# TP 1 : Rappels des bases

2 BCPST 2 - TP Informatique

CONSEIL : à chaque nouvelle question faites une nouvelle cellule en tapant "#%%" .
Vous pourrez ainsi traiter toutes les questions dans un même fichier.

Objectif du TP :
- Réviser les conditions `if, elif, else`
- Réviser les boucles `for, while`
- Réviser les fonctions

## I Exercice : Suites récurrentes

**Q.1.** On pose $u_{n+1} = \dfrac{1}{1+u_n^2}$, coder une fonction python `suite(u0,n)` qui prend en entrée un premier terme de la suite `u0`, et un entier `n` et qui afficher les $n$ premiers termes de la suite $(u_n)_n$.
Que remarque-t-on après plusieurs essais du code ?

**Q.2.** La suite de Fibonacci est définie par $u_0 = 0$, $u_1 = 1$ et $u_{n+2} = u_{n+1} + u_n$. Coder une fonction `fibo(n)` qui prend en entrée un entier `n` et qui affiche les $n$ premiers termes de la suite de Fibonacci.

## II Exercice : Nombres parfaits

Un nombre en mathématiques et dit "parfait" si on l'obtient en faisant la somme de ses diviseurs (sans le compter lui-même). Par exemple les diviseurs de 6 sont 1,2,3,6. En faisant 1+2+3, on obtient 6. Donc 6 est parfait.

Il y a en tout 52 nombres parfaits connus. Aujourd'hui encore on ne sait pas s'il en existe une infinité.

En Grèce antique les quatre premiers nombres parfaits étaient connus, nous proposons dans cet exercice de les retrouver avec l'informatique.

**Q.3.** Coder une fonction `somme_diviseurs(n)` qui prend en entrée un entier `n` et qui renvoie la somme des diviseurs de `n`.

Astuce : On rappelle que la commande '`%`' permet de calculer le reste dans la division euclidienne.
Si le reste vaut zéro dans '`a % b`' c'est que le nombre `a` est divisible par `b`.

**Q.4.** En déduire les quatre premiers nombre parfais.

## III Exercice : Les folles années

Les années 2022, 2023, 2024, 2025 étaient tout à fait uniques en leur genre. En effet, elles étaient toutes divisibles par la somme de leur chiffres ! Hormis le cas évident des années 1,2,3,4,5,6,7,8,9,10, c'était la première fois que ça arrivait. L'objectif de cet exercice est de trouver quand la prochaine occurrence arrivera.

**Q.5.** Créer une fonction `somme_chiffres(n)` qui prend en entrée un entier `n` et qui calcule la somme de ses chiffres.
Exemple :

```
print(  somme_chiffres(186) )
-----------------------------------
-> 15
```

Astuce : on rappelle qu'on peut convertir un nombre `n` en chaine de caractère en écrivant : `s = str(n)`.
On peut ensuite récupérer les chiffres sous forme de caractères en via `s[0]`, `s[1]`,....
On peut faire la conversion inverse : `int(s[0])` pour obtenir à nouveau des nombres.

**Q.6.** Ecrire une fonction `divisible_somme(n)` qui prend en entrée un nombre `n` et qui renvoie le booléan `True` si $n$ est divisible par la somme de ses chiffres et `False` sinon.

**Q.7.** En déduire un code python permettant de trouver la prochaine occurrence de quatre années consécutives divisibles par la somme de leur chiffres.

**Q.8.** Quelle sera la première occurrence de 5 années consécutives divisibles par la somme de leur chiffres ?

## IV Exercice : Les dimanches du XXe siècle

On donne les informations suivantes :

- Le premier janvier 1900 était un lundi.
- Les mois de trente jours sont Avril, Juin, Septembre, Novembre. Tous les autres ont 31 jours...
- ...Sauf évidemment Février, qui lui a 28 jours en général, sauf les années bissextiles pour lesquelles il a alors 29 jours.
- Les années bissextiles sont les années divisibles par 4, sauf quand elles tombent pile sur un siècle à moins que ce siècle ne soit divisible par 400.

**Q.9.** Sachant tout cela, donner le nombre de dimanches qui sont tombés pile sur le premier du mois pendant le XXe siècle, c'est à dire entre le 1er janvier 1901 et le 31 décembre 2000.

Astuce 1 : On pourra commencer par faire une fonction python `bissextile(a)` qui prend en entrée une année `a` et qui retourne `True` si l'année `a` est bissextile, et `False` sinon.

Astuce 2 : On pourra créer une liste qui contient le nombre de jours de chaque mois.