



Fonctions Récursives

✍ Étudier le programme ci-dessous, en créant un tableau qui montre l'évolution des variables pour un nombre n petit, puis expliquer le résultat obtenu en montrant la récurrence et valider votre démarche avec un nombre n supérieur à 20, en testant le programme :

```
n = int(input("Saisir un nombre entier: "))
def somme(n):
    resultat = 0
    for i in range(n):
        resultat = resultat + i*i
    return resultat

print("le résultat est:", somme(n+1))
```

L	n	somme (n+1)	résultat	i
1	2	\emptyset	\emptyset	\emptyset
2	2	$2+1=3$	\emptyset	\emptyset
3	2	"	0	\emptyset
4	"	"	0	0
5	"	"	$0+0*0=0$	0
4	"	"	0	1
5	"	"	$0+1*1=1$	1
4	"	"	1	2
5	"	"	$1+2*2=5$	2



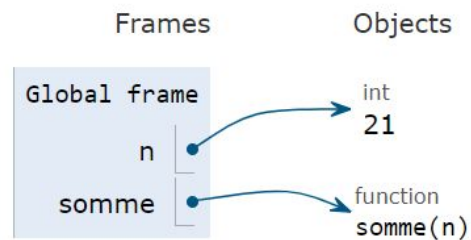


Python 3.6
[known limitations](#)

```
1 n = int(input("Saisir un nombre entier: "))
2 def somme(n):
3     resultat = 0
4     for i in range (n):
5         resultat = resultat + i*i
6     return resultat
7
→ 8 print("le résultat est:", somme(n+1))
```

Print output (drag lower right corner to resize)

```
Saisir un nombre entier: 21
le résultat est: 3311
```



Tester le programme suivant avec les deux nombres que vous avez pris précédemment :

```
n = int(input("Saisir un nombre entier: "))
def somme(n):
    if n == 0:
        resultat = 0
    else:
        resultat = somme(n-1) + n*n
    return resultat
print("le résultat est:", somme(n))
```

Python 3.6
[known limitations](#)

```
1 n = int(input("Saisir un nombre entier: "))
2 def somme(n):
3     if n == 0:
4         resultat = 0
5     else:
6         resultat = somme(n-1) + n*n
7     return resultat
8 print("le résultat est:", somme(n))
```

[Edit this code](#)

just executed
to execute

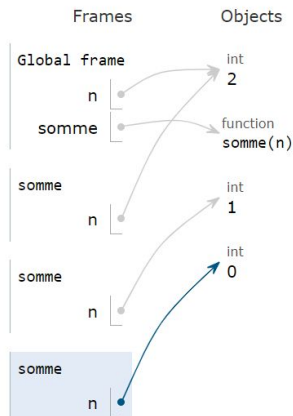
<< First < Prev Next > Last >>

Step 11 of 18

n/Pandas puzzles based on current events at [Bamboo Weekly](#)

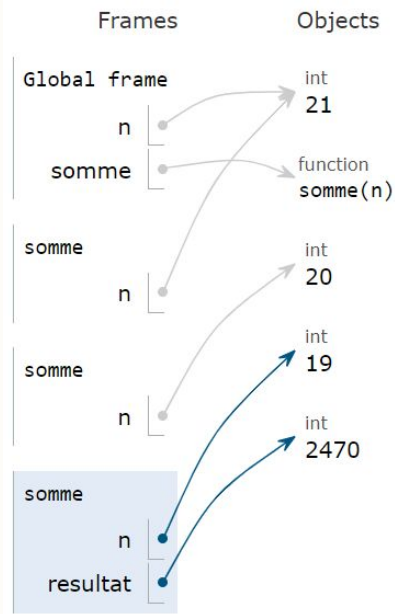
Print output (drag lower right corner to resize)

Saisir un nombre entier: 2



Print output (drag lower right corner to resize)

Saisir un nombre entier: 21



On peut voir que le deuxième programme utilise la récursivité avec “somme (n-1)” à l’instar du premier qui utilisait la méthode “tant que”





Tester le programme ci-après sur **PYTHONTUTOR** :

```
n = int(input("Saisir un nombre entier: "))  
def somme(n):  
    resultat = somme(n-1) + n*n  
    return resultat  
print("le résultat est:", somme(n))
```



