***misterio***

pc

1

5

n

2

5

x

3

-

**GLOBAL**

pc

10

x

-

misterio

Proc

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

**GLOBAL**

pc

10

x

-

misterio

Proc

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

4

5

n

5

4

x

6

-

***misterio***

pc

1

5

n

2

5

x

3

-

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

7

5

n

8

3

x

9

-

***misterio***

pc

4

5

n

5

4

x

6

-

***misterio***

pc

1

5

n

2

5

x

3

-

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

10

5

n

11

2

x

12

-

***misterio***

pc

1

5

n

2

5

x

3

-

***misterio***

pc

7

5

n

8

3

x

9

-

***misterio***

pc

4

5

n

5

4

x

6

-

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

13

5

n

14

1

x

15

-

***misterio***

pc

1

5

n

2

5

x

3

-

***misterio***

pc

10

5

n

11

2

x

12

-

***misterio***

pc

4

5

n

5

4

x

6

-

***misterio***

pc

7

5

n

8

3

x

9

-

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

16

3

n

17

0

***misterio***

pc

7

5

n

8

3

x

9

-

***misterio***

pc

13

5

n

14

1

x

15

-

***misterio***

pc

4

5

n

5

4

x

6

-

***misterio***

pc

10

5

n

11

2

x

12

-

***misterio***

pc

1

5

n

2

5

x

3

-

***misterio***

pc

13

7

n

14

1

x

15

- [1]

r

16

- []

y

17

-

1 def misterio(n):

2     if n == 0:

3         yield [1]

4     else:

5         for x in misterio(n - 1):

6             r = []

7             for y in suspenso(0, x):

8                 r = [\*r, y]

9             yield r

10 for x in misterio(5):

11    print(x)

**Python**

***misterio***

pc

1

5

n

2

5

x

3

-

***misterio***

pc

10

5

n

11

2

x

12

-

***misterio***

pc

7

5

n

8

3

x

9

-

***misterio***

pc

4

5

n

5

4

x

6

-

A este punto de ejecución, por motivos de espacio y tiempo, no se mostrará la ejecución interna del iterador **suspenso**. En cambio, se mostrarán inmediatamente el estado de las variables afectadas por el iterador en determinado marco de pila y al momento de hacer el **yield** para retornar un valor al marco anterior.

Por ejemplo, observe el marco de pila más reciente.

***misterio***

pc

13

7

n

14

1

x

15

- [1]

r

16

- []

y

17

-

Cuando llegamos a la línea 7, comenzamos a generar en demanda los elementos del iterador **suspenso** y los almacenamos en **yi**, donde **yi** es el valor de y en la i-esima iteración de **suspenso.**

y1 = 0 + 1

r = [1]

y2 = 1

r = [1]

**yield** r

Y bajamos al siguiente marco.

***misterio***

pc

10

5

n

11

2

x

12

-

x = [1, 1]

r = []

y1 = 0 + 1

r = [1]

y2 = 1 + 1

r = [1, 2]

y3 = 1

r = [1,2,1]

**yield r**

Y al siguiente,

x = [1 ,2 ,1]

r = []

y1 = 0 + 1 = 1

r = [1]

y2 = 1 + 2 = 3

r = [1, 3]

y3 = 2 + 1 = 3

r = [1, 3, 3]

y4 = 1

r = [1, 3, 3, 1]

**yield r**

***misterio***

pc

7

5

n

8

3

x

9

-

Y al siguiente,

x = [1, 3, 3, 1]

r = []

y1 = 0 + 1 = 1

r = [1]

y2 = 1 + 3 = 4

r = [1, 4]

y3 = 3 + 3 = 6

r = [1, 4, 6]

y4 = 3 + 1 = 4

r = [1, 4, 6, 4]

y5 = 1

r = [1, 4, 6, 4, 1]

**yield r**

***misterio***

pc

4

5

n

5

4

x

6

-

Hasta el primer marco de pila del iterador **misterio** que llamamos.

x = [1,4,6,4,1]

r = []

y1= 0 + 1 = 1

r = [1]

y2 = 1 + 4 = 5

r = [1,5]

y3 = 4 + 6 = 10

r = [1,5,10]

y4 = 6 + 4 = 10

r = [1,5,10,10]

y5 = 4 + 1 = 5

r = [1,5, 10, 10, 5]

**yield r**

***misterio***

pc

1

5

n

2

5

x

3

-

Finalmente, volvemos al alcance global del programa y, teniendo ya el valor de **x**, lo imprimimos su valor en la línea 24, lo que resulta en,

[1,5, 10, 10, 5]

Salida