em>.\n"
11.   ),
12.   "copy_label": "Copy",
13.   "copy_copied_label": "Copied",
14. }
```

- line 2: the name of the first link in the table of contents. Its content includes everything in the document before the first Level 1 heading (Heading 1). This is the beginning of the document;
- line 3: the document title. It is displayed in the top banner of the generated website;
- line 4: also the document title, this time intended for search engines;
- lines 5–6: the tooltips displayed when you hover over the icon in the top bar of the site to switch from light mode to dark mode or vice versa;
- lines 7–11: the site's license. It is displayed in the footer of the generated site;
- lines 12–13: the code displayed on the site, accompanied by a button that allows you to copy the code to the clipboard. Line 12 is the button label before copying, line 13 is the label after the code has been copied;

Here is what this file looks like for an English version of the document:

```

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": "Convert a Word or ODT document into a static HTML website using
   Gemini 3 and ChatGPT 5.2 AI",
5.   "toggle_to_dark": "Switch to dark mode",
6.   "toggle_to_light": "Switch to light mode",
7.   "footer_license_sentence": (
8.     'This tutorial written by <strong>Serge Tahé</strong> is made available to the public
   under the terms of the\n'
9.     '<em>Creative Commons Attribution - Non-Commercial -\n'
10.    '<em>ShareAlike 3.0 Unported</em>.'
11.   ),
12.   "copy_label": "Copy",
13.   "copy_copied_label": "Copied",
14. }
```

config styles

This file is used to specify the style name for the document title. For a DOCX document, this is usually "Title," but for an ODT document, it varies by document.

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

- Line 2: [styles_names] is an array. You can put the names of the styles that the document title can have in it;
- line 3: the document [word-odt-to-html-jan-2026.odt] has a title with the P1 style;

config code

Code blocks in the DOCX or ODT document can be found in two forms:

- rich text (colors, bold, italics, etc.). In this case, it is copied as-is into the HTML page;
- plain text. In this case, instructions are provided to the converters so they can identify the language used in the code block themselves. To do this, keywords are provided:

```
1. CODE = {
2.   # Automatic language detection rules based on content
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", "getDoctor", "@Entity", "@Autowired", "@Bean",
13.      "Serializable", "getClient", "getTimeSlot", "getAppointment", "PostAddAppointment",
14.      "PostDeleteAppointment", "@EnableJpaRepositories", "@Component",
15.      "getDoctorScheduleToday",
16.      "getResponse", "getMessagesForException", "getBase64", "addAppointment",
17.      "Response", "getPartialViewAgenda", "setModelForAgenda", "ActionContext",
18.      "getActionContext", "PostLang", "PostUser", "PostGetAgenda", "@NotNull",
19.      "@EnableAutoConfiguration", "HttpSecurity"
20.    ],
21.    "html": [
22.      "<html>", "</div>", "<body>", "<script>", "href=", "<span>", "<p>",
23.      "<h2>", "<form", "<table", "<input"
24.    ],
25.    "sql": [
26.      "SELECT", "INSERT INTO", "UPDATE", "DELETE FROM", "WHERE",
27.      "CREATE TABLE", "ALTER TABLE"
28.    ],
29.    "python": [
30.      "def", "import", "print(", "from"
31.    ],
32.    "xml": [
33.      "<?xml", "<project", "<version>", "<configuration>", "<build>",
34.      "<dependency>", "<properties>", "<configuration>", "<start-class>"
35.    ],
36.    "javascript": [
37.      "use strict", "console.log", "let", "constructor", "async", "export"
38.    ],
39.    "php": [
40.      "<?php", "declare", "require"
41.    ],
42.    "vbscript": [
43.      "Option", "Dim", "Explicit"
44.    ],
45.    "markdown": [
46.      "# ", "## ", "### ", "****", "___", "![", "]" # Typical Markdown markers
```

```
47.   },
48. }
```

Once again, these detection rules apply only to plain text code blocks and not to rich text code blocks. If you find that code blocks on your site haven't been syntax-highlighted, this is where you should look.

For each language, you specify the strings that identify that language. Some keywords may appear in multiple languages. The converter selects the language for which it found the most keywords. The process is simple: review your code, select characteristic keywords, and enter them in these fields for the appropriate language. Note that you cannot use just any name for the language: you must use the names recognized by MkDocs;

config presentation

This file sets the appearance of various elements of the generated website. This is not something you'll need to change often:

```
1. PRESENTATION = {
2.   # -----
3.   # Document title
4.   # -----
5.   "document_title": {
6.     "font_size": "28px",
7.     "font_weight": "bold",
8.     "margin_bottom": "1em",
9.     "line_height": "1.2"
10.  },
11.
12.  # -----
13.  # Rich code blocks
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.  # Copy button
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.      "line-height: 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: 0.85;"
45.  },
46.
47.  # -----
48.  # Images
49.  # -----
50.  "images": {
51.    "shadow": {
52.      "enabled": True,
```

```

53.         "border_radius": "8px"
54.     }
55. },
56.
57. # -----
58. # Frame / header border
59. # -----
60. "frame": {
61.     "header_top_border_width": "4px"
62. }
63. }

```

If you wish, you can modify the appearance:

- lines 5–10: the document title;
- lines 15–19: the rich text lines;
- lines 24–45: the [Copy] button, which allows you to copy code to the clipboard;
- lines 50–55: the shadows of the images;
- lines 60–62: the border around the top bar of the site's pages;

config_colors

This is the file where you control your site's colors, primarily the background color of the top banner on the pages:

```

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.
19.     "frame": {
20.         # "header_bg_light": "#1E88E5", (blue)
21.         "header_bg_light": "",
22.         "header_bg_dark": "",
23.         "header_top_border_light": "",
24.         "header_top_border_dark": ""
25.     },
26.
27.     "document_title": {
28.         "color": "#2c3e50"
29.     },
30.
31.     "copy_button": {
32.         "border": "rgba(0,150,136,.45)",
33.         "background": "rgba(0,150,136,.12)",
34.         "text": "rgb(0,150,136)",
35.         "background_hover": "rgba(0,150,136,.20)"
36.     },
37.
38.     "images": {
39.         "shadow": {
40.             "zoomable": "0 8px 24px rgba(0,0,0,.28), 0 18px 60px rgba(0,0,0,.20)",
41.             "zoomable_hover": "0 12px 30px rgba(0,0,0,.32), 0 26px 80px rgba(0,0,0,.22)",
42.             "lightbox": "0 24px 80px rgba(0,0,0,.65)"
43.         }

```

```
44.     }
45. }
```

Here, you need to know a little bit about CSS, which I don't. I only used the colors from lines 7 and 13, which set the color of the top banner.

config files to copy

This file lists the files that must be copied to the root of the generated website:

```
1. FILES_TO_COPY =[
2.     "google5179c0eaff293e02.html",
3.     "robots.txt",
4.     "word-odt-to-html-jan-2026.pdf",
5.     "word-odt-to-html-jan-2026.zip"
6. ]
```

- lines 2-3: are required for Google to track the generated site (see section 13);
- lines 4-5: the files you want to copy to the root of your website to make them available to your visitors;

config

The [config.py] file is the configuration file used by both converters. It contains all the information they need. This is a file that should normally never be modified. It is complex, which is why we decided to extract the configuration parameters that are likely to change and place them in external files.

