T6.4 Spécification

Commençons par une devinette: que fait la fonction suivante:

“‘python linenums=‘1’ def doublon(l, c): n = 0 for e in c: if e == l: n += 1 return n > 1

````
![](../images/documentation.jpeg){: .center}

??? check "Mieux?"
    ```python linenums='1'
    def verifie_doublon(l: str, c: str) -> bool:
        '''
        Vérifie si la chaîne c contient au moins deux occurences
        de la chaîne l
        '''
        nb_occurences = 0
        for element in c:
            if element == l:
                nb_occurences += 1
        return nb_occurences > 1

    ```

## 6.4.1 Spécifier et documenter

!!! abstract "Spécification"
    De manière générale la **spécification** est un ensemble de d’exigences à satisfaire par

    En programmation, **spécifier** un programme/une fonction revient à décrire explicitemen

    Pour cela:

    - on décrit ce que fait la fonction dans une documentation appelée **docstring**, placée
    - on utilise des noms de variables explicites (voir T6.1.1);
    - on précise le type de(s) paramètre(s), et éventuellement les conditions d'utilisations
    - on précise le type de la valeur renvoyée par la fonction: on parle de **postconditions
    - ces types apparaissent dans la docstring ou en *annotations de types* (depuis Python 3
    - on intègre éventuellement des exemples d'utilisation qui peuvent servir de tests.


La docstring est alors disponible en utilisant la fonction `help`:

```python linenums='1'
>>> help(verifie_doublon)
Help on function verifie_doublon in module __main__:
````
````

```
verifie_doublon(l: str, c: str) -> bool
    Vérifie si la chaîne c contient au moins deux occurences
    de la chaîne l

>>>
```

!!! question “Intérêt” - programmer : documenter **avant** d’écrire le code donne un objectif clair; - relire : les programmes complexes sont difficiles à comprendre. La documentation simplifie cette étape; - collaborer : travailler à plusieurs demande de l’organisation et une documentation claire est indispensable.

!!! note “PEP8 & Zen” Pour une meilleure lecture du code et une meilleure communication entre les différents collaborateurs d’un projet, il est donc nécessaire de prendre de bonnes habitudes de programmation et de respecter des conventions d’écriture:

```
- La [PEP8](https://openclassrooms.com/fr/courses/4425111-perfectionnez-vous-en-python/44642
- Le Zen Python :

```python
>>> import this
```
```

6.4.2 Exemples

!!! example “Exemple sans tests” La fonction `maximum` vue aux chapitres précédentes:

```
```python linenums='1'
def maximum(n1, n2):
    if n1 > n2:
        return n1
    else:
        return n2
```
devient:

```python linenums='1'
def maximum(n1: int, n2: int) -> int:
    '''
    Renvoie l'entier maximum entre n1 et n2
    n1 et n2 : deux entiers
    '''
    if n1 > n2:
        return n1
    else:
        return n2
```
```

```

!!! example “Exemple avec tests” Voici la fonction `bissextile` spécifiée et documentée. On peut y ajouter des exemples/tests.

```python linenums='1'
def is_leap(year: int) -> bool:
    '''
    Renvoie True si l'entier year correspond à une année bissextile
    et False sinon.
    >>> is_leap(2021)
    False
    >>> is_leap(2020)
    True
    >>> is_leap(2000)
    True
    >>> is_leap(1900)
    False
    '''
    if year%4 == 0 and year%100 != 0:
        return True
    elif year%400 == 0:
        return True
    else:
        return False
```

On peut ensuite lancer les tests à l'aide du module `doctest`:

```python
>>> import doctest
>>> doctest.testmod()
TestResults(failed=0, attempted=4)
>>>
```

Testez cette fonction comme ci-dessus, puis en passant en argument `verbose=True` à la fonct

6.4.3 Exercices

{{ initexo(0) }}

!!! example “{{ exercice() }}” === “Énoncé” Documenter les fonctions `compte_voyelles` et `fizzbuzz` des chapitres précédents. === “Correction” {{ correction(False, " 1. “‘python linenums=‘1’ def compte_voyelles(phrase: str) -> int: ’’’ Renvoie le nombre de voyelles contenues dans la chaîne de caratères
```

phrase ’’’ voyelles = ‘aeiouy’ nb_voyelles = 0 for lettre in phrase: if lettre in voyelles: nb_voyelles += 1 return nb_voyelles

```
```

2.

```python linenums='1'
def fizzbuzz(nombre: int) -> str:
    '''
    Renvoie la chaîne de caractère correspondant à l'entier nombre selon
    les règles du jeu du FizzBuzz:
    - 'Fizz' si divisible par 3
    - 'Buzz' si divisible par 5
    - 'FizzBuzz' si divisible par 3 et par 5
    - le nombre dans les autres cas. Dans ce cas on convertit le nombre en chaîne de ca
    pour homogénéiser le type de sortie.
    '''

    if nombre%3 == 0 and nombre%5 == 0:
        return 'FizzBuzz'
    elif nombre%3 == 0:
        return 'Fizz'
    elif nombre%5 == 0:
        return 'Buzz'
    else:
        return str(nombre)
```

"

) }}