Dans ce TP, on écrira d'abord chaque programme sur papier avant de les tester sur machines.

Exercice 1 :

Écrire un programme qui lit deux entiers et renvoie la valeur absolue de la différence de ceux-ci.

Exercice 2 :

Écrire un programme qui demande la saisie d'un nombre N et renvoie N étoiles. Par exemple si $N = 3$, le programme renvoie

```
*
*
*
```

Exercice 3 :

Écrire un programme qui demande la saisie d'un nombre N et renvoie N étoiles. Par exemple si $N = 3$, le programme renvoie

```
***
```

Exercice 4 :

Écrire un programme qui demande la saisie d'un nombre N et renvoie un triangle isocèle rectangle de coté N étoiles. Par exemple si $N = 3$, le programme renvoie :

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

Exercice 5 :

Soit l'algorithme suivant, où `tab` est une liste de nombre :

```
i=0
NbElem = len(tab)
tmp = tab[0]
while i < NbElem :
        if tmp < tab[i]:
                tmp = tab[i]
    i=i+1
print(tmp)
```

(1) Dire à quoi sert l'algorithme.

(2) Réécrire cet algorithme en utilisant une boucle **for**.

(3) Tester sur machine ces deux algorithmes et vérifier qu'ils calculent la même valeur.

(4) Modifier l'algorithme pour qu'il affiche l'indice du maximum de la liste

Exercice 6 :

Soit l'algorithme suivant, où `tab` est une liste de nombre :

```
tmp = 0
i=0
NbElem = len(tab)
while i < NbElem :
        tmp = tmp + tab[i]
    i=i+1
print(tmp)
```

(1) Dire à quoi sert l'algorithme.

(2) Réécrire cet algorithme en utilisant une boucle **for**.

(3) Tester sur machine ces deux algorithmes et vérifier qu'ils calculent la même valeur.

Exercice 7 :

On se donne la suite u définie par :

$$\begin{cases} u_{n+1} & = 2u_n + 1 \\ u_0 & = 1 \end{cases}$$

(1) Proposer un algorithme qui permet d'afficher les 100 premiers termes.

(2) Proposer un algorithme qui permet d'afficher le 100ème terme.

(3) Proposer un algorithme qui permet d'afficher le plus petit indice n tel que $u_n > 1000000$.

Exercice 8 :

Écrire un programme qui génère un nombre aléatoire entre 0 et 99 et fait deviner celui-ci à l'utilisateur. Le programme devra dire si l'utilisateur a trouvé ou pas le nombre, si l'utilisateur ne le trouve pas, le programme devra indiquer si le nombre saisi est trop grand ou trop petit. Une fois le nombre trouvé, le programme indiquera en combien d'essais l'utilisateur l'a trouvé. Par exemple, si le nombre est 62, on aura :

```
-Ordinateur : Saisir un nombre
-Utilisateur : 50
-Ordinateur : Votre nombre est trop petit.
-Utilisateur : 75
-Ordinateur : Votre nombre est trop grand.
-Utilisateur : 62
-Ordinateur : Vous avez trouvé le nombre en 2 essais.
```

En important la bibliothèque random (import random), on a accès à la fonction randint (exemple random.randint(0,99)

Exercice 9 :

La suite de Syracuse d'un nombre entier N est définie par récurrence, de la manière suivante : $u_0 = N$ et pour tout entier $n \geqslant 0$:

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

Une conjecture affirme que, pour tout $N > 0$, il existe un indice n tel que $u_n = 1$. Dans la suite de cet exercice, on suppose que cette conjecture est vraie.

(1) Écrire un algorithme qui étant donné un nombre rentré par l'utilisateur, affiche tous les éléments de la suite.

(2) Modifier ce programme pour que chaque élément de la liste soit stocké dans une liste.

(3) Effectuer un test pour savoir s'il existe n tel que $u_n = 2$.

(4) Le temps de vol est le plus petit indice n tel que $u_n = 1$. Écrire les instructions qui permettent d'afficher le temps de vol de cette suite.

(5) L'altitude maximale est la valeur maximale de la suite.

   a. Écrire les instructions qui permettent d'afficher cette valeur maximale.

   b. Vérifier la terminaison et la correction de ce programme.

(6) Le temps de vol en altitude est le plus petit indice n tel que $u_{n+1} < u_0$. Afficher ce temps de vol.

(7) Modifier cet algorithme pour calculer les temps de vol des suites de Syracuse des N premiers entiers où N est entré par l'utilisateur.