



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.

1. Introduction

Le PDF de ce document est disponible [[ICI](#)].

Les convertisseurs Gemini 3 / ChatGPT 5.2 [Word ou ODT → HTML] sont disponibles [[ICI](#)].

Ce document vous donne deux convertisseurs :

- un convertisseur LibreOffice ODT vers HTML ;
 - un convertisseur Word DOCX vers HTML ;
-

On se propose dans cet article de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML. Ce convertisseur a été construit initialement par l'IA [Gemini 3](#) puis par [ChatGPT 5.2](#). Il a fallu 356 itérations à ces deux IA pour produire le convertisseur de ce document. Cela a représenté un débogage de plusieurs semaines. Gemini 3 a fait tout le travail au début. Il a fallu d'abord plusieurs dizaines d'itérations pour obtenir une première version à peu près correcte. Puis régulièrement j'ajoutais un nouveau problème posé par de nouveaux documents ODT. Et là, Gemini a souvent régressé. C'est-à-dire que ce qui marchait à l'étape N ne marchait plus aux étapes suivantes. J'ai alors procédé ainsi : dès que Gemini produisait un convertisseur qui résolvait un de mes problèmes, j'en faisais une version de référence et je le disais à Gemini. Je sauvegardais cette référence localement. Puis lorsque je voyais Gemini régresser trop longtemps je lui demandais de revenir à la dernière version de référence connue en la lui donnant. C'est comme ça que peu à peu, on a construit ensemble ce convertisseur : moi en lui exprimant ce que je voulais, essentiellement en lui signalant les anomalies que je trouvais dans le site HTML produit, lui en produisant le code demandé.

J'ai utilisé [Gemini 3](#) avec une licence pro à 22 euros / mois et [ChatGPT 5.2](#) de même.

Gemini / ChatGPT va générer deux scripts Python :

- [convert] pour convertir le document ODT de [LibreOffice](#) ou le document DOCX de Word en site [MkDocs](#) ;
- [build] pour convertir le site MkDocs en site HTML statique ;

Je n'ai jamais regardé le code généré. Je voulais les considérer comme des boîtes noires. Il n'y a pas besoin d'être un développeur Python pour suivre ce tutoriel ni même un développeur tout court.

En proposant des améliorations du convertisseur à l'IA Gemini 3, celle-ci s'est parfois bloquée. Elle n'arrivait pas à faire les améliorations demandées. On en était alors à la version stable V316. Pour pouvoir avancer, j'ai alors donné cette version opérationnelle à [ChatGPT 5.2](#) en lui demandant les améliorations désirées. ChatGPT a correctement modifié le code de Gemini pour satisfaire mes nouvelles demandes. C'est pourquoi je considère que ce sont ces deux IA qui ont généré le convertisseur.

Par la suite j'ai continué à utiliser cette technique. Lorsqu'une IA bloquait sur une fonctionnalité, je donnais à l'autre IA la dernière version stable connue.

Les deux IA ont des méthodes différentes pour délivrer le code Python demandé :

- Gemini donne le code du script généré sur la page d'interrogation. Il faut ensuite copier / coller ce code ;
- ChatGPT propose un lien de téléchargement du script généré ;

Au-delà de 1000 lignes de code généré Gemini a montré de sévères faiblesses. A cause de contraintes techniques propres à cette IA, celle-ci n'arrivait pas à afficher, dans la page d'interrogation, la totalité du code généré. Très souvent il manquait des lignes de code. A cause de cette contrainte, à partir d'un moment il n'a plus été possible d'utiliser Gemini 3. Et c'est donc ChatGPT 5.2 qui a terminé l'écriture du script.

C'est également ChatGPT qui a généré le convertisseur Word vers HTML. Je lui ai donné le convertisseur ODT vers HTML qui marchait et lui ai demandé de l'adapter pour un document Word. Il l'a fait en 18 itérations. C'est une constante de ces deux IA : elles comprennent très bien les scripts Python qu'on leur donne et peuvent y apporter des

améliorations. Pour moi c'est la meilleure façon de travailler avec elles.

Avec ces convertisseurs, j'ai produit plusieurs sites web à partir de documents ODT de LibreOffice, et de documents Word :

- [\[Générer un script Python avec des outils d'IA \(2025\)\]](#) ;
- [\[Introduction au langage Python et au framework web Flask par l'exemple \(2020\)\]](#) ;
- [\[Introduction au langage PHP7 par l'exemple \(2019\)\]](#) ;
- [\[Introduction au framework VUE.JS par l'exemple \(2019\)\]](#) ;
- [\[Introduction au framework NUXTJS par l'exemple \(2019\)\]](#) ;
- [\[Introduction au langage ECMASCIPT6 par l'exemple \(2019\)\]](#) ;
- [\[Introduction au langage VBScript \(2002\)\]](#) ;
- [\[Introduction au langage Java et à l'écosystème Spring au travers d'une étude de cas \(2015\)\]](#) ;
- [\[Apprentissage du langage C# Version 3.0 avec le Framework .NET 3.5 \(2008\)\]](#) ;
- [\[Introduction par l'exemple à Entity Framework 5 Code First \(2012\)\]](#) ;
- [\[Introduction par l'exemple au framework ASPNET MVC par l'exemple \(2013\)\]](#) ;
- [\[Introduction à Spring MVC et Thymeleaf par l'exemple \(2015\)\]](#)
- [\[Exploiter une base de données relationnelle avec l'écosystème Spring \(2015\)\]](#)

Ce sont tous des cours de plusieurs centaines de pages. La taille du document ODT ou DOCX n'importe pas au convertisseur Gemini / ChatGPT. Ce qui lui importe ce sont les structures que l'on y trouve (cf. chapitre : [Les exemples de ce document](#)).

Je voudrais faire un article court. Les interactions entre une IA et un utilisateur sont présentés dans l'article [\[Générer un script Python avec des outils d'IA\]](#) que j'appellerais désormais [\[ref1\]](#). Les interactions avec Gemini et ChatGPT ne seront présentées qu'à la marge. De toute façon il était impossible de présenter toutes les itérations.

2. Les exemples de ce document

Je vais présenter maintenant des exemples des particularités de mes documents ODT / DOCX que le convertisseur Gemini / ChatGPT gère correctement. C'est ce propre document que nous proposerons à la conversion HTML au script de Gemini / ChatGPT. Nous verrons ce qu'il en fait.

2.1. Les listes

Le convertisseur Gemini / ChatGPT sait gérer les listes à puces et les listes numérotées même si elles sont imbriquées :

2.1.1. 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 ;

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 ;

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 ;

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 ;

2.2. Les blocs de code

Mes cours contiennent beaucoup de blocs de code. Ce sont souvent des codes enrichis (gras, couleurs des mots clés) par les IDE (Eclipse, PyCharm, WebStorm, Netbeans). Ces codes enrichis sont rendus à l'identique par le convertisseur.

Quand le code n'est pas enrichi (code venant du Bloc-Notes ou Notepad ou ...), le convertisseur Gemini / ChatGPT le reconnaît (Java, C#, XML, HTML, ...) à l'aide de mots clés du langage mis dans un fichier de configuration. Lorsqu'il reconnaît un langage il insère un marqueur (fence) pour MkDocs afin que celui-ci adapte la coloration syntaxique du code au langage utilisé dans le bloc de code.

2.2.1. Blocs de code enrichi (Eclipse, Visual Studio, ...)

Voici des blocs de code enrichis par différents IDE :

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.         ...

```

C#

```

11. using System;
12.
13. namespace Chap1 {
14.     class Impots {
15.         static void Main(string[] args) {
16.             // tableaux de données nécessaires au calcul de l'impôt
17.             decimal[] limites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
};
18.             decimal[] coeffR = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M,
0.481M };
19.             decimal[] coeffN = { 0M, 291.09M, 1322.92M, 2668.39M, 4846.98M,
6883.66M, 9505.54M };
20.
21.             // on récupère le statut marital
22.             bool OK = false;
23.             string reponse = null;
24.             while (!OK) {
25.                 Console.Write("Etes-vous marié(e) (O/N) ? ");
26.                 reponse = Console.ReadLine().Trim().ToLower();
27.                 if (reponse != "o" && reponse != "n")
28.                     Console.Error.WriteLine("Réponse incorrecte.
Recommencez");
29.                 else OK = true;
30.             }//while
31.             bool marie = reponse == "o";
32.         ...

```

Python

```

21. # -----
22. def affiche(chaine):
23.     # affiche chaine
24.     print("chaine=%s" % chaine)
25.
26.
27. # -----
28. def affiche_type(variable):
29.     # affiche le type de variable
30.     print("type[%s]=%s" % (variable, type(variable)))
31.
32.
33. # -----

```

```

34. def f1(param):
35.     # ajoute 10 à param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # rend un tuple de 3 valeurs
42.     return "un", 0, 100
43.
44.
45. # ----- programme principal -----
46. ...

```

PHP

```

31. <?php
32.
33. // types stricts pour les paramètres de fonctions
34. declare(strict_types=1);
35.
36. // constantes globales
37. define("PLAFOND_QF_DEMI_PART", 1551);
38. define("PLAFOND_REVENUS_CELIBATAIRE_POUR_REDUCTION", 21037);
39. define("PLAFOND_REVENUS_COUPLE_POUR_REDUCTION", 42074);
40. define("VALEUR_REDUC_DEMI_PART", 3797);
41. define("PLAFOND_DECOTE_CELIBATAIRE", 1196);
42. define("PLAFOND_DECOTE_COUPLE", 1970);
43. define("PLAFOND_IMPOT_COUPLE_POUR_DECOTE", 2627);
44. define("PLAFOND_IMPOT_CELIBATAIRE_POUR_DECOTE", 1595);
45. define("ABATTEMENT_DIXPOURCENT_MAX", 12502);
46. define("ABATTEMENT_DIXPOURCENT_MIN", 437);
47.
48. // définition des constantes locales
49. $DATA = "taxpayersdata.txt";
50. $RESULTATS = "resultats.txt";
51. $limites = 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. // lecture des données
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // ceci est un commentaire
43. // constante
44. const nom = "dupont";
45. // un affichage écran
46. console.log("nom : ", nom);
47. // un tableau avec des éléments de type différent
48. const tableau = ["un", "deux", 3, 4];
49. // son nombre d'éléments
50. let n = tableau.length;
51. // une boucle
52. for (let i = 0; i < n; i++) {
53.     console.log(`tableau[${i}] = ${tableau[i]}`);
54. }
55. // initialisation de 2 variables avec le contenu d'un tableau
56. let [chaine1, chaine2] = ["chaine1", "chaine2"];

```

```

57. // concaténation des 2 chaînes
58. const chaine3 = chaine1 + chaine2;
59. // affichage résultat
60. console.log([chaine1, chaine2, chaine3]);
61. ...

```

VBScript

```

51.      ' calcul de l'impôt d'un contribuable
52.      ' le programme doit être appelé avec trois paramètres : marié enfants salaire
53.      ' marié : caractère O si marié, N si non marié
54.      ' enfants : nombre d'enfants
55.      ' salaire : salaire annuel sans les centimes
56.
57.      ' déclaration obligatoire des variables
58.      Option Explicit
59.      Dim erreur
60.
61.      ' on récupère les arguments en vérifiant leur validité
62.      Dim marie, enfants, salaire
63.      erreur=getArguments(marie,enfants,salaire)
64.      ' erreur ?
65.      If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit erreur(0)
66.
67.      ' on récupère les données nécessaires au calcul de l'impôt
68.      Dim limites, coeffR, coeffN
69.      erreur=getData(limites,coeffR,coeffN)
70.      ' erreur ?
71.      If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit 5
72.
73.      ' on affiche le résultat
74.      wscript.echo "impôt=" & calculerImpot(marie,enfants,salaire,limites,coeffR,coeffN)
75.
76.      ' on quitte sans erreur
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 limite, coeffr, coeffn from tranches" />
80.                  </object>
81.                  <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
82.                      <constructor-arg index="0" ref="dao"/>
83.                  </object>
84.              </objects>
85.          </spring>
86.      </configuration>

```

2.2.2. Blocs de code brut (plain text)

Voici des exemples de code brut :

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\venv\Scripts\python.exe
   C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/bases/bases_01.py
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'>
19. res1=14s
20. (res1,res2,res3)=[un,0,100]
21. liste[0]=un
22. liste[1]=0
23. liste[2]=100
24. liste[0]=8
25. liste[1]=5
26. somme=13
27.
28. Process finished with exit code 0
```

Ce cas a nécessité des dizaines d'itérations. Le convertisseur ne trouvait jamais le numéro de la première ligne de code. Finalement j'ai demandé à Gemini comment formater le bloc de code pour qu'il le reconnaisse. Voici ce qu'il m'a répondu :

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.

J'ai suivî cette démarche et ça a marché. Moi j'utilisais la méthode via l'icône de liste (Personnaliser). Donc on peut être amené à modifier certains éléments du Document ODT / DOCX pour obtenir ce qu'on veut.

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

```

Voici les mêmes exemples que dans le paragraphe 2.2.1 mais non enrichis. Ce sont alors les mots clés du fichier de configuration qui vont guider le convertisseur vers le bon langage.

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.         ...

```

C#

```

11.     using System;
12.
13.     namespace Chap1 {
14.         class Impots {
15.             static void Main(string[] args) {
16.                 // tableaux de données nécessaires au calcul de l'impôt
17.                 decimal[] limites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
18.                 };
19.                 decimal[] coeffR = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M,
0.481M };
20.                 decimal[] coeffN = { 0M, 291.09M, 1322.92M, 2668.39M, 4846.98M,
6883.66M, 9505.54M };
21.
22.                 // on récupère le statut marital
23.                 bool OK = false;
24.                 string reponse = null;
25.                 while (!OK) {
26.                     Console.Write("Etes-vous marié(e) (O/N) ? ");
27.                     reponse = Console.ReadLine().Trim().ToLower();
28.                     if (reponse != "o" && reponse != "n")
Console.Error.WriteLine("Réponse incorrecte.
Recommencez");
29.                     else OK = true;
30.                 }//while
31.                 bool marie = reponse == "o";
32.             ...

```

Python

```

21.     # -----

```

```

22.     def affiche(chaine):
23.         # affiche chaine
24.         print("chaine=%s" % chaine)
25.
26.
27.     # -----
28.     def affiche_type(variable):
29.         # affiche le type de variable
30.         print("type[%s]=%s" % (variable, type(variable)))
31.
32.
33.     # -----
34.     def f1(param):
35.         # ajoute 10 à param
36.         return param + 10
37.
38.
39.     # -----
40.     def f2():
41.         # rend un tuple de 3 valeurs
42.         return "un", 0, 100
43.
44.
45.     # ----- programme principal -----
46. ...

```

PHP

```

31. <?php
32.
33. // types stricts pour les paramètres de fonctions
34. declare(strict_types=1);
35.
36. // constantes globales
37. define("PLAFOND_QF_DEMI_PART", 1551);
38. define("PLAFOND_REVENS_CELIBATAIRE_POUR_REDUCTION", 21037);
39. define("PLAFOND_REVENS_COUPLE_POUR_REDUTION", 42074);
40. define("VALEUR_REDUC_DEMI_PART", 3797);
41. define("PLAFOND_DECOTE_CELIBATAIRE", 1196);
42. define("PLAFOND_DECOTE_COUPLE", 1970);
43. define("PLAFOND_IMPOT_COUPLE_POUR_DECOTE", 2627);
44. define("PLAFOND_IMPOT_CELIBATAIRE_POUR_DECOTE", 1595);
45. define("ABATTEMENT_DIXPOURCENT_MAX", 12502);
46. define("ABATTEMENT_DIXPOURCENT_MIN", 437);
47.
48. // définition des constantes locales
49. $DATA = "taxpayersdata.txt";
50. $RESULTATS = "resultats.txt";
51. $limites = 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. // lecture des données
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "Impossible d'ouvrir en lecture le fichier des données [$DATA]\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41.     'use strict';
42.     // ceci est un commentaire
43.     // constante
44.     const nom = "dupont";

```

```

45.    // un affichage écran
46.    console.log("nom : ", nom);
47.    // un tableau avec des éléments de type différent
48.    const tableau = ["un", "deux", 3, 4];
49.    // son nombre d'éléments
50.    let n = tableau.length;
51.    // une boucle
52.    for (let i = 0; i < n; i++) {
53.        console.log("tableau[", i, "] = ", tableau[i]);
54.    }
55.    // initialisation de 2 variables avec le contenu d'un tableau
56.    let [chaine1, chaine2] = ["chaine1", "chaine2"];
57.    // concaténation des 2 chaînes
58.    const chaine3 = chaine1 + chaine2;
59.    // affichage résultat
60.    console.log([chaine1, chaine2, chaine3]);
61.    ...

