

Exercice 1 :

1.

$$somme(n) = \begin{cases} 0 & \text{si } n = 0 \\ n + somme(n-1) & \text{si } n > 0 \end{cases}$$

2. Programme

```
1 def somme(n: int)->int:
2     if n == 0:
3         return 0
4     else:
5         return n + somme(n-1)
6
7 print(somme(10))
```

Exercice 2 :

1.

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{si } n > 0 \end{cases}$$

2. Programme

```
1 def factorielle(n: int)->int:
2     if n == 0:
3         return 1
4     else:
5         return n * factorielle(n-1)
6
7 print(factorielle(10))
```

Exercice 3 : La première version affiche les termes à chaque appel.

```
1 def syracuse(u: int)->None:
2     print(u, end=" ")
3     if u > 1:
4         if u % 2 == 0:
5             syracuse(u // 2)
6         else:
7             syracuse(3 * u + 1)
8
9 syracuse(5)
```

Une seconde version enregistre les termes dans une liste et renvoie cette liste. Il faut noter l'utilisation d'une variable supplémentaire initialisée par défaut.

```
1 def syracuse2(u: int, l: list = [])->list:
2     l.append(u)
3     if u > 1:
4         if u % 2 == 0:
5             syracuse2(u // 2, l)
6         else:
7             syracuse2(3 * u + 1, l)
8     return l
9
10 print(syracuse2(5))
```

Conjecture de Syracuse : quelle que soit la valeur de u_0 il existe un n tel que $u_n = 1$. Toujours pas prouvée à ce jour.

Il faut noter que la fonction ne renvoie rien à chaque appel cette fois.

Exercice 4 :

1. Programme

```
1 def entiers(i: int, k: int)->None:
2     if i <= k:
3         print(i, end=" ")
4         entiers(i+1, k)
5
6 entiers(0,3)
```

2. Programme

```
1 def impairs(i: int, k: int)->None:
2     if i <= k:
3         if i%2 == 1:
4             print(i, end=" ")
5             impairs(i+2, k)
6
7 impairs(1,8)
```

Exercice 5 : Appliquons la méthode d'Euclide :

$$pgcd(a,b) = \begin{cases} b & \text{si } a = 0 \\ pgcd(b\%a, b) & \text{sinon} \end{cases}$$

```
1 def pgcd(a: int, b: int)->int:
2     if a == 0:
3         return b
4     else:
5         return pgcd(b%a, a)
6
7 print(pgcd(20,35))
```

Exercice 6 :

```
1 def nombre_chiffres(n: int)->int:
2     if n <= 9:
3         return 1
4     else:
5         return 1 + nombre_chiffres(n//10)
6
7 print(nombre_chiffres(123))
```

Exercice 7 :

1. Fonction

```
1 def C(n: int, p: int)->int:
2     if p == 0 or n == p:
3         return 1
4     else:
5         return C(n-1, p-1) + C(n-1, p)
```

2. Programme

```
1 for i in range(10):
2     for j in range(i+1):
3         print(C(i, j), sep=" ", end=" ")
4     print()
```