```
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.                 + STYLE["copy_button"]["btn_hover"]
139.             ),
140.             "btn_copied": STYLE["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. }

```

- lines 1-7: all configuration files are imported;
- line 9: the configuration file is a Python script. It defines a single variable [config]. This is a dictionary that will contain all the configuration values;
- line 11: the label for the "Home" link;
- line 14: the [mkdocs] dictionary configures the [mkdocs.yml] file generated by the Gemini/ChatGPT converter. This file is then used by the [build] script to build an HTML site from the MkDocs site generated by the converter;
- lines 15–20: configure the GitHub site that will host the HTML site created from the ODT/DOCX document (see section12)

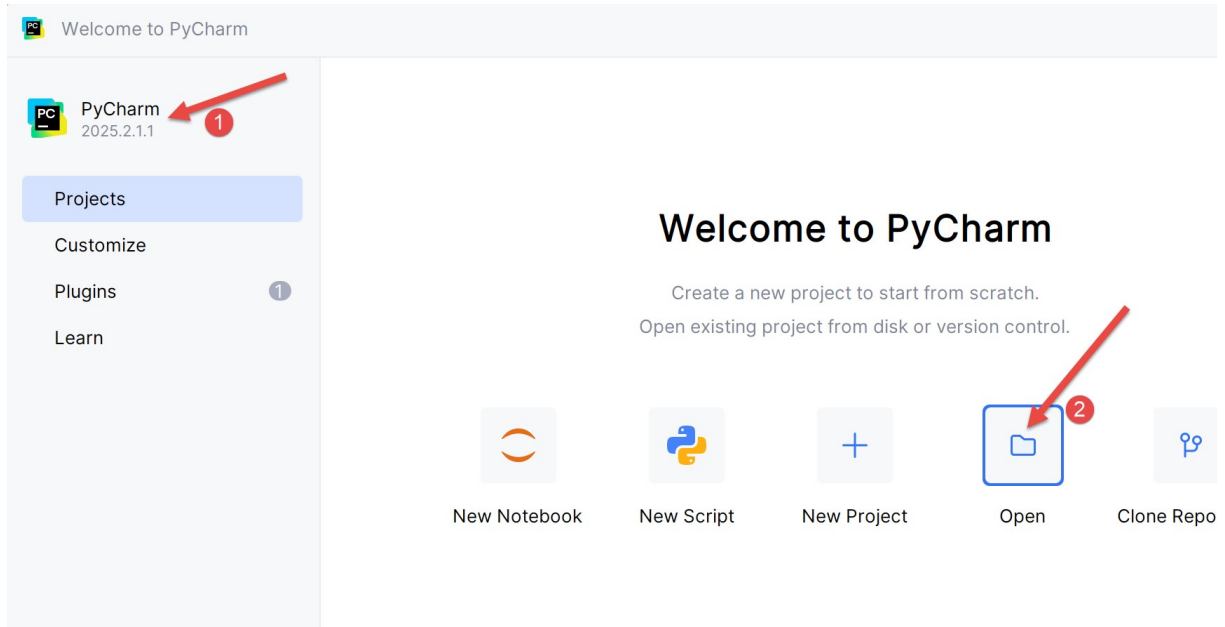
- line 21: this line is important. If it is missing, instead of displaying an HTML page, the browser will open the folder for that page;
- line 24: MkDocs offers several themes for a MkDocs/HTML site. Here we have chosen the “material” theme, line 24;
- lines 27–31: configuration of site navigation. This will be done using a table of contents located in the left column of the displayed page (line 30);
- lines 33–54: define two color palettes: light mode (lines 35–42) or dark mode (lines 45–52). An icon in the top bar of the displayed pages allows you to switch between them;
- lines 57–70: extensions to the Markdown language used by MkDocs. These extensions were generated by Gemini following some of my requests;
- line 73: the [focus.js] script is a JavaScript script generated by Gemini. It is associated with the button in the site’s top bar to show or hide the table of contents;
- line 76: the CSS used by this button;
- lines 80–95: the definition of the HTML site’s footer. It was generated by Gemini based on a sample text I provided;
- lines 97–100: definition of the Google Analytics (GA) tag, which will track site visits;
- line 99: insert your GA tag here;
- lines 103–112: the style of the document title, the one present in the ODT/DOCX document before the first level-1 heading. The style specified on line 104 changes with each ODT document. However, it can be constant (“Title”) for DOCX documents. By default, the converter logs the styles of all paragraphs in the ODT/DOCX document that precede the first level-1 heading. You must therefore run the script once, identify the style of the paragraph you want to use as the title, and then add it to the [config_styles] file;
- lines 105–111: the CSS style you want to apply to the document title. This title appears on the “Home” page, the first page displayed when the site opens;
- line 115: enter case-insensitive keywords here (upper/lowercase);
- line 116: code blocks are identified by their styles in the ODT/DOCX document. There may be several. On line 116, list all the keywords that allow a code style to be detected. In my documents, all my code styles have the word “code” in their names. And no other styles have this word in their names. So I can simply use a single keyword. If there are multiple, list them separated by commas on line 116;
- line 117: sets the default language. In the case of a “plain text” code block, if no language is detected, the default language will be “text.” For MkDocs, this means a code block without syntax highlighting. This is the case, for example, with execution results;
- line 121: this configuration is used for “plain text” code that does not initially have syntax highlighting. These lines are intended to apply highlighting based on the language used in the code block. **Note that if all your code is formatted because it comes from an IDE such as Eclipse, Visual Studio, etc., and you have no “plain text” code blocks associated with a language, then you have nothing to enter in this part of the configuration. You can leave the tables associated with the languages empty.** All of this is done in the code configuration file [config_code];
- lines 118–120: these lines concern formatted code blocks (bold, italics, underlining, highlighting, character color). These blocks are not processed in the same way as “plain text” code blocks. They are rendered exactly as they appear in the HTML;
 - line 118: sets the line height for code lines;
 - line 119: sets the CSS font for the code block;
 - line 120: sets the font size;
- lines 144–152: set the appearance of images;
- lines 154–160: set the appearance of the frame in which the web page is displayed;
- Line 162: the list of files to copy to the root directory of the HTML site;
- line 164: setting [debug] to True enables logging of the styles of the paragraphs preceding the first Level 1 heading in the ODT/DOCX document. These logs will reveal the exact style of the paragraph to be used as the title on the site’s home page;

Ultimately, when switching from one ODT/DOCX document to another, you will not need to change anything in the [config.py] file. You will only need to modify the other configuration files.

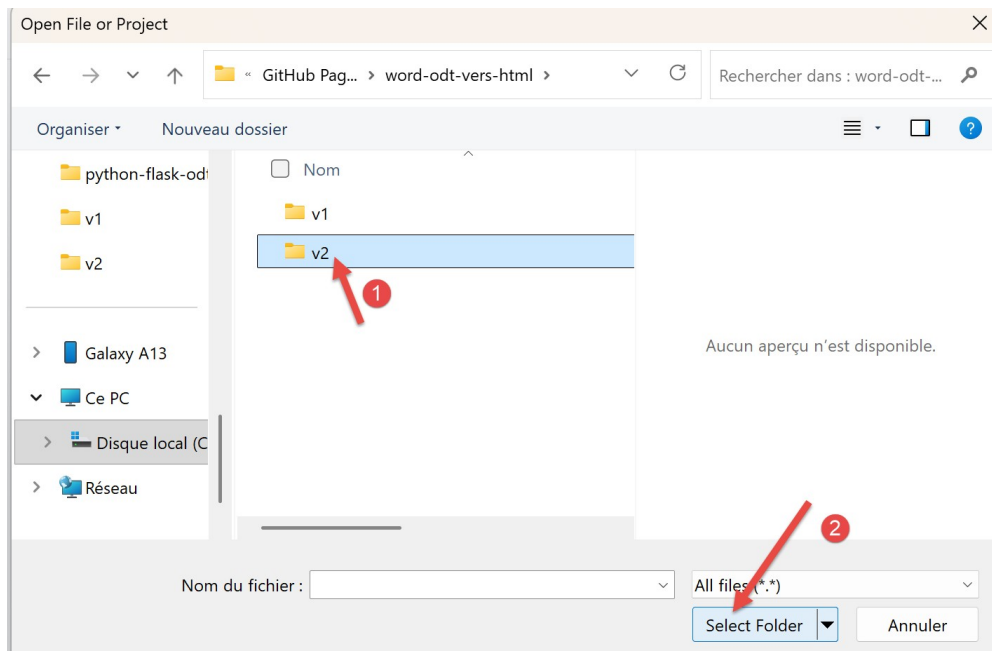
8. Using the ODT → HTML Converter

Remember that the working directory can be found |[HERE](#)|.

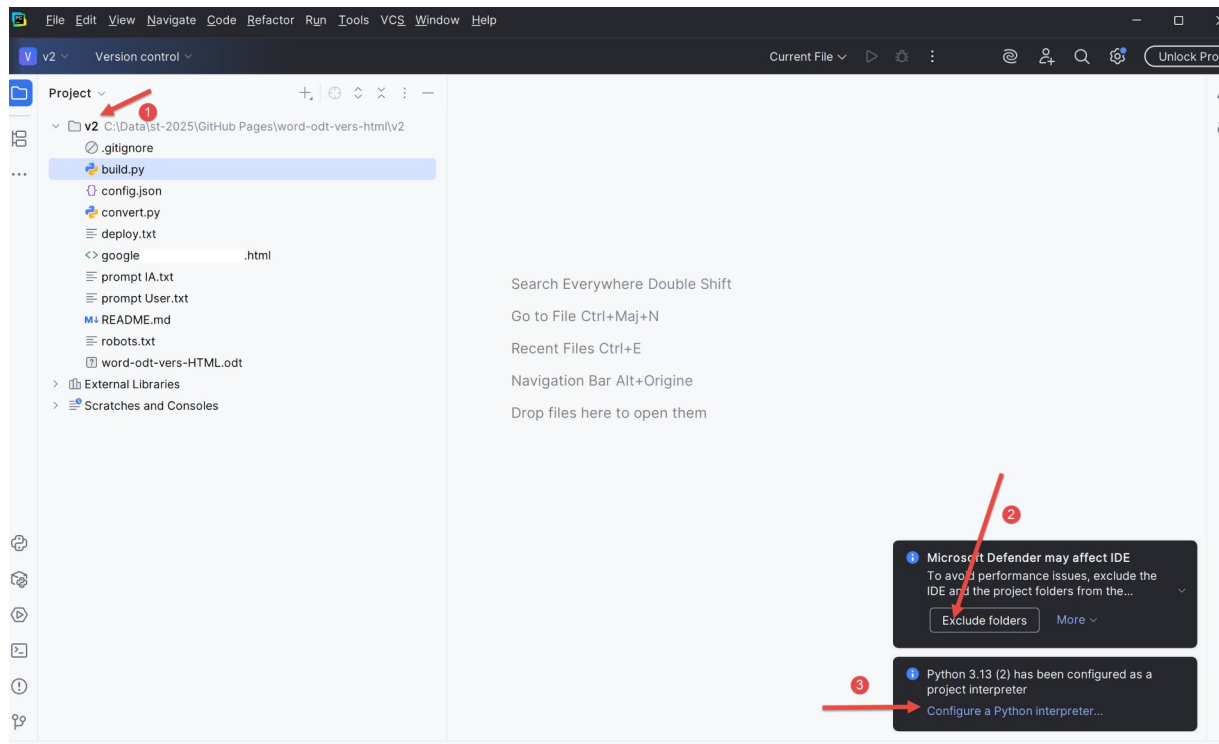
Launch PyCharm:



- In [2], open a new project;

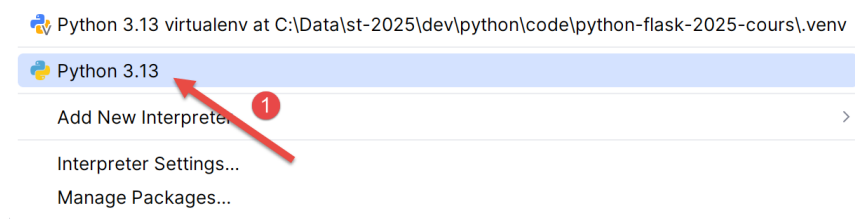


- In [1], select the folder where you saved the working directory;
- In [2], confirm your selection;



- In [1], select your Python project;
- In [2], confirm to avoid being bothered by the Windows security system;
- In [3], choose a Python interpreter;

Python Interpreter



- In [1], select the Python interpreter you installed in the section5 ;

The Gemini/ChatGPT converter uses Python dependencies that you need to install. We'll ask Gemini which ones:

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

Gemini's response is as follows:

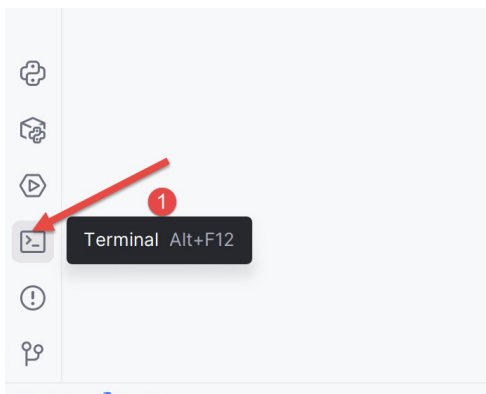
Commande d'installation rapide

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

```
Bash
```

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

We'll install these dependencies:



- In [1], open a terminal;



- In [2], type the command provided by Gemini;

The response is as follows:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> pip install odfpy unicode mkdocs
mkdocs-material
2. Defaulting to user installation because normal site-packages is not writable
3. Requirement already satisfied: odfpy in c:\users\serge\appdata\roaming\python\python313\site-
packages (1.4.1)
4. Requirement already satisfied: unicode in c:
\users\serge\appdata\roaming\python\python313\site-packages (1.4.0)
```

5. Requirement already satisfied: mkdocs in c:\users\serge\appdata\roaming\python\python313\site-packages (1.6.1)
6. Requirement already satisfied: mkdocs-material in c:\users\serge\appdata\roaming\python\python313\site-packages (9.7.0)
7. Requirement already satisfied: defusedxml in c:\users\serge\appdata\roaming\python\python313\site-packages (from odfpy) (0.7.1)
8. Requirement already satisfied: click>=7.0 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (8.3.1)
9. Requirement already satisfied: colorama>=0.4 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.4.6)
10. Requirement already satisfied: ghp-import>=1.0 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (2.1.0)
11. Requirement already satisfied: jinja2>=2.11.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.1.6)
12. Requirement already satisfied: markdown>=3.3.6 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.10)
13. Requirement already satisfied: markupsafe>=2.0.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (3.0.2)
14. Requirement already satisfied: mergedeep>=1.3.4 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (1.3.4)
15. Requirement already satisfied: mkdocs-get-deps>=0.2.0 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.2.0)
16. Requirement already satisfied: packaging>=20.5 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (25.0)
17. Requirement already satisfied: pathspec>=0.11.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (0.12.1)
18. Requirement already satisfied: pyyaml-env-tag>=0.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (1.1)
19. Requirement already satisfied: pyyaml>=5.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (6.0.2)
20. Requirement already satisfied: watchdog>=2.0 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs) (6.0.0)
21. Requirement already satisfied: babel>=2.10 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.17.0)
22. Requirement already satisfied: backrefs>=5.7.post1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (6.1)
23. Requirement already satisfied: mkdocs-material-extensions>=1.3 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (1.3.1)
24. Requirement already satisfied: paginate>=0.5 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (0.5.7)
25. Requirement already satisfied: pygments>=2.16 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.19.2)
26. Requirement already satisfied: pymdown-extensions>=10.2 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (10.17.2)
27. Requirement already satisfied: requests>=2.26 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-material) (2.32.5)
28. Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from ghp-import>=1.0->mkdocs) (2.9.0.post0)
29. Requirement already satisfied: platformdirs>=2.2.0 in c:\users\serge\appdata\roaming\python\python313\site-packages (from mkdocs-get-deps>=0.2.0->mkdocs) (4.5.0)
30. Requirement already satisfied: six>=1.5 in c:\users\serge\appdata\roaming\python\python313\site-packages (from python-dateutil>=2.8.1->ghp-import>=1.0->mkdocs) (1.17.0)
31. Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (3.4.3)
32. Requirement already satisfied: idna<4,>=2.5 in c:\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (3.10)
33. Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (2.5.0)
34. Requirement already satisfied: certifi>=2017.4.17 in c:\users\serge\appdata\roaming\python\python313\site-packages (from requests>=2.26->mkdocs-material) (2025.8.3)
35. PS C:\Data\st-2025\GitHub Pages\word-odt-to-html\v2>

On my PC, everything was already installed. If that's not the case for you, all the required dependencies will be installed.

Now we can use the converter. In the open terminal, type the following command:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-to-html\v2> python .\convert_odt_v356.py .\word-odt-
to-html-jan-2026.odt .\config.py
2. --- ODT to MkDocs Converter V356 ---
3. Processing .\word-odt-to-html-jan-2026.odt...
4. Copied: google5179c0eaff293e02.html
5. Copied: robots.txt
6. Copied: word-odt-to-html-jan-2026.pdf
7. Copied: word-odt-to-html-jan-2026.zip
8. Scanning references...
9. Generating Markdown...
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='P4' Text='Convert a Word or ODT document to a sta...'
22. >>> DOCUMENT TITLE FOUND: Convert a Word or ODT document to a MkDocs-compatible static HTML
site using Gemini 3 and ChatGPT 5.2
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, January 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='This site was created with the [Word or ...'
34. [DEBUG PRE-H1] Tag=text:h Style='P5' Text='Introduction...'
35. >>> CHAPTER 1: 1 Introduction
36. >>> CHAPTER 1: 2 Examples in this document
37. >>> CHAPTER 2: 2.1 Lists
38. >>> CHAPTER 3: 2.1.1 Bulleted lists
39. >>> CHAPTER 3: 2.1.2 Numbered lists
40. >>> CHAPTER 3: 2.1.3 Mixed lists 1
41. >>> CHAPTER 3: 2.1.4 Mixed lists 2
42. >>> CHAPTER 2: 2.2 Code Blocks
43. >>> CHAPTER 3: 2.2.1 Rich text code blocks (Eclipse, Visual Studio, ...)
44. >>> CHAPTER 3: 2.2.2 Plain text code blocks
45. >>> CHAPTER 2: 2.3 Links
46. >>> CHAPTER 2: 2.4 Text Formatting
47. >>> CHAPTER 2: 2.5 A title can also be formatted.
48. >>> CHAPTER 2: 2.6 Images
49. >>> CHAPTER 2: 2.7 Special characters
50. >>> CHAPTER 2: 2.8 Tables
51. >>> CHAPTER 2: 2.9 Footnotes
52. >>> CHAPTER 1: 3 What's on the Internet
53. >>> CHAPTER 1: 4 The Initial Prompt for Gemini 3
54. >>> CHAPTER 1: 5 Setting up a Python development environment
55. >>> CHAPTER 1: 6 The converter's working directory
56. >>> CHAPTER 1: 7 The converter configuration file
57. >>> CHAPTER 1: 8 Using the ODT → HTML converter
58. >>> CHAPTER 1: 9 Using the DOCX → HTML converter
59. >>> CHAPTER 1: 10 Building the Static HTML Site
```

