Exemples de sommes et de produits avec `Python`

Lycée Richelieu
Rueil Malmaison

décembre 2024

L'objectif de cette séance est de répondre aux questions en complétant le script présent dans le même répertoire que ce sujet. Si la question requiert de rédiger en langage naturel, utiliser les commentaires pour répondre.

À la fin, on déposera le script dans ce répertoire en ligne avec le mot de passe *TP_sommes* et on veillera à ce que le nom du fichier permette d'identifier le candidat. Par exemple `script_sommes.py` n'est pas explicite tandis que `script_sommes_pa_fournie.py` l'est.

On rappelle que le script doit pouvoir être exécuté sans erreur de compilation par l'examinateur. Si certaines des fonctions conduisent à des erreurs, il faut commenter les lignes correspondantes.

1 Des sommes

1.1 Nombres triangulaires

- *Algorithme n° 1:* Écrire une fonction nommée `Triangulaire` qui :
  - prend pour argument un entier n ;
  - renvoie en sortie le nombre triangulaire $T_n = 1 + 2 + \cdots + n$.
- *Algorithme n° 2:* Écrire un algorithme qui affiche la plus petite valeur de n telle que $T_n \geq 1\,000\,000$ en testant successivement les valeurs de T_n à l'aide d'une boucle `Tant que`.
- *Algorithme n° 3:* En s'appuyant sur la formule explicite des T_n, résoudre sur papier $T_n \geq 1\,000\,000$ puis réécrire l'algorithme précédent en une seule ligne.

  Indication: On pourra utiliser la fonction `ceil` du module `math` qui renvoie l'approximation par excès à l'entier le plus proche.

1.2 Première approximation de π

Dans toute la suite on pose, pour tout $n \geq 1$, $A_n = \displaystyle\sum_{k=1}^{n} \frac{1}{k^2}$

On admettra que (A_n) *tend vers* $\frac{\pi^2}{6}$ *lorsque n tend vers l'infini. Ainsi une approximation de* π *sera donnée par* $\sqrt{6A_n}$.

- *Algorithme n° 4:* Écrire une fonction appelée `PremiereApproximation` qui :
  - prend pour argument un entier n ;
  - renvoie en sortie l'approximation de π obtenue à l'aide de A_n.

1.3 Seconde approximation de π

Dans toute la suite on pose, pour tout n, $B_n = \sum_{k=0}^{n} \frac{(-1)^k}{(2k+1)}$

On admettra que (B_n) *tend vers* $\frac{\pi}{4}$ *lorsque n tend vers l'infini. Ainsi une approximation de* π *sera donnée par* $4B_n$.

- *Algorithme n° 5:* Écrire une fonction appelée `SecondeApproximation` qui :
  - prend pour argument un entier n ;
  - renvoie en sortie l'approximation de π obtenue à l'aide de B_n.

2 Des produits

2.1 Factorielle

- *Algorithme n° 6:* Écrire une fonction appelée `Factorielle` qui :
  - prend pour argument un entier n ;
  - renvoie en sortie $n!$.

  Indication: Ne pas oublier le cas $n = 0$.

2.2 Nombre de combinaisons

- *Algorithme n° 7:* Écrire une fonction appelée `Combinaisons` qui :
  - prend pour argument deux entiers k et n ;
  - renvoie en sortie $\binom{n}{k}$.

  On décommentera les deux lignes de test puis on corrigera éventuellement la fonction.

2.3 Une dernière approximation de π

Pour tout n, on note $W_n = 2 \times (2n+1) \times \left(\prod_{k=1}^{n} \frac{2k}{2k+1} \right)^2$. On admettra que W_n tend vers π.

- *Algorithme n° 8:* Écrire une fonction appelée `TroisiemeApproximation` qui :
  - prend pour argument un entier n ;
  - renvoie en sortie l'approximation de π obtenue à l'aide de W_n.

3 Comparaison des différentes approximations de π

- *Algorithme n° 9:* Écrire une fonction appelée `Precision` qui :
  - prend en entrée une fonction d'approximation `Approx` et un entier n ;
  - renvoie en sortie le logarithme base 10 de la valeur absolue de la différence entre la vraie valeur de π et son approximation par la fonction `Approx` au rang n.

  Préciser en commentaire la signification d'un tel calcul.
- *Algorithme n° 10:* Décommenter l'algorithme d'exemple puis le tester.

  En exploitant les fonctions de tracé de courbes proposées en exemple, tracer trois courbes correspondant aux valeurs de la précision en fonction de n, pour $n \in [\![1; 10\,000]\!]$, pour les trois approximations de π définies plus haut.

  Préciser en commentaire si l'une des approximations semble être meilleure que les autres.