Récursivité
Première approche

Christophe Viroulaud

Terminale - NSI

Lang 05

Observation

Les constructions élémentaires (variable, condition, boucle,...) permettent de construire n'importe quel programme. Cependant il existe d'autres méthode de programmation, que celle abordée l'année dernière, qui facilite l'implémentation de certains algorithmes.

Découvrir la **programation récursive**.

Sommaire

Méthode de programmation classique

Figure 1 – Le TriNitroToluène dans Minecraft

Contexte

Après avoir été activé, le bloc de TNT commence à fumer et à clignoter. Il s'agit à présent d'une entité (bloc non-solide de 0,98m d'arête). Son fusible dure 80 ticks d'horloge (environ quatre secondes) ou un nombre aléatoire entre 10 à 30 ticks si elle est détruite par une explosion.

Extrait de https://minecraft.fandom.com

Activité 1 : Écrire une fonction `exploser_iteratif(duree:int)` $\rightarrow$ `None` qui affiche la durée restante à chaque tour, puis BOUM quand la durée atteint 0.

Correction

```
1  def exploser_iteratif(duree: int) -> None:
2      while duree > 0:
3          print(duree)
4          duree = duree - 1
5
6      print("boum")
7
8  # programme principal
9  exploser_iteratif(4)
```

Sommaire

Programmation récursive

```python
def exploser_iteratif(duree: int) -> None:
    while duree > 0:
        print(duree)
        duree = duree - 1

    print("boum")
```

Observation

Dans la fonction itérative, un code est exécuté tant que la **condition** est vérifiée.

À retenir

Une fonction récursive est une fonction qui s'appelle elle-même, à l'intérieur de son propre code.

```python
def exploser_recursif(duree: int) -> None:
    if duree > 0:
        print(duree)
        exploser_recursif(duree - 1)
    else:
        print("boum")
```

Code 1 – Version récursive de la fonction précédente.

Activité 2 : Recopier la fonction récursive et vérifier qu'elle se comporte comme la version itérative.

Correction

```
1 exploser_recursif(4)
```

Code 2 – Appel principal

```
1 4
2 3
3 2
4 1
5 BOUM
```

Code 3 – Affichage

Sommaire

Pile d'appels

```
1 def exploser_recursif(duree: int) -> None:
2     if duree > 0:
3         print(duree)
4         exploser_recursif(duree - 1)
5     else:
6         print("boum")
```

Code 4 – Lors du premier appel de la fonction, la valeur `4` est passé au paramètre **`duree`**.

Observation

La valeur étant positive, la condition est vérifiée. L'exécution de l'appel **`exploser_recursif(4)`** se met en pause et l'appel **`exploser_recursif(3)`** est lancé.

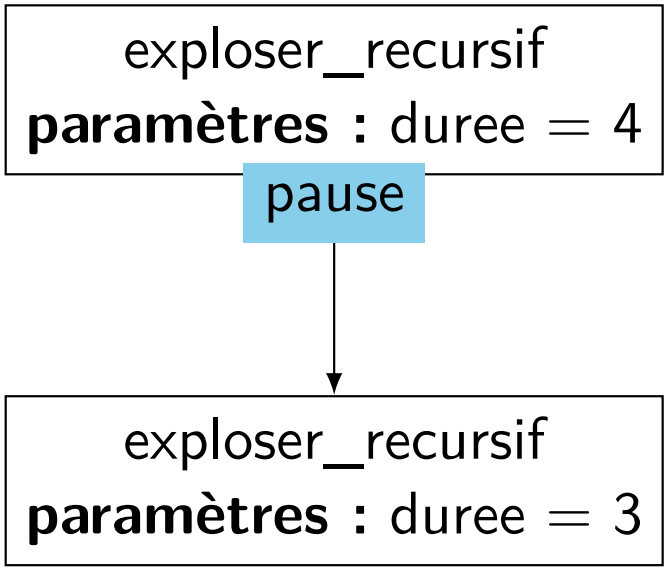

Figure 2 – La **pile d'appels** enregistre l'état de chaque appel.

