Étape III – Programmation dynamique

- Cours 10 Récursivité
- Cours 11 Examen 2 PL (en classe)
- Cours 12 Algorithme Glouton
- Cours 13 Programmation Dynamique
- Cours 14 Programmation Dynamique
- Cours 15 Examen final

Étape III – Notes de cours – NSI



Auteur: David Roche

Récursivité -

https://dav74.github.io/site_nsi_term/c4c/

```
NSI terminale
                                                                                                                                                                        Q Rechercher
NSI terminale
                                  Cours
Partie 1
Partie 2
 C4 · récursivité
                                 Considérons le programme suivant :
   Cours
                                                                                                                                                                                                 Activités
                                       print ("Début fonction fctA")
   Exercices
   Révision
                                       while i<5:
                                          print(f"fctA {i}")
  C5: listes-piles-files
                                            i = i + 1
  C6: les dictionnaires
                                       print ("Fin fonction fctA")
  C7: les arbres
  C8: algo arbres binaires
                                       print ("Début fonction fctB")
  *C9: les graphes
                                       while i<5:
  *C10: algo graphes
Partie 3
                                                print("Retour à la fonction fctB")
Partie 4
                                            print(f"fctB {i}")
Partie 5
                                            i = i + 1
                                       print ("Fin fonction fctB")
Partie 6
                                   fctB()
```

1) Pile d'exécution

```
1     def fcta():
2         i = 0
3         while i < 5:
4         print(f"fctA {i}")
5         i = i + 1</pre>
8         def fctb():
9         i = 0
while i < 5:
10         while i < 5:
11         if i == 3:
               fcta()
print(f"fctB {i}")
14         i = i + 1</pre>
```

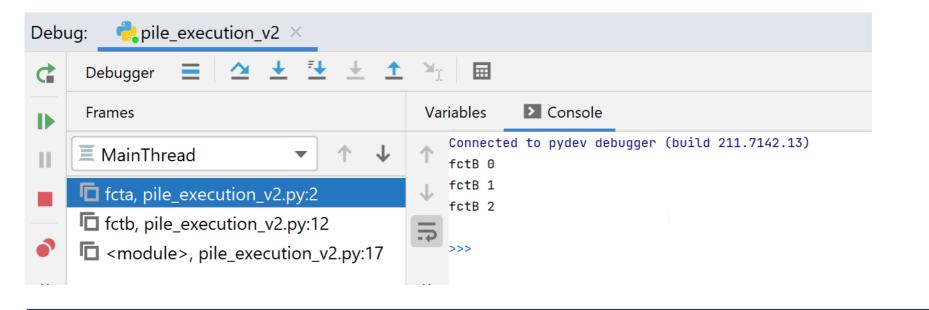
 Lecture des notes de cours de la section 1 Pile d'exécution P 2-4

Pile d'exécution dans le débogueur

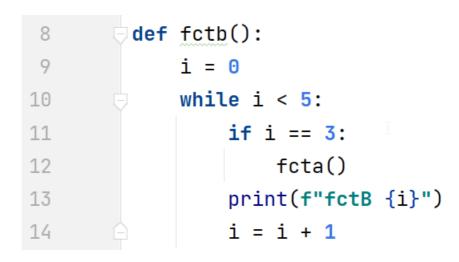
- Que se passe-t-il à la ligne 12 ?
- Combien de variable i existe en mémoire lorsque l'on exécute la ligne 5, et 13 ?