```

VBScript

```

51.    ' calcul de l'impôt d'un contribuable
52.    ' le programme doit être appelé avec trois paramètres : marié enfants salaire
53.    ' marié : caractère O si marié, N si non marié
54.    ' enfants : nombre d'enfants
55.    ' salaire : salaire annuel sans les centimes
56.
57.    ' déclaration obligatoire des variables
58.    Option Explicit
59.    Dim erreur
60.
61.    ' on récupère les arguments en vérifiant leur validité
62.    Dim marie, enfants, salaire
63.    erreur=getArguments(marie,enfants,salaire)
64.    ' erreur ?
65.    If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit erreur(0)
66.
67.    ' on récupère les données nécessaires au calcul de l'impôt
68.    Dim limites, coeffR, coeffN
69.    erreur=getData(limites,coeffR,coeffN)
70.    ' erreur ?
71.    If erreur(0)<>0 Then wscript.echo erreur(1) : wscript.quit 5
72.
73.    ' on affiche le résultat
74.    wscript.echo "impôt=" & calculerImpot(marie,enfants,salaire,limites,coeffR,coeffN)
75.
76.    ' on quitte sans erreur
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 limite, coeffr, coeffn from tranches" />
80.      </object>
81.      <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
82.          <constructor-arg index="0" ref="dao" />
83.      </object>
84.  </objects>
85. </spring>
86. </configuration>
```

HTML

```

71. <!DOCTYPE HTML>
72. <HTML>
73.   <head>
74.     <title>Laragon</title>
75.
76.     <link href="https://fonts.googleapis.com/css?family=Karla:400" rel="stylesheet"
type="text/css">
77.
78.     <style>
79.       HTML, body {
80.         height: 100%;
81.       }
82.
83.       body {
84.         margin: 0;
85.         padding: 0;
86.         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.          <div class="info"><br />
126.          Apache/2.4.35 (Win64) OpenSSL/1.1.0i PHP/7.2.11<br />
127.          PHP version: 7.2.11 <span><a title="phpinfo()" href="/"?
128.          q=info">info</a></span><br />
129.          Document Root: C:/myprograms/laragon-lite/www<br />
130.
131.      </div>

```

2.3. Les liens

Le convertisseur Gemini / ChatGPT sait conserver les liens externes du Document ODT / DOCX. 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 : 2.1.1.

Un renvoi vers un repère de texte qui précède : Gemini 3

Un renvoi vers un repère de texte qui suit : GitHub

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](#)].

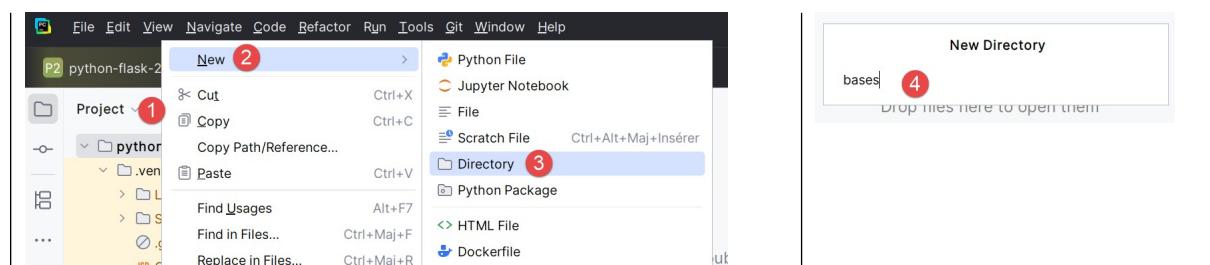
Le **convertisseur** gère également la **couleur** des **caractères**.

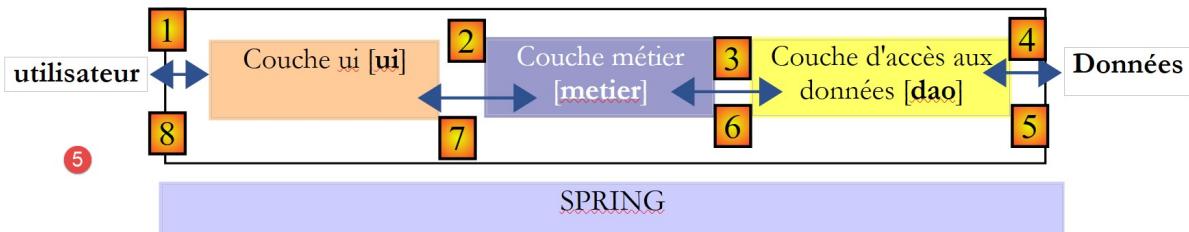
Il gère également les bordures supérieure et inférieure des paragraphes.

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

2.6. Les images

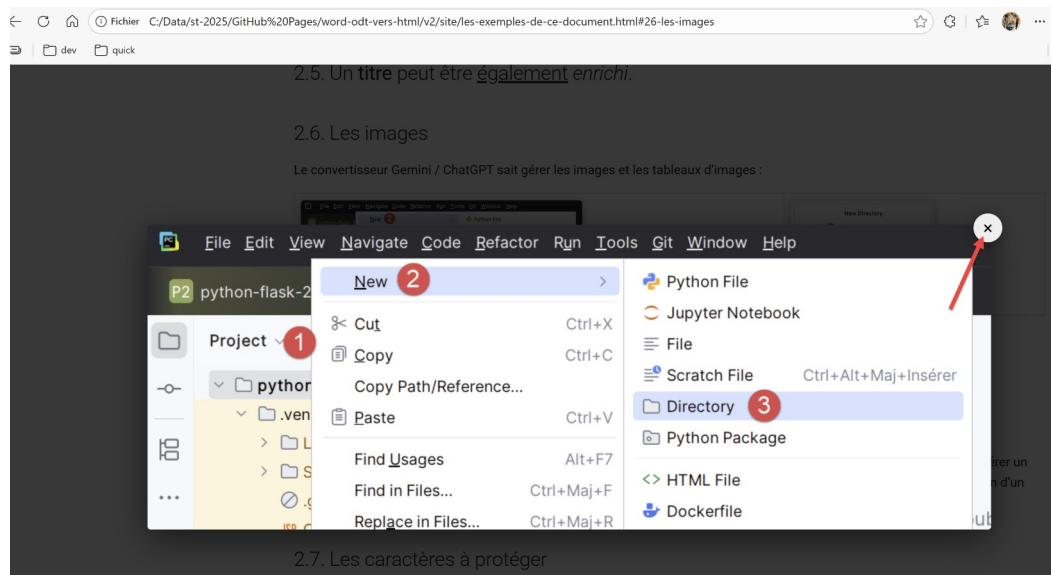
Le convertisseur Gemini / ChatGPT sait gérer les images et les tableaux d'images :





Dans les documents ODT il est fréquent d'avoir des dessins. Malgré des dizaines de tentatives, Gemini n'a pas réussi à générer un script qui générerait l'image (comme une copie d'écran) du dessin. Ainsi ci-dessus, l'image 5 est la copie d'écran d'un dessin d'un document ODT.

Toutes les images sont cliquables pour être agrandies. Si ci-dessus on clique sur l'image [1-3], on obtient l'agrandissement suivant :



2.7. Les caractères à protéger

Un site MkDocs¹ a des pages dont le contenu n'est pas du HTML mais du MarkDown. Si le Document ODT / DOCX contient des caractères qui existent en MarkDown, ils risquent d'être interprétés par MkDocs et donc ne pas avoir le rendu attendu. Voici deux exemples :

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

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

Un autre exemple est lorsque vous voulez insérer un bloc de code MarkDown dans votre document comme celui-ci :

```

1.      # Convertisseur Word/ODT vers Site HTML (MkDocs)
2.
3.      **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-vers-html-janv-2026/)***
4.
5.      ---
6.

```

¹MkDocs

```

7.      ## 📄 Description
8.
9.      On se propose dans ce projet de mettre à disposition du lecteur un convertisseur Python de
documents Word ou ODT vers un site statique HTML.
10.
11.     Lorsque le Document ODT / DOCX convient, le convertisseur produit un site HTML via **MkDocs**
qui a l'aspect professionnel des sites produits par Pandoc.
12.
13.     ## 🎨 Contexte de création
14.
15.     Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abonnement pro). Il
est le résultat d'itérations successives pour gérer finement la structure des documents ODT
(OpenDocument Text).
16.
17.     ## ✨ Fonctionnalités
18.
19.     Le script `convert.py` effectue les actions suivantes :
20.
21.     * **Conversion ODT vers Markdown** : Analyse le fichier `*.odt` (XML) pour en extraire la
structure.
22.     * **Gestion des Titres** : Génère automatiquement la Table des Matières (TOC) et la navigation
latérale.
23.     * **Blocs de Code** : Détection automatique des langages, coloration syntaxique et **gestion
précise de la numérotation des lignes** (attributs `start-value`).
24.     * **Listes** : Support des listes à puces et numérotées avec indentation correcte.
25.     * **Mise en forme** : Support du *gras*, *italique*, *souligné* et du *surlignage* (avec
respect des couleurs d'origine).
26.     * **Images** : Extraction et intégration automatique des images contenues dans le document.
27.     * **Configuration** : Personnalisation via un fichier `config.py` (Bas de page, Google
Analytics, etc.).
28.
29.     ## 🛠 Installation
30.
31.     ### Prérequis
32.
33.     * Python 3.x
34.     * Les bibliothèques suivantes :
35.
36.     ```bash
37.     pip install odfpy unidecode mkdocs mkdocs-material
38.
39.     ```


```

2.8. Les tableaux

Un tableau peut contenir différents contenus :

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.ClassPathXmlApplication
Context;
8.
9. public class Demo01 {
10.
11.
```

3 4



```
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,
11.        type(variable)))
12.
```

2.9. Les notes de bas de page

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

3. Ce qui existe sur internet

Avant de commencer, on peut demander à l'IA ce qui existe sur le marché pour convertir un document Word ou ODT en site statique HTML et qui serait gratuit.

Posons la question à Gemini :

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

Voici sa réponse :

Oui, il existe plusieurs solutions gratuites, mais le choix dépend de votre définition d'un "site statique" (une simple page HTML vs un site complet avec navigation) et de vos compétences techniques.

Voici les meilleures options classées par méthode :

- **1. La solution "Tout-en-un" (Pour créer un vrai site navigable)**

Si vous voulez transformer un long document Word (avec chapitres et sous-chapitres) en un site web complet avec un menu de navigation à gauche, c'est la meilleure option.

²Google Gemini

³La note de bas de page

- **Logiciel : HelpNDoc** (Version gratuite pour usage personnel)
- **Comment ça marche :** C'est un logiciel de création de documentation. Vous importez votre fichier Word, il analyse la structure (Titres 1, Titres 2...) et génère automatiquement un site HTML5 statique et "responsive" (adapté aux mobiles).
- **Les + :** Crée un vrai site avec menu, barre de recherche et index sans écrire une ligne de code.
- **Le - :** La version gratuite ajoute une petite bannière "Généré par HelpNDoc" en bas de page.
- **2. La solution "Nettoyeur en ligne" (Pour convertir page par page)**

Si vous avez un document court et que vous voulez juste obtenir le code HTML pour le mettre sur une page existante, les convertisseurs en ligne sont idéaux pour nettoyer le code "sale" généré par Word.

- **Outil : WordToHTML.net** (ou Word2CleanHTML)
- **Comment ça marche :**
 1. Vous copiez votre texte dans Word.
 2. Vous le collez dans l'éditeur visuel de gauche sur le site.
 3. Le site génère un code HTML propre (nettoyé des balises inutiles de Microsoft) dans la fenêtre de droite.
- **Les + :** Très simple, visuel, retire le "gras" du code Microsoft.
- **Le - :** Idéal pour une page unique, pas pour générer l'architecture d'un site entier.
- **3. La solution "Pro / Développeur" (La meilleure qualité)**

C'est la méthode recommandée par les experts pour obtenir un code ultra-propre et pérenne.

- **Logiciel : Pandoc** (Ligne de commande)
- **Comment ça marche :** C'est un petit programme sans interface graphique (à utiliser via le terminal/invite de commande). Il convertit les formats de manière très précise.
- **Commande type :** `pandoc mon_document.docx -t html -o mon_site.html`
- **Les + :** Code HTML5 parfait, gratuit, open-source, gère très bien les images et les formules mathématiques (LaTeX).
- **Le - :** Nécessite d'être à l'aise avec une invite de commande (écran noir).
- **4. La solution "Système D" (Microsoft Word)**

Vous pouvez le faire directement depuis Word, mais attention au résultat.

- **Méthode :** Fichier > Enregistrer sous > Choisir **"Page Web filtrée (*.htm; *.html)"**.
- **Important :** Ne choisissez pas "Page Web à fichier unique (.mht)", choisissez bien "Filtrée". L'option "Filtrée" réduit (un peu) le code superflu que Word ajoute habituellement.
- **Le - :** Le code reste souvent lourd et difficile à modifier par la suite.
- **Tableau Comparatif Rapide**

Solution	Type de résultat	Qualité du code	Facilité	Idéal pour...
HelpNDoc	Site complet (avec menu)	Bonne	Moyenne	Manuels, thèses, documentation
WordToHTML	Page unique	Excellente	Très Facile	Articles de blog, pages simples
Pandoc	Page unique (brute)	Parfaite	Difficile	Développeurs, archivage propre
Word (Natif)	Page unique	Mauvaise	Facile	Dépannage rapide

• **Mon conseil**

Si votre objectif est de publier ce document en ligne de façon professionnelle, je vous suggère **HelpNDoc** si vous êtes débutant et que vous voulez une structure de site immédiate.

Si vous souhaitez simplement le contenu propre pour le coller ailleurs (WordPress, etc.), utilisez **WordToHTML.net**.

Souhaitez-vous que je vous détaillle la procédure pas-à-pas pour l'un de ces outils (par exemple WordToHTML) ?

J'ai utilisé parfois [Pandoc] mais j'avais trouvé son usage difficile. Je me rappelle qu'il fallait « préparer » le document Word pour qu'il soit correctement géré par Pandoc. Cependant le résultat était excellent. **HelpNDoc** semble un logiciel attrayant puisqu'il génère un site avec menu. Je l'ai testé sur un de mes documents Word et le résultat n'a pas été satisfaisant.

Maintenant cherchons s'il existe un logiciel de conversion pour les documents ODT :

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

La réponse de Gemini est la suivante :

Oui, tout à fait. Comme pour Word, le format **ODT** (utilisé par LibreOffice et OpenOffice) dispose de ses propres solutions.

Voici les meilleures options gratuites spécifiques aux fichiers ODT, classées par méthode :

- **1. La solution "Extension" (La meilleure qualité intégrée)**

Si vous utilisez LibreOffice ou OpenOffice, l'export HTML de base est (comme pour Word) assez médiocre. Cependant, il existe une extension gratuite formidable conçue spécifiquement pour cela.

- **Outil : Writer2xhtml** (Extension pour LibreOffice)
- **Comment ça marche :**
 1. Vous téléchargez et installez l'extension dans LibreOffice.
 2. Vous faites Fichier > Exporter et une nouvelle option "XHTML 1.1 + MathML" (ou similaire) apparaît.
- **Les + :** Le code est beaucoup plus propre que l'export standard, et il gère très bien les formules mathématiques et la mise en forme complexe.

- **Le - :** Il faut installer une extension (manipulation technique unique).
- **2. La solution "Pro / Ligne de commande" (Universelle)**

C'est le même outil que pour Word, car il excelle aussi avec le format ODT.

- **Logiciel : Pandoc**
- **Comment ça marche :** Via le terminal de commande.
- **Commande :** pandoc mon_fichier.odt -t html -o mon_site.html
- **Les + :** C'est sans doute le convertisseur le plus puissant du marché pour le format ODT vers HTML. Il "comprend" la structure du document ODT mieux que quiconque.
- **3. La solution "Site Complet avec Navigation" (Via HelpNDoc)**

HelpNDoc (mentionné précédemment) est toujours la meilleure option gratuite pour créer un vrai site avec menu à gauche.

Cependant, HelpNDoc n'importe pas toujours nativement le format .odt aussi bien que le .docx.

- **L'astuce :**
 1. Ouvrez votre fichier dans LibreOffice.
 2. Faites Enregistrer sous... et choisissez "**Word 2007-365 (.docx)**".
 3. Importez ce fichier .docx dans HelpNDoc pour générer votre site.
- **Pourquoi ?** Cela garantit que la hiérarchie de vos chapitres (Titres 1, Titres 2) est parfaitement reconnue pour créer le menu du site.
- **4. La solution en ligne (Nettoyage rapide)**

Certains outils en ligne acceptent directement le ODT pour en sortir du HTML propre.

- **Outil : WordToHTML.net** (Accepte aussi les ODT)
- **Comment ça marche :** Vous pouvez uploader votre fichier ODT directement ou coller votre texte.
- **Les + :** Très pratique pour convertir un article unique sans rien installer.
- **Résumé de mon conseil pour du ODT :**
 1. **Pour un vrai site (Documentation/Thèse) :** Convertissez votre ODT en **DOCX** via LibreOffice, puis utilisez **HelpNDoc** (c'est la méthode la plus fiable pour avoir un menu de navigation).
 2. **Pour une page simple et propre :** Utilisez l'extension **Writer2xhtml** si vous faites cela souvent, ou **Pandoc** si vous voulez automatiser la tâche.

Voulez-vous que je vous explique comment installer l'extension **Writer2xhtml** dans LibreOffice ?

J'ai testé l'extension de LibreOffice [Writer2xhtml] sur un de mes documents. Ca donne quelque chose de lisible mais pas suffisamment professionnel. On n'a pas de table des matières sur la gauche. Les blocs de code perdent leur coloration syntaxique.

Au final je pense que le convertisseur généré par Gemini est une alternative à ce qui est proposé ci-dessus. Le résultat est comparable à celui produit par Pandoc si le document ODT convient au convertisseur.

4. Le prompt initial à Gemini 3

On suppose ici que le lecteur a une première connaissance de l'IA [Gemini](#). Si ce n'est pas le cas, il lui est conseillé de lire le document [ref1]. Le prompt initial donné à Gemini l'a été sous la forme du fichier texte [Prompt User.txt] suivant :