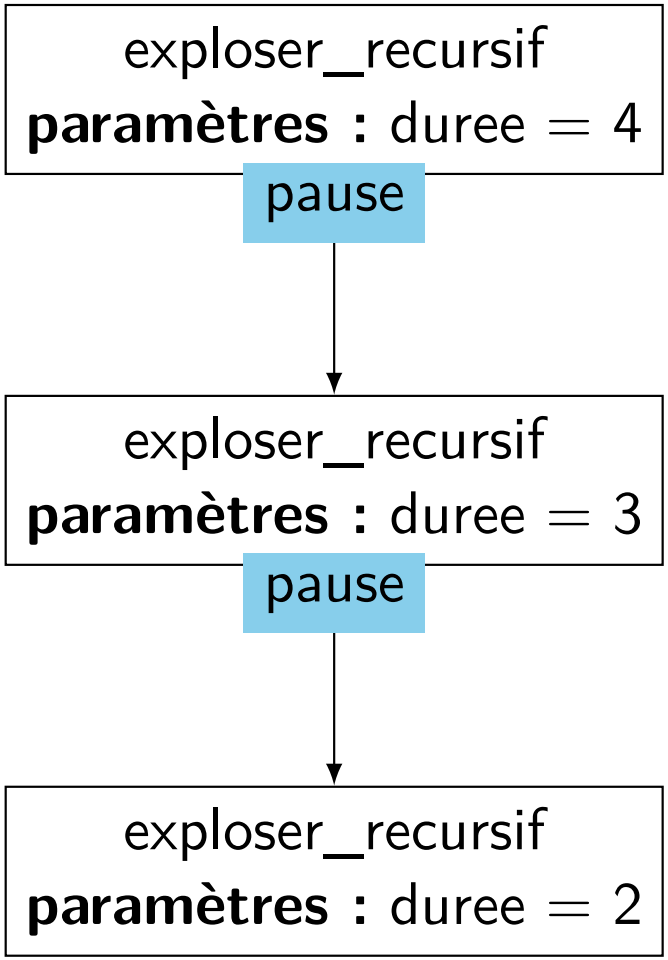

Figure 3 – La condition est encore vérifiée : appel suivant.

