Exercice 5. Méthode `join` pour les chaînes de caractères

```
def join(sep, L):
    """join(sep: str, L: str list) -> str"""
    if L == []:
        return ""
    res = L[0]
    for s in L[1:]:
        res += sep + s
    return res
```

Exercice 6. Méthode `split` pour les chaînes de caractères

Question 1 –

```
def debut_mot(s, i):
    j = i
    while j < len(s) and s[j] == " ":
        j += 1
    return j
```

Question 2 –

```
def fin_mot(s, i):
    j = i
    while j < len(s) and s[j] != " ":
        j += 1
    return j
```

Question 3 –

```
def split(s):
    L = []
    i = 0
    while i < len(s):
        deb = debut_mot(s, i)
        fin = fin_mot(s, deb)
        if deb != len(s):
            L.append(s[deb:fin])
        i = fin
    return L
```

Exercice 7. Approximation de $\sqrt{2}$ par dichotomie

Question 1 –

```
def approx_sqrt2(epsilon):
    """approx_sqrt2(epsilon: float) -> float"""
    a = 1
    b = 2
    while b-a > 2*epsilon:
        m = (a+b)/2
        f_m = m**2-2
        if f_m == 0:
            a = m
            b = m
        elif f_m > 0:
            b = m
        else:
            a = m
    return (a+b)/2
```

Question 2 – On suppose qu'à chaque étape, $f_m \neq 0$. Pour chaque k, on note l_k la taille de l'intervalle I_k. Alors :

$$l_0 = 1 \text{ et } l_{k+1} = l_k/2$$

Ainsi, par une récurrence immédiate :

$$\forall k : l_k = 1/2^k$$

On cherche le plus petit entier k tel que $l_k \leqslant 2\varepsilon$, c'est à dire le plus petit k tel que $k \geqslant log_2(1/\varepsilon) - 1$. Il s'agit de l'entier :

$$k_0 = \left\lceil log_2(1/\varepsilon) \right\rceil - 1.$$

Finalement le nombre de tours de boucle est K vérifie :

- $\rightarrow$ $K = 0$ si $\varepsilon \geqslant 1$.
- $\rightarrow$ $K = k_0$ si $f_m \neq 0$ à chaque étape et $\varepsilon < 1$.
- $\rightarrow$ $K \leqslant k_0$ si $f_m = 0$ à une certaine étape et $\varepsilon < 1$.