

TP3 : Structures Conditionnelles et Boucles For

Exercice 1 : Conversion

En utilisant les propriétés de la division Euclidienne, spécifier et écrire un programme qui convertisse un nombre entier n de secondes en heures, minutes et secondes. n sera indiqué par l'utilisateur.

Exemple : un athlète court un marathon en 20533 secondes, ou en 5h42m13s.

Exercice 2 : Année bissextile

On souhaite un programme en Python qui prenne en entrée un numéro n d'année indiqué par l'utilisateur et affiche le booléen `True` si cette année est bissextile et `False` sinon. On rappelle qu'une année est bissextile si elle divisible par 4 sauf si elle est divisible par 100 (auquel cas elle n'est pas bissextile) sauf si elle est divisible par 400 (auquel cas elle l'est).

Q1 : Définir 3 variables booléennes correspondantes aux trois conditions logiques :

- n divisible par 4
- n divisible par 100
- n divisible par 400

Q2 : Ecrire le programme demandé.

Exercice 3 : Résolution d'une équation polynomiale d'ordre 2

Spécifier, écrire et vérifier un programme qui résolve dans $\mathbb{R}$ les équations du second ordre.

Exercice 4 : Somme de nombres

Q1 : En utilisant une boucle `for`, calculer la somme des entiers positifs inférieurs ou égaux à 1000.

Q2 : Modifier le programme pour réaliser la somme des entiers positifs inférieurs ou égaux à 1000 qui ne sont divisibles ni par 3 ni par 5.

Exercice 5 : Table de multiplication

Construire un programme qui affiche des chaînes de caractères représentant la table de multiplication d'un entier compris entre 1 et 10 renseigné par l'utilisateur. On rappelle que l'instruction `print(objet1, objet2, objet3)` affichera les 3 objets bout à bout (les objets peuvent être des constants ou des variables)

Par exemple :

```
+-----+
| Table du 3 |
+-----+
  1 x 3 = 3
+-----+
  ...
+-----+
 10 x 3 = 30
+-----+
```

Exercice 6 : Diviseurs

Q1 : Ecrire un programme qui demande à l'utilisateur de saisir un nombre entier strictement positif, et qui affiche successivement ses diviseurs (entiers strictement positifs).

Q2 : Modifier le programme pour qu'il affiche également le nombre de diviseurs de cet entier.

Exercice 7 : Nombre premier

On souhaite un programme qui demande à l'utilisateur de saisir un nombre entier strictement positif, et qui affiche `True` si ce nombre est premier, `False` sinon.

Q1 : Ecrire ce programme en utilisant une boucle `for`.

Exercice 8 : Intégration numérique**Partie 1 : Représentation d'une fonction**

On cherche à étudier la fonction $f(x) = \cos^2 x$ sur l'intervalle $[2; 4]$.

Nous avons besoin de la fonction $\cos x$. Pour l'importer dans la mémoire, il faut introduire en début de programme :

```
from math import cos
```

- Créer un programme qui, à l'aide d'une boucle, affiche les entiers de 0 à 10.
- Vérifier le résultat.
- Modifier ce programme pour afficher les nombres décimaux de 0 à 2 par pas de 0,2.
- Vérifier le résultat.
- Modifier ce programme pour afficher les nombres décimaux de 2 à 4 par pas de 0,2.
- Vérifier le résultat.

Q1: Analyser la discrétisation de l'intervalle $[2; 4]$ effectuée : nombre de points et pas de discrétisation

- Modifier ce programme pour afficher ces mêmes décimaux (x_i) et leur image par la fonction $f(x_i)$
- Vérifier le résultat.

Pour tracer la courbe représentative de $f(x)$, on utilise un module dédié à l'affichage de graphe : matplotlib. Dans ce module, nous utiliserons la fonction plot.

En début de programme, il faut introduire :

```
from matplotlib.pyplot import plot
```

La courbe sera tracée point par point en utilisant l'instruction (qui permet d'afficher un point de coordonnées x et y , sous forme de croix bleue) :

```
plot(x, y, ' +b')
```

- Introduire la ou les instruction(s) nécessaire(s) dans le programme pour afficher la courbe.

Partie 2 : Intégration par les rectangles à gauche ou par Euler explicite

Q1: Après avoir pris connaissance du document d'accompagnement, mettre en place la relation de récurrence permettant de calculer une approximation de l'intégrale de $f(x)$ sur $[2; 4]$.

Q2: Justifier que le résultat numérique est une approximation du résultat exact.

Partie 3 : Analyse des performances

Q1: Rappeler la solution exacte de l'intégrale recherchée.

Q2: Comparer le résultat du programme avec la solution exacte (quantifier l'erreur).

Modifier l'algorithme pour avoir un pas de discrétisation dix fois plus petit.

Q3: Quel est alors le nombre de point ?

Q4: Quelle est la nouvelle erreur avec la solution exacte ?

Modifier l'algorithme pour utiliser le schéma d'intégration par trapèze.

Q5: Quelle est la nouvelle erreur avec la solution exacte ? Commenter.