```
1. Demande utilisateur
2.
3. - crée un script Python [convert] pour créer un site MkDocs avec le thème material à partir d'un document ODT. Le nom du document ODT sera l'unique argument de convert.
4.
5. - puis crée un script Python [build] qui exploitera le fichier [mkdocs.yml] créé par convert pour créer un site statique HTML
6. - à la fin, le script [build] affichera l'accueil du site dans un navigateur.
7.
8. - le site aura deux colonnes indépendantes. Celle de gauche contiendra la table des matières, celle de droite le contenu du document ODT.
9.
10. - la table des matières reprendra la totalité des titres du document ODT (Titre1, Titre2, Titre3, etc...), leur numérotation et leur hiérarchie
11.
12. - les images sont la plupart du temps dans des tableaux et ont été redimensionnées.
13. Tu exploreras les tableaux à la recherche de ces images et tu respecteras leur redimensionnement.
14.
15. - il y a des blocs de code dans le document ODT. Ils peuvent être repérés par leurs styles [Source Code numéroté résultats, Source Code]
16. Les autres attributs du paragraphe de code doivent être ignorés. Par exemple, si le paragraphe de code est également une liste numérotée, tu dois ignorer cet attribut.
17. Tous ces blocs de code doivent être traités comme du code dans le site MkDocs. Les lignes de code doivent être numérotées dans le résultat HTML.
18.
19. - pour tous les blocs de code tu utiliseras la coloration syntaxique de Python.
20.
21. - tu ne dois pas toucher aux lignes de code. Notamment tu ne dois pas supprimer les indentations ou espaces qui précèdent le premier caractère de la ligne de code.
22.
23. - tu dois trouver toutes les listes à puces. ODT a plusieurs façons de les créer. Tu dois toutes les explorer. Les listes de puces sont
24. parfois imbriquées les unes dans les autres. Tu dois respecter cette imbrication.
25.
26. - le titre du site sera "Générer un script Python avec des outils d'IA"
27.
28. - tu logueras chaque image extraite du document ODT ainsi que chaque chapitre MkDocs créé
29.
30. - tu dois respecter les liens que tu trouves dans le document ODT. Ils doivent rester des liens dans le document HTML généré.
31.
32. - tu dois respecter les textes en gras, soulignés ou en italiques. ils doivent être rendus tels quels dans le document HTML
33.
```

- ligne 19 : le test initial a été fait sur un cours publié précédemment qui contenait des codes Python ;
- ce prompt a permis de démarrer le chat. Au bout de quelques dizaines d'itérations il est devenu obsolète ;

5. Créer un environnement de travail Python

Gemini et ChatGPT vont générer deux scripts Python. Il nous faut donc un environnement Python :

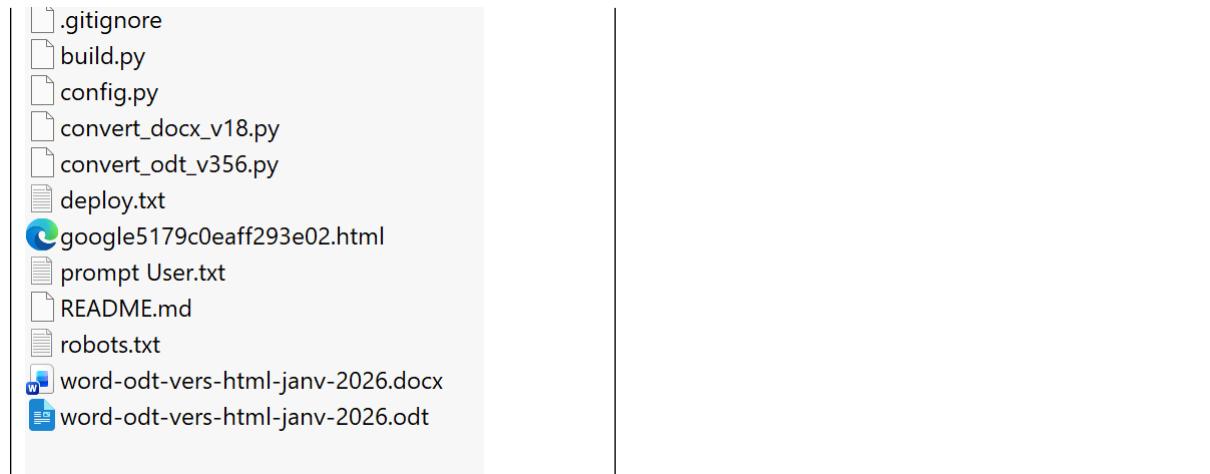
- L'installation de Python est présenté au [paragraphe 3.1](#) de [ref1] ;
- L'installation de PyCharm est présenté au [paragraphe 3.2](#) de [ref1]. [ref1] présente une longue configuration de PyCharm. Ici, PyCharm ne sera utilisé que pour exécuter les scripts Python générés par Gemini et

ChatGPT. Donc toute configuration de PyCharm est inutile. PyCharm sera utilisé également pour présenter les fichiers nécessaires au convertisseur ;

6. Le dossier de travail du convertisseur

Vous pouvez télécharger le dossier de travail [[ICI](#)].

Le dossier de travail est le suivant :



- Nous allons ignorer pour l'instant les fichiers suivants [.gitignore, deploy.txt, google*.html, README.md, robots.txt, deploy.txt]. Ce sont des fichiers qui nous serviront au déploiement sur GitHub du site HTML généré localement ;
- [prompt User.txt] est le prompt avec lequel j'ai commencé les itérations avec Gemini ;
- Le fichier [word-odt-vers-html-janv-2026.odt] est le document ODT de cet article. Il est converti en HTML par le script [convert_odt_v356.py] ;
- Le fichier [word-odt-vers-html-janv-2026.docx] est le document DOCX de ce même article. Il est converti en HTML par le script [convert_docx_v18.py] ;
- Les scripts [convert*.py] et [build.py] sont les deux scripts Python générés par les IA ;
 - [convert*] convertit un document ODT ou DOCX en site [[MkDocs](#)]. Un site MkDocs est un site statique dont les pages sont écrites selon la syntaxe [[MarkDown](#)]. [[MkDocs](#)] fournit un serveur capable d'afficher les sites MkDocs ;
 - [build] convertit le site MkDocs en site statique HTML standard. A la fin de cette conversion, il ouvre un navigateur pour afficher la racine du site ;

A aucun moment nous ne regarderons le code Python généré. On considérera les deux scripts comme deux boîtes noires. A aucun moment je n'ai modifié du code à la main. J'ai toujours demandé à Gemini / ChatGPT de corriger lui-même son script. C'est pourquoi il n'est pas nécessaire de connaître Python pour utiliser le convertisseur.

- [config.py] est un fichier de configuration pour les scripts [convert*]. Au départ [convert*] n'avait pas de fichier de configuration. Puis en changeant de documents à convertir, je me suis aperçu peu à peu qu'il y avait des éléments que je demandais à l'IA de générer (par exemple le nom du site qui change pour chaque document) qui seraient mieux placés dans un fichier de configuration que l'utilisateur pourrait modifier lui-même. Je l'ai construit lui aussi par itérations avec l'IA ;

7. Le fichier de configuration des convertisseurs

Le fichier de configuration [config*.py] rassemble les informations qui peuvent changer d'un document à l'autre. Il commence ainsi :

```
1.     config = {
2.         # -----
3.         # Configuration MkDocs
4.         # -----
5.         "mkdocs": {
6.             # Titre du site qui apparaîtra dans l'onglet du navigateur
7.             "site_name": "Convertir un document Word ou ODT vers un site statique HTML à l'aide de
8.             l'IA Gemini 3",
9.             # URL de publication du site (ex: GitHub Pages)
10.            "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
11.
12.            # Méta-description pour le référencement (SEO)
13.            "site_description": "Convertir un document Word ou ODT vers un site statique HTML à
14.             l'aide de l'IA Gemini 3",
15.            # Auteur du site
16.            "site_author": "Serge Tahé",
17.
18.            # Lien vers le dépôt de code source (icône GitHub en haut à droite)
19.            "repo_url": "https://stahe.github.io/word-odt-vers-html-janv-2026",
20.            "repo_name": "GitHub",
```

- ligne 1 : le fichier de configuration est un script Python. Il définit une unique variable [config]. Celle-ci est un dictionnaire qui va contenir toutes les valeurs de la configuration ;
- ligne 5 : le dictionnaire [mkdocs] va configurer le fichier [mkdocs.yml] généré par le convertisseur Gemini / ChatGPT. Ce fichier sert ensuite au script [build] pour construire un site HTML à partir du site MkDocs généré par le convertisseur ;
- lignes 5-20 : configurent le site GitHub qui va héberger le site HTML créé à partir du document ODT / DOCX (cf paragraphe 12)

Le fichier de configuration [config.py] se poursuit ainsi :

```
21.        # Important : à False pour que les liens fonctionnent correctement sur GitHub Pages
22.        # (évite les liens du type dossier/ au lieu de dossier/index.html)
23.        "use_directory_urls": False,
24.
25.        # Configuration du thème Material pour MkDocs
26.        "theme": {
27.            "name": "material",
28.            # Dossier contenant les surcharges (footer personnalisé, etc.)
29.            "custom_dir": "overrides",
30.
31.            # Fonctionnalités de navigation activées
32.            "features": [
33.                "navigation.sections", # Regroupe les sections dans le menu
34.                "navigation.indexes", # Permet de cliquer sur le titre de section
35.                "navigation.expand", # Ouvre les menus par défaut
36.                "toc.integrate", # Intègre la table des matières à gauche (si supporté)
37.                "navigation.top" # Bouton "Retour en haut"
38.            ],
39.
40.            # Palette de couleurs (Mode Clair / Mode Sombre)
41.            "palette": [
42.                {
43.                    "media": "(prefers-color-scheme: light)",
44.                    "scheme": "default",
45.                    "primary": "teal",
46.                    "accent": "purple",
47.                    "toggle": {
48.                        "icon": "material/brightness-7",
```

```

49.          "name": "Passer au mode sombre"
50.      },
51.      {
52.        "media": "(prefers-color-scheme: dark)",
53.        "scheme": "slate",
54.        "primary": "teal",
55.        "accent": "purple",
56.        "toggle": {
57.          "icon": "material/brightness-4",
58.          "name": "Passer au mode clair"
59.        }
60.      }
61.    ],
62.  ],
63.

```

- ligne 23 : cette ligne est importante. Si elle est absente, au lieu d'afficher une page HTML, le navigateur ouvrira le dossier de cette page ;
- ligne 26 : MkDocs propose plusieurs thèmes pour un site MkDocs / HTML. Ici nous avons choisi le thème « material », ligne 27 ;
- lignes 31-38 : configuration de la navigation dans le site. Celle-ci se fera à l'aide d'une table des matières située dans la colonne de gauche de la page affichée (ligne 36) ;
- lignes 40-63 : définissent deux palettes de couleurs : le mode clair (lignes 43-49) ou le mode sombre (lignes 52-61). Une icône dans la barre supérieure des pages affichées permet de passer de l'un à l'autre ;

La suite du fichier de configuration est la suivante :

```

64.      # Extensions Markdown utilisées pour enrichir le contenu
65.      "markdown_extensions": [
66.        "admonition", # Blocs d'avertissement/info
67.        "attr_list", # Attributs CSS personnalisés {: .class }
68.        "pymdownx.superfences", # Blocs de code avancés
69.        "pymdownx.mark", # Surlignage ==texte==
70.        {
71.          "pymdownx.highlight": {
72.            "anchor_linenums": True,
73.            "linenums": None # La numérotation est gérée dynamiquement par le script
74.          }
75.        },
76.        "md_in_html", # Autorise le Markdown dans les balises HTML (crucial pour les
listes et tableaux)
77.        "footnotes" # Notes de bas de page [^1]
78.      ],

```

- lignes 64-78 : des extensions au langage MarkDown utilisé par MkDocs. Ces extensions ont été générées par Gemini à la suite de certaines de mes demandes ;

La suite du fichier de configuration est la suivante :

```

79.      # Scripts et CSS additionnels
80.      "extra_javascript": [
81.        "javascripts/focus.js" # Script pour le mode "Focus" (plein écran)
82.      ],
83.      "extra_css": [
84.        "stylesheets/focus.css"
85.      ],
86.    },

```

- ligne 81 : le script [focus.js] est un script Javascript généré par Gemini. Il est associé au bouton de la barre supérieure du site pour afficher ou cacher la table des matières ;
- ligne 84 : le CSS utilisé par ce bouton ;

La suite du fichier de configuration est la suivante :

```
87.      # -----
```

```

88.      # Pied de page (Footer)
89.      #
90.      "footer": (
91.          "{% block footer %}\n"
92.          "  <div class=\"md-footer-meta md-typeset\">\n"
93.          "    <div class=\"md-footer-meta__inner\">\n\n"
94.          "      <div>\n"
95.          "        <a href=\"https://tahe.developpez.com\" target=\"_blank\">\n"
96.          "          https://tahe.developpez.com\n"
97.          "        </a>\n"
98.          "        <br>\n"
99.          "        Ce cours-tutoriel écrit par <strong>Serge Tahé</strong> est mis à disposition
du public selon les termes de la\n"
100.         "        <em>Licence Creative Commons Attribution - Pas d'Utilisation Commerciale -\n"
101.         "        Partage dans les Mêmes Conditions 3.0 non transposé</em>.\n"
102.         "      </div>\n\n"
103.         "    </div>\n"
104.         "  </div>\n"
105.         "{% endblock %}"
106.     ),

```

- lignes 87-106 : la définition du pied de page du site HTML. A été généré par Gemini à partir d'un exemple texte que je lui ai donné ;

La suite du fichier de configuration est la suivante :

```

107.      #
108.      # Configuration supplémentaire (Analytics, etc.)
109.      #
110.      "extra": {
111.          "analytics": {
112.              "provider": "google",
113.              "property": "G-XXXXXXX"
114.          }
115.      },
116.
117.      #
118.      # Détection du Titre du Document
119.      #
120.      "document_title": {
121.          # Styles ODT à considérer comme le titre principal du document (H1 global)
122.          "style_names": [
123.              "P1"
124.          ],
125.          # CSS appliquée à ce titre dans le Markdown généré
126.          "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
color: #2c3e50;"
127.      },

```

- lignes 110-115 : définition du marqueur Google Analytics (GA) qui va permettre de suivre les visites du site ;
- ligne 113 : mettez-là votre marqueur GA ;
- lignes 120-127 : le style du titre du document, celui qui est présent dans le document ODT /DOCX avant le premier titre de niveau 1. Le style indiqué ligne 123 change avec chaque Document ODT. En revanche il peut être constant ("Titre") pour les documents DOCX. Par défaut, le convertisseur logue les styles de tous les paragraphes du document ODT / DOCX qui précède le premier titre de niveau 1. Il faut donc faire une première exécution, repérer le style du paragraphe qu'on veut mettre en titre, puis mettre ce style ligne 123 ;
- ligne 126 : le style CSS que vous voulez donner au titre du document. Ce titre est dans la page « Accueil », la première page affichée lorsque le site s'ouvre ;

La suite du fichier de configuration est la suivante :

```

128.      #
129.      # Gestion du Code Source
130.      #
131.      "code": {

```

```

132.      # Mot-clé dans le style ODT pour identifier un bloc de code
133.      "style_keywords": [
134.          "code"
135.      ],
136.      # Langage par défaut si aucune détection ne fonctionne
137.      "default_language": "text",
138.
139.      # Style visuel du code enrichi (gras/italique/couleur préservés depuis ODT)
140.      "rich_line_height": "20px",
141.      "rich_font_family": "Consolas, 'Courier New', monospace",
142.      "rich_font_size": "15px",
143.
144.      # Règles de détection automatique du langage basé sur le contenu
145.      "detection_rules": {
146.          "csharp": [
147.              "using", "Console.WriteLine", "public static void Main", "WebMethod",
148.              "TryParse", "EventArgs", "String.Format", "System.Web.Services"
149.          ],
150.          "java": [
151.              "System.out.println", "public static void main(String", "package",
152.              "JUnitTest", "Class.forName", "PreparedStatement", "private static void",
153.              "private void", "getAgendaMedecinJour", "@PostConstruct", "@ResponseBody",
154.              "@RequestMapping", "getMedecin", "@Entity", "@Autowired", "@Bean",
155.              "Serializable", "getClient", "getCreneau", "getRv", "PostAjouterRv",
156.              "PostSupprimerRv", "@EnableJpaRepositories", "@Component",
157.              "getAgendaMedecinJour", "getResponse", "getMessagesForException", "getBase64", "ajouterRv",
158.              "Reponse", "getPartialViewAgenda", "setModelforAgenda", "ActionContext",
159.              "getActionContext", "PostLang", "PostUser", "PostGetAgenda", "@NotNull",
160.              "@EnableAutoConfiguration", "HttpSecurity"
161.          ],
162.          "html": [
163.              "<html>", "</div>", "<body>", "<script>", "href=", "<span>", "<p>",
164.              "<h2>", "<form>", "<table>", "<input"
165.          ],
166.          "sql": [
167.              "SELECT", "INSERT INTO", "UPDATE", "DELETE FROM", "WHERE",
168.              "CREATE TABLE", "ALTER TABLE"
169.          ],
170.          "python": [
171.              "def", "import", "print(", "from"
172.          ],
173.          "xml": [
174.              "<?xml", "<project", "<version>", "<configuration>", "<build>",
175.              "<dependency>", "<properties>", "<configuration>", "<start-class>"
176.          ],
177.          "javascript": [
178.              "use strict", "console.log", "let", "constructor", "async", "export"
179.          ],
180.          "php": [
181.              "<?php", "declare", "require"
182.          ],
183.          "vbscript": [
184.              "Option", "Dim", "Explicit"
185.          ]
186.      },
187.  },

```

- lignes 128-187 : cette configuration est utilisée pour les codes en « plain text » qui n'ont pas initialement de coloration syntaxique. Ces lignes visent à leur donner une selon le langage utilisé dans le bloc de code ;
- lignes 133-135 : les blocs de code sont repérés par leurs styles dans le document ODT / DOCX. Il peut y en avoir plusieurs. Ligne 134, on met tous les mots clés qui permettent de détecter un style de code. Dans mes documents, tous mes styles de code ont le mot « code » dans leur nom. Et aucun autre style n'a ce mot dans son nom. Donc je peux me contenter de mettre un unique mot clé. S'il y en a plusieurs on les met en les séparant par des virgules ligne 134 ;
- lignes 145-187 : les chaînes de caractères qui permettent d'identifier le langage d'un bloc de code.. Certains mots clés peuvent se retrouver dans divers langages. Le convertisseur sélectionne le langage pour lequel il a trouvé le plus de mots clés. Le principe est simple : vous regardez vos codes, sélectionnez des mots clés

caractéristiques et vous les mettez dans ces lignes pour le bon langage. Notez que vous ne pouvez pas mettre n'importe quoi comme nom de langage : il faut utiliser les noms reconnus par MkDocs ;

- lignes 139-142 : ces lignes concernent les blocs de code enrichis (gras, italique, soulignage, surlignage, couleur des caractères). Ces blocs ne subissent pas le traitement des blocs de code « plain text ». Ils sont rendus à l'identique dans le HTML ;
 - ligne 140 : fixe la hauteur de ligne des lignes de code ;
 - ligne 141 : fixe la police CSS du bloc de code ;
 - ligne 142 : fixe la taille des caractères ;
- ligne 137 : fixe le langage par défaut. Dans le cas d'un bloc de code « plain text », si aucun langage n'est détecté, le langage par défaut sera « text ». Pour MkDocs, cela signifie un bloc de code sans coloration syntaxique. C'est par exemple, le cas de résultats d'exécution ;

La configuration se termine de la façon suivante :