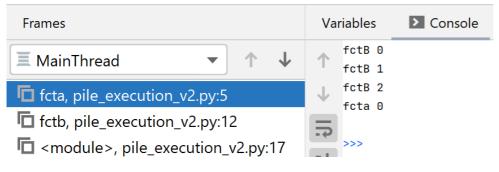
L12; appel de fcta() et empilement de fctb()

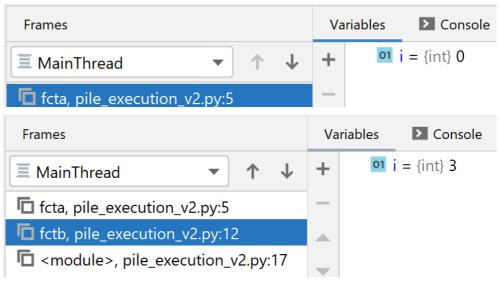
```
def fcta():
                                                  def fctb():
             i = 0
                                          9
                                                      i = 0
             while i < 5:
                                                      while i < 5:
                                         10
                 print(f"fcta {i}")
                                                           if i == 3:
                                         11
                                                               fcta()
                 i = i + 1
                                         12
5
                                                           print(f"fctB {i}")
                                         13
                                                           i = i + 1
                                         14
```



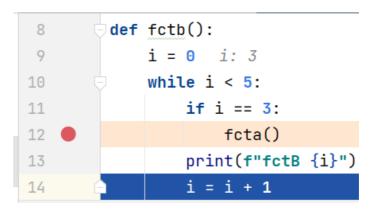
L5; 2 variables i distinctes sont en mémoire

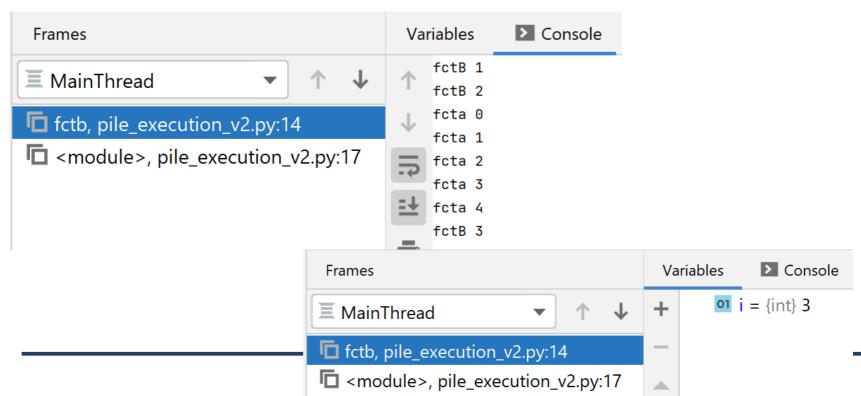






L13; 1 seule variable i est en mémoire





Pile d'exécution

Une pile d'exécution permet d'enregistrer des informations sur les fonctions en cours d'exécution dans un programme.

2) Fonction récursive

 Lecture des notes de cours de la section 2 « Fonction récursive »

Fonction récursive

Une fonction peut s'appeler elle-même, on parle alors de fonction récursive.

Fonction récursive

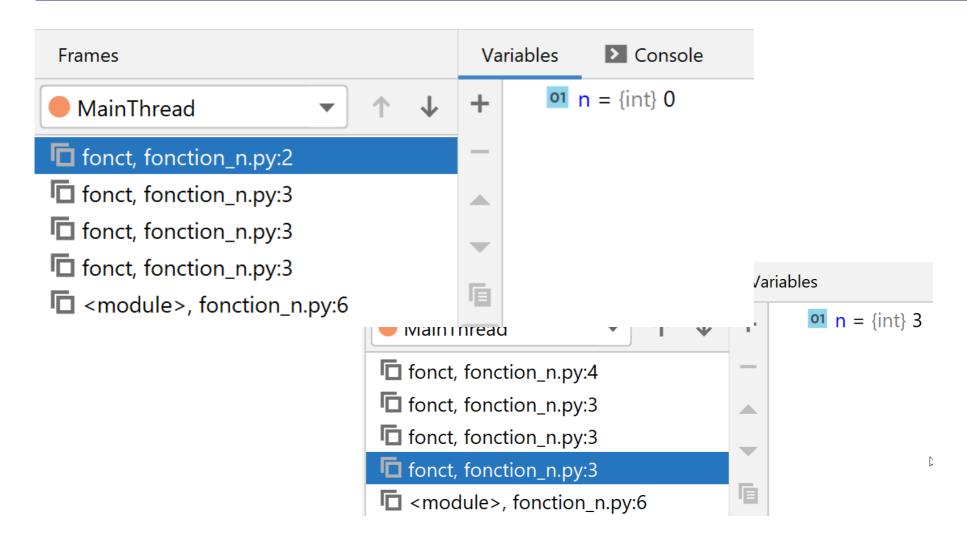
Une fonction récursive doit disposer d'une condition d'arrêt sinon Python interrompt le programme en générant une erreur quand la pile d'exécution dépasse une certaine taille.

