Boucle Tant que

Voici le problème 14 du site **Projet Euler**.

Nous ne donnerons pas la solution, mais nous allons utiliser ce problème pour illustrer l'intérêt de la boucle Tant que.

> The following iterative sequence is defined for the set of positive integers:
>
> $n \to n/2$ (n is even)
> $n \to 3n + 1$ (n is odd)
>
> Using the rule above and starting with 13, we generate the following sequence:
>
> $$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$$
>
> It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.
>
> Which starting number, under one million, produces the longest chain?
>
> **NOTE:** Once the chain starts the terms are allowed to go above one million.

Dans un premier temps on va définir sur l'ensemble des entiers $\mathbb{N}$ une fonction f définie par :

Si n est pair alors $f(n) = n//2$, sinon $f(n) = 3 * n + 1$

En Python :

```
def f(n):
    if n % 2 == 0:
        return n // 2
    else:
        return 3*n + 1
```

La chaîne de calcul 13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1 s'obtient par l'**algorithme suivant** :

"En partant de n valant 13, répéter n prend la valeur de f(n) **tant que n est différent de 1**"

Cependant ce qui nous intéresse plus que les valeurs c'est le nombre de valeurs dans la chaîne, ici 10 valeurs.

D'où un nouvel algorithme :

"En partant de n valant 13 et d'un compteur nb_valeurs, répéter n prend la valeur de f(n) et nb_valeurs prend la valeur nb_valeurs + 1 **tant que n est différent de 1**"

On traduit cet algorithme en une fonction `longueur` en Python :

```
def longueur(n):
    nb_valeurs = 1
    while n != 1:
        n = f(n)
        nb_valeurs = nb_valeurs + 1
    return nb_valeurs
```

Maintenant **explorons** les longueurs des valeurs de n de 1 jusqu'à 100 !

```
for n in range(1,101):
    print("n = ",n," longueur = ",longueur(n))
```

On obtient :

```
n =  1  longueur =  1
n =  2  longueur =  2
n =  3  longueur =  8
n =  4  longueur =  3
n =  5  longueur =  6
n =  6  longueur =  9
n =  7  longueur =  17
n =  8  longueur =  4
n =  9  longueur =  20
n =  10  longueur =  7
n =  11  longueur =  15
n =  12  longueur =  10
n =  13  longueur =  10
n =  14  longueur =  18
n =  15  longueur =  18
n =  16  longueur =  5
n =  17  longueur =  13
n =  18  longueur =  21
n =  19  longueur =  21
n =  20  longueur =  8
n =  21  longueur =  8
n =  22  longueur =  16
n =  23  longueur =  16
n =  24  longueur =  11
n =  25  longueur =  24
n =  26  longueur =  11
n =  27  longueur =  112
```

Ce n'est pas joli à l'affichage ! donc on va **formater la chaîne de caractères** en paramètre de la fonction print()

Voir ici la documentation Python pour les f- strings

`https://docs.python.org/fr/3/tutorial/inputoutput.html#formatted-string-literals`

```
for n in range(1,101):
    print(f"n = {n:3} longueur = {longueur(n):3}")
```

On obtient maintenant :

```
n =   1 longueur =   1
n =   2 longueur =   2
n =   3 longueur =   8
n =   4 longueur =   3
n =   5 longueur =   6
n =   6 longueur =   9
n =   7 longueur =  17
n =   8 longueur =   4
n =   9 longueur =  20
n =  10 longueur =   7
n =  11 longueur =  15
n =  12 longueur =  10
n =  13 longueur =  10
n =  14 longueur =  18
n =  15 longueur =  18
n =  16 longueur =   5
n =  17 longueur =  13
n =  18 longueur =  21
n =  19 longueur =  21
n =  20 longueur =   8
n =  21 longueur =   8
n =  22 longueur =  16
n =  23 longueur =  16
n =  24 longueur =  11
n =  25 longueur =  24
n =  26 longueur =  11
n =  27 longueur = 112
```

A vous de jouer maintenant en trouvant quelle valeur de n avant 1 million donne la longueur la plus grande ?