```
188.      # -----
189.      # Fichiers statiques à copier à la racine du site
190.      #
191.      "files_to_copy": [
192.          "googleXXXXXX.html",
193.          "robots.txt",
194.          "word-odt-vers-html-janv-2026.pdf"
195.      ],
196.
197.      # Mode debug : affiche les styles rencontrés avant le premier titre
198.      "debug": True
199.  }
```

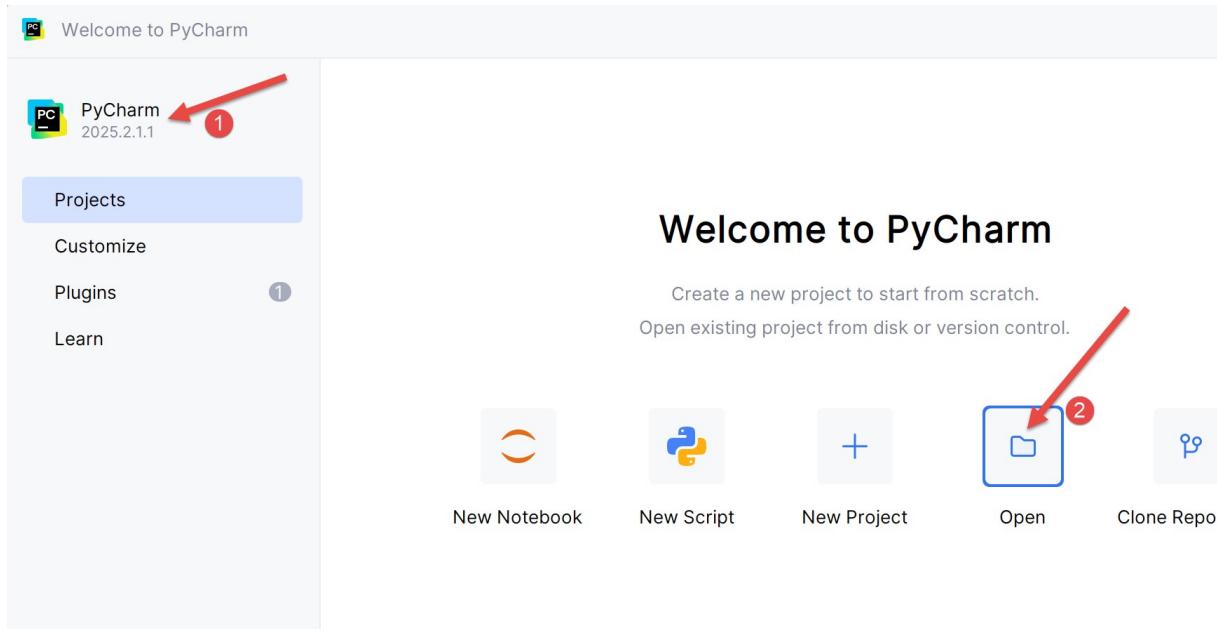
- lignes 191-195 : la liste des fichiers à copier à la racine du site HTML ;
 - lignes 192-193 : le marqueur Google Analytics du site et le fichier [robots.txt] qui autorise l'exploration du site par les moteurs de recherche ;
 - ligne 194 : le PDF de ce document rendu disponible au téléchargement ;
- ligne 198 : [debug] à True autorise les logs des styles des paragraphes qui précèdent le premier titre de niveau 1 du Document ODT / DOCX. Ce sont dans ces logs qu'on trouvera le style exact du paragraphe devant servir de titre dans la page d'accueil du site ;

Au final, lorsqu'on passe d'un Document ODT / DOCX à l'autre, on modifiera les lignes [6-20, 127-129, 201] et certaines lignes des chaînes clés des langages. Le reste ne change pas.

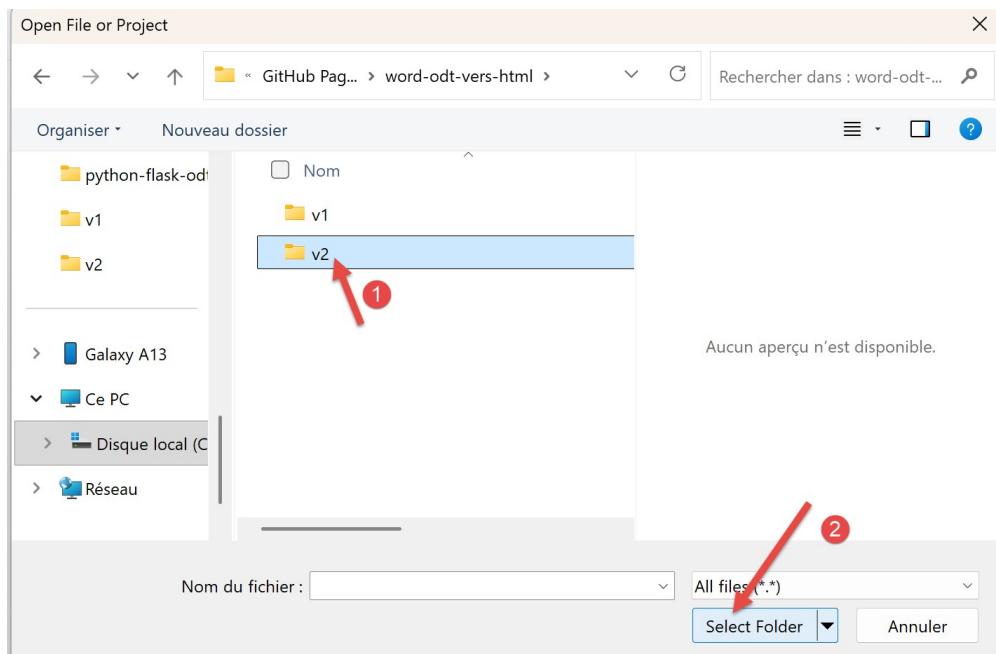
8. Utilisation du convertisseur ODT → HTML

On rappelle que le dossier de travail peut être trouvé |[ICI](#)|.

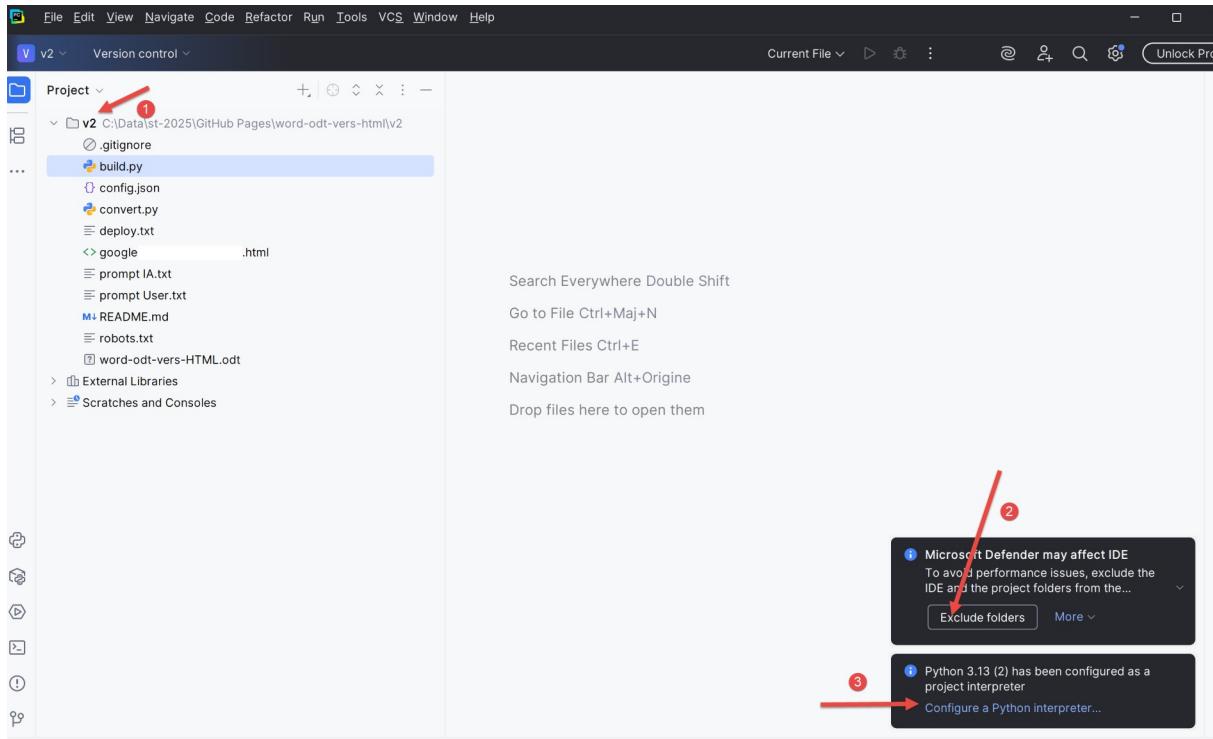
Lançons PyCharm :



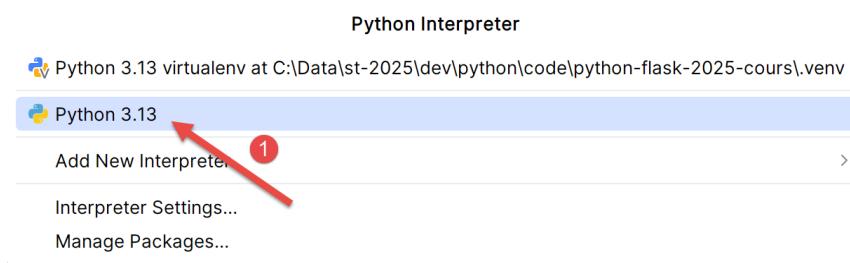
- En [2], ouvrez un nouveau projet ;



- En [1], sélectionner le dossier où vous avez sauvegardé le dossier de travail ;
- En [2], validez ce choix ;



- En [1], votre projet Python ;
- En [2], validez pour ne pas être embêté par le système de sécurité de Windows ;
- En [3], choisissez un interpréteur Python ;



- En [1], choisissez l'interpréteur Python que vous avez installé au paragraphe 5 ;

Le convertisseur Gemini / ChatGPT utilise des dépendances Python que vous devez installer. On va demander lesquelles à Gemini :

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

La réponse de Gemini est la suivante :

<https://tahe.developpez.com>

28/85

Commande d'installation rapide

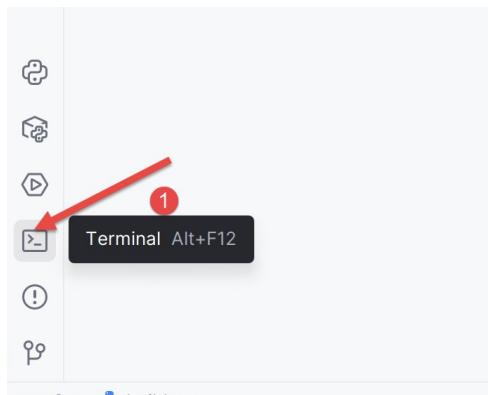
Vous pouvez installer tout le nécessaire en une seule commande dans votre terminal :

Bash



```
pip install odfpy unidecode mkdocs mkdocs-material
```

On va installer ces dépendances :



- En [1], ouvrez un terminal ;



- En [2], tapez la commande donnée par Gemini ;

La réponse est la suivante :

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

```

Sur mon PC, tout était déjà installé. Si ce n'est pas votre cas, toutes les dépendances demandées vont être installées.

Maintenant, nous pouvons utiliser le convertisseur. Dans le terminal ouvert, tapez la commande suivante :

```

1.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_odt_v356.py .\word-odt-
vers-html-janv-2026.odt .\config.py
2.      --- ODT to MkDocs Converter V356 ---
3.      Traitement de .\word-odt-vers-html-janv-2026.odt...
4.      Copié : google5179c0eaff293e02.html
5.      Copié : robots.txt
6.      Copié : word-odt-vers-html-janv-2026.pdf
7.      Copié : word-odt-vers-html-janv-2026.zip
8.      Scan des renvois...
9.      Génération 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='P1' Text='Convertir un document Word ou ODT vers un site
statique'
22.     >>> TITRE DOCUMENT TROUVÉ : Convertir un document Word ou ODT vers un site statique HTML
compatible MkDocs avec les IA Gemini 3 et 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, janvier 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='Ce site a été créé avec le convertisseur [Word
ou ...'
34.     [DEBUG PRE-H1] Tag=text:h Style='P5' Text='Introduction...'
35.     >>> CHAPITRE 1: 1 Introduction
36.     >>> CHAPITRE 1: 2 Les exemples de ce document
37.     >>> CHAPITRE 2: 2.1 Les listes
38.     >>> CHAPITRE 3: 2.1.1 Listes à puces
39.     >>> CHAPITRE 3: 2.1.2 Listes numérotées
40.     >>> CHAPITRE 3: 2.1.3 Listes mixtes 1
41.     >>> CHAPITRE 3: 2.1.4 Listes mixtes 2
42.     >>> CHAPITRE 2: 2.2 Les blocs de code
43.     >>> CHAPITRE 3: 2.2.1 Blocs de code enrichi (Eclipse, Visual Studio, ...)
44.     >>> CHAPITRE 3: 2.2.2 Blocs de code brut (plain text)
45.     >>> CHAPITRE 2: 2.3 Les liens
46.     >>> CHAPITRE 2: 2.4 L'enrichissement de texte
47.     >>> CHAPITRE 2: 2.5 Un titre peut être également enrichi.
48.     >>> CHAPITRE 2: 2.6 Les images
49.     >>> CHAPITRE 2: 2.7 Les caractères à protéger
50.     >>> CHAPITRE 2: 2.8 Les tableaux
51.     >>> CHAPITRE 2: 2.9 Les notes de bas de page
52.     >>> CHAPITRE 1: 3 Ce qui existe sur internet
53.     >>> CHAPITRE 1: 4 Le prompt initial à Gemini 3
54.     >>> CHAPITRE 1: 5 Créer un environnement de travail Python
55.     >>> CHAPITRE 1: 6 Le dossier de travail du convertisseur
56.     >>> CHAPITRE 1: 7 Le fichier de configuration du convertisseur
57.     >>> CHAPITRE 1: 8 Utilisation du convertisseur ODT → HTML
58.     >>> CHAPITRE 1: 9 Utilisation du convertisseur DOCX → HTML
59.     >>> CHAPITRE 1: 10 Construction du site HTML statique
60.     >>> CHAPITRE 1: 11 Examen du site HTML généré
61.     >>> CHAPITRE 2: 11.1 La barre supérieure du site
62.     >>> CHAPITRE 2: 11.2 Le bas de page du site

```

```

63.      >>> CHAPITRE 2: 11.3 La page d'accueil
64.      >>> CHAPITRE 2: 11.4 Les listes à puces
65.      >>> CHAPITRE 2: 11.5 Les listes numérotées
66.      >>> CHAPITRE 3: 11.5.1 Listes mixtes 1
67.      >>> CHAPITRE 3: 11.5.2 Listes mixtes 2
68.      >>> CHAPITRE 2: 11.6 Les blocs de code enrichis
69.      >>> CHAPITRE 3: 11.6.1 Exemple 1
70.      >>> CHAPITRE 3: 11.6.2 Exemple 2
71.      >>> CHAPITRE 3: 11.6.3 Exemple 3
72.      >>> CHAPITRE 2: 11.7 Les blocs de code brut (plain text)
73.      >>> CHAPITRE 3: 11.7.1 Exemple 1
74.      >>> CHAPITRE 3: 11.7.2 Exemple 2
75.      >>> CHAPITRE 3: 11.7.3 Exemple 3
76.      >>> CHAPITRE 2: 11.8 Autres blocs de code
77.      >>> CHAPITRE 2: 11.9 Les liens
78.      >>> CHAPITRE 2: 11.10 L'enrichissement de texte
79.      >>> CHAPITRE 2: 11.11 Les images
80.      >>> CHAPITRE 2: 11.12 Les caractères protégés
81.      >>> CHAPITRE 2: 11.13 Les tableaux
82.      >>> CHAPITRE 2: 11.14 Notes de bas de page
83.      >>> CHAPITRE 2: 11.15 Anomalies connues
84.      >>> CHAPITRE 2: 11.16 Autres cas
85.      >>> CHAPITRE 1: 12 Héberger le site HTML sur GitHub
86.      >>> CHAPITRE 1: 13 Assurer le suivi du site avec Google Analytics et Google Search Console
87.      >>> CHAPITRE 1: 14 Conclusion
88.      Terminé.

