Fiche d'exercices : Bonnes pratique de développement

En-tête de fonctions

Une en-tête de fonction permet de spécifier les types de paramètres en entrée et le type du résultat renvoyé par la fonction.

Elle est constitué ainsi :

```
def nom_de_fonction(parametre_1 : type, parametre_2 : type , ... ) -> type_du_renvoi :
```

Cela permet de se rendre compte du premier coup d'oeil de ce que la fonction doit faire.

Écrire l'en-tête d'une fonction `multiplier` qui prend en paramètre deux nombres entier et renvoie leur produit.

Écrire l'en-tête d'une fonction `concatenation` qui prend en paramètres deux chaînes de caractères et renvoie leur concaténation.

Écrire l'en-tête d'une fonction `est_pair` qui prend en paramètre un nombre entier et renvoie True si le nombre est pair, False impair.

Écrire l'en-tête d'une fonction `afficher_somme` qui prend en paramètres deux nombres entiers et affiche seulement leur somme.

Écrire l'en-tête d'une fonction `aire_rectangle` qui prend en paramètres deux nombres réels (float) et renvoie leur produit.

Documentation de fonctions

La documentation permet à un utilisateur qui fait une revue de code de comprendre ce que la fonction réalise pour : soit comprendre le code, soit le débugger.
Elle est souvent conjointe à la définition d'en-tête mais peut la remplacer.

La documentation est primordiale lors d'un travail en projet ou en entreprise. Cela permet aussi de savoir revenir sur un projet après ne pas y avoir travaillé après une certaine durée.

Enfin, cette documentation permet aussi de générer de la documentation automatiquement, souvent utilisé en projets.
Elle est souvent de la forme :

```
    def nom_de_fonction(parametre_1, parametre_2):
        '''
        Paramètres :
            parametre_1 : type du paramètre
            parametre_2 : type du paramètre
        Retourne :
            type de la sortie
        Explication courte de ce que fait la fonction
        '''
```

Écrire la spécification des fonctions suivantes.

```
def est_voyelle(lettre):
    if lettre == 'a' or lettre == 'e' or lettre == 'o' or lettre == 'i' or lettre == 'y':
        return True
    else :
        return False

def mots(phrase):
    compteur = 0
    for caractere in phrase:
        if caractere == ' ':
            compteur = compteur + 1
    return compteur

def surface(rayon):
    pi = 3.14159
    aire = pi * rayon ** 2
    return aire
```

Nommage de variables

Un code peut être documenté correctement mais si les variables et le nom de fonction ne sont pas compréhensibles, cela peut impacter négativement la compréhension du code.
Renommer les variables et les fonctions des différents algorithmes pour les rendre plus compréhensibles.

```python
def f(l):
    """
    Calcule la somme des éléments d'une liste de nombres.
    Paramètres :
        l (list) : Liste contenant des nombres.
    Retourne :
        int : La somme des éléments de la liste.
    """
    s = 0
    for i in l:
        s += i
    return s

def m(a, b):
    """
    Calcule la moyenne de deux nombres.
    Paramètres :
        a (int/float) : Le premier nombre.
        b (int/float) : Le deuxième nombre.
    Retourne :
    float : La moyenne des deux nombres.
    """
    return (a + b) / 2

def p(l, L):
    """
    Calcule le périmètre d'un rectangle.
    Paramètres :
    l (float) : La longueur du rectangle.
    L (float) : La largeur du rectangle.
    Retourne :
    float : Le périmètre du rectangle.
    """
    return 2 * (l + L)
```