```

60. >>> CHAPTER 1: 11 Reviewing the Generated HTML Site
61. >>> CHAPTER 2: 11.1 The Site's Top Bar
62. >>> CHAPTER 2: 11.2 The Site Footer
63. >>> CHAPTER 2: 11.3 The Home Page
64. >>> CHAPTER 2: 11.4 Bulleted Lists
65. >>> CHAPTER 2: 11.5 Numbered lists
66. >>> CHAPTER 3: 11.5.1 Mixed lists 1
67. >>> CHAPTER 3: 11.5.2 Mixed Lists 2
68. >>> CHAPTER 2: 11.6 Rich Code Blocks
69. >>> CHAPTER 3: 11.6.1 Example 1
70. >>> CHAPTER 3: 11.6.2 Example 2
71. >>> CHAPTER 3: 11.6.3 Example 3
72. >>> CHAPTER 2: 11.7 Plain Text Code Blocks
73. >>> CHAPTER 3: 11.7.1 Example 1
74. >>> CHAPTER 3: 11.7.2 Example 2
75. >>> CHAPTER 3: 11.7.3 Example 3
76. >>> CHAPTER 2: 11.8 Other code blocks
77. >>> CHAPTER 2: 11.9 Links
78. >>> CHAPTER 2: 11.10 Text Formatting
79. >>> CHAPTER 2: 11.11 Images
80. >>> CHAPTER 2: 11.12 Special characters
81. >>> CHAPTER 2: 11.13 Tables
82. >>> CHAPTER 2: 11.14 Footnotes
83. >>> CHAPTER 2: 11.15 Known Issues
84. >>> CHAPTER 2: 11.16 Other Cases
85. >>> CHAPTER 1: 12 Hosting the HTML Site on GitHub
86. >>> CHAPTER 1: 13 Tracking the Site with Google Analytics and Google Search Console
87. >>> CHAPTER 1: 14 Conclusion
88. Done.

```

- Line 1: the command that converts the ODT document into a MkDocs site [`python convert_odt_v356.py .\word-odt-to-html-jan-2026.odt .\config.py`]. Adjust the version number (here 356) to the version you downloaded. The first parameter of the converter is the ODT document to be converted, the second is the converter's configuration file;
- lines 4–7: the files that the converter copies to the root of the MkDocs site it creates;
- lines 10–34: debugging the styles of the paragraphs preceding the first Level 1 heading. These paragraphs will form the home page. One of the paragraphs serves as the title of the home page and thus of the site. This is the paragraph on line 21. Note its style, P1. You must include this style in the configuration file:

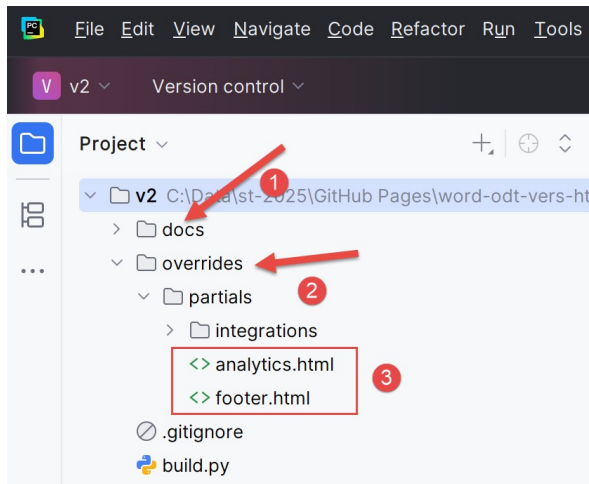
```

1.     STYLES = {
2.         "style_names": [
3.             "p4"
4.         ]
5.     }

```

- lines 35-88: I asked the AI to log all the chapters it encountered;

This execution has modified your working directory:



- In [1], [docs] is the MkDocs site that the Gemini/ChatGPT converter created. You might be curious to visit it;
- in [2], an [overrides] folder has been created. It will be used by the [build] tool for the HTML site;
- in [3]: [analytics.html] will be used for tracking the site via Google Analytics. [footer.html] is the footer you defined in the [config.py] file;

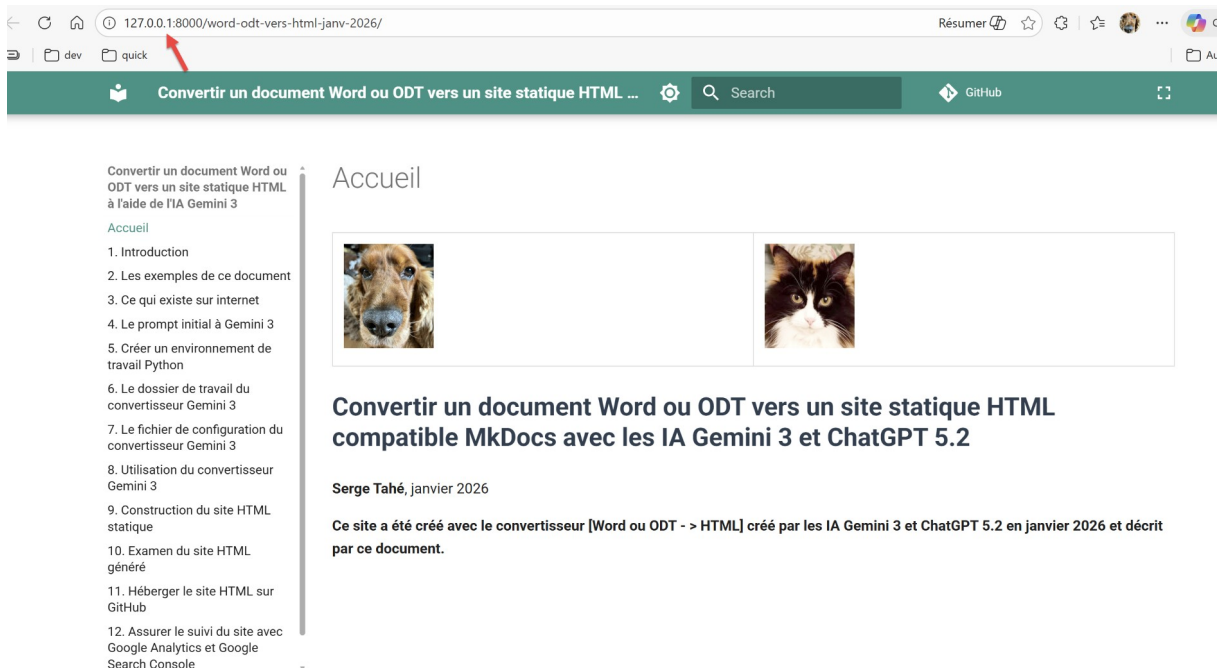
We could use the MkDocs site right now. The command [python -m mkdocs serve] lets you view it. You can try:

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2. INFO - Building documentation...
3. INFO - Cleaning site directory
4. INFO - Doc file 'les-exemples.md' contains a link '#_Les_exemples', but there is no such anchor on this page.
5. INFO - Documentation built in 0.41 seconds
6. INFO - [15:46:06] Serving on http://127.0.0.1:8000/word-odt-vers-html-janv-2026/

```

Ctrl-click on the link on line 6. The MkDocs site should appear:

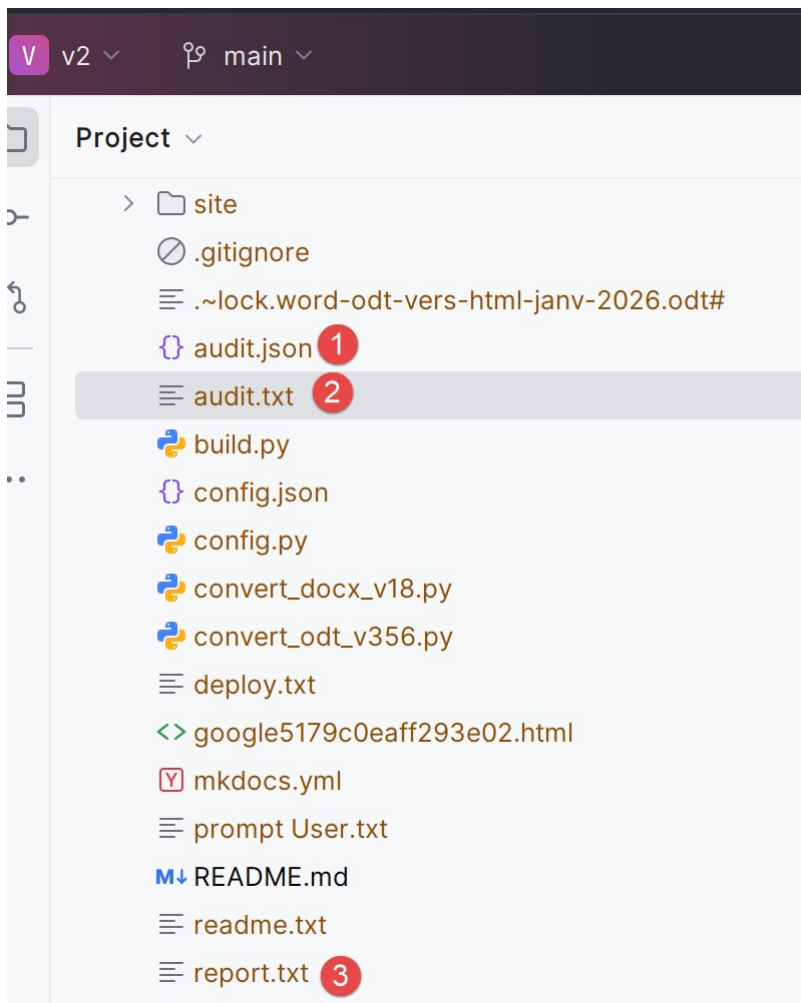



```

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>This site was created using the &#91;Word or ODT - &gt; HTML&#93; converter cr...'
32. [DEBUG PRE-H1] style=Title1 heading=1 rebase=0 numId=1 ilvl=0 list=numPr:1/ordered
txt='<span>Introduction</span>...'
33. Done. (audit.json, audit.txt, report.txt generated)

```

- line 1: the command is as follows: `[python .\convert_docx_v18.py .\word-odt-to-html-jan-2026.docx .\config.py]` (Adjust the version number (here 18) to the version you downloaded.):
 - the first parameter `[\convert_docx_v18.py]` is the DOCX → HTML converter
 - the second parameter `[\word-odt-to-html-jan-2026.docx]` is the name of the DOCX document to be converted;
 - the third parameter `[\config.py]` is the configuration file;
- Line 33: The converter reports that three files have been generated:



The [audit.txt] file is as follows:

```
1. Version: V16
2. Paragraphs: 2029
3. Tables: 97
4. Images (blips): 2
5. Headings detected (raw): 53
6. Minimum heading level detected (raw): 1
7. Rebase offset applied: 0
8.
9. Top paragraph styles:
10. - SourceCodenumrot: 1054
11. - StandardWW: 594
12. - Standard: 146
13. - Paragraphedeliste: 113
14. - SourceCodenumRotResults: 33
15. - new code: 28
16. - Title2: 25
17. - Heading 1: 14
18. - Title 3: 14
19. - StandardWWW: 6
20. - Plain text: 1
21. - Heading: 1
22.
23. List paragraphs:
24. - with numbering: 1329
25. - by style fallback: 49
26. - Not recognized: 0
```

- line 2: the number of paragraphs in the Word document;
- line 3: the number of tables;
- lines 9–21: the styles found in the document;
 - lines 10, 14, 15: the style of the code blocks. A single style would probably have sufficed;
 - lines 11–12, 19: the style of standard paragraphs. A single style would likely have sufficed;
 - lines 16–18, 21: the styles of the document's headings. On line 21, we see that only one paragraph has the 'Heading' style. This is the document title that precedes the first 'Heading 1';

This audit of the Word document is a good way to assess the document's quality. Here I can see that I've used too many different styles for the same thing in my Word document.

The [audit.json] file is identical to the [audit.txt] file but in JSON format:

```
1. {
2.   "version": "V16",
3.   "file": "word-odt-to-html-jan-2026.docx",
4.   "counts": {
5.     "paragraphs": 2029,
6.     "tables": 97,
7.     "image_blips": 2,
8.     "headings_raw": 53
9.   },
10.  "lists": {
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.     "ListParagraph",
34.     113
35.   ],
36.   [
37.     "SourceCodeNumberResults",
38.     33
39.   ],
40.   [
41.     "newcode",
42.     28
43.   ],
44.   [
45.     "Title2",
46.     25
47.   ],
48.   [
49.     "Title1",
50.     14
51.   ],
52.   [
53.     "Title3",
54.     14
55.   ],
56.   [
57.     "StandardWWW",
58.     6
59.   ],
60.   [
61.     "PlainText",
62.     1
63.   ],
64.   [
65.     "Title",
66.     1
67.   ]
68. ]
69. }

```

The [report.txt] file is as follows:

```

1. [SUMMARY] Lists detected via "by style" fallback (aggregated)
2.   - Paragraphedeliste -> level=1 type=unordered: 49
3.
4. [SUMMARY] Ignored Word blocks (aggregated)

```

```
5. - <w:sectPr>: 1
```

I didn't understand that...

It is possible to request only an audit of the Word document to assess its quality using the [--audit] parameter:

```
python .\convert_docx_v18.py .\word-odt-to-html-jan-2026.docx .\config.py --audit
```

In this case, only the document audit is performed. The MkDocs site is not generated.