```

- Ligne 1 : la commande qui convertit le document ODT en site MkDocs [`python convert_odt_v356.py .\word-odt-vers-html-janv-2026.odt \config.py`]. Adaptez le numéro de version (ici 356) à la version que vous avez téléchargée. Le premier paramètre du convertisseur est le document ODT à convertir, le second le fichier de configuration du convertisseur ;
- lignes 4-7 : les fichiers que le convertisseur copie à la racine du site MkDocs qu'il crée ;
- lignes 10-34 : le débogage des styles des paragraphes qui précèdent le premier titre de niveau 1. Ces paragraphes vont constituer la page d'accueil. Un des paragraphes joue le rôle de titre de la page d'accueil et donc du site. C'est le paragraphe de la ligne 21. On note son style P1. On doit mettre ce style dans le fichier de configuration :

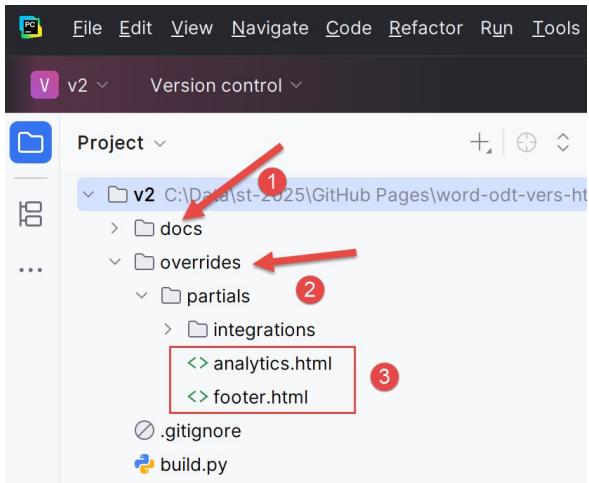
```

# -----
# Détection du Titre du Document
#
"document_title": {
    # Styles ODT à considérer comme le titre principal du document (H1 global)
    "style_names": [
        "P1"
    ],
    # CSS appliquée à ce titre dans le Markdown généré
    "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2; color: #2c3e50;"
},

```

- lignes 35-88 : j'ai demandé à l'IA de loguer tous les chapitres qu'il rencontrait ;

Cette exécution a modifié votre dossier de travail :



- En [1], [docs] est le site MkDocs que le convertisseur Gemini / ChatGPT a créé. Vous pouvez avoir la curiosité de le visiter ;
- en [2], un dossier [overrides] a été créé. Il va être utilisé par le constructeur [build] du site HTML ;
- en [3] : [analytics.html] sera utilisé pour le suivi du site par Google Analytics. [footer.html] est le bas de page que vous avez défini dans le fichier [config.py] ;

On pourrait utiliser le site MkDocs dès maintenant. La commande [python -m mkdocs serve] permet de le visualiser. Vous pouvez essayer :

```

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-cliquez sur le lien de la ligne 6. Le site MkDocs doit apparaître :

Convertir un document Word ou ODT vers un site statique HTML ... GitHub Search

Accueil

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini 3

Accueil

- 1. Introduction
- 2. Les exemples de ce document
- 3. Ce qui existe sur internet
- 4. Le prompt initial à Gemini 3
- 5. Créer un environnement de travail Python
- 6. Le dossier de travail du convertisseur Gemini 3
- 7. Le fichier de configuration du convertisseur Gemini 3
- 8. Utilisation du convertisseur Gemini 3
- 9. Construction du site HTML statique
- 10. Examen du site HTML généré
- 11. Héberger le site HTML sur GitHub
- 12. Assurer le suivi du site avec Google Analytics et Google Search Console

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.





Voici le fruit d'un long travail. Pour arrêter le serveur de MkDocs, faites simplement Ctrl-C dans le terminal qui l'a lancé.

9. Utilisation du convertisseur DOCX → HTML

La commande de conversion d'un document Word DOCX est très analogue à celle de la conversion d'un document LibreOffice ODT. Nous allons modifier le style du titre du document dans [config.py] :

```

1.      # -----
2.      # Détection du Titre du Document
3.      #
4.      "document_title": {
5.          # Styles ODT à considérer comme le titre principal du document (H1 global)
6.          "style_names": [
7.              "Titre"
8.          ],
9.          # CSS appliquée à ce titre dans le Markdown généré
10.         "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
color: #2c3e50;"
11.     },

```

- ligne 7 : mettez "Titre". C'est le style du document DOCX que vous allez convertir. Nous le verrons dans les lignes de débogage du convertisseur.

Toujours dans le terminal PyCharm, tapez la commande suivante :

```

1.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_docx_v18.py .\word-odt-
vers-html-janv-2026.docx .\config.py
2.      C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\convert_docx_v18.py:976: SyntaxWarning:
invalid escape sequence '\h'
3.          - REF Bookmark \h
4.          --- DOCX to MkDocs Converter V16 ---
5.          Copié : google5179c0eaff293e02.html
6.          Copié : robots.txt
7.          Copié : word-odt-vers-html-janv-2026.pdf
8.          Copié : word-odt-vers-html-janv-2026.zip

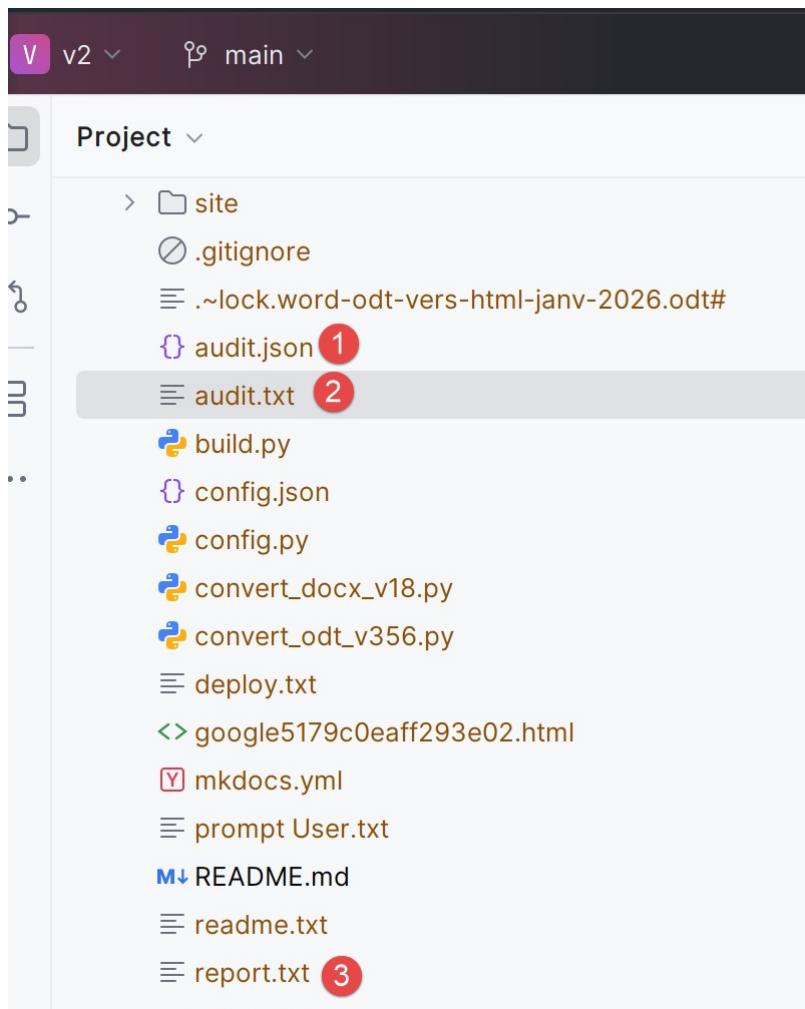
```

```

9.      [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
10.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
11.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
12.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
13.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
14.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
15.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
16.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
17.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
18.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
19.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
20.     [DEBUG PRE-H1] style=Titre heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<span>Convertir un document Word ou ODT vers un site statique HTML compatible Mk...'
21.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
22.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
23.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
24.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Serge Tahé</b><span>, janvier 2026</span>...'
25.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
26.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
27.     [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
28.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
29.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
30.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
31.     [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Ce site a été créé avec le convertisseur &#91;Word ou ODT - &gt; HTML&#93; cr...'
32.     [DEBUG PRE-H1] style=Titre1 heading=1 rebase=0 numId=1 ilvl=0 list=numPr:1/ordered
txt='<span>Introduction</span>...'
33.   Terminé. (audit.json, audit.txt, report.txt générés)

```

- ligne 1 : la commande est la suivante : [**python .\convert_docx_v18.py .\word-odt-vers-html-janv-2026.docx .\config.py**] (Adaptez le numéro de version (ici 18) à la version que vous avez téléchargée.) :
 - le premier paramètre [**.\convert_docx_v18.py**] est le convertisseur DOCX → HTML
 - le second paramètre [**.\word-odt-vers-html-janv-2026.docx**] est le nom du document DOCX à convertir ;
 - le troisième paramètre [**.\config.py**] est le fichier de configuration ;
- ligne 33 : le convertisseur signale que trois fichiers ont été générés :



Le fichier [audit.txt] est le suivant :

```
1. Version: V16
2. Paragraphs: 2029
3. Tables: 97
4. Images (blips): 2
5. Headings detected (raw): 53
6. Min heading level detected (raw): 1
7. Rebase offset applied: 0
8.
9. Top paragraph styles:
10.   - SourceCodenumrrot: 1054
11.   - StandardWW: 594
12.   - Standard: 146
13.   - Paragraphhedeliste: 113
14.   - SourceCodenumrotrsultats: 33
15.   - codenouveau: 28
16.   - Titre2: 25
17.   - Titre1: 14
18.   - Titre3: 14
19.   - StandardWWWW: 6
20.   - Textebrut: 1
21.   - Titre: 1
22.
23. List paragraphs:
24.   - with numPr: 1329
25.   - by style fallback: 49
```

```
26.      - not recognized: 0
```

- ligne 2 : le nombre de paragraphes du document Word ;
- ligne 3 : le nombre de tables ;
- lignes 9-21 : les styles trouvés dans le document ;
 - lignes 10, 14, 15 : le style des blocs de code. Un seul style aurait probablement suffi ;
 - lignes 11-12, 19 : le style des paragraphes standard. Un seul style aurait probablement suffi ;
 - lignes 16-18, 21 : les styles des titres du document. Ligne 21, on voit qu'un seul paragraphe a le style 'Titre'. C'est le titre du document qui précède le premier 'Titre1' ;

Cet audit du document Word est un bon moyen de juger de la qualité du document. Là je vois que j'ai utilisé trop de styles différents pour la même chose dans mon document Word.

Le fichier [audit.json] est identique au fichier [audit.txt] mais sous une forme JSON :

```
1.      {
2.          "version": "V16",
3.          "file": "word-odt-vers-html-janv-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.                 "Paragraphedeliste",
34.                 113
35.             ],
36.             [
37.                 "SourceCodenumrotrsultats",
38.                 33
39.             ],
40.             [
41.                 "codenouveau",
42.                 28
43.             ],
44.             [
45.                 "Titre2",
46.                 25
47.             ],
48.             [

```

```

49.         "Titre1",
50.         14
51.     ],
52.     [
53.         "Titre3",
54.         14
55.     ],
56.     [
57.         "StandardWWWW",
58.         6
59.     ],
60.     [
61.         "Textebrut",
62.         1
63.     ],
64.     [
65.         "Titre",
66.         1
67.     ]
68.   ]
69. }
```

Le fichier [report.txt] est celui-ci :

```

1.      [SUMMARY] Listes détectées via fallback "par style" (agrégé)
2.          - Paragraphedeliste -> level=1 type=unordered: 49
3.
4.      [SUMMARY] Blocs Word ignorés (agrégé)
5.          - <w:sectPr>: 1
```

Je ne l'ai pas compris...

Il est possible de ne demander que l'audit du document Word pour juger de sa qualité avec le paramètre [--audit] :

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

Dans ce cas, seul l'audit du document est fait. Le site MkDocs n'est pas généré.

Comme il a été montré précédemment, vous pouvez visualiser le site MkDocs généré par le convertisseur :

```

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    - 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-Cliquez sur l'URL de la ligne 5 :

Convertir un document Word ou ODT vers un site statique HTML à l'aide des IA Gemini 3 et ChatGPT 5.2

Accueil

1. Introduction

2. Les exemples de ce document

3. Ce qui existe sur internet

4. Le prompt initial à Gemini 3

5. Créer un environnement de travail Python

6. Le dossier de travail du convertisseur Gemini 3

7. Le fichier de configuration du convertisseur Gemini 3

8. Utilisation du convertisseur Gemini 3

9. Construction du site HTML statique

10. Examen du site HTML générée

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.

10. Construction du site HTML statique

A partir de maintenant il n'y a plus de différence entre documents ODT et DOCX. On travaille sur le site MkDocs construit par l'un ou l'autre des convertisseurs.

Maintenant construisons le site HTML. Cela se fait avec le script [build]. Toujours dans le terminal, tapez la commande suivante :

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\build.py
2. Démarrage de la construction du site MkDocs...
3. Construction réussie ! Le site est dans : C:\Data\st-2025\GitHub Pages\word-odt-vers-
html\v2\site
4. Ouverture de C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site\index.html dans le
navigateur...
```

- ligne 1 : la commande [python build] crée le site HTML à partir du site MkDocs ;
- ligne 4 : le site est affiché dans un navigateur ;

Le résultat est le même que pour le site MkDocs :

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini 3

Accueil

1. Introduction
2. Les exemples de ce document
3. Ce qui existe sur internet
4. Le prompt initial à Gemini 3
5. Créer un environnement de travail Python
6. Le dossier de travail du convertisseur Gemini 3
7. Le fichier de configuration du convertisseur Gemini 3
8. Utilisation du convertisseur Gemini 3
9. Construction du site HTML statique
10. Examen du site HTML généré
11. Héberger le site HTML sur GitHub
12. Assurer le suivi du site avec Google Analytics et Google Search Console

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.

- En [1], on voit que c'est une page HTML qui est affichée ;
- En [2], cette page correspond à la page d'accueil du site ;
- En [3], la table des matières du document HTML ;
- En [4], le titre du document a été formaté selon sa configuration dans [config.py] ;
- En [5], la première page affichée est [Accueil]. Le contenu de cette page est le contenu du Document ODT / DOCX qui **précède le premier chapitre de niveau Titre 1** ;

L'exécution de [build] a modifié votre dossier de travail :

Project

v2 C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2

- > docs
- > overrides
- > site 1
- > assets
- > images
- > javascripts
- > search
- > stylesheets

2

- <> 404.html
- <> assurer-le-suivi-du-site-avec-google-analytics.html
- <> ce-qui-existe.html
- <> conclusion.html
- <> construction-du-site-html.html
- <> creer-un-environnement-de-travail.html
- <> examen-du-site-html-genere.html

- En [1], un nouveau dossier [site] a été créé. Il contient votre site statique HTML ;
- En [2], les pages de votre site ;

11. Examen du site HTML généré

Nous allons examiner maintenant le rendu HTML de ce Document ODT / DOCX. On a déjà vu que le convertisseur respectait la table des matières.

11.1. La barre supérieure du site

Regardons la barre supérieure du site :



- En [1], le nom du site défini dans [config.py] ;
- En [2], l'icône qui permet de passer au mode sombre ou clair ;
- En [3], l'icône qui est un lien sur le dépôt GitHub où sera exporté le site HTML. Aussi défini dans [config.py] ;
- En [4], l'icône qui permet de cacher / montrer la table des matières ;

11.2. Le bas de page du site

Regardons maintenant le bas de page :



C'est le bas de page défini dans le fichier [config.py].

11.3. La page d'accueil

La page de titre du Document ODT / DOCX était la suivante :



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.

Cette page de titre du Document ODT / DOCX devient la page d'accueil du site HTML :

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.

Le convertisseur Gemini 3 / ChatGPT met dans la page Accueil tout ce qui précède, dans le Document ODT / DOCX, le premier titre de niveau 1, de style ‘Titre 1’ donc. Si vous y mettez des images comme ci-dessus, il va les afficher. Vous pouvez donc imaginer donner une couverture à votre site comme avec un livre réel. En [1], c'est le titre principal du document. Son rendu est contrôlé par les lignes suivantes du fichier de configuration [config.py] :

```

1.      # -----
2.      # Détection du Titre du Document
3.      #
4.      "document_title": {
5.          "# Styles ODT à considérer comme le titre principal du document (H1 global)
6.          "style_names": [
7.              "P1"
8.          ],
9.          "# CSS appliquée à ce titre dans le Markdown généré
10.         "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
color: #2c3e50;"
11.     },

```

- lignes [6-8] : la liste des styles possibles pour le titre de votre document. Quand je regarde ce document, le style LibreOffice de mon titre est ‘Titre principal’. Or le convertisseur Gemini ne le trouvait pas. Il a fait des logs des styles qu'il rencontrait et cela a affiché [P1]. C'est un gros souci avec LibreOffice : les noms affichés des styles ne correspondent pas aux noms internes utilisés par le logiciel. Ils ne sont là que pour s'adapter à la langue de l'utilisateur ;
- ligne 10 : une fois que le titre principal a été détecté, vous pouvez paramétrier son rendu. Moi je voulais une police de taille 28 (font-size: 28px;) et du gras (font-weight: bold) ;

Avec les images et le style du titre, vous pouvez faire une page d'accueil attrayante.

Le titre principal de votre document n'aura peut-être pas un des styles définis aux lignes [6-8]. Pour trouver le style de votre titre principal, utilisez la ligne suivante du fichier [config.py]

