# Boucles

Exercice 1 : **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 2 : **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 3 : **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 4 : **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 5 : **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.

# Listes

Exercice 6 : **Manipulation de listes**

Une liste est définie entre crochets, avec un séparateur entre les éléments : la virgule.

```
maliste = ["Lun", "Mar", "Sam", "Dim"]
```

**Q1 :** Définir la liste `maliste` en mémoire.

**Q2 :** Dans un programme, à l'aide d'une boucle, construire une nouvelle liste `newliste`, de même contenu que `maliste`, mais d'ordre inversée. (attention, il est parfois nécessaire de définir une nouvelle variable de liste avant de l'utiliser. Par exemple : `newliste=[]`).

Exercice 7 : **Recherche d'intrus**

On considère la liste hétérogène ['lun','mar',3,'jeu','ven','sam','dim']

**Q1 :** Proposer un programme qui, à l'aide d'une boucle, recherche l'intrus (élément de type différent) dans une liste sachant que l'intrus est unique et qu'il n'est pas en première position. Faire afficher l'index de l'intrus

**Q2 :** Compléter le programme pour qu'il affiche la liste dont il aura retiré l'intrus.

Exercice 8 : **l'algorithme de Luhn**

Certains numéros administratifs (carte bancaire, numéro d'identification SIREN d'une entreprise, plaque d'immatriculation dans certains pays, IMEI d'un téléphone, carte SNCF) doivent respecter certaines règles contraignantes qui limitent les contrefaçons. L'algorithme de Luhn en est une. Il procède en trois étapes.

- L'algorithme remplace un chiffre sur deux, en commençant par l'avant dernier et en se déplaçant de droite à gauche. Il est remplacé par son double. Si un chiffre qui est multiplié par deux est plus grand que neuf, on y retranche neuf, pour qu'il soit compris entre 0 et 9.
- La somme de tous les chiffres obtenus est effectuée.
- Si le reste de la division par 10 est égal à zéro, alors le code original est valide.

**Q1 :** A la main, vérifier que le numéro d'identification 972-487-086 est valide selon la méthode de Luhn.

**Q2 :** Proposer un programme qui, à une liste d'entiers correspondant au code, renvoie vrai si l'algorithme de Luhn vérifie le code.

**Q3 :** Vérifier avec le numéro d'identification 972-487-086, ou avec votre numéro de carte bleue.

# String

Exercice 9 : **Caractères spéciaux**

Taper dans la console les instructions suivantes. Justifier les résultats.

- `print(len('Hello world'))`
- `print('a\nb')`
- `print('c\td')`
- `print('e\bf')`

Exercice 10 : **Tracé de triangles**

**Q1 :** Ecrire un programme qui affiche à l'écran un triangle de 20 lignes remplies avec le caractère * comme ci-dessous :

```
*
**
***
****
```

Exercice 11 : **Chaine de caractères**

Soit une chaine de caractère : `ch='bonjour à tous'`

Ecrire un programme qui :

- Affiche la longueur de la chaîne.
- Remplace toutes les occurrences de la lettre `o` par la lettre `a`.
- Crée la liste de tous les mots de cette chaîne.

# Arrays

Exercice 12 : **Tracé de courbe : comparaison de syntaxes**

Les trois programmes ci-dessous permettent de tracer la même courbe, avec des outils et des syntaxes différentes.

```
from math import cos,pi
from matplotlib import pyplot as plt
for x in range(11):
    y=2*cos(pi*2*x/10)
    plt.plot(x/10,y,'+b')
```

```
from math import cos,pi
from matplotlib import pyplot as plt
X=[t/10 for t in range(11)]
Y=[2*cos(pi*2*x) for x in X]
plt.plot(X,Y)
```

```
from numpy import array,arange,cos,pi
from matplotlib import pyplot as plt
X=arange(0,1.1,0.1)
Y=2*cos(2*pi*X)
plt.plot(X,Y)
```

Q1 : Expliquer les différences entre chaque programme.

Q2 : Quelle est la complexité algorithmique de chaque programme, en fonction de $n$ le nombre de points de la courbe.