As shown earlier, you can view the MkDocs site generated by the converter:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-to-html\v2> python -m mkdocs serve
2. INFO - Building documentation...
3. INFO - Cleaning site directory
4. INFO - Documentation built in 0.59 seconds
5. INFO - [06:05:48] Serving on http://127.0.0.1:8000/word-odt-vers-html-janv-2026/
```

Ctrl-Click on the URL in line 5:



10. Building the static HTML site

From now on, there is no difference between ODT and DOCX documents. We work on the MkDocs site built by either converter.

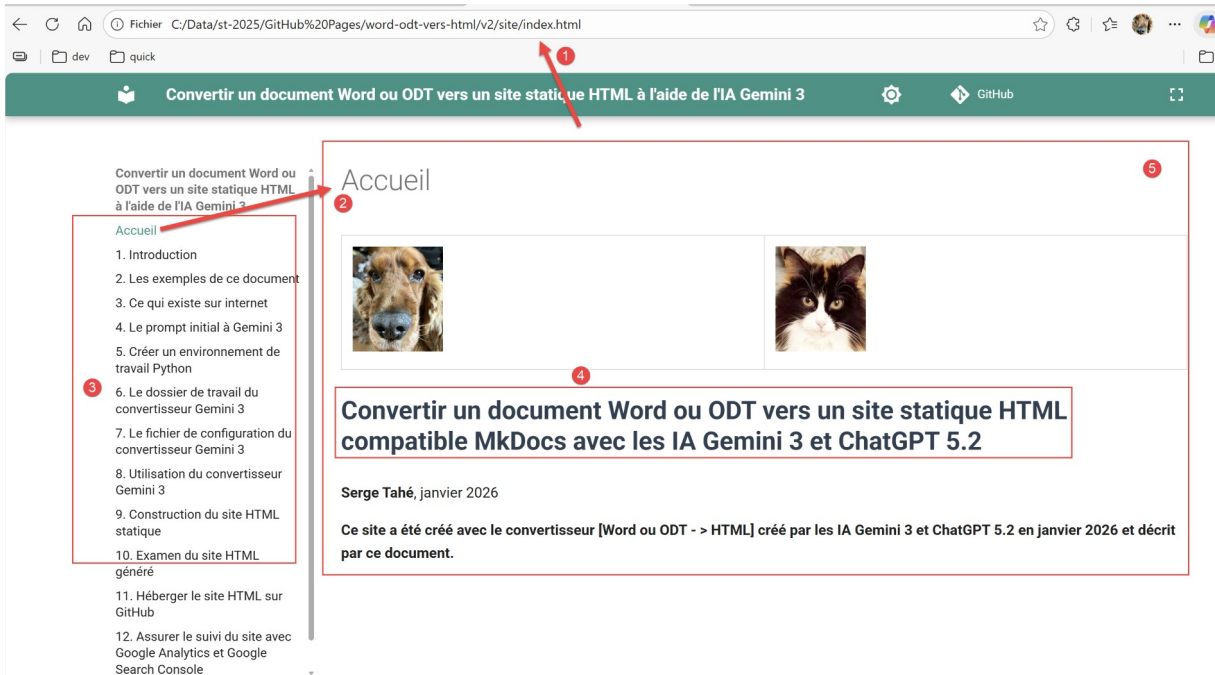
Now let's build the HTML site. This is done using the [build] script. Still in the terminal, type the following command:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-to-html\v2> python .\build.py
2. Starting to build the MkDocs site...
3. Build successful! The site is located at: C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site
4. Opening C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site\index.html in the browser...
```

- line 1: the [python build] command creates the HTML site from the MkDocs site;

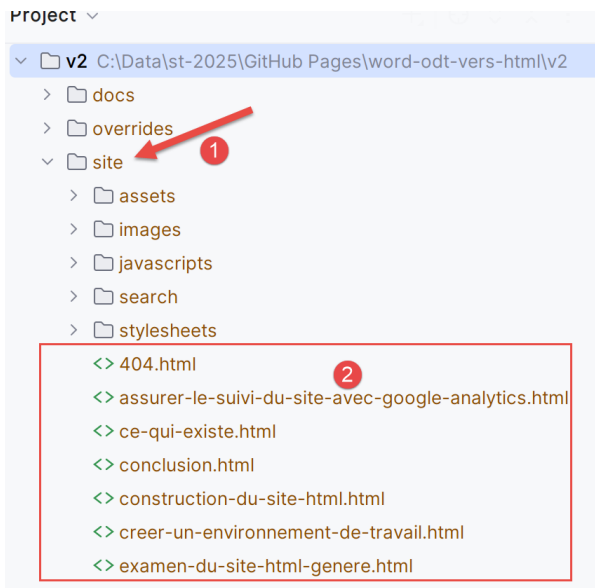
- line 4: the site is displayed in a browser;

The result is the same as for the MkDocs site:



- In [1], we can see that an HTML page is being displayed;
- In [2], this page corresponds to the site's home page;
- In [3], the table of contents for the HTML document;
- In [4], the document title has been formatted according to its configuration in [config.py];
- In [5], the first page displayed is [Home]. The content of this page is the content of the ODT/DOCX document **preceding the first Chapter 1-level chapter**;

Running [build] has modified your working directory:



- In [1], a new [site] folder has been created. It contains your static HTML site;
- In [2], the pages of your site;

11. Reviewing the generated HTML site

We will now examine the HTML output of this ODT/DOCX document. We have already seen that the converter preserves the table of contents.

11.1. The top bar of the site

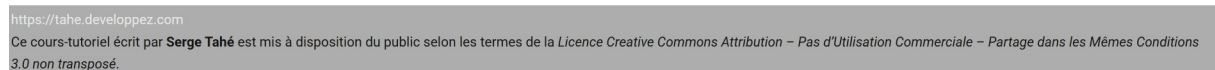
Let's look at the site's top bar:



- In [1], the site name defined in [config.py];
- At [2], the icon that lets you switch between dark and light modes;
- At [3], the icon that links to the GitHub repository where the HTML site will be exported. Also defined in [config.py];
- At [4], the icon that lets you hide or show the table of contents;

11.2. The site's footer

Now let's look at the footer:



This is the footer defined in the [config.py] file.

11.3. The home page

The title page of the ODT/DOCX document was as follows:



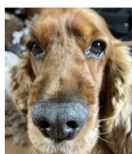
Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec les IA Gemini 3 et ChatGPT 5.2

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.

This title page of the ODT/DOCX document becomes the HTML site's home page:

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.

The Gemini 3 / ChatGPT converter places everything above—in the ODT / DOCX document, the first Level 1 heading, i.e., the “Title 1” style—on the Home page. If you include images as shown above, it will display them. You can therefore imagine giving your site a cover, just like a real book. In [1], this is the document’s main title. Its rendering is controlled by the following lines in the configuration file [config.py]:

```

1. # -----
2. # Document Title Detection
3. # -----
4. "document_title": {
5.     # ODT styles to be considered as the main title of the document (global H1)
6.     "style_names": [
7.         "P1"
8.     ],
9.     # CSS applied to this heading in the generated Markdown
10.    "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
11.    color: #2c3e50;"
12. },

```

- Lines [6–8]: the list of possible styles for your document’s title. When I look at this document, the LibreOffice style for my title is “Main Title.” However, the Gemini converter couldn’t find it. It logged the styles it encountered, and this displayed [P1]. This is a major issue with LibreOffice: the displayed style names do not match the internal names used by the software. They are only there to adapt to the user’s language;
- Line 10: Once the main title has been detected, you can configure its rendering. I wanted a font size of 28 (font-size: 28px;) and bold (font-weight: bold);

With the images and the title style, you can create an attractive home page.

The main title of your document may not have one of the styles defined in lines [6-8]. To find the style of your main title, use the following line in the [config.py] file

```

1. "debug": True

```

With the value [true], the style of the paragraphs preceding the first Level 1 heading—i.e., the paragraphs on the homepage—will be displayed when the Gemini/ChatGPT converter runs. So, for a document other than this one, I obtained the following logs:

```

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='Introduction to PHP7 through examples...'
11. >>> DETECTED DOCUMENT TITLE: Introduction to the PHP7 language through examples
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, July 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='Introduction to PHP 7...'

```

- line 10, the document title has the ‘P4’ style;

In the [config.py] file, I then added the following lines:

```

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:  
        #2c3e50;"  
6.     },
```

- line 3, the style I was looking for;

This is why the debugger displays the lines:

```
1. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Text='Introduction to the PHP7 language through  
    examples...'  
2. >>> DOCUMENT TITLE DETECTED: Introduction to the PHP7 language through examples
```

It encountered the 'P4' style and then displays that the document title has been found. Once it has been found, you can set the [debug] key to [false] in [config.py]:

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

Now let's look at the conversion of the [Examples] chapter, which contains the examples that the Gemini/ChatGPT converter can handle:

11.4. Bulleted lists

ODT / DOCX document:

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 ;

Above, the text [Bulleted lists] is highlighted because it is a reference associated with a cross-reference.

HTML document:

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 ;

Note that while the ODT/DOCX document uses different bullet styles, the HTML document uses only one type of bullet.

11.5. Numbered lists

ODT/DOCX document:

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 document:

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. Mixed lists 1

ODT/DOCX document

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 document

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 ;

Here too, there are sometimes differences between the types of bullets used.

11.5.2. Mixed lists 2

ODT / DOCX document

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 document

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 code blocks

Rich code blocks are rendered identically in HTML (except for the background color). Here are three examples:

11.6.1. Example 1

ODT / DOCX document

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 rendering

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. Example 2

ODT / DOCX file:

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 output

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. Example 3

ODT / DOCX document

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 diffère
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 conter
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 rendering

<https://tahe.developpez.com>

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. Plain text code blocks

Plain text code blocks found in the ODT / DOCX document are syntax-highlighted by MkDocs based on the language found in the code block. To help the converter identify the correct language, "key strings" have been added to the [config.py] file for each language. The converter counts the "key strings" found. It then associates the code block with the language that has the most "key strings" found.

Let's look at a few examples.

11.7.1. Example 1

ODT / DOCX document (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 Output

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         ...
    }
```

In the HTML document, we can see that the Java code has been syntax-highlighted.

11.7.2. Example 2

ODT / DOCX document (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 output

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. Example 3

ODT / DOCX document (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 output

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. Other code blocks

ODT / DOCX document

An execution result with a first line that does not start at 1:

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. [chaîne1,chaîne2,chaîne1chaîne2]
14. chaîne=chaîne1
15. type[4]=<class 'int'>
16. type[chaîne1]=<class 'str'>
17. type[['un', 'deux', 3, 4]]=<class 'list'>
18. type[a changé]=<class 'str'>
```

HTML output

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'>
```

A code block that is not numbered in ODT remains so in the HTML:

ODT / DOCX document

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 rendering

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. Links

ODT / DOCX document

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 rendering

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. Text formatting

ODT / DOCX document

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 rendering

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**.

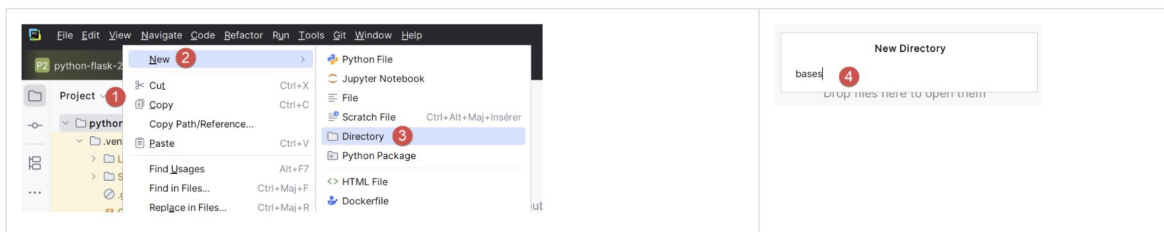
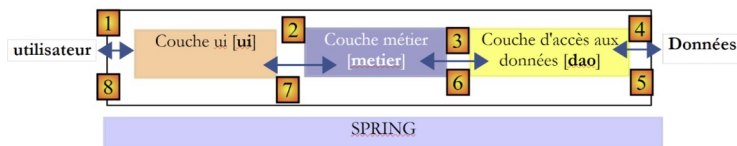
Il gère également les bordures supérieure et inférieure des paragraphes.

2.5. Un titre peut être également enrichi.

11.11. Images

2.5. Les images

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



Note that the Gemini / ChatGPT converter preserves image resizing done in the ODT / DOCX document.

11.12. Protected characters

ODT / DOCX document

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

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

HTML Rendering

<https://tahe.developpez.com>

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 document (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 output

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

```

- Markdown code has been preserved;

11.13. Tables

ODT / DOCX document

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 rendering

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. Footnotes

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. Known issues

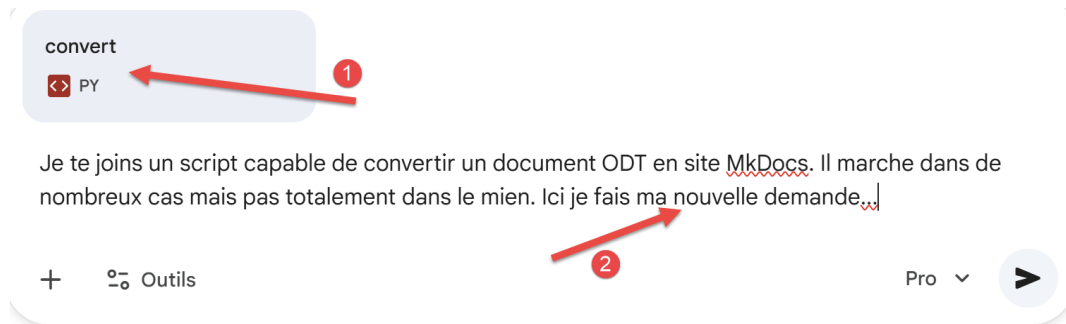
Some issues have been identified, but they can be corrected by modifying the ODT / DOCX:

- **code blocks must be followed by a blank line;** otherwise, the code block may render incorrectly. The identified issue is code immediately followed by a heading without being separated from it by a blank line;
- Bulleted lists cannot have a bottom border. **To add one, insert a blank line after the last list item;**
- **Bulleted lists must be nested.** Thus, a level 2 list must always be included within a level 1 list; otherwise, the level 2 list is rendered as code;

As new versions of the converter are released, some of these issues will be resolved. The three mentioned above can be avoided by correcting the source document.