RecursionError: maximum recursion depth exceeded while calling a Python object

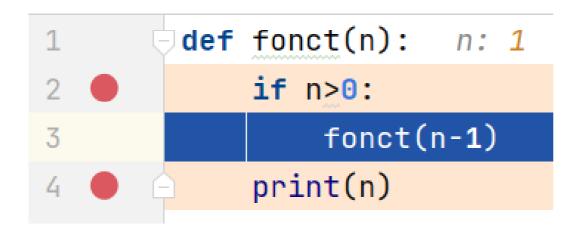
Fonction récursive

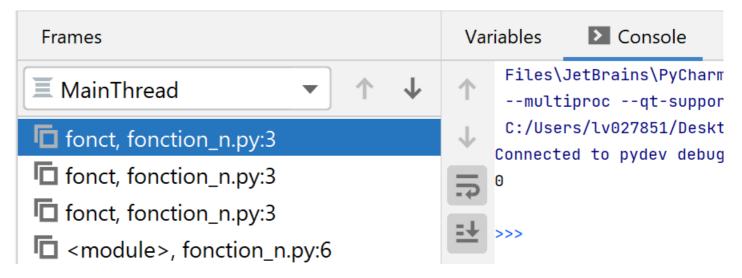
- Combien y-a-t-il d'appel(s) de fonction?
- Avec quelle valeur de n la pile contient toutes ces fonctions?
- Quelle est la condition d'arrêt qui permet le dépilement des fonctions?

4 fonctions sont empilées lorsque n = 0



Lorsque n = 0, on dépile après le print()





Exercice – Corriger la fonction fcta()

```
1 def fcta():
    print("Hello")
    fcta()
    fcta()
    fcta()
    fcta()
    fcta()
    fcta()
    fcta()
    fcta(3)
```

Fcta(n) doit afficher récursivement n fois « hello »

Repoduisez ce résultat

2) Hello
3) Hello

Fonction récursive – fact(n)

3) Récurrence mathématique

 En mathématiques une suite définie par récurrence est une suite définie par son premier terme et par une relation de récurrence, qui définit chaque terme à partir du précédent (ou des précédents lorsqu'ils existent)

 Lecture des notes de cours de la section 3

Relation de récurrence fact(n)

Condition d'arrêt;

$$u_0 = 1$$

- Récurrence;
 n entier et n > 0, u_n = n * u_{n-1}
- $u_0 = 1$ $u_1 = 1 * u_0 = 1*1 = 1$ $u_2 = 2 * u_1 = 2*1 = 2$ $u_2 = 3 * u_2 = 3*2 = 6$

Exercice – Coder la suite de Fibonacci

Condition d'arrêt;

$$u_0 = 0$$
 et $u_1 = 1$

Récurrence;
 n entier et n > 1, u_n = u_{n-1} + u_{n-2}

•
$$u_0 = 0$$
 $fib(0) = 0$ $u_1 = 1$ $fib(1) = 1$ $u_2 = u_{1+} u_0 = 0 + 1 = 1$ $fib(2) = 1$ $u_3 = u_{2+} u_1 = 1 + 1 = 2$ $fib(3) = 2$ $fib(4) = 3$