
	Fiche information	01/02/2024
	Génération automatique de documentation du code source avec Doxygen	
	Lycée Jules Haag	



Doxygen est un outil logiciel permettant de générer automatiquement la documentation du code source d'un programme à partir des commentaires placés dans les différents fichiers.

Il permet de générer la documentation dans des codes source en C++, Python, Java, ... et bien d'autres.

Il est multiplateforme (Windows, Linux, Mac).

La documentation peut être obtenue dans différents formats :

- Un ensemble de pages HTML
- Un document PDF
- Au format XML (pour être exploité dans un autre logiciel, comme Sphinx par exemple...)

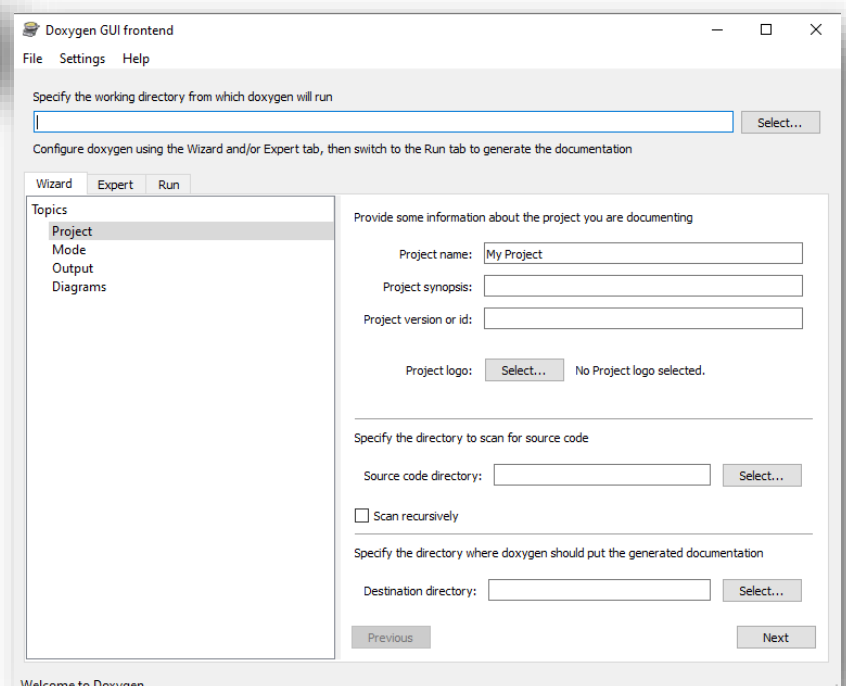
Utilisation sous Windows

1. Télécharger et installer le logiciel pour Windows

version System Installer pour une installation simple.

<https://www.doxygen.nl/download.html>

2. Lancer DoxyWizard



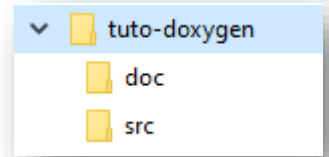
3. Télécharger les fichiers pour le tuto