11.16. Other Cases

If your document uses features other than those mentioned above, it is highly likely that these will not be supported by the Gemini/ChatGPT converter. What should you do then? You can submit your new request to one of the AIs by providing it with the current converter:



- In [1], I attach the converter for this document;
- In [2], I make my new request;

You will likely go through many iterations. When a version is stable, note its number so you can provide it to the AI in case of regression. It is also advisable to make a copy of each stable version. A major drawback of both AIs is that they regress quite easily. Simply asking for a new feature can cause the AI to break code that previously worked. Hence the importance of noting the version numbers of stable versions so you can revert to them. In January 2026, it seemed to me that ChatGPT 5.2 was less prone to regression than Gemini 3.

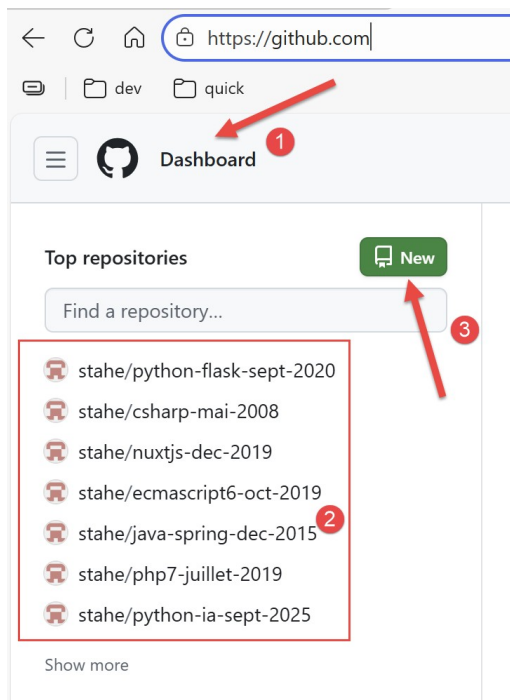
12. Hosting the HTML site on GitHub

It was Gemini himself who suggested that I host the HTML site generated by his two scripts on GitHub⁴. I didn't know that was possible. GitHub is a site that hosts development projects. Uploading programming courses there seems like a natural fit.

First, you need to have a [GitHub](#) account. If necessary, create one.

Log in to your GitHub account:

⁴Footnote for GitHub



- In [2], your existing repositories, if you have any;
- In [3], create a new 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 * staha / 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

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

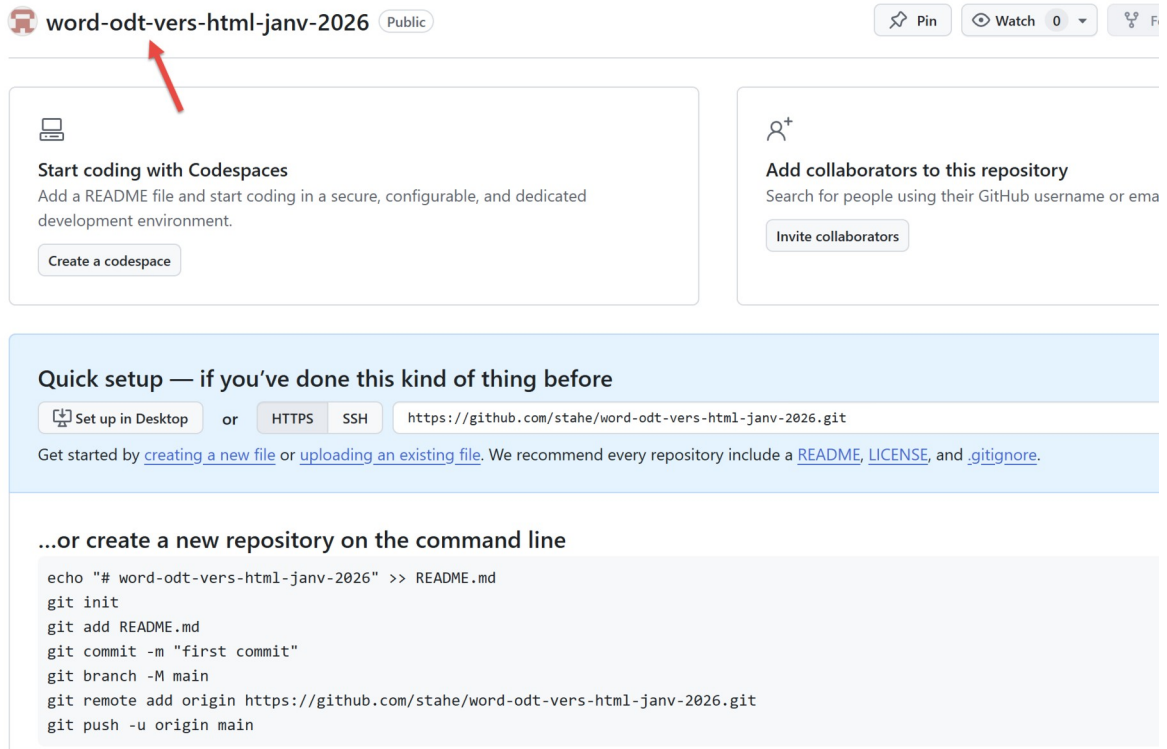
- In [1], use the name you entered in [config.py]:

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

- In [2], you can also enter the same information as in [config.py]:

```
1. "site_description": "Convert a Word or ODT document into a static HTML site using Gemini 3 and ChatGPT 5.2 AI,"
```

- In [3], confirm the creation of your GitHub repository;

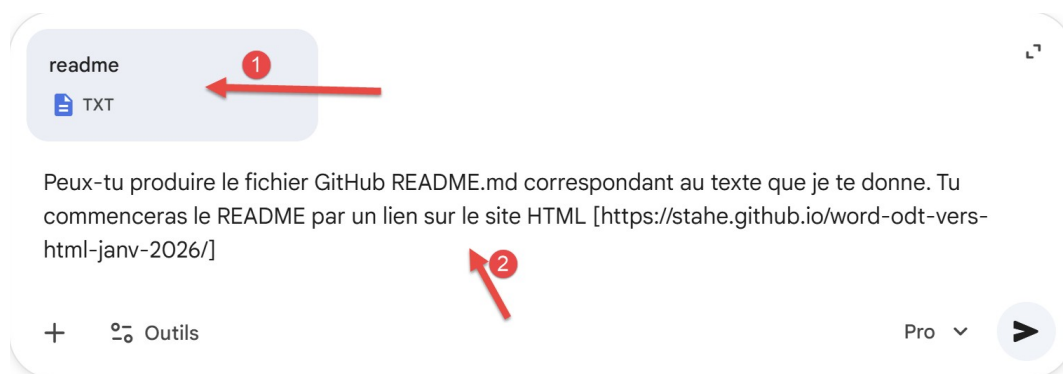


It is recommended that you create a [README.md] file for each GitHub repository, which will be displayed in place of the screen above. It must be written in Markdown, which isn't exactly straightforward. So we'll write a text file containing what we'd like to see in the README. Then we'll ask the AI to generate the corresponding README.md file.

The text file [readme.txt] will be as follows:

1. In this article, we aim to provide readers with a Python converter for converting Word or ODT documents into a static HTML site.
- 2.
3. This converter was built by the Gemini 3 AI.
- 4.
5. When the ODT document is suitable, the converter generates an HTML site via MkDocs that has the professional look of sites produced by Pandoc.
- 6.
7. I used Gemini 3 with a Pro subscription.







The prompt for Gemini is as follows:



- In [1], the attached file [readme.txt];
- In [2], the prompt;

Copy the response Gemini gives you into the [README.md] file in your working folder. Here is the response Gemini gave me:

```

1. # Word/ODT to HTML Site Converter (MkDocs)
2.
3.  [View the generated demo site](https://stahe.github.io/word-odt-vers-html-janv-2026/)
4.
5. ---
6.
7. ##  Description
8.
9. This project aims to provide readers with a Python converter for converting Word or ODT
10. documents into a static HTML website.
11. When the ODT document is suitable, the converter generates an HTML site using MkDocs that
12. has the professional look of sites produced by Pandoc.
13.
14. ##  Development Background
15.
16. This converter was built entirely by the Gemini 3 AI (with a pro subscription). It is the
17. result of successive iterations to finely manage the structure of ODT (OpenDocument Text)
18. documents.
19.
20. ##  Features
21.
22. The `convert.py` script performs the following actions:
23.
24. * ODT to Markdown Conversion: Parses the `.odt` (XML) file to extract its structure.
25. * Heading Management: Automatically generates the Table of Contents (TOC) and sidebar
26. navigation.
27. * Code Blocks: Automatic language detection, syntax highlighting, and precise line
28. numbering (`start-value` attributes).
29. * Lists: Support for bulleted and numbered lists with correct indentation.
30. * Formatting: Support for bold, italic, underline, and highlighting (preserving
31. original colors).
32. * Images: Automatic extraction and embedding of images contained in the document.
33. * Configuration: Customization via a `config.py` file (Footer, Google Analytics, etc.).
34.
35. ##  Installation
36.
37. ### Prerequisites
38.
39. * Python 3.x
40. * The following libraries:
41.
42. ```bash
43. pip install odfpy unicode mkdocs mkdocs-material
44. ```
45.
46. ### Project Structure
47.
48. Make sure you have the following files:
49.
50. * `convert.py`: The conversion script.
51. * `config.py`: Your configuration file.
52. * `your-document.odt`: The source document.
53.
54. ##  Usage
55.
56. 1. Conversion
57. Run the script by specifying the source ODT file and the configuration file:
58.
59. ```bash
60. python convert.py your-document.odt config.py

```

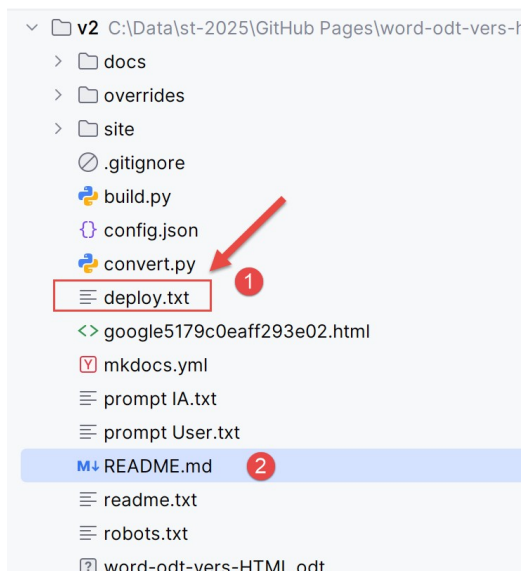
```

55.
56. ```
57.
58.
59. *This will generate a `docs/` folder containing the Markdown files and an `mkdocs.yml` file.*
60. 2. Preview
61. To view the site locally:
62. ```bash
63. mkdocs serve
64.
65. ```
66.
67.
68. 3. Generation
69. To build the static site (folder `site/`):
70. ```bash
71. mkdocs build
72.
73. ```
74.
75.
76.
77. ## ⚙️ Configuration (`config.py`)
78.
79. The `config.py` file lets you control the site's appearance:
80.
81. * mkdocs: General site settings (title, description, Material theme).
82. * footer: Complete HTML code for customizing the footer.
83. * code: Language detection rules for syntax highlighting.
84. * extra: Google Analytics (GA4) configuration.
85.
86. ## 📄 License
87.
88. This tutorial written by Serge Tahé is made available to the public under the terms of
  the:
89. *Creative Commons Attribution - Non-Commercial - ShareAlike 3.0 Unported License.*

```

He provided me with a very comprehensive README. That's because Gemini is very familiar with this project, which we've been working on for weeks. You'll likely have a less detailed [README.md].

