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'incrémente sans sortir de la boucle jusqu'à 79. On fait donc $79 - 2 = 77$ tours.)

Exercice 6

```python
somme = 0
nb = 0
while nb < 100: # 0 à 99 => 100 nombres différents
    carre = nb**2
    somme = somme + carre
    nb = nb + 1
print(somme)
```

Exercice 7

```python
# FONCTIONS 1)
def somme_inverses(n):
    somme = 0
    denominateur = 1
    while denominateur <= n:
        somme = somme + 1/denominateur
        denominateur = denominateur + 1
    return somme

# SCRIPT
# tests : on vérifie que la fonction fait ce qu'on attend
assert somme_inverses(1) == 1
assert somme_inverses(2) == 1 + 1/2
assert somme_inverses(4) == 1 + 1/2 + 1/3 + 1/4

# 2) plus petit entier n tel que Hn > 10
n = 0
while somme_inverses(n) < 10:
    n = n + 1
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

```
# script 1
n = 0
s = 0
while n < 10:
    s = s + n

print(s)
```

```
# script 2
n = 0
s = 0
while s < 1000:
    n = n + 1

print(n)
```

```
# script 3
x = 0
n = 0
while x != 10:
    x = x + 0.1
    n = n + 1
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`

```
# script 1
n = 0
s = 0
while n < 10:
    s = s + n
    n = n + 1
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).

```
# script 2
n = 0
s = 0
while s < 1000:
    n = n + 1
    s = s + 1
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).

```
# script 3
x = 0
n = 0
while x < 10:
    x = x + 0.1
    n = n + 1
print(n)
```