<https://github.com/Olivier-hac/tuto-doxygen/tree/main>

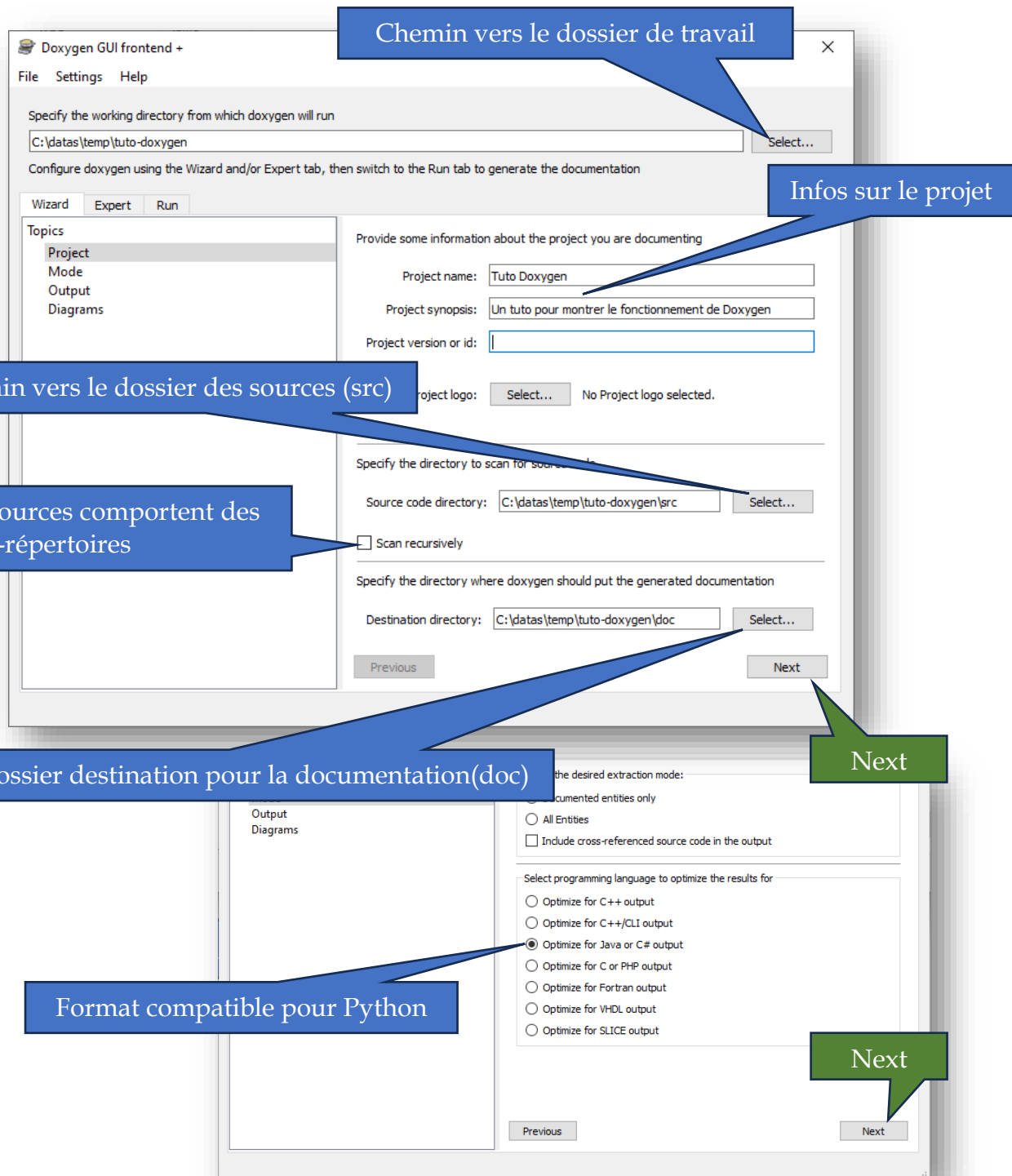
Créer un dossier de travail **tuto-doxygen**.