1 Ecriture en binaire d'un entier

En Français :

"Diviser n par 2 puis le quotient par 2 ainsi de suite jusqu'à ne plus pouvoir le faire lorsque le quotient deviendra nul. La suite des restes des divisions par 2, dans l'ordre inverse de leur arrivée, est la représentation binaire de n."

En Python :

```
def bin(n:int)->str:
    ch = ''
    while n > 0:
        ch = str(n % 2) + ch
        n = n //2
    return ch
```

Exercices

Ex 1

1. Ecrire un algorithme d'abord en **français** puis sous une forme un peu plus structurée, qui permet d'obtenir le nombre de chiffres d'un entier naturel écrit en base 10
2. Définir une fonction Python `nb_chiffres(n)` qui retourne le nombre de chiffres (on n'utilisera pas `len(str(n))`)

Ex 2

1. Ecrire un algorithme d'abord en **français** puis sous une forme un peu plus structurée, qui permet d'obtenir le nombre de bits contenue dans la représentation binaire d'un entier naturel écrit en base 10 par exemple si n = 7 alors $7 = (111)_2$ est représenté par 3 bits
2. Définir une fonction Python `nb_bits(n)` qui retourne le nombre de bits de la représentation binaire de n

Ex 3

Ecrire une fonction Python `exposant_2(n)` qui renvoie l'exposant de la plus grande puissance de 2 contenue dans l'entier naturel `n`

Par exemple `exposant_2(10)` renvoie 3

Ex 4 : Exercice corrigé

On modélise l'évolution de la pression atmosphérique en fonction de l'altitude de la manière suivante :

On considère que la pression au sol à l'altitude 0 est de 1013 hPa (HectoPascal) et que la pression diminue de 1 % tous les 100m

1. Ecrire une fonction Python `f(p)` qui prend en paramètre une pression p et qui renvoie la pression diminuée de 1 %
2. Ecrire une fonction `seuil(p2)` qui prend en paramètre une pression p2 et qui renvoie le nombre d'hectomètres à partir duquel la pression est inférieure à p2.

   A partir de quelle altitude en mètres est on sûr d'avoir une pression inférieure à 900 hPa ?

Corrigé :

1. Voir le cours de maths une diminution de t% correspond à la fonction linéaire $x \to x \times (1 - \frac{t}{100})$

   d'où a fonction Python

```
def f(p):
    return 0.99*p
```

2. pour $n = 0$ (au niveau du sol) $p = 1013$

   ensuite pour $n = 1$ (à 100 m d'altitude) $p = 1013 * 0.99$

   ensuite pour $n = 2$ (à 200 m d'altitude) $p = 1013 * 0.99 * 0.99$

   On va calculer **tant que** p > p2 d'où la fonction Python.

   A partir de 1200 m on est sûr d'avoir une pression inférieure à 900 hPa **avec ce modèle qui n'est pas la réalité !**

```
def seuil(p2):
    """
    >>> seuil(900)
    12
    """
    p = 1013
    n = 0
    while p < p2:
        p *= 0.99
        n += 1
    return n
```

Ex 5

On modélise l'évolution de la pression atmosphérique en fonction de la profondeur dans l'eau

On considère que la pression au sol à l'altitude 0 est de 1013 hPa (HectoPascal) = 1 bar et que la pression augmente sous l'eau de 1 bar tous les 10m

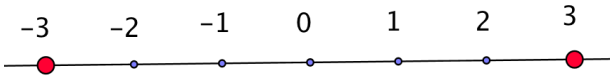
Ecrire une fonction seuil(p) qui prend en paramètre une pression p et qui renvoie le nombre de décamètres (dizaine de mètres) à partir duquel la pression sous l'eau sera supérieure à p.

A partir de quelle profondeur en mètres est on sûr d'avoir une pression supérieure à 5 bars ?