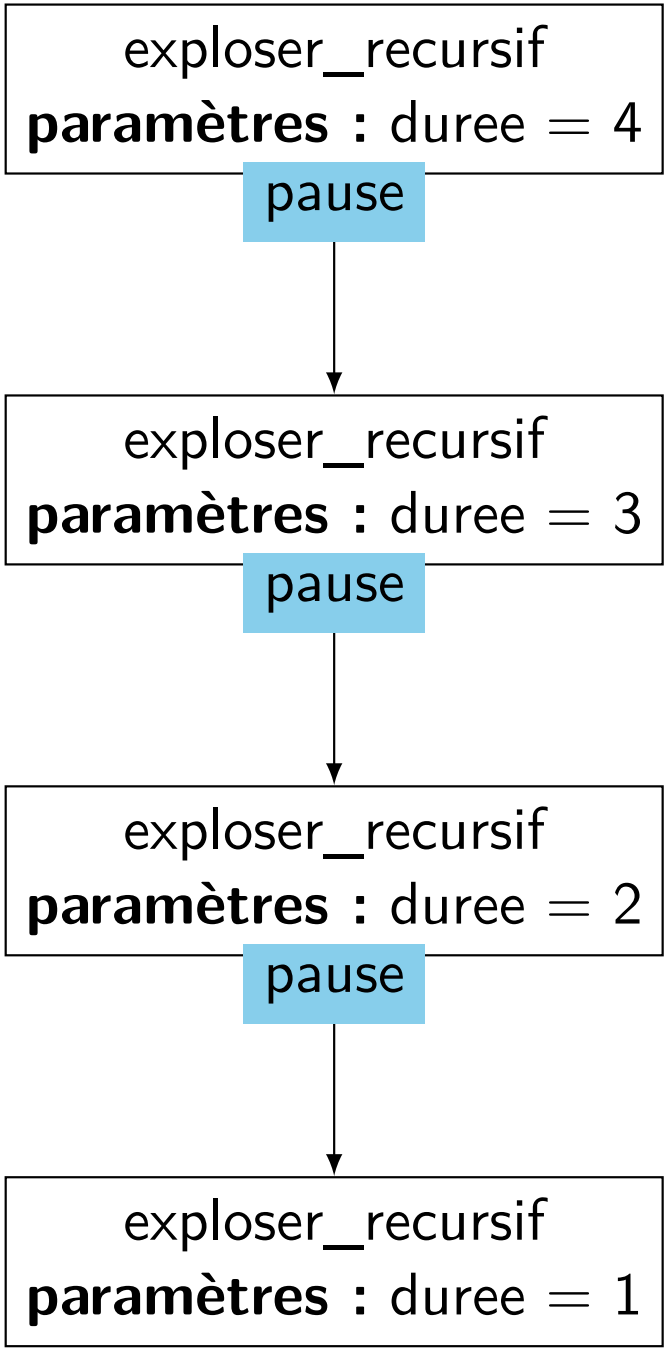

Figure 4 – La condition est encore vérifiée : appel suivant.

```python
def exploser_recursif(duree: int) -> None:
    if duree > 0:
        print(duree)
        exploser_recursif(duree - 1)
    else:
        print("boum")
```

Code 5 – Que se passe-t-il quand **`duree = 0`** ?

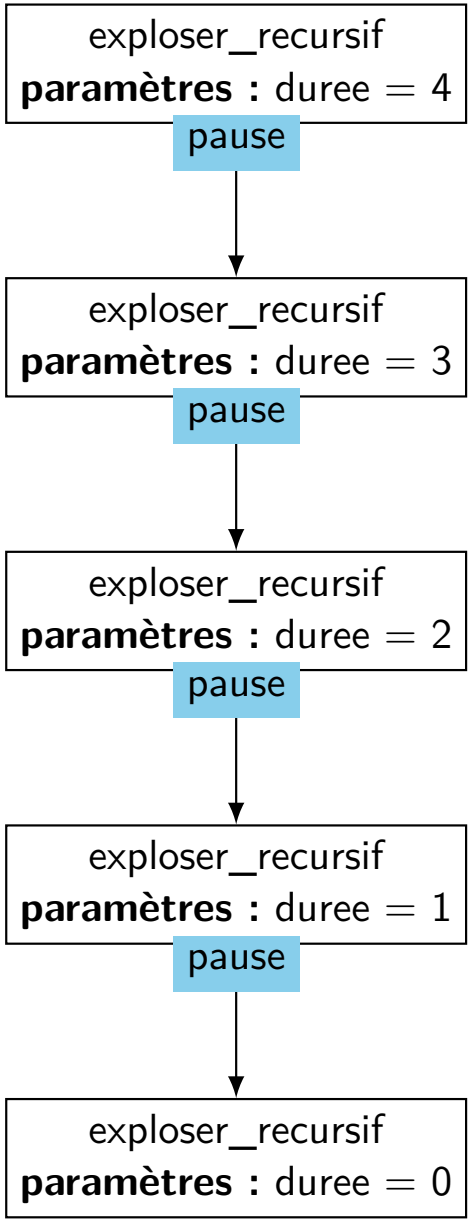

Figure 5 – La condition n'est plus vérifiée : la fonction poursuit l'exécution de son code.