# Tracés

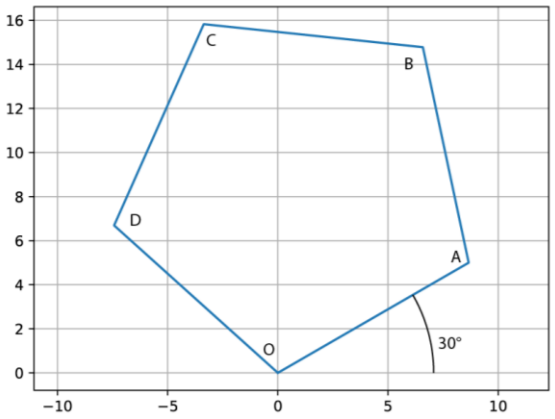
Exercice 13 : **Tracés de polygones**

Dans un repère orthonormé $(O, \vec{x}, \vec{y})$ :

- Donner les coordonnées des 4 sommets d'un carré de côté 50, dont un sommet est confondu avec l'origine $O$, et dont tous les points ont des coordonnées positives ou nulles.
- Tracer ce carré en utilisant `matplotlib`.
- Tracer l'image de ce carré par une translation de vecteur $(25,10)$. On déterminera au préalable les coordonnées des nouveaux sommets.
- Tracer l'image du premier carré par une rotation de centre $O$ et d'angle $+30°$. On déterminera au préalable les coordonnées des nouveaux sommets.

Exercice 14 : **Tracé de pentagone**

Écrire programme permettant de tracer le pentagone régulier comme ci-contre. Ses côtés ont une longueur 10. On pourra utiliser les fonctions `cos` et `sin` après les avoir importées en mémoire, ainsi que la constante `pi`.




```
from numpy import cos,sin,pi
```

# Fonctions

Exercice 15 : **Tracé de fonction**

Créer la liste X=[0,0.01,0.02,...,2]
Créer la fonction :

$$f(x) = \frac{x}{1+x^3}$$

Créer la liste Y=[f(0),f(0.01),f(0.02),...,f(2)]

Au début de votre programme, taper :
`import matplotlib.pyplot as plt`
puis en fin de programme, taper :
`plt.plot(X,Y)`

Exercice 16 : **Analyse statistique**
La fonction `randint` du module `random` permet de générer un nombre entier aléatoire.

- Importer la fonction randint en mémoire vive.
- Consulter l'aide de cette fonction

**Q1.** Quelle instruction permet de reproduire le tirage aléatoire d'un lancer de dé ?

On souhaite un programme qui réalise 30 000 lancers de dé, et qui compte le nombre d'occurrences de chacun des résultats. Il affichera la fréquence statistique de chaque résultat.

- Ecrire ce programme.

**Q2.** L'affichage est-il conforme à la théorie ?

- Faire de même avec cette fois un lancer de deux dés.

Exercice 17 : **Moyenne et variance**
Le fichier `notes.txt` contient les notes obtenues par les élèves de PCSI à un DS.
Pour charger le fichier sous forme d'une liste, utiliser le programme `lecture.py`

Q1 : Ecrire le programme qui crée une liste L d'entiers, correspondant aux notes contenues dans le fichier.

```
print(L)
```

Il est possible d'afficher un graphe des notes (éléments de la liste) en fonction de l'index dans la liste :

```
plot(L)
```

La courbe représentée n'a aucune signification : les grandeurs tracées ne sont pas continues. On préfèrera :

```
plot(L,'o')
```

Il est aussi possible de tracer l'histogramme associé à la répartition de ces notes :

```
hist(L,range(20))
```

On cherche à extraire les données statistiques de cette liste.

Q2 : Spécifier et écrire une fonction `moyenne` qui, à une liste d'entiers, renvoie la moyenne de ces entiers. Vérifier cette fonction sur la liste `L`.

Q3 : Spécifier et écrire une fonction `ecart_type` qui, à une liste d'entiers, renvoie l'écart type de ces entiers. Vérifier cette fonction sur la liste `L`.

Exercice 18 : **Suivi de compte bancaire**

L'objectif de cet exercice est un programme de suivi de solde bancaire. Ce programme s'appuie sur trois fonctions :

- fonction « solde » : affiche le solde du compte
- fonction « debit(somme à retirer) » : retire la somme du compte, et affiche le solde modifié
- fonction « credit(somme à ajouter) » : ajoute la somme au compte, et affiche le solde modifié

Le corps du programme est décrit ci-dessous :

```
1  Solde=1000
2  n='init'
3  while n!=0:
4      n=int(input('faite votre choix : '))
5      if n==1:
6          solde()
7      elif n==2:
8          debit(int(input('quel débit ?')))
9      elif n==3:
10         credit(int(input('quel crédit ?')))
11     elif n==0:
12         print('fin de programme')
13     else:
14         print('choix non valide')
```

Partie 1 : Compréhension du programme

**Q 1:** Quelles sont les variables utilisées par ce programme ?
**Q 2:** Décrire les opérations réalisées à la ligne 8.
**Q 3:** Quelle est l'utilité de la structure conditionnelle ici ?
**Q 4:** Quelle est l'utilité de la structure « while » ?
**Q 5:** Quelle est la différence entre « Solde » et « solde » ?

Partie 2 : Finalisation

**Q 6:** Rédiger les fonctions « solde », « credit », « debit ».
**Q 7:** Vérifier le bon fonctionnement en exécutant le programme.