ISM MPSI/PCSI, Informatique
Année 2025/2026

TP Informatique 5

⚠ Tous les exercices seront traités dans la même feuille de script nommée `TP05.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 :
- 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 TP05.py`.

Exercice 1

1. Dans l'éditeur, saisir la fonction

```
def f(x):
    if x>0:
        return x
    else:
        return -x
```

   puis la tester dans la console sur différentes valeurs de flottants. Que réalise la fonction `f` ?
2. Écrire une nouvelle version de la fonction `f2` avec un test sans alternative.

Exercice 2

1. Écrire une fonction `arr1(x)` d'argument `x` un flottant qui renvoie x si $|x| > \varepsilon$ et 0 sinon avec $\varepsilon = 10^{-6}$. On écrira un test avec alternative.
2. Écrire une nouvelle version `arr2(x)` qui renvoie le même résultat que `arr1(x)` mais avec un test sans alternative.

Exercice 3

1. Dans l'éditeur, saisir la fonction :

```
def somme1(n):
    res=0
    for k in range(n+1):
        res+=k
    return res
```

   puis la tester sur différentes valeurs d'entiers. Que réalise la fonction `somme1` ?
2. En s'inspirant de l'exemple, écrire une fonction `somme2(n)` qui calcule $\sum_{k=0}^{n} k^2$ pour n entier naturel non nul.
3. Écrire une fonction `test_somme2()` permettant de tester la fonction `somme2` en utilisant l'instruction `assert`.

Exercice 4

1. Dans l'éditeur, saisir la fonction :

```
def somme_liste(L):
    res=0
    for x in L:
        res+=x
    return res
```

   puis la tester sur différentes listes de nombres. Que réalise la fonction `somme_liste` ?
2. En s'inspirant de l'exemple, écrire une fonction `prod_liste(L)` d'argument `L` une liste de nombres et qui renvoie le produit de ceux-ci.
3. Écrire une fonction `test_prod_liste()` permettant de tester la fonction `prod_liste` en utilisant l'instruction `assert`.

Exercice 5

1. Écrire une fonction `somme_cube(n)` d'argument `n` un entier naturel non nul et qui renvoie la somme des cubes parfaits inférieurs stricts à n. Le code proposé ne manipulera que des entiers et utilisera une boucle `for`.

   Par exemple `somme_cube(65)` donne $1+8+27+64=100$, et `somme_cube(64)` donne $1+8+27=36$.
2. Écrire une fonction `test_somme_cube()` permettant de tester la fonction `somme_cube` en utilisant l'instruction `assert`.

Exercice 6

Écrire une fonction `mult(n,p)` d'arguments `n` et `p` des entiers non nuls et qui renvoie le multiple de `p` le plus proche de `n`. Par exemple, l'appel `mult(31,4)` renvoie `32` tandis que l'appel `mult(29,4)` renvoie `28`. L'appel de `mult(30,4)` peut renvoyer indifféremment `28` ou `32`, les deux réponses étant correctes.

Exercice 7

Un *triplet pythagoricien* est un triplet d'entiers naturels (m,n,p) **non nuls** tels que $m \leqslant n$ et $m^2+n^2=p^2$. Par exemple, le triplet $(3,4,5)$ est pythagoricien car on a $3^2+4^2=5^2$.

Écrire une fonction `pytha(N)` d'argument N un entier naturel qui renvoie la liste des triplets pythagoriciens `[m,n,p]` tels que $p \leqslant$ N. On les affichera par ordre croissant selon p.
On vérifiera que l'appel `pytha(10)` renvoie `[[3, 4, 5], [6, 8, 10]]`.