Exercice 1 :

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

2. Programme

```
def somme(n: int)->int:
    if n == 0:
        return 0
    else:
        return n + somme(n-1)

print(somme(10))
```

Exercice 2 :

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

2. Programme

```
def factorielle(n: int)->int:
    if n == 0:
        return 1
    else:
        return n * factorielle(n-1)

print(factorielle(10))
```

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

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

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.

```
def syracuse2(u: int, l: list = [])->list:
    l.append(u)
    if u > 1:
        if u % 2 == 0:
            syracuse2(u // 2, l)
        else:
            syracuse2(3 * u + 1, l)
    return l

print(syracuse2(5))
```


Christophe Viroulaud

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

```
def entiers(i: int, k: int)->None:
    if i <= k:
        print(i, end=" ")
        entiers(i+1, k)

entiers(0,3)
```

2. Programme

```
def impairs(i: int, k: int)->None:
    if i <= k:
        if i%2 == 1:
            print(i, end=" ")
        impairs(i+2, k)

impairs(1,8)
```

Exercice 5 : Appliquons la méthode d'Euclide :

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

```
def pgcd(a: int, b: int)->int:
    if a == 0:
        return b
    else:
        return pgcd(b%a, a)

print(pgcd(20,35))
```

Exercice 6 :

```
def nombre_chiffres(n: int)->int:
    if n <= 9:
        return 1
    else:
        return 1 + nombre_chiffres(n//10)

print(nombre_chiffres(123))
```

Exercice 7 :

Christophe Viroulaud

1. Fonction

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

2. Programme

```python
for n in range(10):
    for p in range(n+1):
        print(C(n, p), end=" ")
    print()
```


Christophe Viroulaud