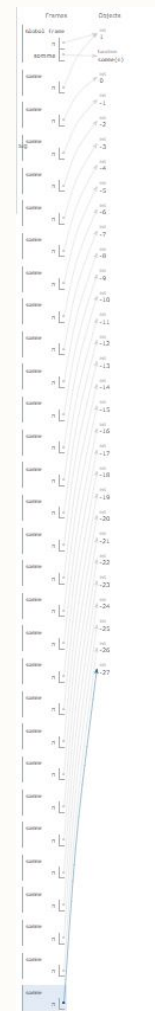
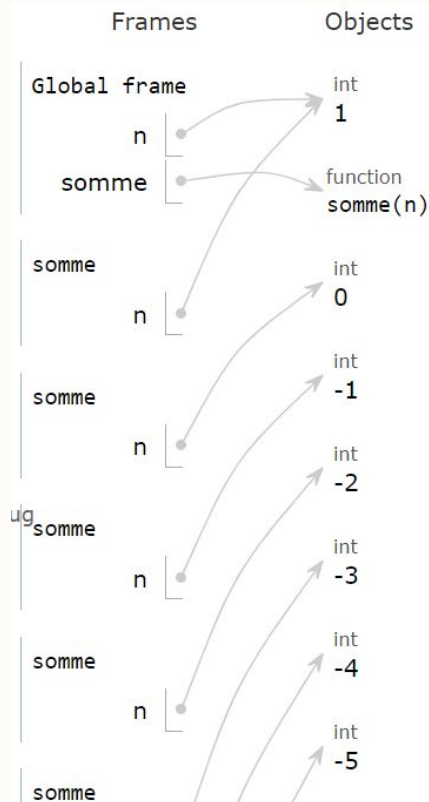
Expliquer le message d'erreur :

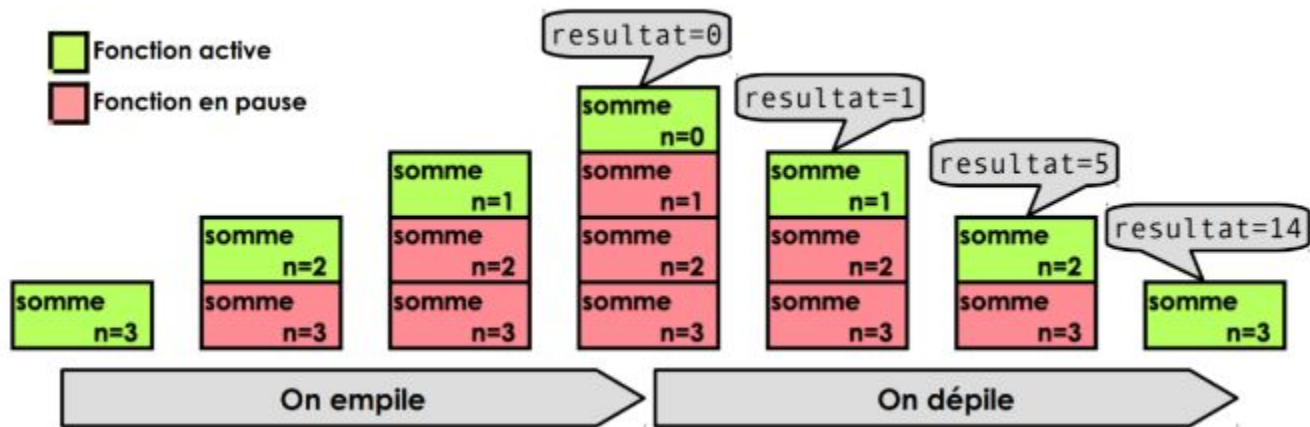
Stopped since stack has 30 functions on it.
You may have infinite recursion [#RecursionError]



Expliquer à quoi est due cette erreur et comment l'éviter.

Le message d'erreur nous indique que pythontutor a arrêté notre programme car cela a dépassé les 30 fonctions "somme" à cause de la récursivité. Cela nous explique que le programme a sans doute une récursion infinie et qu'il ne s'arrêtera jamais.