Now let's return to our working directory:



- In [2], the README file you just modified;

- In [1], the [deploy.txt] file explains how to export your HTML site to your GitHub repository;

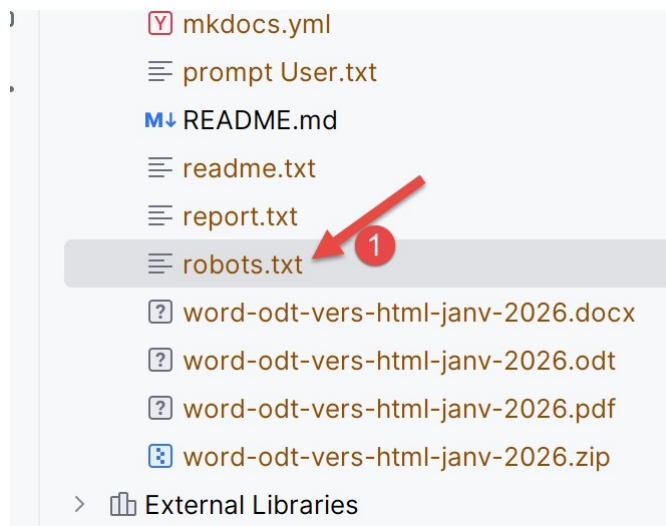
The contents of the [deploy.txt] file are as follows:

```
1. git init
2. git branch -M main
3. # Check .gitignore first!
4. git add .
5. # first commit
6. git commit -m "Initial commit: MkDocs source"
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
```

This sequence of commands will export your HTML site to your GitHub repository. You will need to modify line 7 with the URL of your own GitHub repository, found in the [config.py] file:

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

You should also check another URL in the [robots.txt] file:



The contents of the [robots.txt] file are as follows:

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

On line 3, enter your site's URL, the same one used in the [config.py] file:

```
1. # Site publication URL (e.g., GitHub Pages)
2. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

The [robots.txt] file is not used during local site build, but it will be used when the site is hosted on GitHub.

The command sequence in [deploy.txt] uses a program called Git. You must install it [[Git - Install for Windows](#)].

Once that's done, check the file named [.gitignore] in your working directory. It tells Git which files to ignore. My [.gitignore] file is as follows:

```
1. # 1. Ignore ALL project files and folders
2. *
3.
```

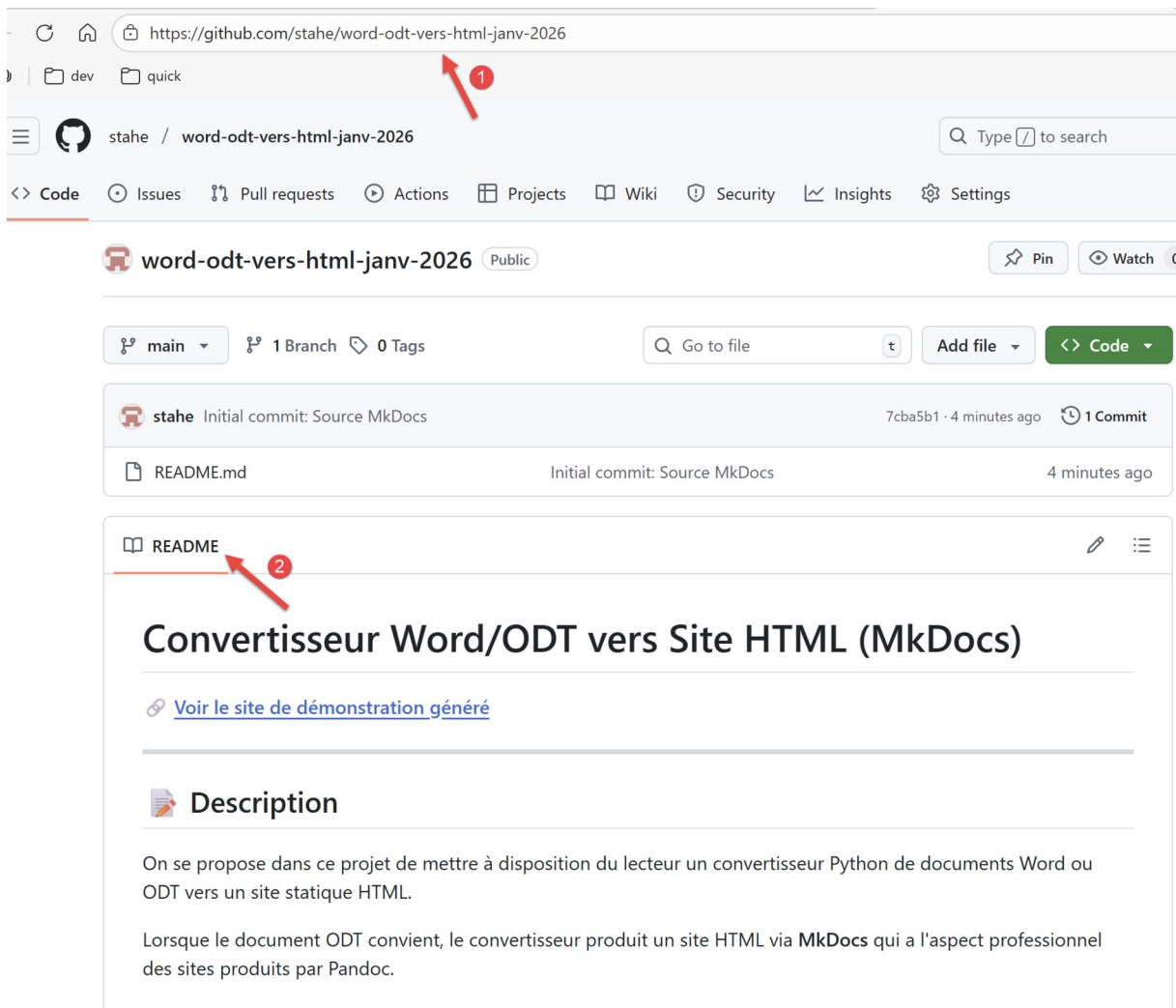
```
4. # 3. We make an exception (!) for the single file you want to keep
5. !README.md
```

It's extremely simple. We ignore all files (line 2) except the [README.md] file (line 5). GitHub is designed to host development projects. Typically, the entire development project is uploaded to GitHub. We're simply looking to upload an HTML site, not a development project. The only file we want to export to our GitHub repository is the [README.md] file, which explains to visitors what our HTML site contains.

Now, in your terminal, type the following sequence of commands in the order specified by [deploy.txt], up to command 8. Do not run command 9 for now.

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git init
2. Initialized empty Git repository in C:/Data/st-2025/GitHub Pages/word-odt-vers-html/v2/.git/
3. PS C:\Data\st-2025\GitHub Pages\word-odt-to-html\v2> git branch -M main
4. PS C:\Data\st-2025\GitHub Pages\word-odt-to-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] Initial commit: Source MkDocs
7. 1 file changed, 89 insertions(+)
8. create mode 100644 README.md
9. PS C:\Data\st-2025\GitHub Pages\word-odt-to-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-to-html\v2> git push -u origin main
11. Enumerating objects: 3, done.
12. Counting objects: 100% (3/3), done.
13. Delta compression using up to 8 threads
14. Compressing objects: 100% (2/2), done.
15. Writing objects: 100% (3/3), 1.70 KiB | 1.70 MiB/s, done.
16. Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
17. To https://github.com/stahe/word-odt-vers-html-janv-2026.git
18. * [new branch] main -> main
19. branch 'main' set up to track 'origin/main'.
```

Ctrl-click on the URL in line 17. This will take you to your GitHub repository:



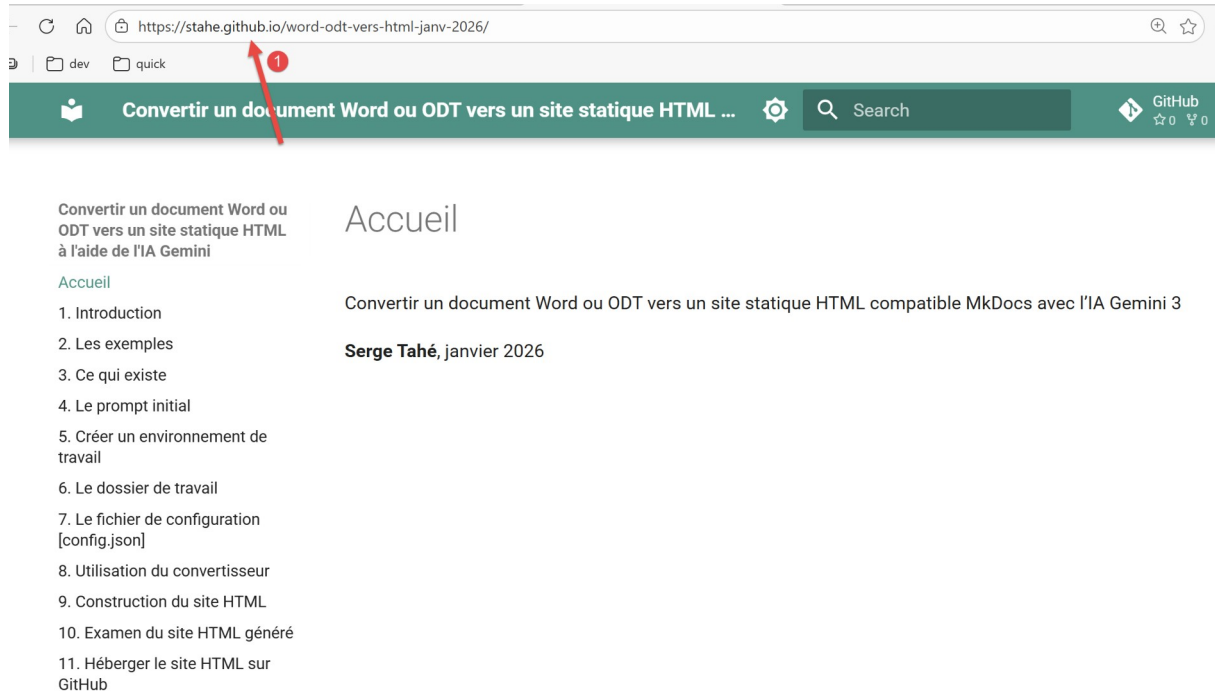
- In [1], the URL of your GitHub repository;
- In [2], the new README.md;