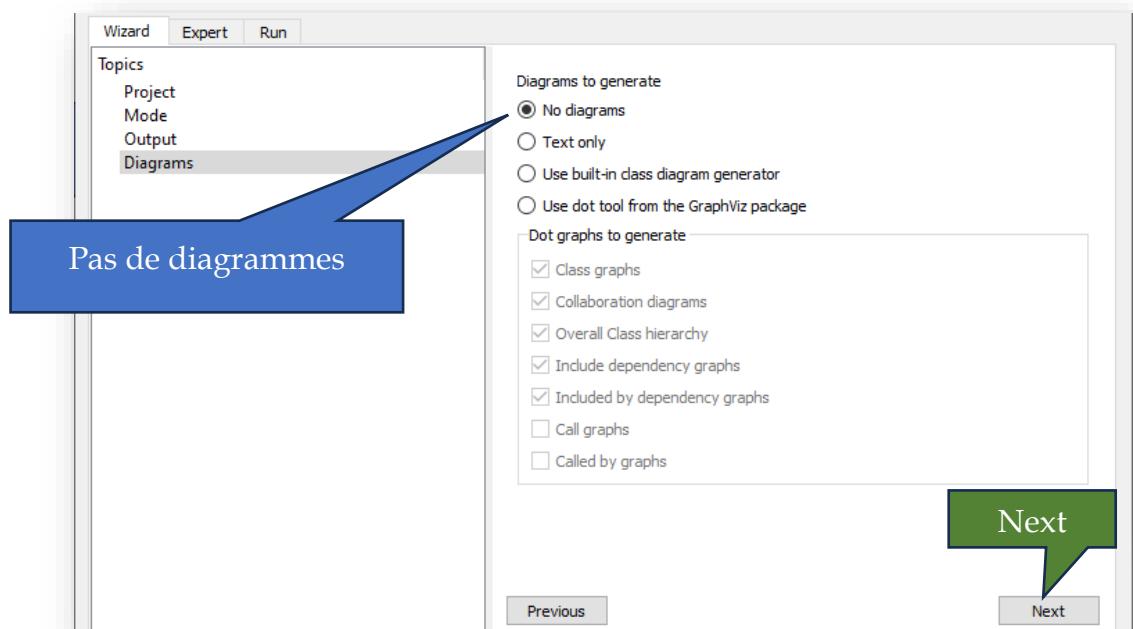
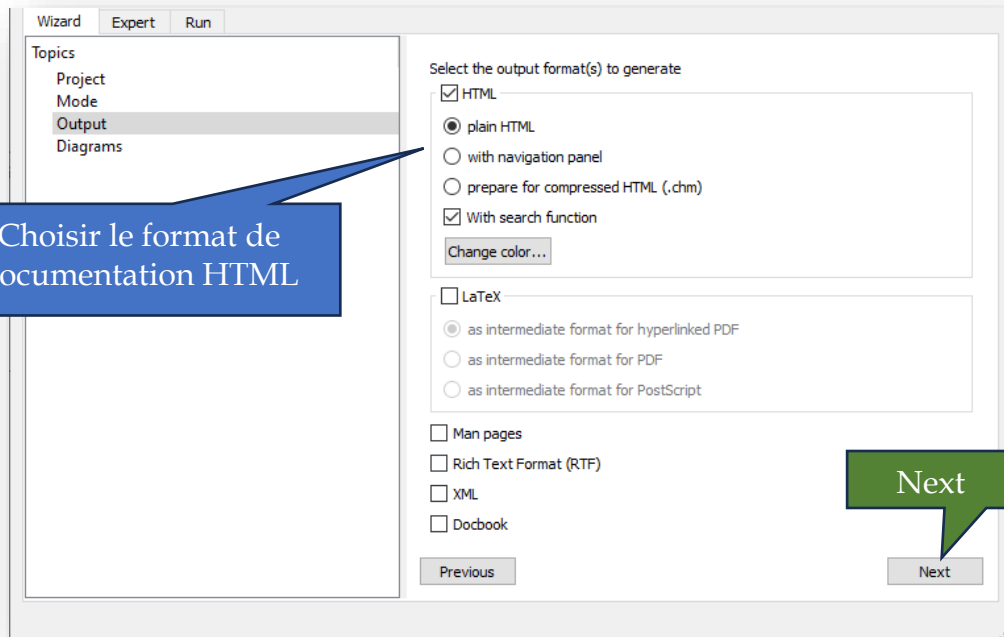
Copier le dossier **src** , précédemment téléchargé, dans ce dossier.