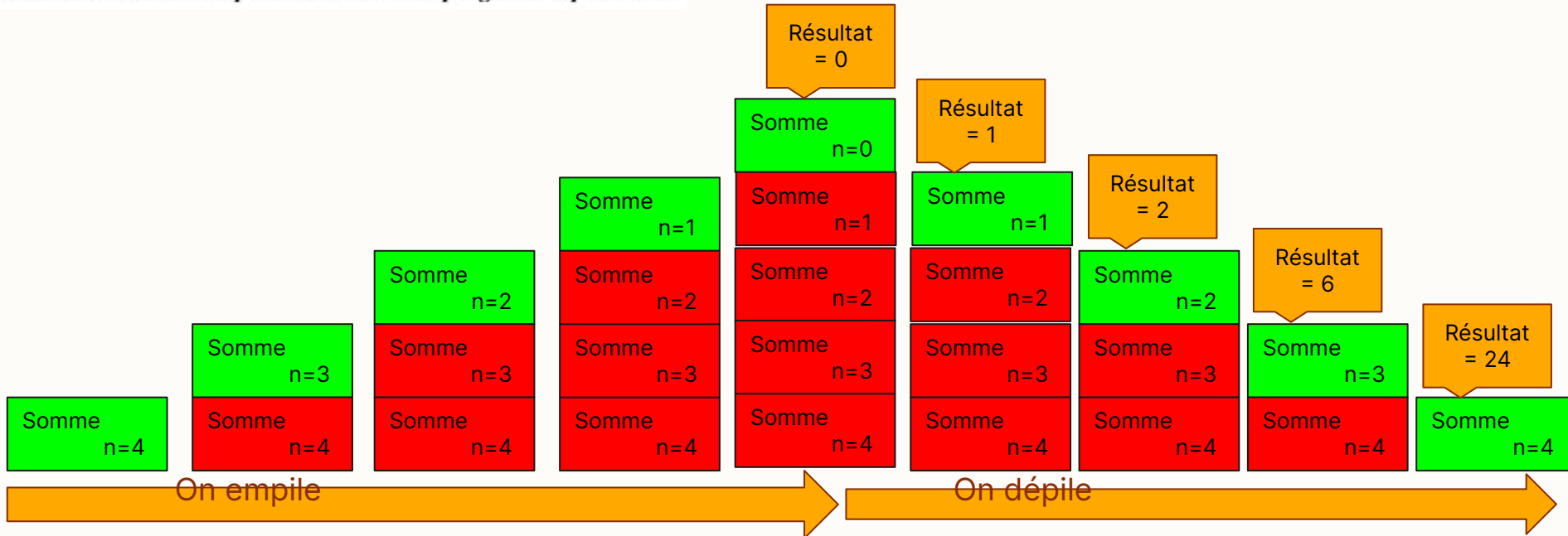
Écrire la fonction `facto(n)` qui demande une valeur `n` et renvoie le résultat du calcul suivant :

$$n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$$

```
n = int(input("Saisir un nombre entier: "))
def facto(n):
    if n == 1 or n == 0:
        return n
    else:
        return n*facto(n-1)
```

Saisir un nombre entier: 4
24

Faite un schéma de la pile d'exécution du programme pour `n=4`.



Créer une fonction récursive « *appel(n)* » qui permet d'afficher :


```
>>> appel(3)
Allô ?
Allô ?
Allô ?
Allô ?
```

Print output (drag lower right corner to resize)

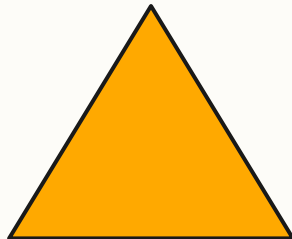
```
Saisir un nombre entier: 3
Allô ?
Allô ?
Allô ?
Allô ?
```

```
n = int(input("Saisir un nombre entier: "))
def appel(n):
    if n == 1 or n == 0:
        return "Allô ?"
    else:
        return "Allô ? \n"*(n+1)
print(appel(n))
```


5. FRACTALE ! UN EXEMPLE IDÉAL DE FONCTION RECURSIVE !

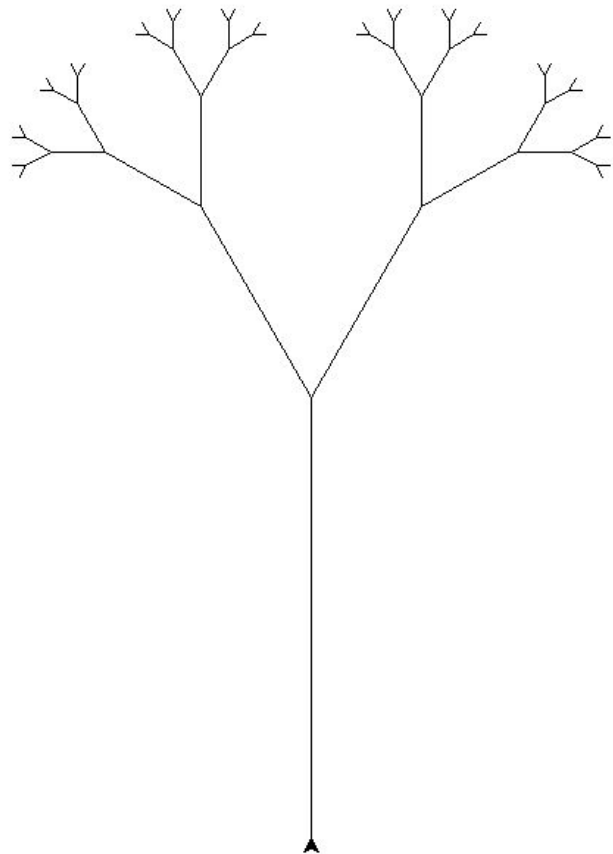
 Dessiner le résultat de l'exécution du programme :

