Nom :	Prénom :	Classe :

NSI 1re — Documenter une fonction (Docstring)

Objectif :
- Savoir documenter une fonction grâce à une Docstring.

Introduction

La **documentation** d'une **fonction**, c'est le **texte** (le commentaire) qui explique ce que fait une fonction, comment l'utiliser, et dans quelles conditions elle renvoie un résultat. En pratique, c'est une aide à la lecture du code, à la maintenance et à la réutilisation.

Voici un exemple de fonction documentée (commentée) :

```
1  def tva(p, t):
2      """Calcule le prix TTC et renvoie ce prix"""
3      ttc = p + p * t / 100
4      return ttc
```

Dans l'exemple ci-dessus, on comprend rapidement le rôle de cette fonction `tva`.

Dans la vie courante, c'est un peu comme la notice d'un appareil ou l'étiquette d'un médicament : on comprend à quoi ça sert, comment l'utiliser, et ce qu'il faut éviter. Pour une développeuse ou un développeur, une bonne documentation (ou « doc ») évite de devoir relire tout le code pour savoir quoi appeler, avec quels valeurs de paramètres, et avec quels effets attendus.

La documentation facilite la compréhension, accélère la maintenance, et réduit les erreurs quand plusieurs personnes travaillent sur le même projet. Elle est aussi utile plus tard, quand le code est repris après plusieurs mois, parce qu'elle sert de mémoire du projet.

Le besoin de documenter est né d'un problème simple : un code sans explication devient vite difficile à comprendre, à partager et à maintenir, surtout dès qu'il grandit ou qu'il change souvent. En d'autres termes, plus un programme est complexe, plus il devient risqué de laisser les fonctions « parler toutes seules » sans indication claire.

➤ **Exercice 1** — Réfléchir à haute voix

Parmi les deux programmes ci-dessous, lequel contient une documentation utile ?

Programme 1	Programme 2

`def vs(a):`
`    """fonction pour calculer"""`
`    return (4*pi*a**3)/3`
	`def vs(a):`
	`    """Renvoie le volume d'une sphère`
	`    de rayon 'a'`
	`    """`
	`    return (4*pi*a**3)/3`

Comment documenter une fonction ?

Une bonne documentation de fonction répond généralement à quelques questions précises : que fait la fonction, quels paramètres elle attend, quel type de résultat elle produit, et quelles erreurs ou cas particuliers peuvent arriver.

Techniquement, la documentation d'une fonction est placée **au tout début** du bloc d'instruction de la fonction. C'est un **commentaire Python** qui commence et se termine par **trois guillemets doubles** :

```
1  def plus_un(a):
2      """Renvoie la valeur de 'a' additionné de 1"""
3      return a + 1
```

La documentation d'une fonction est baptisée « **Docstring** » dans le langage de programmation Python.

Documentation sur une ligne

Voici un exemple de Docstring tenant sur **une seule ligne** :

```
1  def c(r):
2      """Renvoie l'aire d'un cercle de rayon 'r'"""
3      return pi*r**2
```

Documentation sur plusieurs lignes

Voici un exemple de Docstring tenant sur **plusieurs lignes** :

```
1  def tarif(a, p):
2      """Calcule et renvoie le tarif final à payer par le client.
3
4      Paramètres :
5      a (int): L'âge du client.
6      p (bool): Option popcorn.
7      """
8      f = 13.2
9      if a < 18:
10         f = 7.8
11     if p == True:
12         f += 2.5
13     return f
```

Le programme ci-dessus profite du *large espace* offert par une Docstring multi-lignes pour apporter des informations sur les paramètres attendus par la fonction.

➤ **Exercice 2** — Documenter une fonction existante

Documenter la fonction des programmes ci-dessous.

La première ligne est donnée à titre d'exemple.

N°	Programme	Documentation de la fonction (à compléter)

0	`def plus_un(a):`
	`  """❓"""`
	`  return a + 1`
		"""Renvoie la valeur de 'a' additionnée de 1."""

1	`def multi(a, b):`
	`  """❓"""`
	`  return a * b`
		Renvoie le résultat du calcul correspondant à “a” multiplié par “b”.

2	`def div(a, b):`
	`  """❓"""`
	`  return a / b`
		Renvoie le résultat (un flottant) du calcul correspondant à “a” divisé par “b”.

3	`def volume_pave(lar, lon, hau):`
	`  """❓`
	`  ❓`
	`  ❓`
	`  ❓`
	`  ❓`
	`  """`
	`  return lar * lon * hau`
		Renvoie le volume d'un pavé.
		
		Param :
		- lar (int / float) : Largeur du pavé
		- lon (int / float) : Largeur du pavé
		- hau (int / float) : Largeur du pavé

Écrire une fonction à partir de sa documentation

À l'inverse, il est possible d'écrire le code d'une fonction à partir d'une documentation fournie.

Exemple :

N°	Documentation de la fonction	Programme (à compléter)
0	"""Calcule et renvoie l'aire d'un carré. Paramètre : c (float): Le côté d'un carré. """	`def aire_carre(c):` `  return c * c`

➤ **Exercice 3** — Écrire un programme, à partir de sa documentation

Écrire les programmes correspondant aux documentations ci-dessous.

N°	Documentation de la fonction	Programme (à compléter)
1	"""Calcule et renvoie le reste de la division euclidienne d'un nombre A par un nombre B. Paramètres : a (float): Dividende. b (float): Diviseur. """	def euclide_reste(a, b): return a % b
2	"""Calcule et renvoie le volume d'un cube, à partir de la formule C^3 (où C est la longueur d'une arête du cube). Paramètre : c (float): Le côté d'un carré. """	def vol_cube(c): return c ** 3
3	"""Calcule et renvoie l'aire d'une sphère, à partir de la formule $4 \times \pi \times R^2$ (où R est le rayon de la sphère). Paramètres : r (float): Rayon de la sphère. pi (float): Valeur de π. """	def aire_sphere(r, pi): return 4 * pi * r ** 2

Déjà terminé ?

Ok, un dernier pour la route :

N°	Documentation de la fonction	Programme (à compléter)

4	"""Calcule et renvoie le quotient et le reste de la division euclidienne d'un nombre A par un nombre B.
	
	Paramètres :
	a (float): Dividende.
	b (float): Diviseur.
	
	Renvoie :
	Un tuple contenant deux éléments
	"""
		def euclide_quotient_reste(a, b):
		  quotient = a // b
		  reste = a % b
		  return (quotient, reste)