ISM MPSI/PCSI, Informatique
Année 2025/2026

# TP Informatique 6

**Évaluation automatique de l'exercice 1 (20 minutes)**

Télécharger le fichier `TP06_EX01` sur cahier de prépa, l'ouvrir et le compléter. Pour cela, supprimer l'instruction `pass` et compléter les fonctions.
Au bout de 20 minutes au maximum, vous déposerez votre fichier dans le répertoire prévu sur `cahier de prépa` (il faudra vous connecter). **Le fichier déposé doit pouvoir être exécuté sans message d'erreur et sans effet de bord (pas d'appel de la fonction `print`).** Puis vous poursuivrez le sujet de TP.

## Exercice 1

**Vous respecterez l'utilisation d'une boucle conditionnelle `while` ou inconditionnelle `for` imposée dans chaque question.**

1. On souhaite écrire une fonction `somme(n)` renvoyant la valeur de la somme $\sum_{k=0}^{n} k$. On propose :

```
def somme(n):
    """n : entier"""
    k,res=0,0
    while k<n:
        res+=k
        k+1
    return res
```

   Identifier la (ou les) erreur(s) de ce code et écrire une version corrigée de la fonction `somme`. Cette fonction utilisera nécessairement une boucle `while`.
2. En vous inspirant du code précédent (toujours avec une boucle `while`), écrire une fonction `fact(n)` renvoyant $n!$.
3. Écrire une fonction `somme1(n)`, avec boucle conditionnelle, d'argument `n` un entier et qui renvoie $\sum_{0\leqslant 2k+1\leqslant n} (2k+1)^2$.
4. Écrire une fonction `geom(n,q)`, avec boucle inconditionnelle, d'arguments `n` un entier et `q` un flottant et qui renvoie la valeur de $\sum_{k=0}^{n} q^k$. On veillera à calculer efficacement les puissances successives de $q^k$ en s'appuyant sur la valeur de $q^{k-1}$ déjà calculée.

---

⚠ Tous les exercices seront traités dans la même feuille de script nommée `TP06.py`, à enregistrer dans votre espace personnel sur le disque `U:`.

On prévoira d'exécuter `pytest` avant la fin du TP. La démarche est la suivante pour les salles D206 et D207 :

— depuis l'explorateur de fichiers, se placer dans le répertoire `C:\Winpython` puis exécuter `WinPython Command Prompt` qui ouvre une fenêtre appelée *invite de commande* ;

— Saisir `U:` dans l'invite de commande puis `pytest TP06.py`.

La démarche est la suivante pour la salle A202 :
— lancer *l'invite de commande* en tapant `cmd` dans la recherche windows ;
— saisir `pip install pytest` ;
— saisir `U:` dans l'invite de commande puis `pytest TP06.py`.

## Exercice 2

On considère la suite de Héron définie par $u_0 = 1$ et la relation suite :

$$\forall n \in \mathbb{N}\,,\ u_{n+1} = \frac{1}{2}\left(u_n + \frac{2}{u_n}\right)$$

On admet que cette suite tend vers $\sqrt{2}$.

1. Écrire une fonction `heron(n)` d'argument `n` un entier naturel et qui renvoie le terme $u_n$ de la suite. Le code proposé utilisera une boucle `for`.
2. Écrire une fonction `test_heron()` utilisant l'instruction `assert` testant la fonction `heron` pour différentes valeurs de `n`.
3. Écrire une fonction `heron2(n)` d'argument `n` un entier naturel et qui renvoie le terme $u_n$ de la suite. Le code proposé utilisera une boucle `while`.
4. Écrire une fonction `test_heron2()` utilisant l'instruction `assert` testant la fonction `heron2` pour différentes valeurs de `n`.
5. Écrire une fonction `seuil_heron(e)` prenant en argument un flottant `e` et qui renvoie le plus petit entier naturel $n_0$ pour lequel $|u_{n_0+1} - u_{n_0}| < e$. Vérifier notamment que `seuil_heron(1e-3)` renvoie 3.

## Exercice 3

On a
$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2} \xrightarrow[n\to\infty]{} +\infty$$

Par conséquent, on peut définir

$$\forall p \in \mathbb{R} \qquad \varphi(p) = \min\left\{n \in \mathbb{N} \mid \sum_{k=0}^{n} k \geqslant p\right\}$$

1. Écrire une fonction `seuil(p)` d'argument `p` un flottant et qui renvoie le couple $\left(\sum_{k=0}^{\varphi(p)} k, \varphi(p)\right)$.
2. Écrire une fonction `test_seuil()` utilisant l'instruction `assert` testant la fonction `seuil` pour différentes valeurs de $p$.

## Exercice 4

Dans l'éditeur, effectuer l'importation `import numpy as np`.

1. Écrire une fonction `somme_log(n)`, avec boucle conditionnelle, d'argument `n` un entier et qui renvoie la valeur de $\sum_{1\leqslant k,\ k\ln k\leqslant n} k$. On rappelle que la fonction ln correspond à la fonction `log` dans `numpy`.
2. Peut-on écrire une version avec boucle inconditionnelle ? Si oui, le faire en nommant cette fonction `somme_log_bis(n)`