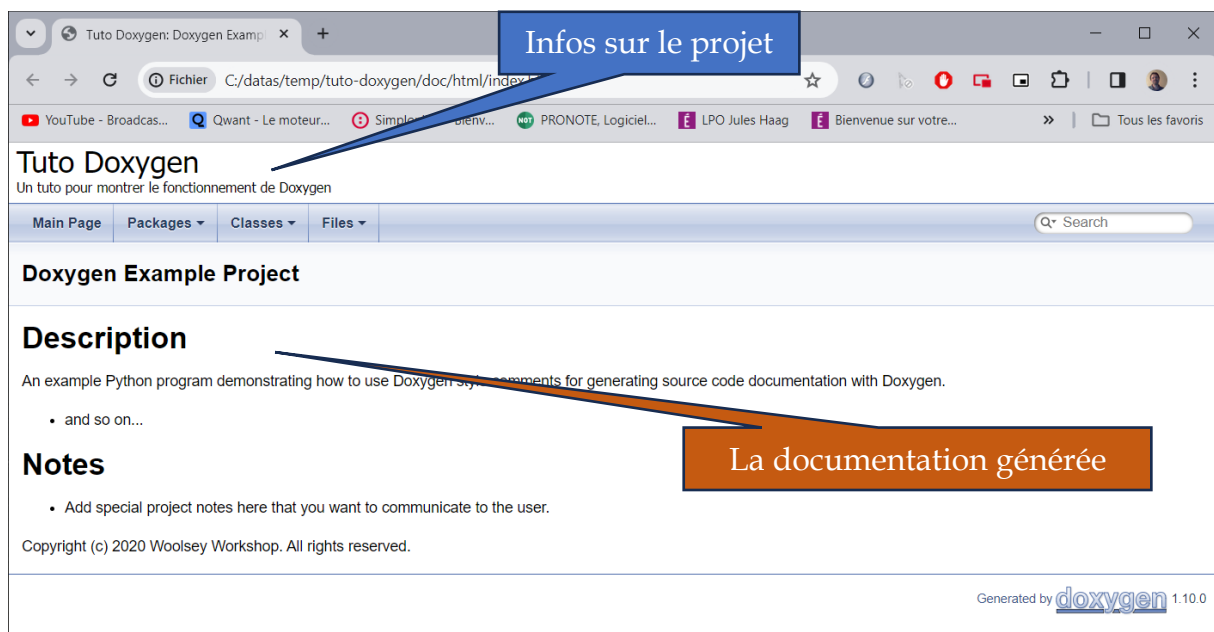
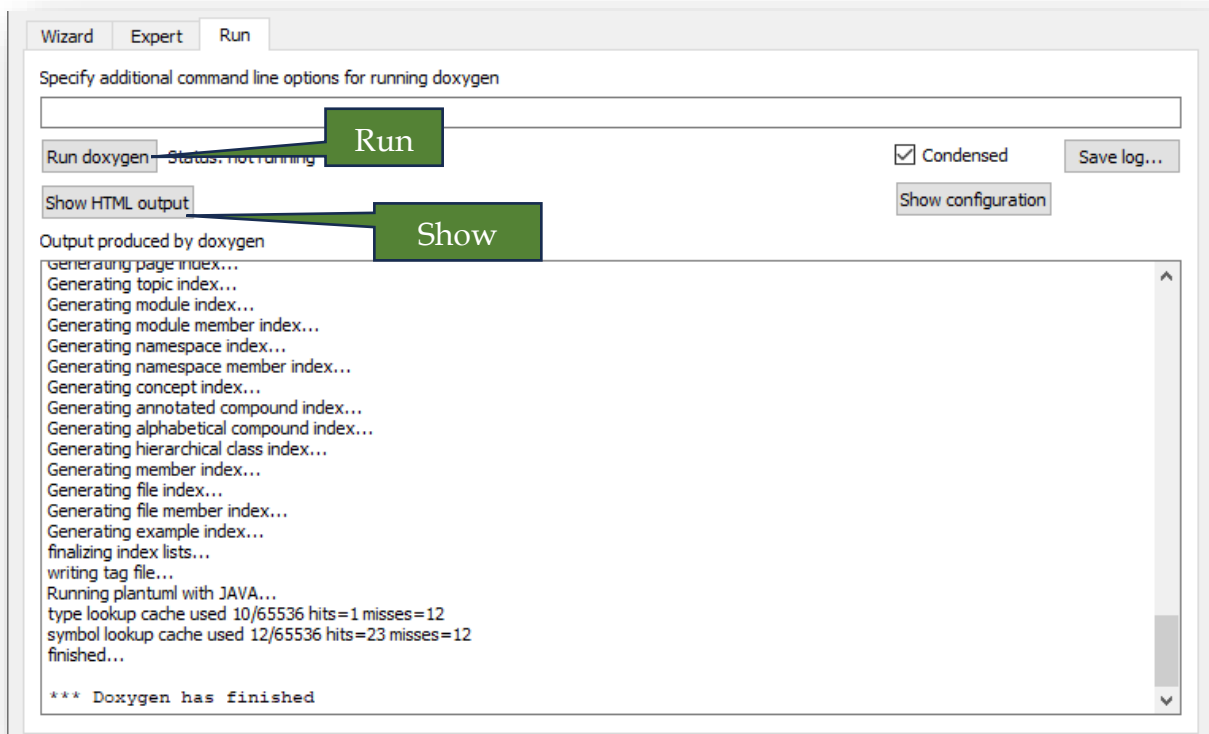
Créer un dossier **doc** dans le dossier de travail.



4. Préciser les paramètres pour une première génération de la documentation







5. Détails du format des commentaires pour Doxygen

Les commentaires placés dans le code permettent de définir les indications que l'on trouvera dans la documentation.