Now let's move on to line 9 of the [deploy.txt] file. This is the command that exports the HTML site to GitHub:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs gh-deploy
2. INFO - Cleaning site directory
3. INFO - Building documentation to directory: C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site
4. INFO - The 'les-exemples.md' file contains a link '#_Les_exemples', but there is no such anchor on this page.
5. INFO - Documentation built in 1.79 seconds
6. WARNING - Version check skipped: No version specified in previous deployment.
7. INFO - Copying 'C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site' to the 'gh-pages' branch and pushing to GitHub.
8. Enumerating objects: 91, done.
9. Counting objects: 100% (91/91), done.
10. Delta compression using up to 8 threads
11. Compressing objects: 100% (85/85), done.
12. Writing objects: 100% (91/91), 1.64 MiB | 2.01 MiB/s, done.
13. Total 91 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
14. remote: Resolving deltas: 100% (9/9), done.
15. remote:
16. remote: Create a pull request for 'gh-pages' on GitHub by visiting:
17. remote: https://github.com/stahe/word-odt-vers-html-janv-2026/pull/new/gh-pages
```

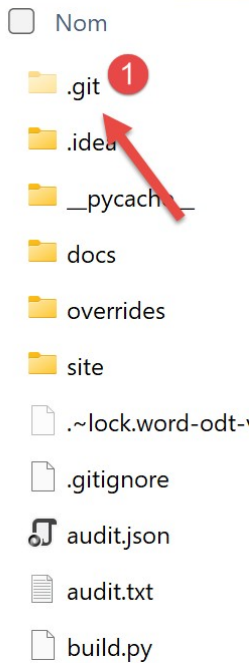
```
18. remote:
19. Tohttps://github.com/stahe/word-odt-vers-html-janv-2026.git
20. * [new branch]      gh-pages -> gh-pages
21. INFO    - Your documentation should shortly be available at: https://stahe.github.io/word-odt-vers-html-janv-2026/
```

Ctrl-click on the URL in line 21. This will take you to your new HTML site on GitHub:



- In [1], you can see that you are displaying a website on GitHub;

It is extremely easy to make a mistake when running the commands in [deploy.txt]. It is then difficult to undo the changes because Git keeps a record of what you've done (wrong). To start over, view the working directory:



- In [1], delete the [.git] folder;

Then go back to PyCharm and run the series of commands from [deploy.txt] again.

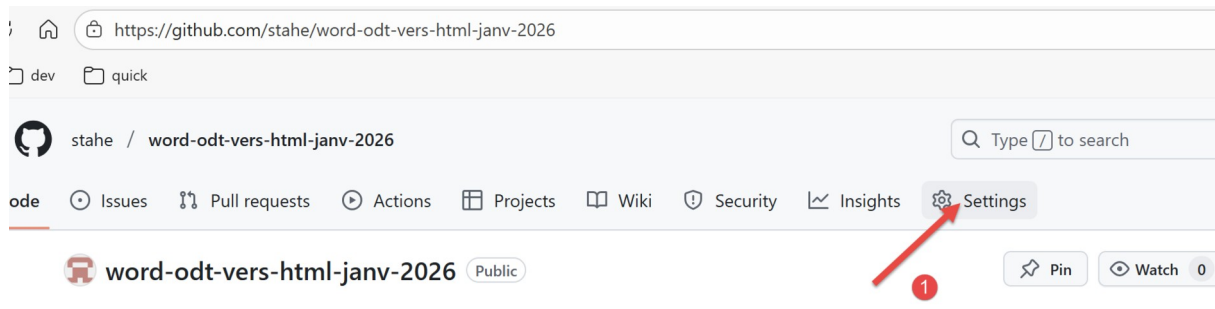
What should you do if you modify your ODT/DOCX document? Do the following 3 things:

1. Convert your ODT/DOCX document again using [convert];
2. build the HTML site with [build];
3. export the HTML site to GitHub using the command [python -m mkdocs gh-deploy]. This command is sufficient as long as you do not modify the [README.md] file. If you modify the [README.md] file, you will need to run a few more commands:

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

If you've only modified the README, only commands 1, 3, and 4 are necessary. Command 5 is unnecessary if you've already deployed the HTML site and it hasn't changed since then.

What if you want to start from scratch because things have gone haywire? You can delete your GitHub repository and repeat all the steps in the chapter12 . The option to delete a GitHub repository is well hidden:



- In [1], go to the repository settings;

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](#)

Scroll all the way to the bottom of the [settings] page. There you'll find the button to delete your repository [2].

13. Track your site with Google Analytics and Google Search Console

If you host your site on GitHub, you'll likely want to know how many visitors your site gets. To do this, you'll need a Google Analytics (GA) tag. We'll ask [ChatGPT](#) how to create a GA tag:

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. 

- In [1], enter the URL of your website on GitHub;

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.

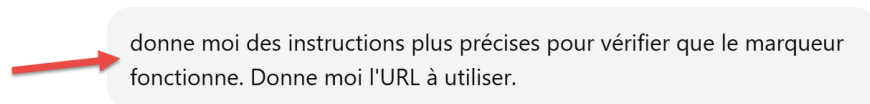
- In [2], note your site's tag;

Add this tag to the configuration file [config.py]:

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

On line 4, enter your tracking code. MkDocs will automatically insert it into all your HTML pages when it generates the HTML site. Therefore, you do not need to follow the other instructions from ChatGPT that show how to install the GA tracking code in your HTML pages.

We'll ask ChatGPT for specific instructions to verify that the tag is working:



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/ ↗](https://analytics.google.com/)

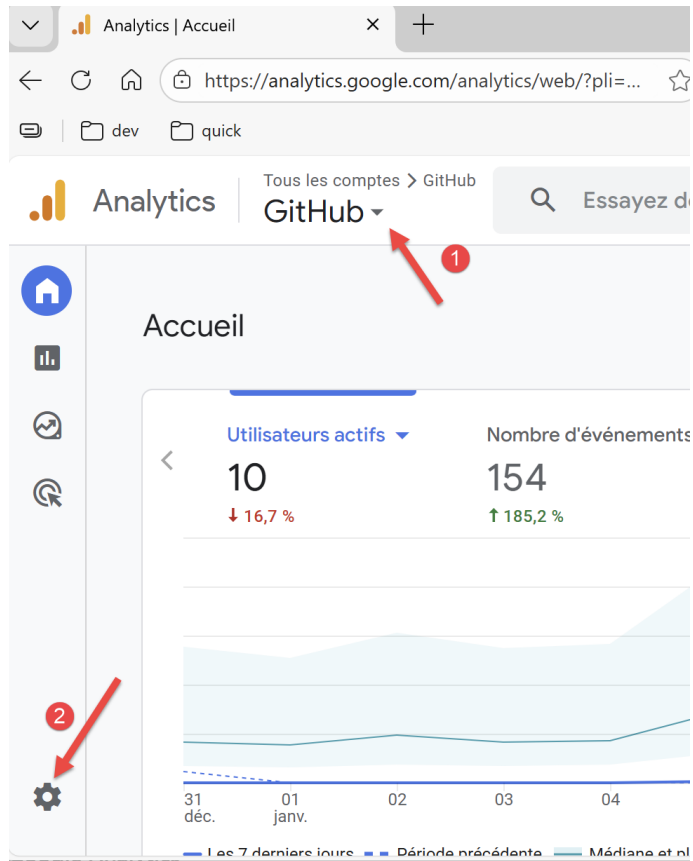
Connectez-vous avec le compte Google qui possède la propriété GA4.

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.

The previous step isn't clear enough. Here are some screenshots to help you:



- In [1], select the account you want to track (in case you have multiple accounts);
- In [2], select the admin button;

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		🔗

- In [3], select [Data Stream];

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.	>
---	-----------------------------------	---	-------------	---	---

- In [4], click on the data stream;

Détails du flux Web

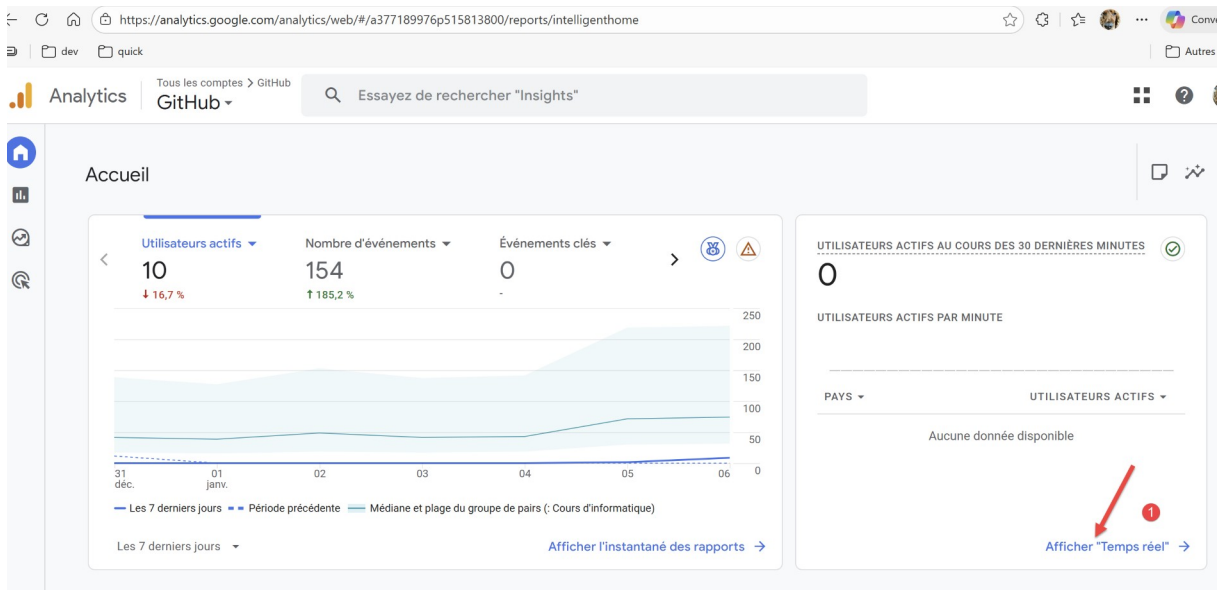
✔️ La collecte de données a été active au cours des 48 dernières heures.

Détails du flux ✎

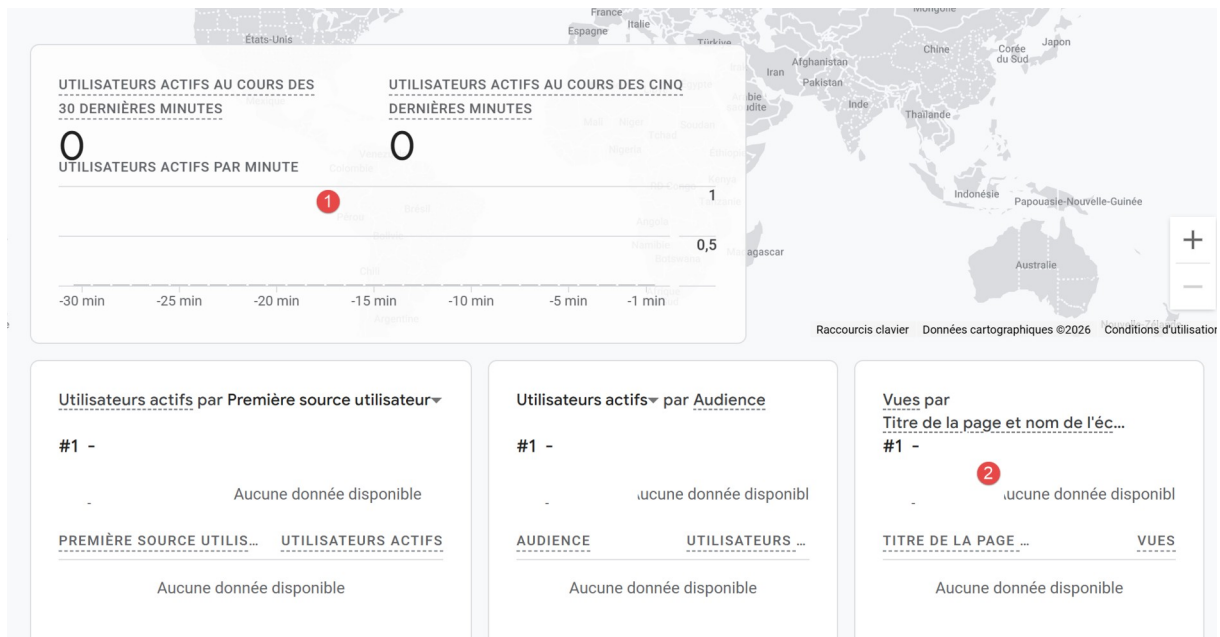
NOM DU FLUX	URL DE FLUX	ID DE FLUX	ID DE MESURE
GitHub	https://stahe.github.io	13112486249	G-EBZC: 5

- In [5], you'll find the GA4 tag you created;
- In [6], the stream ID. This is different from the measurement ID;

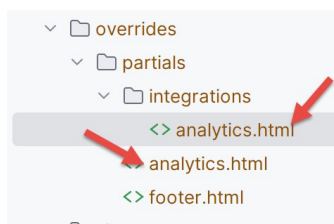
Let's go back to the GA homepage:



- In [1], select the real-time view of visits;



- In [1], you'll see your visitors;
- In [2], the page that was visited. I asked the Gemini converter to associate the site name rather than the page name with the GA4 tag. The [analytics.html] file handles this renaming associated with GA4:



Now we can continue with ChatGPT's explanations:

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.



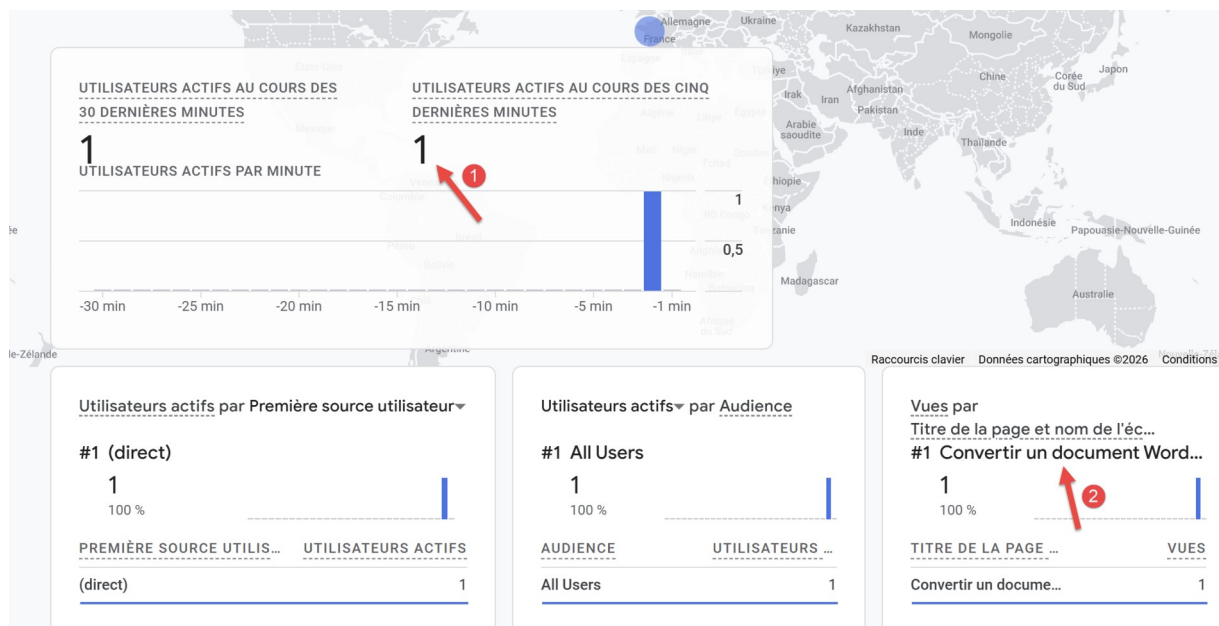
5. Résultat attendu

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.

I did what was requested in [4] without using an incognito window. And I got this:



The Google Analytics GA4 tag is working properly.

Now we'll introduce another tool for monitoring your site [Google Search Console]. Let's ask ChatGPT what this tool is used for:

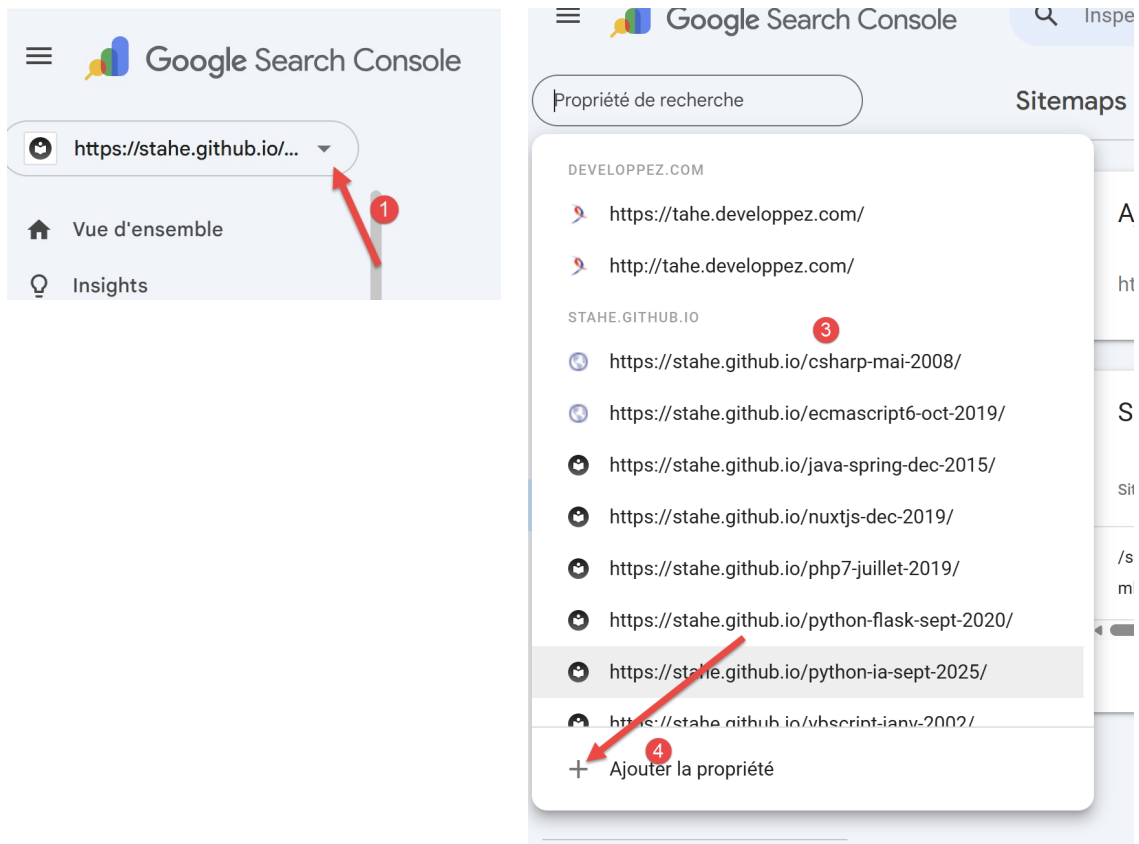
Conclusion

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.

I almost never use this tool. I only use it to encourage the Google search engine to crawl and index the site so that users can find it.

The tool's URL [<https://search.google.com/search-console>]:



- In [1], the list of sites that have been submitted to Google;
- In [3], the list of your sites;
- In [4], add your new site;

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

- In [5], enter your site's URL. It's in the [config.py] file:

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

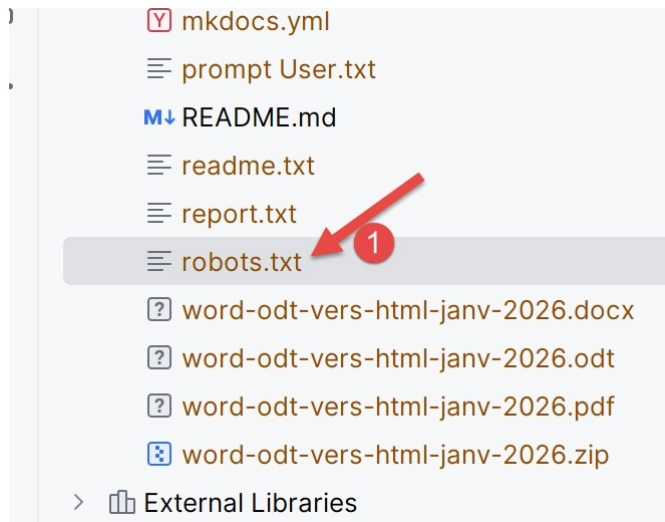
- In [6], proceed to the next step;

Normally, at this step, Google Search Console will offer you a [googlexxxxx.html] file that you need to download. I didn't see this screen because I already had this file on the deployed site. It is one of the two files declared in [config.py]

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

Place the [googlexxxxx.html] file in the root of your working directory and enter its name on line 2 above in the [config.py] file. The [convert] script handles placing the two files above in the root of the MkDocs site it generates. The [build] script will place them in the root of the generated HTML site.

Once you have copied the [googlexxxxx.html] file to the root of your working directory, check the contents of the [robots.txt] file:



The contents of the [robots.txt] file are as follows:

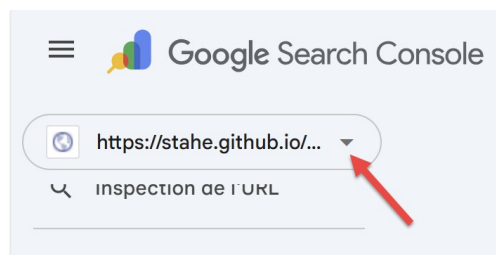
```
1. User-agent: *
2. Allow: /
3. Sitemap: https://stahe.github.io/word-odt-vers-html-janv-2026/sitemap.xml
```

On line 3, make sure that the URL is your site's URL, the same as in the [config.py] file.

Deploy your website to GitHub by entering the following three commands in your Python terminal:

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

Once this is done, return to [Google Search Console] and select the property you created earlier.



- In [1], select the [Sitemaps] option;
- In [2], type [sitemap.xml];
- In [3], validate the URL;

The [sitemap.xml] file is a file that Mkdocs placed in the root directory of the HTML site exported to GitHub. You can verify its presence by typing its URL directly:

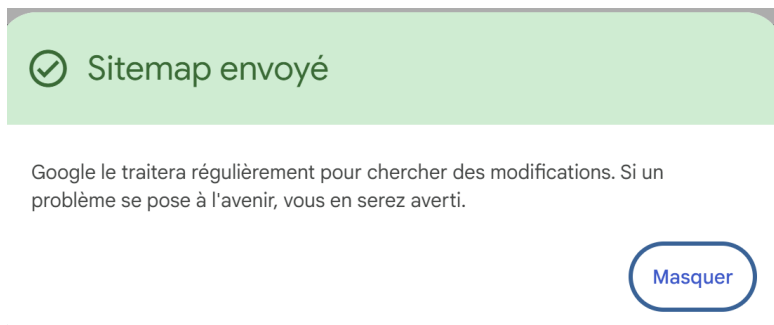


This XML file does not appear to have any style information associated with it. The document tree is shown below:

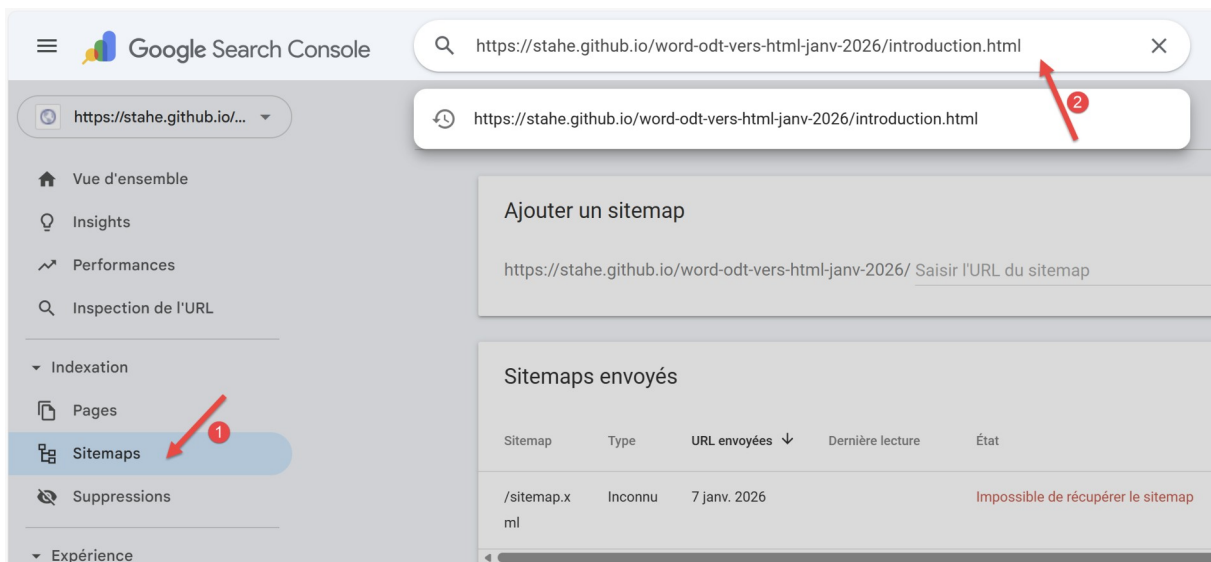
```
<?xml version="1.0" encoding="UTF-8" ?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/index.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  <url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/ce-qui-existe.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  <url>
    <loc>https://stahe.github.io/word-odt-vers-html-janv-2026/construction-du-site-html.html</loc>
    <lastmod>2026-01-07</lastmod>
  </url>
  <url>
```

The [sitemap.xml] file lists all the HTML pages on the site.

If everything goes well, you'll see the following screen:

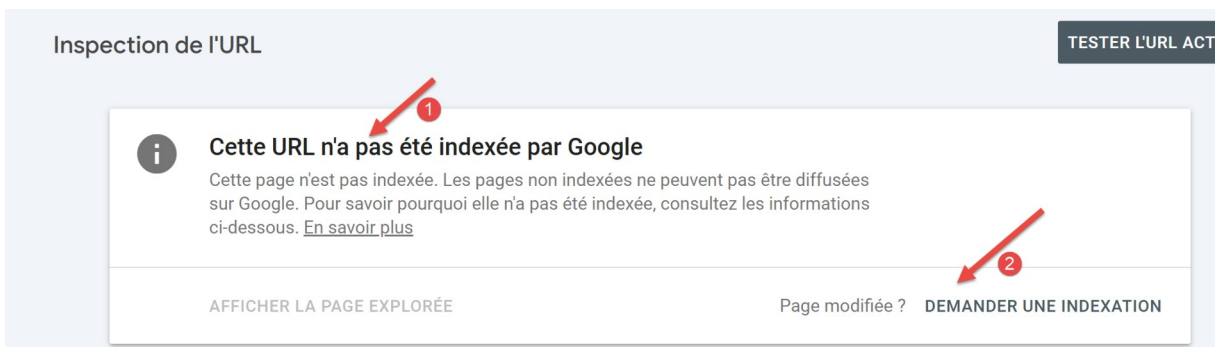


That's it. You'll need to index the site's pages yourself:



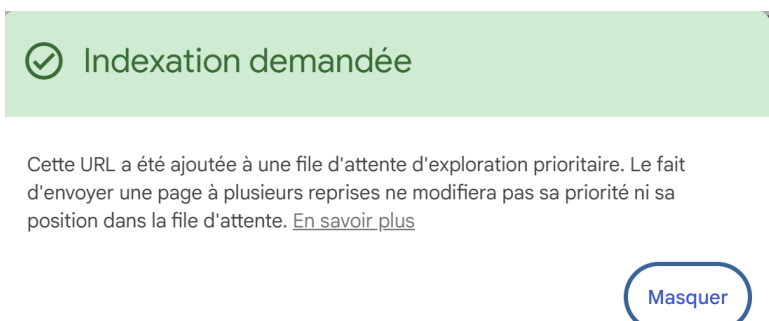
- In [2], enter the URL of one of your site's pages;

Here, the result is as follows:



- In [1], Google says the page isn't indexed yet;
- In [2], you can request that it be indexed;

Google Search Console then checks whether the requested page can be indexed. If so, you'll see the following message:



You can do this for all the pages on your site to ensure that it is properly indexed by Google's search engine.

14. Conclusion

In this document, we have demonstrated a converter that transforms ODT documents created by [LibreOffice](#) or DOCX documents from Word into a high-quality static HTML site.

In the chapter [[Examples in this document](#)], we presented the structures correctly handled by the Gemini/ChatGPT converter.

We have also shown how to export the HTML sites created by the ODT and DOCX converters to GitHub (see the chapter "[Hosting the HTML Site on GitHub](#)")

Apart from the structures in the examples chapter, nothing is guaranteed. It is likely that the converter will display anomalies. You will then need to start a new chat with Gemini or ChatGPT to resolve these anomalies. The simplest approach is to provide the current converter to the AI along with its configuration file. Then ask the AI if it understands the code you've provided. This forces it to analyze the code. It will always respond that it fully understands the code. That's when you should ask it for the desired changes. For both Word and LibreOffice, you can assist the AI with macros. The AI does not "see" the ODT or DOCX document being processed. It therefore lacks certain information. As a result, the DOCX-to-HTML converter couldn't properly handle the numbering of rich code blocks. It couldn't read the line number of the first line in a code block to render it in HTML. It always started the code at line 1, even if the code in the DOCX file started at line 12. So I used a macro to help. I placed the document cursor on the numbered line and asked ChatGPT to generate a VBA macro that would provide it with the characteristics of that line, enabling it to generate the HTML numbering correctly. ChatGPT did so, and I gave it the result of its macro. That solved the problem: the retrieved information showed it that the line number was being propagated by its style, not by the numbering icon. It then changed its algorithm, and it worked. So we can help AI make progress.

Another, simpler approach is to modify your document. When the converter produces an error, try modifying your document so that it uses only the structures from the examples chapter.