Boucle `while`

1. Principe de la boucle `tant que`

Si l'on ne sait pas à l'avance le nombre de tours de boucle à effectuer, la boucle `for` est difficile à manipuler.

Pour cela, nous allons voir dans ce chapitre la boucle **tant que**, notée `while` en Python qui permet de répondre à ce problème.

Nous avons déjà rencontré :

- d'une part le mécanisme de la boucle `for` qui permet d'effectuer un nombre prédéfini d'itérations d'un bloc d'instructions
- d'autre part le branchement conditionnel `if` qui permet d'effectuer un bloc d'instruction en fonction d'un test effectué sur le moment.

Les boucle *tant que*, ou boucle `while`, combinent ces deux principes d'une nouvelle manière en conditionnant la poursuite des itérations à un test effectué préalablement *à chaque tour de boucle*.

1.1 État des variables

Exécuter le code ci-dessous dans Python Tutor et noter l'état des variables pour chaque ligne exécutée ainsi que l'éventuel affichage.

ligne	`n`	`i`	affichage
1	`11`		
2	`11`	`2`	
3			
4			
5			
3			

ligne	`n`	`i`	affichage
…			

```
[ ]: n = 11
     i = 2
     while i <= n:
         print(i)
         i = 2 * i
     print("Fin")
```

1.2 Lien avec la boucle `for`

Transformer la boucle `for` suivante en boucle `while` :

```
for i in range(n):
    print(i)
print("Fin")
```

1.3 Divergence

2. Sortir d'une boucle

Proposer un algorithme pour le problème suivant :

demander à l'utilisateur de saisir un nombre positif ou nul, et continuer à lui demander de nouvelles saisies tant que la donnée saisie n'est pas valide.

3. Chercher dans un tableau

Écrire un programme `appartient` prenant en paramètre un élément `elem` et un tableau `tab`, et renvoyant `True` si l'élément `elem` apparaît au moins une fois dans le tableau `tab`, et renvoyant `False` sinon.

4. Terminaison