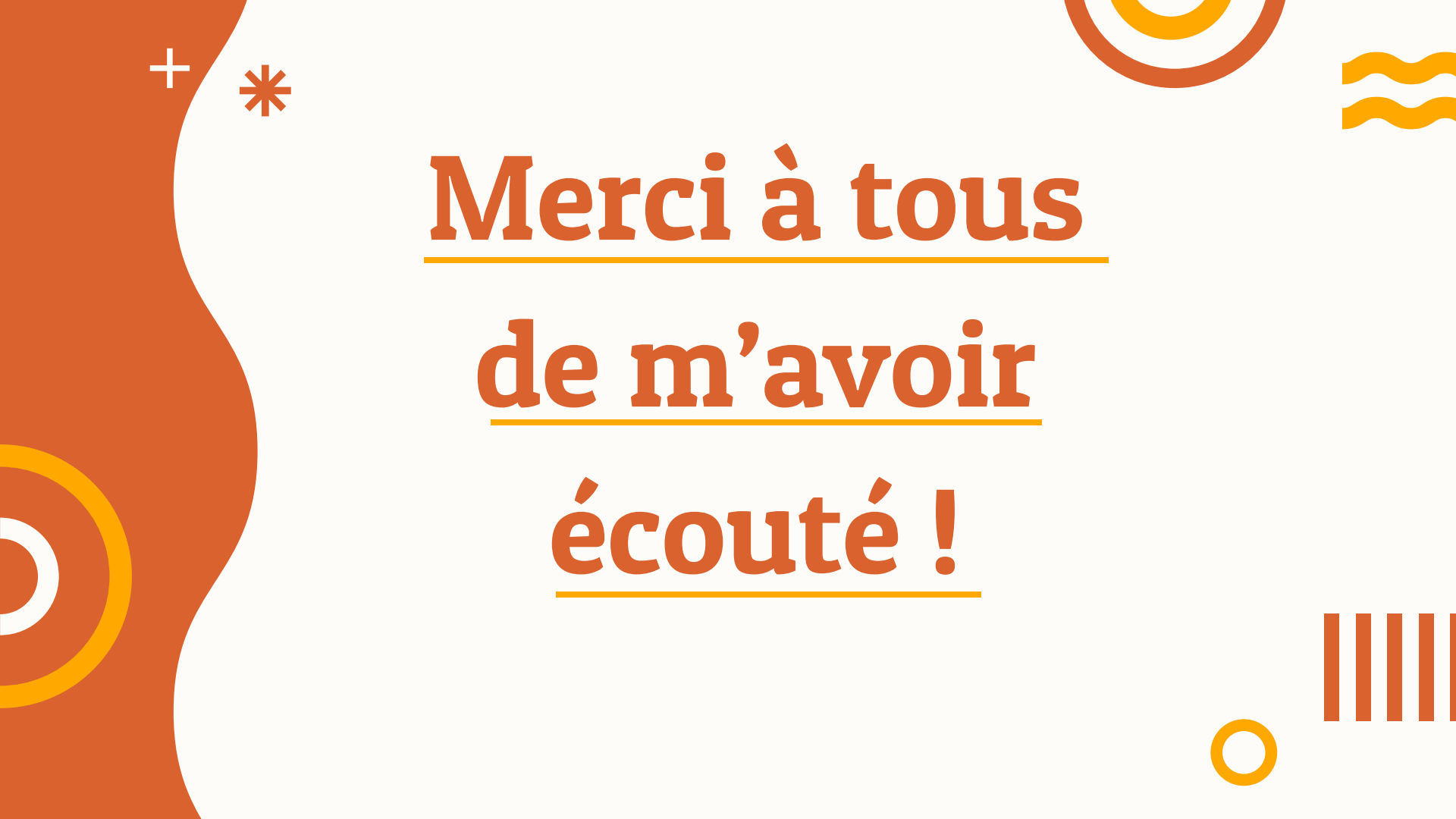
```
import turtle as t
t.forward(100)
t.left(120)
t.forward(100)
t.left(120)
t.forward(100)
```



```
from turtle import *
def arbre_fractale(l):
    if l<15:
        forward(l)
        backward(l)
    else:
        forward(l)
        left(30)
        arbre_fractale(l/2)
        right(60)
        arbre_fractale(l/2)
        left(30)
        backward(l)

left(90)
arbre_fractale(300)
mainloop()
```





Merci à tous
de m'avoir
écouté !