TP Python n°1
Comment peut-on conjecturer des propriétés avec Python ?

Exercice 1. Suite de Syracuse

On appelle **suite de Syracuse** des nombres qui se suivent en utilisant le principe de calcul suivant. On choisit le premier terme de la suite.

- Si le terme est pair alors le terme suivant est la moitié de ce terme.
- Si le terme est impair alors le terme suivant est le triple de ce terme augmenté de 1.

▶1a) Lorsque le 1er terme est 10, créer un algorithme qui calcule le terme suivant.

b) Lorsque le 1er terme est 11, créer un algorithme qui calcule le terme suivant.

c) Créer un algorithme qui demande la valeur du 1er terme et qui calcule le terme suivant.

▶2. Modifier votre algorithme pour qu'il demande la valeur du 1er terme puis qui calcule les vingt termes suivants.

▶3. Modifier votre algorithme pour qu'il demande la valeur du 1er terme puis qui calcule les termes suivants jusqu'à obtenir 1.

▶4. On appelle ***Temps de vol de la suite de Syracuse***, le nombre de termes nécessaires pour atteindre 1. Modifier l'algorithme précédent pour qu'il compte le nombre de terme jusqu'à obtenir 1 c'est-à-dire le temps de vol.

▶5. Modifier l'algorithme précédent pour qu'il fasse varier le 1er terme de 1 à 100 et qu'il calcule les temps de vol de chaque suite de Syracuse obtenue alors.

Quel est alors le temps de vol maximal ? Pour quel 1er terme apparaît-il ?

Exercice 2. Multiple

▶1. Créer un algorithme qui détermine si un entier naturel a est multiple d'un entier naturel b.

▶2. Créer un algorithme qui, pour des entiers a et b donnés, déterminer le plus grand multiple de a inférieur ou égal à b.

Exercice 3. Diviseur

▶1. Créer un algorithme qui détermine si un entier naturel b est un diviseur d'un entier naturel a.

▶2. Créer un algorithme qui détermine la liste des diviseurs d'un entier naturel a.

▶3. Créer un algorithme qui déterminer si un entier naturel est premier.

▶4. Créer un algorithme qui donne tous les nombres premiers inférieurs à 100.

Exercice 4. Nombre parfait

On dit qu'un entier naturel est un nombre parfait lorsqu'il est égal à la somme de ses diviseurs positifs autres que lui-même (ex : $6 = 1 + 2 + 3$)

▶1. Ecrire un algorithme qui teste si un nombre donné est parfait ou pas.

▶2. Ecrire un second algorithme donnant la liste des nombres parfaits inférieurs ou égaux à un nombre donné.