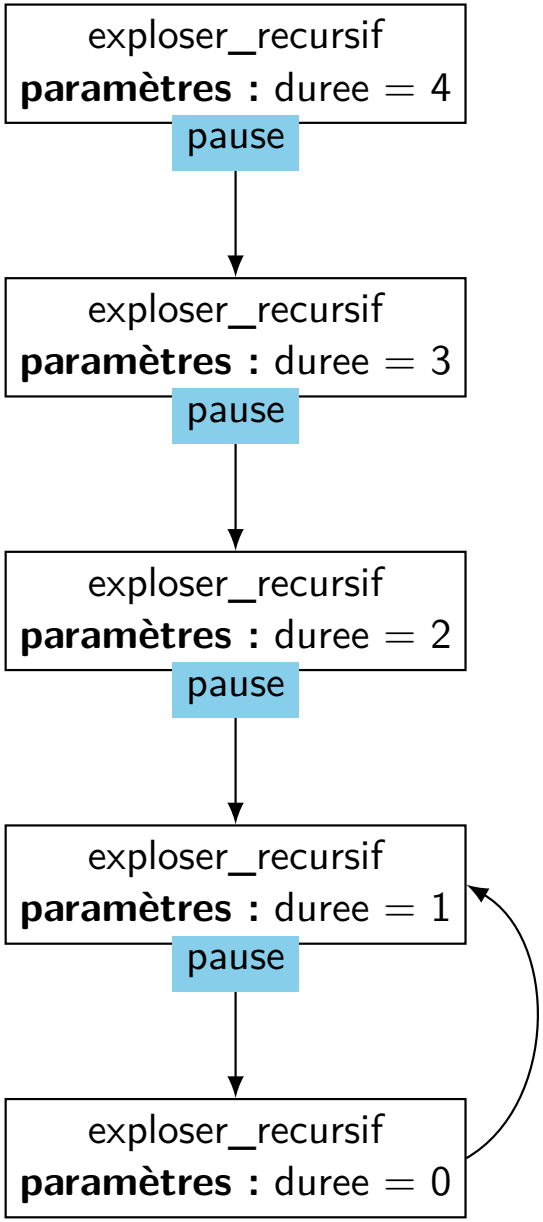

Figure 6 – La fonction se termine et ne renvoie rien ici (d'après sa signature).

```
1 def exploser_recursif(duree: int) -> None:
2     if duree > 0:
3         print(duree)
4         exploser_recursif(duree - 1)
5     else:
6         print("boum")
```

Code 6 – La fonction reprend le cours de son exécution.

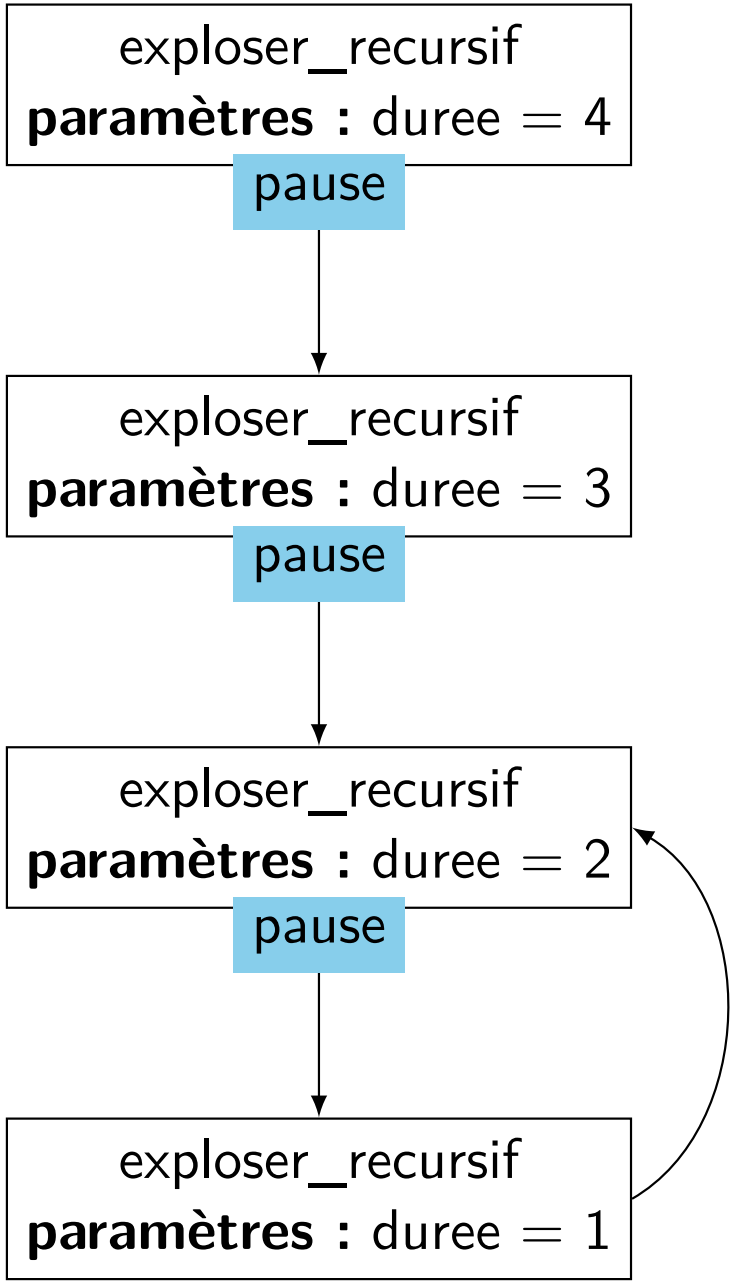

Figure 7 – La fonction reprend où elle s'était arrêtée.