```
1.      "debug": True
```

Avec la valeur [true], le style des paragraphes qui précèdent le premier titre de niveau 1, donc les paragraphes de la page d'accueil, va être affiché lors de l'exécution du convertisseur Gemini / ChatGPT. Ainsi pour un autre document que celui-ci, j'ai obtenu les logs suivants :

```

1.      [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
2.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
3.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
4.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
5.      [DEBUG PRE-H1] Style='P3' (Clean='p3') | Texte='...'
6.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
7.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
8.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
9.      [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texte='...'
10.     [DEBUG PRE-H1] Style='P4' (Clean='p4') | Texte='Introduction au langage PHP7 par l'exemp...'
11.     >>> TITRE DOCUMENT DETECTÉ : Introduction au langage PHP7 par l'exemple
12.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
13.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
14.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
15.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
16.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
17.     [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texte='...'
18.     [DEBUG PRE-H1] Style='P5' (Clean='p5') | Texte='Serge Tahé, juillet 2019...'
19.     [DEBUG PRE-H1] Style='P5' (Clean='p5') | Texte='...'
20.     [DEBUG PRE-H1] Style='P6' (Clean='p6') | Texte='...'
21.     [DEBUG PRE-H1] Style='Heading 1' (Clean='heading 1') | Texte='Introduction au langage PHP 7...'
```

- ligne 10, le titre du document a le style ‘P4’ ;

Dans le fichier [config.py], j'ai alors mis les lignes suivantes :

```
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.     },
```

- ligne 3, le style que je cherchais ;

C'est pour cette raison que le débogueur affiche les lignes :

```
1. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Texte='Introduction au langage PHP7 par
l'exemp...'
2. >>> TITRE DOCUMENT DETECTÉ : Introduction au langage PHP7 par l'exemple
```

Il a rencontré le style 'P4' et affiche alors que le titre du document a été trouvé. Lorsque celui-ci a été trouvé, vous pouvez mettre la clé [debug] à [false] dans [config.py] :

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

Regardons maintenant la conversion du chapitre [Exemples] qui regroupe les exemples que le convertisseur Gemini / ChatGPT sait gérer :

11.4. Les listes à puces

Document ODT / DOCX :

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 ;

Cidessus, le texte [Listes à puces] est surligné parce que c'est une référence associée à un renvoi.

Document HTML :

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 ;

On remarque si le Document ODT / DOCX utilise différentes puces, le document HTML n'utilise qu'une sorte de puce.

11.5. Les listes numérotées

Document ODT / DOCX :

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 ;

Document HTML :

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. Listes mixtes 1

Document ODT / DOCX

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 ;

Document HTML

2.1.3. Listes mixtes 1

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

Là également, il y a parfois des différences entre les types de puces utilisés.

11.5.2. Listes mixtes 2

Document ODT / DOCX

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 ;

Document HTML

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. Les blocs de code enrichis

Les blocs de code enrichi sont rendus à l'identique dans le HTML (en-dehors de la couleur de fond). Voici trois exemples :

11.6.1. Exemple 1

Document ODT / DOCX

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 ClassPathXmlAppli
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>());
19.         Appartement appart01 = ctx.getBean(Appartement.class);
20.         ...
}
```

Rendu HTML

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.clas
17         Personne p02 = ctx.getBean("personne_02", Personne.clas
18         List<Personne> club = ctx.getBean("club", new ArrayList<
19         Appartement appart01 = ctx.getBean(Appartement.class);
```

11.6.2. Exemple 2

Document ODT / DOCX :

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" % (variabl
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. ...
```

Rendu HTML

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
```

11.6.3. Exemple 3

Document ODT / DOCX

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 constructeur
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. ...
```

Rendu HTML

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. Les blocs de code brut (plain text)

Les blocs de code brut trouvés dans le Document ODT / DOCX sont colorés syntaxiquement par MkDocs selon le langage trouvé dans le bloc de code. Pour aider le convertisseur à trouver le bon langage, on a mis des « chaînes clés » dans le fichier `[config.py]` pour chaque langage. Le convertisseur compte les « chaînes clés » trouvées. Il associe alors le bloc de code au langage ayant le plus de « chaînes clés » trouvées.

Voyons quelques exemples.

11.7.1. Exemple 1

Document ODT / DOCX (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>());
19.         Appartement appart01 = ctx.getBean(Appartement.class);
20.         ...
21.     }
22. }
```

Rendu HTML

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>());
19         Appartement appart01 = ctx.getBean(Appartement.class);
20         ...
21     }
22 }
```

Dans le document HTML on voit que le code Java a été coloré syntaxiquement.

11.7.2. Exemple 2

Document ODT / DOCX (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.ContextSectionHandler" />
7.                  <section name="objects" type="Spring.Context.Support.DefaultObjectFactorySectionHandler" />
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" value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;" />
19.                     <constructor-arg index="2" value="select limite, coeffr, coefficient, idmetier from metier where idmetier = " />
20.                 </object>
21.                 <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
22.                     <constructor-arg index="0" ref="dao" />
23.                 </object>
24.             </objects>
25.         </spring>
26.     </configuration>

```

Rendu HTML

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.ContextSectionHandler" />
7              <section name="objects" type="Spring.Context.Support.DefaultObjectFactorySectionHandler" />
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" value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;" />
19                 <constructor-arg index="2" value="select limite, coeffr, coefficient, idmetier from metier where idmetier = " />
20             </object>
21             <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
22                 <constructor-arg index="0" ref="dao" />
23             </object>
24         </objects>
25     </spring>
26 </configuration>

```

11.7.3. Exemple 3

Document ODT / DOCX (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.              }
```

Rendu HTML

HTML

```
1 <!DOCTYPE HTML>
2 <HTML>
3     <head>
4         <title>Laragon</title>
5
6         <link href=<a href="view-source:https://fonts.cse.google.com/cdnfonts.css?l=fr&v=1" style="display:none"></a>>
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. Autres blocs de code

Document ODT / DOCX

Un résultat d'exécution avec une première ligne qui ne commence pas à 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. [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'>
```

Rendu HTML

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'>
```

Un bloc de code non numéroté dans ODT le reste dans le HTML :

Document ODT / DOCX

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
```

Rendu HTML

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. Les liens

Document ODT / DOCX

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](#)

Rendu HTML

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. L'enrichissement de texte

Document ODT / DOCX

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.

Rendu HTML

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**.

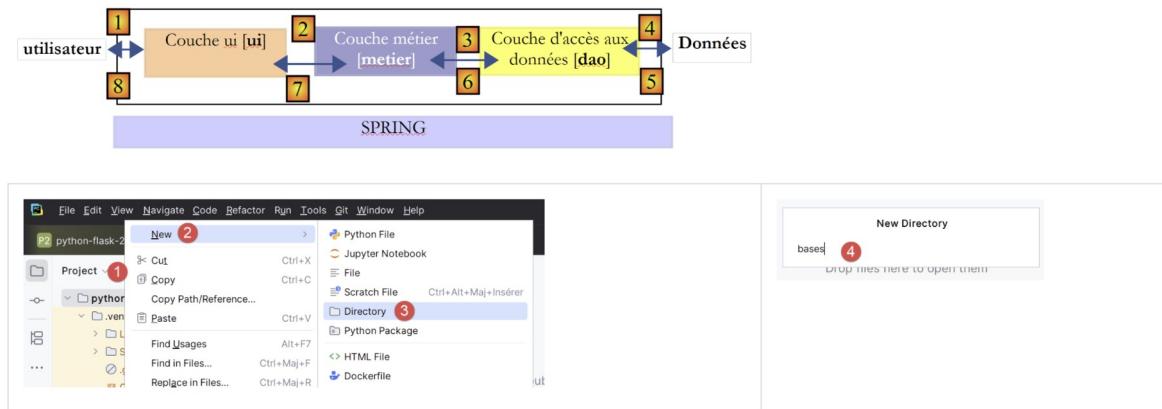
Il gère également les bordures supérieure et inférieure des paragraphes.

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

11.11. Les images

2.5. Les images

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



Il faut noter que le convertisseur Gemini / ChatGPT respecte les redimensionnements des images, faits dans le Document ODT / DOCX.

11.12. Les caractères protégés

Document ODT / DOCX

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

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

Rendu HTML

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

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

Document ODT / DOCX (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).
```

Rendu HTML

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 ab
16
17 ## ✨ Fonctionnalités
18
19 Le script `convert.py` effectue les actions suivantes :
20
```

- le code MarkDown a été préservé ;

11.13. Les tableaux

Document ODT / DOCX

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 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, 11. type(variable))) 12.</pre>

Rendu HTML

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. Notes de bas de page

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. Anomalies connues

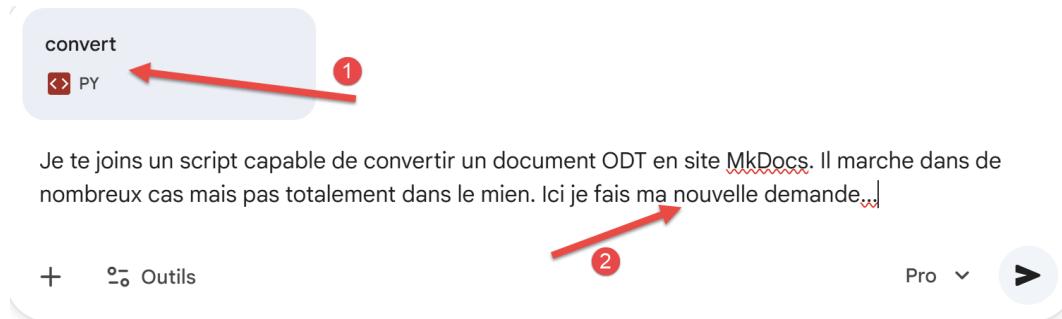
Certaines anomalies ont été repérées mais elles peuvent être corrigées en modifiant l'ODT / DOCX :

- **les blocs de code doivent être suivis d'une ligne vide** sinon le bloc de code peut être mal rendu. Le cas repéré est un code suivi immédiatement d'un titre sans être séparé de celui-ci par une ligne vide ;
- les listes à puces ne peuvent pas voir de bordure inférieure. **Pour avoir celle-ci, il faut ajouter une ligne vide derrière le dernier élément de la liste** ;
- **les listes à puces doivent être hiérarchisées.** Ainsi une liste de niveau 2 doit être toujours incluse dans une liste de niveau 1, sinon la liste de niveau 2 est rendue comme étant du code ;

Au fil des versions du convertisseur, certaines de ces anomalies sont appelées à disparaître. Les trois précédentes sont évitables en corrigeant le document source.

11.16. Autres cas

Si votre document utilise d'autres particularités que celles évoquées précédemment, il est hautement probable que celles-ci ne seront pas prises en compte par le convertisseur Gemini / ChatGPT . Que faire alors ? Vous pouvez faire part de votre nouvelle demande à l'une des IA en lui donnant le convertisseur actuel :



- En [1], je joins le convertisseur de ce document ;
- En [2], je fais ma nouvelle demande ;

Vous allez partir probablement pour de nombreuses itérations. Lorsqu'une version est stable, notez son numéro pour pouvoir le redonner à l'IA en cas de régression. Il est conseillé également de faire une copie de chaque version stable. Un inconvénient majeur des deux IA est qu'elles régressent assez facilement. Il suffit de lui demander une nouvelle fonctionnalité pour que l'IA casse le code qui marchait auparavant. D'où l'importance de noter le numéro de version des versions stables afin de pouvoir y revenir. En janvier 2026, il m'a semblé que ChatGPT 5.2 avait moins tendance à régresser que Gemini 3.

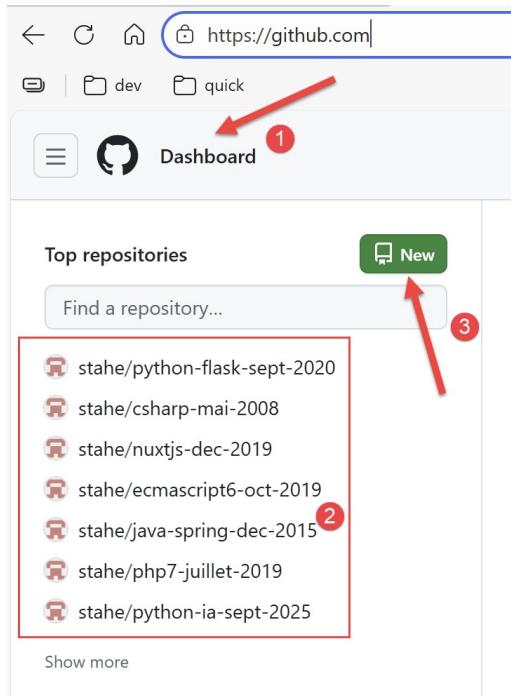
12. Héberger le site HTML sur GitHub

C'est Gemini lui-même qui m'a proposé d'héberger le site HTML produit par ses deux scripts sur GitHub⁴. Je ne savais pas que c'était possible. GitHub est un site qui héberge des projets de développement. Y exporter des cours de programmation paraît naturel.

Il faut d'abord que vous ayez un compte [GitHub](#). Si besoin est, créez le.

Connectez-vous à votre compte GitHub :

⁴Note de bas de page pour GitHub



- En [2], vos dépôts (repository) existants si vous en avez ;
- En [3], créez un nouveau dépôt GitHub ;

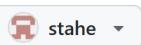
Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

1 General

Owner *



Repository name *

word-odt-vers-html-janv-2026

1

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

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

Description

Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini

84 / 350 characters

2

2 Configuration

Choose visibility *

Choose who can see and commit to this repository

Public

Add README

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

Off

On

Add .gitignore

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

No .gitignore

Add license

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

No license

3

Create repository

- En [1], utilisez le nom que vous avez mis dans [config.py] :

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

- En [2], vous pouvez également mettre la même chose que dans [config.py] :

1. "site_description": "Convertir un document Word ou ODT vers un site statique HTML à l'aide de les IA Gemini 3 et ChatGPT 5.2",

- En [3], validez la création de votre dépôt GitHub ;

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

Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.

[Create a codespace](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/stahe/word-odt-vers-html-janv-2026.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# word-odt-vers-html-janv-2026" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/stahe/word-odt-vers-html-janv-2026.git
git push -u origin main
```

Il est conseillé pour chaque dépôt GitHub de créer un fichier [README.md] qui sera affiché à la place de l'écran ci-dessus. Il doit être écrit en MarkDown, un truc pas simple. On va donc écrire un fichier texte avec dedans ce qu'on voudrait voir dans le README. Puis on demandera à l'IA de nous donner le README.md qui correspond.

Le fichier texte [readme.txt] sera le suivant :

1. On se propose dans cet article de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML.
- 2.
3. Ce convertisseur a été construit par l'IA Gemini 3.
- 4.
5. Lorsque le document ODT convient, le convertisseur produit un site HTML via MkDocs qui a l'aspect professionnel des sites produits par Pandoc.
- 6.
7. J'ai utilisé Gemini 3 avec un abonnement pro.