Ex 6

Un jeton part de l'origine et se déplace aléatoirement ainsi : Si le lancer d'une pièce a donné **Pile** alors le jeton se déplace d'une unité vers la droite sinon il se déplace d'une unité vers la gauche et le déplacement s'arrête lorsque l'abscisse du jeton est 3 ou -3. Il s'agit de compter **le nombre moyen** de lancers de la pièce par déplacement

1. Compléter l'algorithme suivant (on dira que Pile est traduit par 0 et Face par 1)



Algorithme 1 : Déplacement du jeton

Données : Un jeton à l'origine, un intervalle gradué [-3 ;3] et un générateur de nombres aléatoires

Résultat : Le nombre de lancers de la pièce

```
1  début
2      position ← 0
3      nbLancers ← 0
4      tant que ..... faire
5          piece ← entierAleatoire(0,1)
6          .....
7          .....
8          .....
9          .....
10     fin
11     afficher(nbLancers)
12 fin
```

2. Traduire l'algorithme par une fonction Python `nb_lancers()`
3. Exécuter un "grand nombre de fois" , par exemple 1000 fois la fonction précédente et calculer la moyenne des nombres de lancers.

   N'est ce pas remarquable ?Faire une **conjecture** et tester la

Ex 7

Dans le Jeu **Devine un nombre** le joueur doit deviner un nombre choisi au hasard par l'ordinateur entre 0 et 1000

A chaque tour le joueur propose une solution et l'ordinateur répond si la solution proposée est plus petite ou plus grande que le nombre tiré au hasard au départ

Ecrire un programme permettant de jouer à ce jeu

Ex 8

Pour pouvoir écrire en Python une fonction `binaire(n)` qui renvoie l'écriture binaire de `n` sous la forme d'une chaîne de caractères il nous faut travailler **la concaténation de chaînes de caractères**

1. Exécutez à la console

```
>>> chaine = "1"
>>> chaine = chaine + "2"
>>> chaine
?
```

   Qu'obtenez vous ?

2. Exécutez à la console

```
>>> chaine = "1"
>>> chaine = "2" +chaine
>>> chaine
?
```

   Qu'obtenez vous ?

3. Ecrire une fonction `binaire(n)` qui renvoie l'écriture binaire de `n` sous la forme d'une chaîne de caractères

Ex 9

Etant donné un nombre entier naturel a et un nombre entier naturel strictement positif b il existe un unique entier naturel q et un unique entier naturel $0 \leqslant r < b$ tel que

$a = bq + r$ (division euclidienne de a par b)

Voici un algorithme pour calculer q et r étant donnés a et b

Algorithme 2 : Division euclidienne

Données : un entier naturel a, un entier naturel $b > 0$
Résultat : deux entiers q et r tel que $a = bq + r$

```
1 début
2 |  q ← 0
3 |  r ← a
4 |  tant que  r >= b faire
5 |  |  r ← r - b
6 |  fin
7 |  q ← q + 1
8 fin
```

Compléter la fonction Python suivante `div(a,b)` (que renvoie la fonction ?)

```
def div(a,b):
    q = 0
    r = a
    .........
    return q,r
```

Ex 10

Quelles sont les briques élémentaires d'un langage de programmation ?

Ex 11

Dès l'antiquité on connaissait un algorithme pour calculer la racine carrée d'un nombre a > 1

Cet algorithme est de nature géométrique. On part d'un rectangle de longueur a et de largeur 1. Ce rectangle est d'aire a. Ensuite on va créer une suite de rectangles "de plus en plus carrés", toujours d'aire a. Autrement dit un des côtés de ces rectangles, lorsqu'on a suffisamment calculé, aura une valeur proche de $\sqrt{a}$

Algorithme 3 : Racine carrée de a > 0

Données : a un nombre positif, un seuil
Résultat : Une valeur approchée de la racine carrée de a

```
1  début
2      // On part d'un rectangle d'aire a
3      // de longueur L, de largeur w
4      // et tel que Lxw = a
5      L ← a
6      w ← 1
7      tant que abs(L-w) > seuil faire
8          // la largeur w prend pour valeur
9          // la moyenne de w et de L
10         w ← (L + w) / 2
11         // L fois w est égal à a
12         L ← 2 / w
13     fin
14 fin
```

Ecrire une fonction `racine(a)` qui renvoie une valeur approchée de la racine carrée de a