En python, les commentaires peuvent prendre 2 formes :

```
""" un commentaire qui peut être écrit sur plusieurs lignes """  
# un commentaire simple sur une seule ligne
```

Doxygen doit pouvoir repérer dans le code les indications qu'il doit extraire pour constituer la documentation.

Dans un script Python les blocs de commentaires au format Doxygen doivent commencer par

```
## ou """! si sur plusieurs lignes """
```

Les différentes parties de la documentation seront repérées par la syntaxe :

```
# @partie Texte à afficher dans la doc
```

Partie permet de distinguer les différents types de partie de la documentation

Suit le texte à afficher dans la doc

Et pour certaines parties pouvant exister en plusieurs exemplaires, on peut préciser un identifiant

```
# @partie identifiant Texte à afficher dans la doc
```

Partie permet de distinguer les différents types de partie de la documentation

Identifiant permet d'identifier de manière unique cette partie dans la doc

Suit le texte à afficher dans la doc

Dans l'exemple donné, le projet est constitué de 2 fichiers :

- **demo.py** : le programme principal contenant les instructions exécutées (*main*)
- **sensors.py** : fichier module, contenant 2 classes utilisées dans **demo.py**

Commentaires d'ordre général

Les commentaires généraux concernant le projet, sont dans **demo.py**

My Project

Le titre de la page principale

Une section

Une autre section

```
##  
# @mainpage Doxygen Example Project  
#  
# @section description_main Description  
# An example Python program demonstrating how to use Doxygen style comments for  
# generating source code documentation with Doxygen.  
# - and so on...  
#  
# @section notes_main Notes  
# - Add special project notes here that you want to communicate to the user.  
#  
# Copyright (c) 2020 Woolsey Workshop. All rights reserved.
```

Commentaires des variables du programme

Ces commentaires sont visibles via *Namespaces > Namespaces List > demo*

Variables

```
int DEBUG = 1
    The mode of operation; 0 = normal, 1 = debug.

int MIN_BASE = 1
    The minimum number to map.

int MAX_BASE = 10
    The maximum number to map.

int MIN_MAPPED = 0
    The minimum mapped value.

int MAX_MAPPED = 255
    The maximum mapped value.
```

```
# Global Constants
## The mode of operation; 0 = normal, 1 = debug.
DEBUG = 1
## The minimum number to map.
MIN_BASE = 1
## The maximum number to map.
MAX_BASE = 10
## The minimum mapped value.
MIN_MAPPED = 0
## The maximum mapped value.
MAX_MAPPED = 255
```

Le commentaire doit être placé juste avant la définition de la variable

Commentaires détaillant les fonctions

◆ map_range()

```
demo.map_range ( number,
                 in_min,
                 in_max,
                 out_min,
                 out_max )
```

Maps a number from one range to another.

Parameters

number The input number to map.
in_min The minimum value of an input number.
in_max The maximum value of an input number.
out_min The minimum value of an output number.
out_max The maximum value of an output number.

Returns

The mapped number.

```
def map_range(number, in_min, in_max, out_min, out_max):
    """! Maps a number from one range to another.

    @param number The input number to map.
    @param in_min The minimum value of an input number.
    @param in_max The maximum value of an input number.
    @param out_min The minimum value of an output number.
    @param out_max The maximum value of an output number.

    @return The mapped number.
    """
    mapped = (number - in_min) * (out_max - out_min) / (in_max - in_min)
    if out_min <= out_max:
        return max(min(mapped, out_max), out_min)
    return min(max(mapped, out_max), out_min)
```

Le commentaire est un commentaire multiligne au format doxygen ("""! ... """)

Les sections `@param` et `@return` définissant les paramètres à passer à la fonction, et la valeur retournée par la fonction

6. Pour aller plus loin

- Le site de doxygen (<https://www.doxygen.nl/manual/index.html>) documente toute la richesse possible des commentaires permettant d'obtenir une documentation fournie de votre code.
- Il est également possible de régénérer la documentation du code automatiquement, à chaque commit de son code sur github. Comme expliqué dans le tuto ci-dessous :

Création automatique de documentation avec Doxygen et Sphinx, en CI/CD et Gitlab

<https://bioinfo-fr.net/comment-creer-une-documentation-automatique-avec-doxygen-et-sphinx-en-ci-cd-gitlab>