Le prompt à Gemini est alors le suivant :

readme

1

TXT

Peux-tu produire le fichier GitHub README.md correspondant au texte que je te donne. Tu commenceras le README par un lien sur le site HTML [<https://stahe.github.io/word-odt-vers-html-janv-2026/>]

2

+ ⚙️ Outils

Pro ▾ >

- En [1], le fichier attaché [readme.txt] ;
- En [2], le prompt ;

Copiez la réponse que Gemini vous donne dans le fichier [README.md] de votre dossier de travail. Voici la réponse que Gemini m'a donnée :

```

1.      # Convertisseur Word/ODT vers Site HTML (MkDocs)
2.
3.      🔗 **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-vers-html-janv-2026/)***
4.
5.      ---
6.
7.      ## 📄 Description
8.
9.      On se propose dans ce projet de mettre à disposition du lecteur un convertisseur Python de documents Word ou ODT vers un site statique HTML.
10.
11.     Lorsque le document ODT convient, le convertisseur produit un site HTML via **MkDocs** qui a l'aspect professionnel des sites produits par Pandoc.
12.
13.     ## 🎯 Contexte de création
14.
15.     Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abonnement pro). Il est le résultat d'itérations successives pour gérer finement la structure des documents ODT (OpenDocument Text).
16.
17.     ## ✨ Fonctionnalités
18.
19.     Le script `convert.py` effectue les actions suivantes :
20.
21.     * **Conversion ODT vers Markdown** : Analyse le fichier ` `.odt` (XML) pour en extraire la structure.
22.     * **Gestion des Titres** : Génère automatiquement la Table des Matières (TOC) et la navigation latérale.
23.     * **Blocs de Code** : Détection automatique des langages, coloration syntaxique et **gestion précise de la numérotation des lignes** (attributs `start-value`).
24.     * **Listes** : Support des listes à puces et numérotées avec indentation correcte.
25.     * **Mise en forme** : Support du *gras*, *italique*, *souligné* et du *surlignage* (avec respect des couleurs d'origine).
26.     * **Images** : Extraction et intégration automatique des images contenues dans le document.
27.     * **Configuration** : Personnalisation via un fichier `config.py` (Bas de page, Google Analytics, etc.).
28.
29.     ## 🛠 Installation
30.
31.     ### Prérequis
32.
33.     * Python 3.x
34.     * Les bibliothèques suivantes :
35.
36.     ```bash
37.     pip install odfpy unidecode mkdocs mkdocs-material
38.
39.     ```
40.
41.     ### Structure du projet
42.
43.     Assurez-vous d'avoir les fichiers suivants :
44.
45.     * `convert.py` : Le script de conversion.
46.     * `config.py` : Votre fichier de configuration.
47.     * `votre-document.odt` : Le document source.
48.
49.     ## 💡 Utilisation
50.
51.     1. **Conversion**
52.     Lancez le script en indiquant le fichier ODT source et le fichier de configuration :
53.     ```bash

```

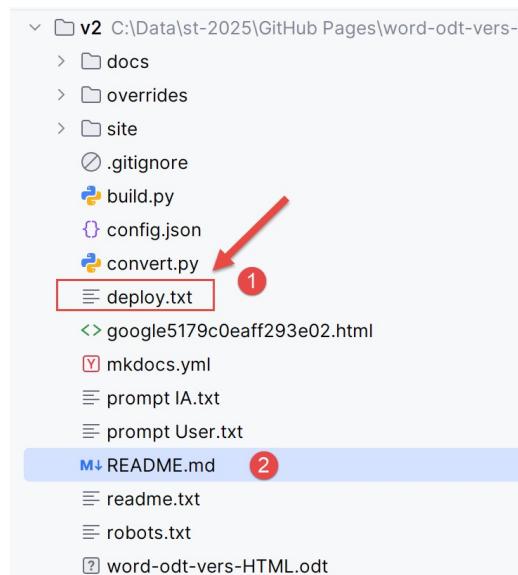
```

54.     python convert.py votre-document.odt config.py
55.
56.     ...
57.
58.
59.     *Cela va générer un dossier `docs/` contenant les fichiers Markdown et un fichier
`mkdocs.yml`.*  

60.     2. **Prévisualisation**
61.         Pour voir le site en local :
62.             ```bash
63.             mkdocs serve
64.
65.             ...
66.
67.
68.     3. **Génération**
69.         Pour construire le site statique (dossier `site/`) :
70.             ```bash
71.             mkdocs build
72.
73.             ...
74.
75.
76.
77.     ## 🛡 Configuration (`config.py`)
78.
79.     Le fichier `config.py` permet de contrôler l'apparence du site :
80.
81.     * **mkdocs** : Paramètres généraux du site (titre, description, thème Material).
82.     * **footer** : Code HTML complet pour personnaliser le pied de page.
83.     * **code** : Règles de détection des langages pour la coloration syntaxique.
84.     * **extra** : Configuration de Google Analytics (GA4).
85.
86.     ## 📄 Licence
87.
88.     Ce cours tutoriel écrit par **Serge Tahé** est mis à disposition du public selon les termes de
la :
89.     *Licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes
Conditions 3.0 non transposé.*
```

C'est un README très complet qu'il m'a fourni. C'est parce que Gemini connaît très bien ce projet sur lequel on travaille depuis des semaines. Vous aurez sans doute un [README.md] moins détaillé.

Maintenant revenons à notre dossier de travail :



- En [2], le fichier README que vous venez de modifier ;
- En [1], le fichier [deploy.txt] explique comment exporter votre site HTML sur votre dépôt GitHub ;

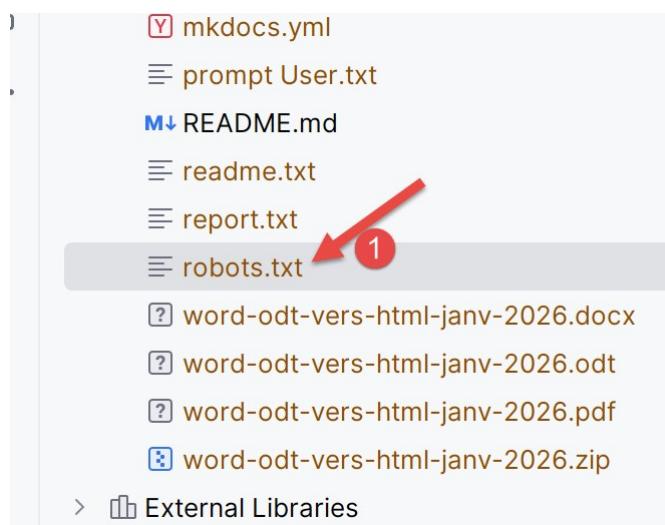
Le contenu du fichier [deploy.txt] est le suivant :

```
1. git init
2. git branch -M main
3. # vérifier .gitignore avant !
4. git add .
5. # premier commit
6. git commit -m "Initial commit: Source MkDocs"
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
```

C'est la suite de commandes qui va exporter votre site HTML sur votre dépôt GitHub. Vous aurez à modifier la ligne 7 avec l'URL de votre propre dépôt GitHub, présente dans le fichier [config.py] :

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

Vous devez vérifier également une autre URL présente dans le fichier [robots.txt] :



Le contenu du fichier [robots.txt] est le suivant :

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

En ligne 3, mettez l'URL de votre site, la même que dans le fichier [config.py] :

```
1. # URL de publication du site (ex: GitHub Pages)
2. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

Le fichier [robots.txt] n'est pas utilisé lors de la construction locale du site, mais il le sera lorsqu'il sera hébergé sur GitHub.

La suite de commandes de [deploy.txt] utilise un logiciel appelé Git. Vous devez l'installer [[Git - Install for Windows](#)].

Ceci fait, vérifiez le fichier s'appelant [.gitignore] dans votre dossier de travail. Il indique à Git les fichiers qu'il doit ignorer. Mon fichier [.gitignore] est le suivant :

```
1.      # 1. On ignore TOUS les fichiers et dossiers du projet
2.      *
3.
4.      # 3. On fait une exception (!) pour le seul fichier que vous voulez garder
5.      !README.md
```

Il est extrêmement simple. On ignore tous les fichiers (ligne 2) sauf le fichier [README.md] (ligne 5). GitHub est fait pour héberger les projets de développement. C'est en général tout le projet de développement qui est exporté sur GitHub. Nous on cherche simplement à exporter un site HTML, pas un projet de développement. Le seul fichier qu'on veut exporter sur notre dépôt GitHub est le fichier [README.md] qui explique aux visiteurs ce que contient notre site HTML.

Maintenant, dans votre terminal, tapez la séquence de commandes suivantes dans l'ordre indiqué par [deploy.txt] en allant jusqu'à la commande 8. N'exécutez pas la commande 9 pour l'instant.

```
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-vers-html\v2> git branch -M main
4.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git add .
5.      PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git commit -m "Initial commit: Source
MkDocs"
6.      [main (root-commit) 7cba5b1] 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-vers-html\v2> git remote add origin
https://github.com/stahe/word-odt-vers-html-janv-2026.git
10.     PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git push -u origin main
11.     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'.
```

Faites un Ctrl-clic sur l'URL de la ligne 17. Cela va vous emmener sur votre dépôt GitHub :

The screenshot shows a GitHub repository page. At the top, the URL <https://github.com/stahe/word-odt-vers-html-janv-2026> is displayed in the browser's address bar. Below the address bar, the repository navigation menu includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Code' tab is selected. On the right side of the header, there are 'Pin' and 'Watch' buttons. The repository name 'word-odt-vers-html-janv-2026' is shown with a 'Public' status. Below the header, there are buttons for 'main' (branch), '1 Branch', '0 Tags', 'Go to file', 'Add file', and 'Code'. The repository history shows one commit by 'stahe' titled 'Initial commit: Source MkDocs' made 4 minutes ago. The repository files list includes 'README.md' with a note 'Initial commit: Source MkDocs' and a timestamp '4 minutes ago'. The main content area displays the 'README' file, which contains the title 'Convertisseur Word/ODT vers Site HTML (MkDocs)' and a link 'Voir le site de démonstration générée'. Below this, there is a 'Description' section with text about the project's purpose and Pandoc's role.

- En [1], l'URL de votre dépôt GitHub ;
- En [2], le nouveau README.md ;

Maintenant passons à la commande 9 du fichier [deploy.txt]. C'est elle qui exporte le site HTML sur GitHub :

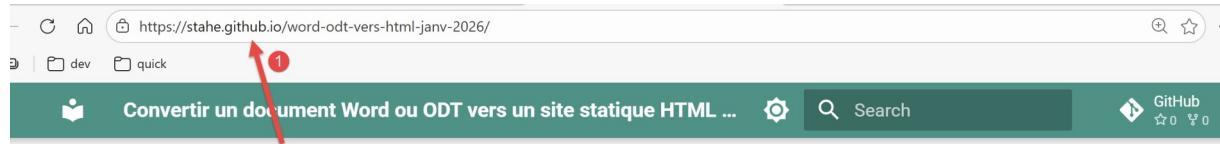
```