À retenir

Lors d'un appel récursif :

- la fonction appelante se met en pause,
- la **pile d'appels** enregistre l'état de chaque appel.

Lors de la remontée d'appels, la fonction reprend son exécution là où elle s'était mise en pause.

Sommaire

Cas limites

Activité 3 : Tester le code suivant :

```
1 def exploser_recursif(duree: int) -> None:
2     print(duree)
3     exploser_recursif(duree - 1)
4
5     print("boum")
```

Expliquer le comportement.

Correction

```
RecursionError: maximum recursion depth exceeded
   while calling a Python object
```

Code 7 – Les appels récursifs ne s'arrêtent jamais. La fonction boucle indéfiniment.

Remarque

Par défaut, Python arrête automatiquement le nombre d'appels récursifs après 1000.

À retenir

Dans un fonction récursive, on a obligatoirement :

- un appel de la fonction elle-même : l'**appel récursif**,
- un **cas limite** : pour arrêter les appels récursifs.

```python
def exploser_recursif(duree: int) -> None:
    if duree == 0:
        print("boum")
    else:
        print(duree)
        exploser_recursif(duree - 1)
```

Code 8 – Il peut être judicieux de faire apparaître clairement le cas limite.

Activité 4 : Prédire et expliquer le comportement de ce code :

```
1 def exploser_recursif(duree: int) -> None:
2     if duree > 0:
3         print(duree)
4         exploser_recursif(duree - 1)
5
6     print("boum")
```

Correction

```
4
3
2
1
boum
boum
boum
boum
boum
```

Code 9 – Résultat obtenu

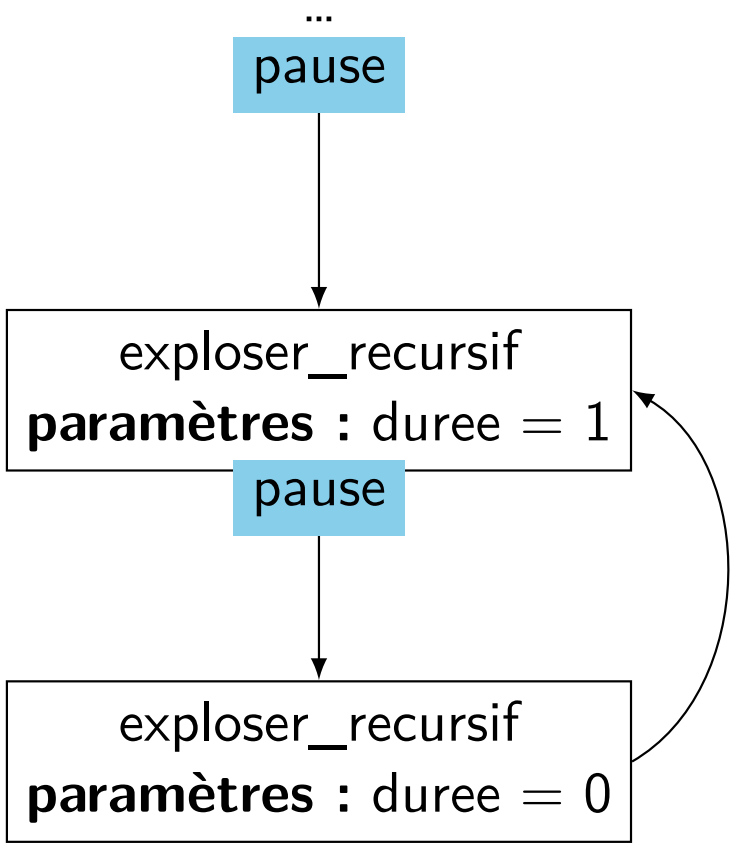

Figure 8 – La fonction reprend où elle s'était arrêtée.

```
1 def exploser_recursif(duree: int) -> None:
2     if duree > 0:
3         print(duree)
4         exploser_recursif(duree - 1)
5     # la fonction reprend à ce point
6     print("boum")
```