

Corrigé – La boucle non bornée

Exercice 1

```
1  compteur = 1
2  while compteur < 10:
3      compteur = compteur * 2
4  print(compteur)
```

1. On observe bien la boucle grâce à la répétition des lignes 2 et 3.

N° de ligne	condition	compteur
1	-	1
2	True	1
3	True	2
2	True	2
3	True	4
2	True	4
3	True	8
2	True	8
3	True	16
2	False	16

2. 16

Exercice 2

```
1  n = 0
2  while n < 5:
3      n = n + 1
4  print(n)
```

Exercice 3

1. mdp est le mot de passe en clair (ce qui n'est pas du tout sécurisé et qui est une mauvaise idée en réalité) et rep est la valeur entrée par l'utilisateur.

- 2.

```
1  mdp = "hj51dpM@"
2  rep = ""
3
4  while rep != mdp:
5      rep = input("Entrer le mot de passe pour accéder à la page.")
6
7  print("Accès autorisé")
```

Exercice 4

1. 0 0
8 2
1

2. • Script 1 : 5 tours ; • Script 2 : 8 tours ; • Script 3 : 10 tours.

3. • Script 1 : `for n in range(5)` • Script 2 : `for n in range(7)`

/!\ L'existence de `n` dans le script 3 est absurde : cette variable ne sert a priori à rien.

Exercice 5

1. 6 tours et renvoie `False`

2. 77 tours et renvoie `True` (on ne va évidemment pas compter manuellement les 77 tours mais il est facile de faire la déduction sachant que 79 est bien un nombre premier : on commence à la valeur 2 et on l'incrmente sans sortir de la boucle jusqu'à 79. On fait donc $79 - 2 = 77$ tours.)

Exercice 6

```
1  somme = 0
2  nb = 0
3  while nb < 100: # 0 à 99 => 100 nombres différents
4      carre = nb**2
5      somme = somme + carre
6      nb = nb + 1
7  print(somme)
```

Exercice 7

```
1  # FONCTIONS 1)
2  def somme_inverses(n):
3      somme = 0
4      denominateur = 1
5      while denominateur <= n:
6          somme = somme + 1/denominateur
7          denominateur = denominateur + 1
8      return somme
9
10 # SCRIPT
11 # tests : on vérifie que la fonction fait ce qu'on attend
12 assert somme_inverses(1) == 1
13 assert somme_inverses(2) == 1 + 1/2
14 assert somme_inverses(4) == 1 + 1/2 + 1/3 + 1/4
15
16 # 2) plus petit entier n tel que Hn > 10
17 n = 0
18 while somme_inverses(n) < 10:
19     n = n + 1
20 print(n)
```

Exercice 8

« [...] un nombre parfait est un entier naturel égal [...] à la somme de ses diviseurs stricts. » *in Wikipedia*

1. True
False
True
2. Ce script tourne tant qu'il n'a pas trouvé un nombre impair parfait.

Exercice 9

```
1  # script 1           8  # script 2           15  # script 3
2  n = 0                9  n = 0                16  x = 0
3  s = 0                10 s = 0                17  n = 0
4  while n < 10:        11 while s < 1000:       18 while x != 10:
5      s = s + n        12     n = n + 1         19     x = x + 0.1
6                        13                        20     n = n + 1
7  print(s)             14 print(n)             21 print(n)
```

- Script 1 : n n'est jamais modifié donc la condition n'est jamais fausse et la boucle ne s'arrêtera pas. Il faudrait ajouter `n = n + 1`

```
1  # script 1
2  n = 0
3  s = 0
4  while n < 10:
5      s = s + n
6      n = n + 1
7  print(s)
```

- Script 2 : s n'est jamais modifié donc la condition n'est jamais fausse et la boucle ne s'arrêtera pas. Il faudrait par exemple ajouter `s = s + 1` (on ne peut pas vraiment savoir ce qui est attendu ici).

```
1  # script 2
2  n = 0
3  s = 0
4  while s < 1000:
5      n = n + 1
6      s = s + 1
7  print(n)
```

- Script 3 : x n'est jamais égal à 10. Ceci sera étudié plus tard dans l'année mais on peut déjà le vérifier en exécutant le programme avec les affichages adéquats. Il faut donc changer le `!=` (différent de) en le remplaçant par un `<` (inférieur).

```
1  # script 3
2  x = 0
3  n = 0
4  while x < 10:
5      x = x + 0.1
6      n = n + 1
7  print(n)
```