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    - Doc file 'les-exemples.md' 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 '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.      To https://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/
```

Faites un Ctrl-clic sur l'URL de la ligne 21. Cela va vous amener à votre nouveau site HTML sur GitHub :



Convertir un document Word ou
ODT vers un site statique HTML
à l'aide de l'IA Gemini

Accueil

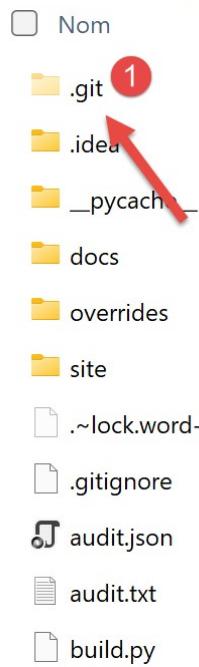
- 1. Introduction
- 2. Les exemples
- 3. Ce qui existe
- 4. Le prompt initial
- 5. Créer un environnement de travail
- 6. Le dossier de travail
- 7. Le fichier de configuration [config.json]
- 8. Utilisation du convertisseur
- 9. Construction du site HTML
- 10. Examen du site HTML généré
- 11. Héberger le site HTML sur GitHub

Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

- En [1], on voit que vous affichez un site web sur GitHub ;

Il est extrêmement facile de se tromper dans l'exécution des commandes de [deploy.txt]. Il est alors difficile de revenir en arrière car Git garde la mémoire de ce qu'on a fait (mal). Pour repartir de zéro, visualisez le dossier de travail :



- En [1], supprimez le dossier [.git] ;

Revenez ensuite dans PyCharm et recommencez la série de commandes de [deploy.txt].

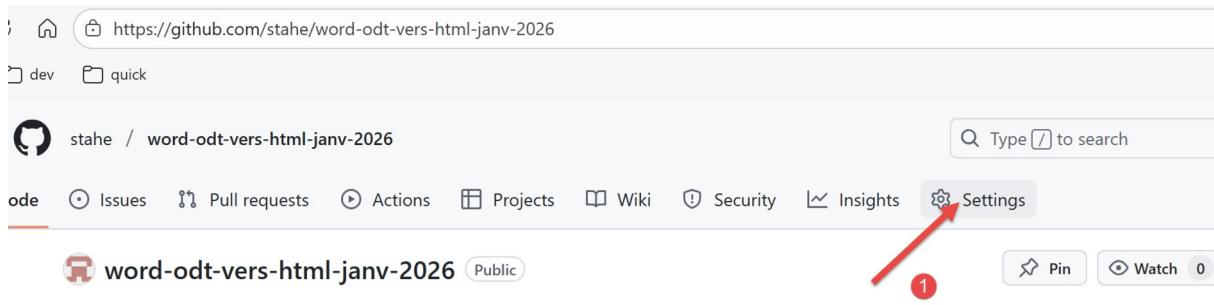
Que faire si vous modifiez votre Document ODT / DOCX ? Faites les 3 choses suivantes :

1. convertissez de nouveau votre Document ODT / DOCX avec [convert] ;
2. construisez le site HTML avec [build] ;
3. exportez le site HTML sur GitHub avec la commande [python -m mkdocs gh-deploy]. Cette commande suffit tant que vous ne modifiez pas le fichier [README.md]. Si vous modifiez le fichier [README.md], vous aurez un peu plus de commandes à émettre :

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

Si vous avez seulement modifié le README, seules les commandes 1, 3, 4 sont nécessaires. La commande 5 est inutile si vous avez déjà déployé le site HTML et qu'il n'a pas bougé depuis.

Que faire si vous voulez repartir de zéro parce que les choses sont parties en vrille ? Vous pouvez supprimer votre dépôt GitHub et refaire toutes les opérations du chapitre 12. L'option de suppression d'un dépôt GitHub est bien cachée :



- En [1], allez dans les paramètres de configuration du dépôt ;

Danger Zone

The 'Danger Zone' section contains the following options:

- 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](#)

Allez tout en bas de la page des [settings]. Vous y trouverez le bouton pour supprimer votre dépôt [2].

13. Assurer le suivi du site avec Google Analytics et Google Search Console

Si vous hébergez votre site sur GitHub vous serez probablement intéressé par connaître le nombre de visiteurs du site. Il vous faut pour ça un marqueur Google Analytics (GA). Nous demandons à [ChatGPT](#) comment créer un marqueur GA :

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-XXXXXX`). 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.
- En [1], mettez l'URL de votre site web sur 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-XXXXXXX

2

👉 C'est ce marqueur Google Analytics

Il identifie de manière unique votre site dans GA4.

- En [2], notez le marqueur de votre site ;

Mettez ce marqueur dans le fichier de configuration [config.py] :

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

Ligne 4, inscrivez votre marqueur. MkDocs le mettra automatiquement dans toutes vos pages HTML lorsqu'il va générer le site HTML. Vous n'avez donc pas besoin de suivre les autres instructions de ChatGPT qui montrent comment installer le marqueur GA dans vos pages HTML.

On va demander à ChatGPT des instructions précises pour vérifier que le marqueur fonctionne :

donne moi des instructions plus précises pour vérifier que le marqueur fonctionne. Donne moi l'URL à utiliser.

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/)

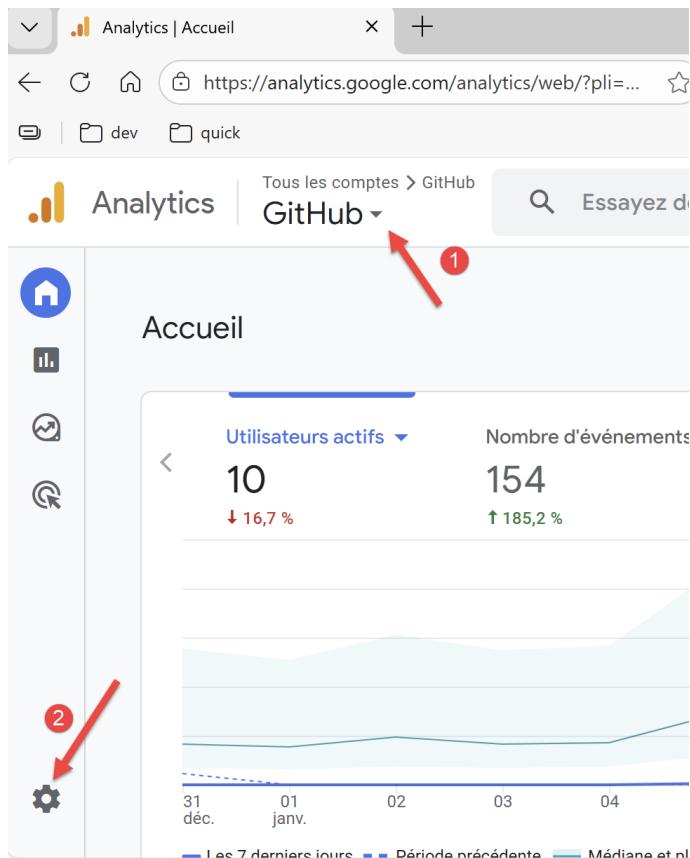
Connectez-vous avec le compte Google qui possède la propriété GA4.

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.

L'étape précédente n'est pas assez claire. Voici quelques copies d'écran pour vous aider :



- En [1], sélectionnez le compte que vous voulez suivre (au cas où vous en auriez plusieurs) ;
- En [2], sélectionnez le bouton d'administration ;

Tous les comptes > GitHub

GitHub

Créer

Administration

Mes préférences

Assistant de configuration

Paramètres du compte

- Compte
- Propriété
- Collecte et modification des ...
- Affichage des données
- Associations de produits

PARAMÈTRES DU COMPTE

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

Détails de la propriété Gestion des accès à la propriété

Collecte et modification des données

Flux de données Collecte des données

- En [3], sélectionnez [Flux de données] ;

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.
--------------------------------	---	-------------	---

- En [4], cliquez sur le flux de données ;

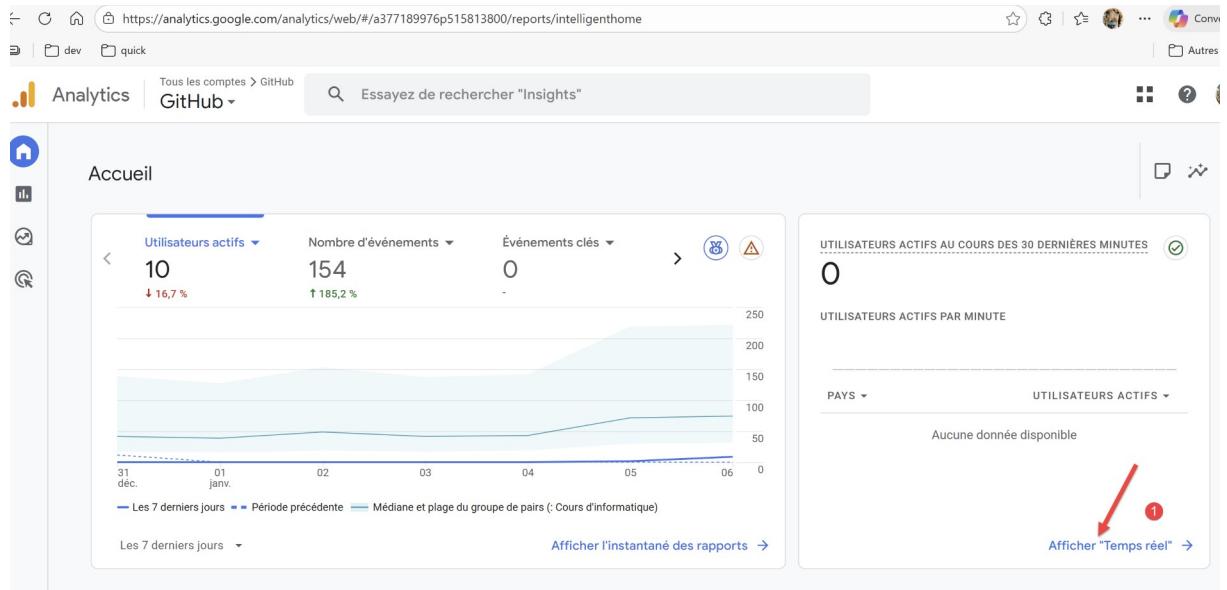
Détails du flux Web

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

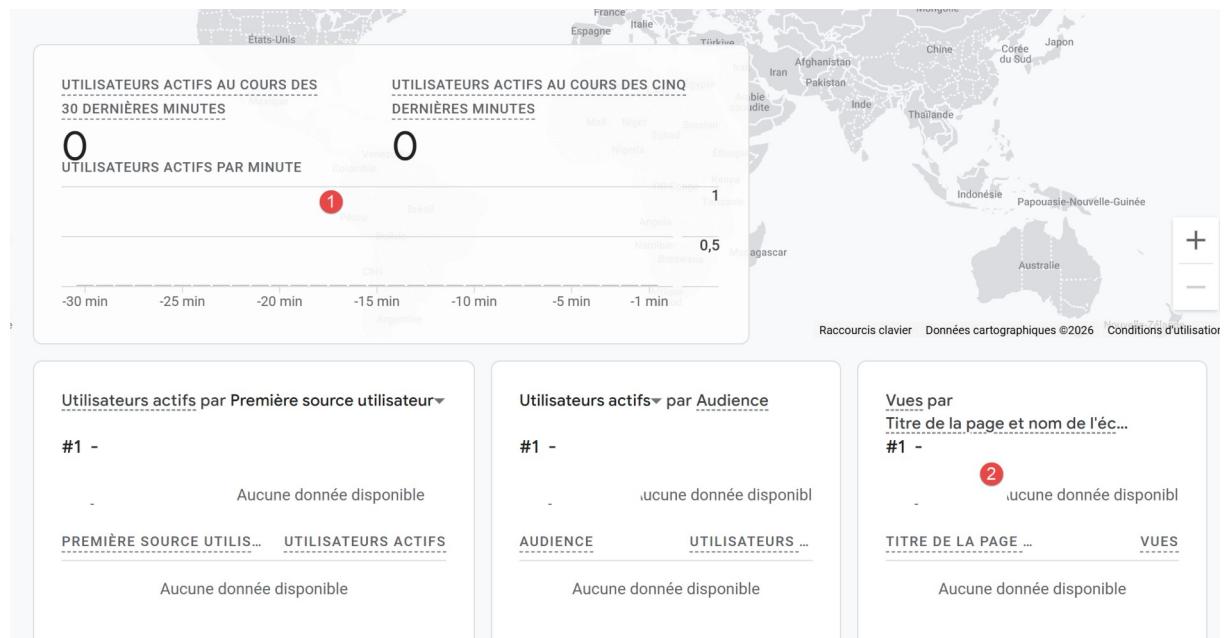
Détails du flux	
NOM DU FLUX	URL DE FLUX
GitHub	https://stahe.github.io
ID DE FLUX	13112486249
ID DE MESURE	G-EBZC: 5

- En [5], vous retrouvez le marqueur GA4 que vous avez créé ;
- En [6], l'ID de flux. C'est une notion différente de l'ID de mesure ;

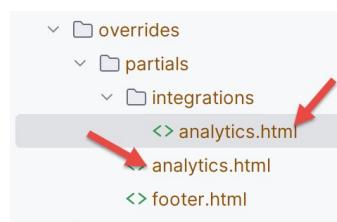
Revenons à l'accueil de GA :



- En [1], demandez l'affichage temps réel des visites ;



- En [1], vous voyez vos visiteurs ;
- En [2], la page qui a été visitée. J'ai demandé au convertisseur Gemini d'associer le nom du site plutôt que le nom de page au marqueur GA4. C'est le fichier [analytics.html] qui assure ce changement de nom associé au GA4 :



Maintenant on peut continuer avec les explications de ChatGPT :

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.



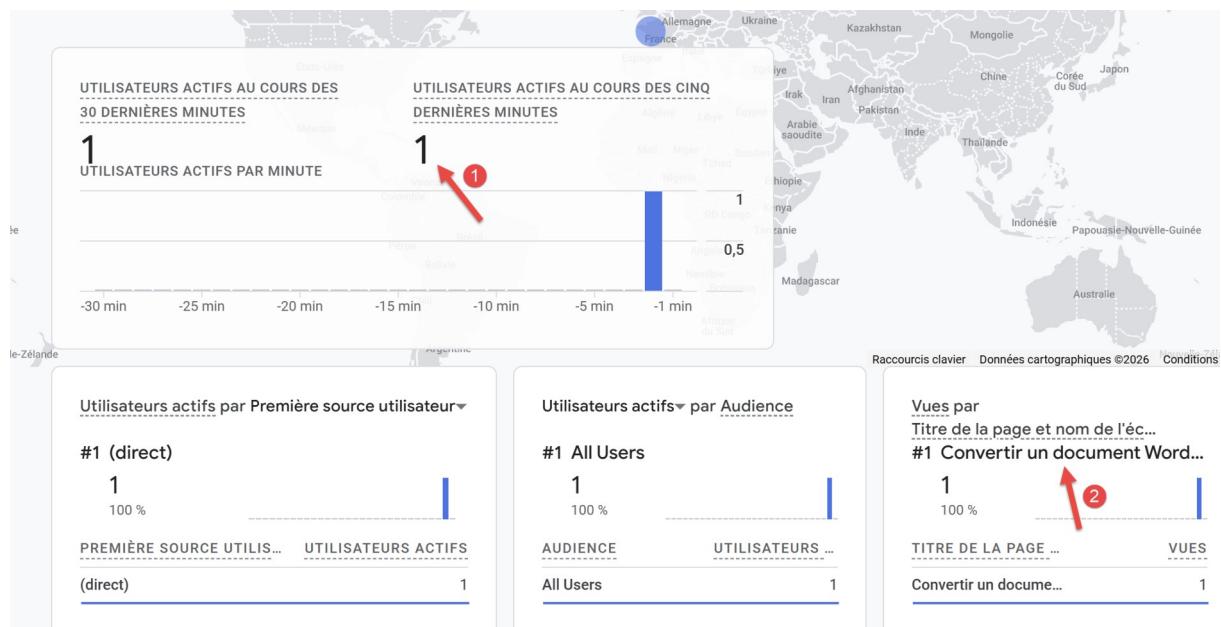
5. Résultat attendu

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.

J'ai fait ce qui est demandé en [4] sans passer par une fenêtre privée. Et j'ai obtenu ceci :



Le marqueur GA4 de Google Analytics est bien fonctionnel.

Maintenant nous allons présenter un autre outil de suivi de votre site [Google Search Console]. Demandons à ChatGPT à quoi sert cet outil :

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.

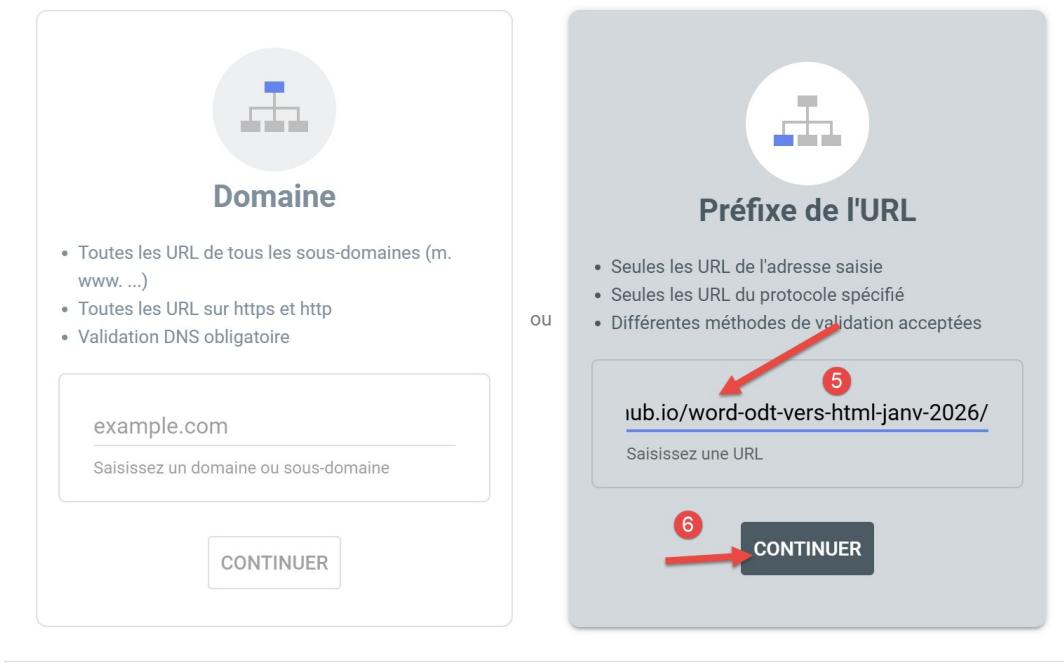
Je ne me sers quasiment jamais de cet outil. Je ne le fais que pour inciter le moteur de recherche Google à l'explorer et à l'indexer afin que des internautes puissent le trouver.

L'URL de l'outil [<https://search.google.com/search-console>] :

The screenshot shows the Google Search Console interface. On the left, there's a sidebar with a dropdown menu set to 'https://stahe.github.io/...', a 'Vue d'ensemble' button, and an 'Insights' button. A red arrow labeled '1' points to the dropdown menu. On the right, the main area has a header 'Google Search Console' and a 'Sitemaps' tab. Below it, a search bar says 'Propriété de recherche'. A dropdown menu is open under 'Propriété de recherche', showing 'DEVELOPPEZ.COM' and 'STAHE.GITHUB.IO'. Under 'STAHE.GITHUB.IO', several URLs are listed, each with a small circular icon. One URL, 'https://stahe.github.io/python-ia-sept-2025/', is highlighted with a red arrow labeled '3'. At the bottom of the dropdown menu, there's a button '+ Ajouter la propriété' with a red arrow labeled '4' pointing to it.

- En [1], la liste des sites qui ont été déclarés à Google ;
- En [3], la liste de vos sites ;
- En [4], ajoutez votre nouveau 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

ou

Préfix de l'URL

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

example.com
Saisissez un domaine ou sous-domaine

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

CONTINUER

CONTINUER

- En [5], tapez l'URL de votre site. Elle est dans le fichier [config.py] :

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

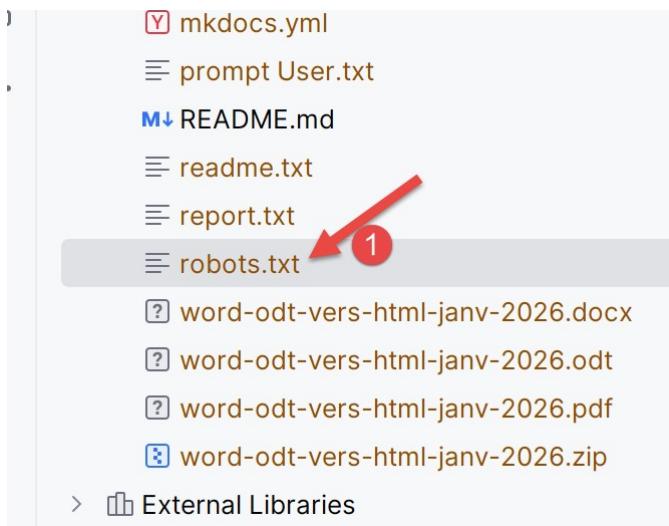
- en [6], passez à l'étape suivante ;

Normalement à cette étape, Google Search Console vous propose un fichier [googlexxxxx.html] que vous devez télécharger. Je n'ai pas eu cet écran car j'avais déjà ce fichier dans le site déployé. C'est l'un des deux fichiers déclarés dans [config.py]

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

Mettez le fichier [googlexxxxx.html] à la racine de votre dossier de travail et inscrivez son nom à la ligne 2 ci-dessus du fichier [config.py]. Le script [convert] s'occupe de mettre les deux fichiers ci-dessus à la racine du site MkDocs qu'il génère. Le script [build] lui les mettra à la racine du site HTML générée.

Lorsque vous avez copié le fichier [googlexxxxx.html] à la racine de votre dossier de travail, vérifiez le contenu du fichier [robots.txt] :



Le contenu du fichier [robots.txt] est le suivant :

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

En ligne 3, vérifiez bien que l'URL est celle de votre site, la même que dans le fichier [config.py].

Redéployez votre site web sur GitHub en tapant les trois commandes suivantes dans votre terminal Python :

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

Ceci fait, retournez dans [Google Search Console] et sélectionnez la propriété que vous avez créée précédemment.

- En [1], sélectionnez l'option [Sitemaps] ;
- En [2], tapez [sitemap.xml] ;
- En [3], validez l'URL ;

Le fichier [sitemap.xml] est un fichier que Mkdocs a mis à la racine du site HTML exporté sur GitHub. Vous pouvez vérifier sa présence en tapant directement son URL :



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

```
▼<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>
```

Le fichier [sitemap.xml] liste toutes les pages HTML du site.

Si tout va bien, vous obtenez l'écran suivant :



Google le traitera régulièrement pour chercher des modifications. Si un problème se pose à l'avenir, vous en serez averti.

[Masquer](#)

C'est tout. Vous devrez indexer les pages du site vous-même :

Google Search Console

Ajouter un sitemap
https://stahe.github.io/word-odt-vers-html-janv-2026/ Saisir l'URL du sitemap

Sitemaps envoyés

Sitemap	Type	URL envoyées	Dernière lecture	État
/sitemap.xml	Inconnu	7 janv. 2026		Impossible de récupérer le sitemap

- En [2], tapez l'URL d'une des pages de votre site ;

Ici, le résultat est le suivant :

Inspection de l'URL

Cette URL n'a pas été indexée par Google

Cette page n'est pas indexée. Les pages non indexées ne peuvent pas être diffusées sur Google. Pour savoir pourquoi elle n'a pas été indexée, consultez les informations ci-dessous. [En savoir plus](#)

[AFFICHER LA PAGE EXPLORÉE](#) Page modifiée ? [DEMANDER UNE INDEXATION](#)

- En [1], Google dit que la page n'est pas encore indexée ;
- En [2], vous pouvez demander son indexation ;

Google Search Console vérifie alors que la page demandée peut être indexée. Si oui, on obtient le message suivant :

Indexation demandée

Cette URL a été ajoutée à une file d'attente d'exploration prioritaire. Le fait d'envoyer une page à plusieurs reprises ne modifiera pas sa priorité ni sa position dans la file d'attente. [En savoir plus](#)

[Masquer](#)

Vous pouvez faire cela pour toutes les pages de votre site afin d'être sûr qu'il est bien indexé par le moteur de recherche de Google.

14. Conclusion

Nous avons montré dans ce document un convertisseur de documents ODT produits par [LibreOffice](#) ou de documents DOCX de Word en un site statique HTML de très bonne qualité.

Nous avons présenté dans le chapitre [[Les exemples de ce document](#)] les structures gérées correctement par le convertisseur Gemini / ChatGPT.

Nous avons montré également comment exporter sur GitHub les sites HTML créés par les convertisseurs ODT et DOCX (cf chapitre [Héberger le site HTML sur GitHub](#))

En dehors des structures du chapitre des exemples, rien n'est garanti. Il est probable que le convertisseur montrera des anomalies. Il faudra alors de nouveau commencer un chat avec Gemini ou ChatGPT pour résoudre ces anomalies. Le plus simple est de donner le convertisseur actuel à l'IA avec son fichier de configuration. Puis de demander à l'IA s'il comprend les codes qu'on lui a donnés. Cela la force à analyser le code. Elle répondra toujours qu'elle comprend parfaitement le code. Ce sera le moment de lui demander les changements désirés. Aussi bien pour Word que LibreOffice, vous pouvez aider l'IA avec des macros. L'IA ne « voit » pas le document ODT ou DOCX exploité. Il lui manque donc des informations. Ainsi le convertisseur DOCX -> HTML n'arrivait pas à gérer correctement la numérotation des codes riches. Elle ne savait pas lire le numéro de la première ligne d'un bloc de code pour le rendre en HTML. Elle commençait toujours le code à 1 même si le code dans le DOCX commençait à 12. Je me suis alors aidé d'une macro. J'ai placé le curseur du document sur la ligne numérotée et j'ai demandé à ChatGPT de générer une macro VBA qui lui donnerait les caractéristiques de cette ligne, qui lui permettraient de générer correctement la numérotation HTML. ChatGPT l'a fait et je lui ai donné le résultat de sa macro. Ca l'a débloqué : les informations récupérées lui montraient que le numéro de la ligne était propagé par son style et non par l'icône de numérotation. Il a alors changé son algorithme qui a fonctionné. Donc on peut aider l'IA à avancer.

Une autre voie plus simple est de changer son document. Quand le convertisseur produit une anomalie, essayez de modifier votre document pour qu'il n'utilise que les structures du chapitre des exemples.