Chapitre 0 - Pseudo-code et révisions Python

Compétences :

- ▷ Améliorer sa pensée logique
- ▷ Décomposer un problème complexe en étapes simples
- ▷ Décrire les étapes d'un programme en une structure simple et lisible
- ▷ Écrire un programme en langage naturel

1. Introduction

Le langage naturel, ou **pseudo-code**, agit comme un pont entre la langue humaine et les langages de programmation. Il permet aux programmeurs de conceptualiser des algorithmes en utilisant une syntaxe informelle, favorisant la planification efficace et la communication des idées complexes avant la mise en œuvre concrète. En simplifiant les concepts de programmation, le pseudo code facilite le développement d'algorithmes logiques et bien conçus.

2. Définition

A retenir !

Le pseudo-code permet de décrire facilement un algorithme avec un vocabulaire simple et sans connaissance à priori du langage de programmation utilisé pour son implémentation machine.

3. Consignes

Pour chacun des exercices ci-dessous, il faudra :

- ▷ Ecrire l'algorithme correspondant en pseudo-code
- ▷ Implémenter la fonction demandée en Python
- ▷ Faire l'appel de la fonction ainsi créée dans votre programme principal
- ▷ Déposer chaque code source dans *Classroom* avec un fichier nommé *TP00-ExN.py*

4. Exercices

Exercice 1 – *Somme d'une liste d'entiers*

Écrire une fonction `somme(L)`, qui prend en paramètre une liste d'entiers et qui retourne la somme de ses éléments.

Algorithme en pseudo-code :

Rappel :

Pour implémenter l'algorithme écrit en pseudo-code, vous pourrez compléter le code Python ci-dessous :

```
from random import *
L = [randint(0,100) for i in range(20)]

def somme(L):

    # ecrire votre code ici

print(somme(L))                          # appel de la fonction dans le PP
```

Exercice 2 – *Maximum d'une liste*

Écrire une fonction `maxi(L)`, qui prend en paramètre une liste d'entiers et qui retourne le plus grand de ses éléments.

Algorithme en pseudo-code :

Exercice 3 – *Minimum d'une liste*

Écrire une fonction `mini(L)`, qui prend en paramètre une liste d'entiers et qui retourne le plus petit de ses éléments.

Algorithme en pseudo-code :

Exercice 4 – *Puissance*

Écrire une fonction `puissance(x,n)`, qui prend en paramètre un réel x et un entier naturel n et qui renvoie x^n.

Algorithme en pseudo-code :

Exercice 5 – *Mot inversé*

Écrire une fonction `inverse(mot)` qui prend en paramètre un mot et qui renvoie le mot écrit à l'envers.

Par exemple :

```
>> mot = bonjour
>> inverse(mot)
>> roujnob
```

Algorithme en pseudo-code :

Exercice 6 – *Tri par insertion*

Implémenter la fonction `tri_insertion(L)` correspondant à l'algorithme de tri ci-dessous vu en classe de Première :

Algorithme 1 : Tri par insertion

Données : Une liste L de n éléments
Sorties : La liste L triée en ordre croissant
pour k *allant de 1* **à** $n-1$ **faire**
  $j \leftarrow k$;
  tant que $j > 0$ *et* $L[j-1] > L[j]$ **faire**
    On échange L[j-1] et L[j]
    $j \leftarrow j-1$
retourner L

Exercice 7 – *Conversion*

Examiner le code Python ci-dessous :

```
def conversion(n,b):
signes = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"
mot = " "
while n! = 0:
    mot = signes[n%b] + mot
n = n//b
return mot
```

1. Que permet de faire la fonction ci-dessus ?
2. Ecrire l'algorithme en pseudo-code.

Algorithme en pseudo-code :

Exercice 8 – *Déconversion*

Écrire une fonction `deconvers(mot,b)` qui fait le contraire de la fonction de l'exercice 7.

Aide : l'instruction `print(signes.index("A"))` affiche 10

Algorithme en pseudo-code :

Exercice 9 – *Suite de Fibonacci*

Ecrire une fonction `fib(n)` qui affiche les n premiers termes de la suite de Fibonacci.

Rappel :

Les deux premiers termes de la suite de Fibonacci sont 0 et 1. Ensuite, chaque terme successif est la somme des deux termes précédents. Ainsi 0+1=1, 1+1=2, 1+2=3, 2+3=5, 3+5=8,..., *etc.*

Algorithme en pseudo-code :

Exercice 10 – *Plus Grand Commun Diviseur*

Ecrire une fonction `pgcd(a,b)` qui renvoie le PGCD de deux entiers naturels a et b.

Rappel :

Pour revoir la notion de PGCD, utiliser le site suivant : https://www.jeuxmaths.fr/cours/calcul-PGCD.php

Algorithme en pseudo-code :

Exercice 11 – *Tri à bulles*

Ecrire une fonction `tri_bulles(L)` qui effectue le tri à bulles d'une liste L passée en paramètres.

Rappel :

Pour découvrir et comprendre le principe du tri à bulles, utiliser l'article Wikipedia correspondant.

Algorithme en pseudo-code :